Exercise Answers
For
The Free SQL Book
(Edition 0.0)

This document should ONLY be downloaded from: www.freesqlbook.com

 Copyright Pending 2020

PART I

The SELECT Statement

Chapter-1 - Getting Started: The SELECT Statement

The following three exercises (1A, 1B, and 1C) reference the PRESERVE table.

1A. Display the row with a preserve number (PNO) of 5.

```
SELECT *
FROM  PRESERVE
WHERE PNO  = 5
```

The result should look like:

PNO	PNAME	STATE	ACRES	FEE
5	HASSAYAMPA RIVER	AZ	660	3.00

1B. Display all information about any nature preserve that does not charge an admission fee (i.e., the admission fee is zero).

```
SELECT *
FROM PRESERVE
WHERE FEE = 0.00
```

The result should contain the following rows.

PNO	PNAME	STATE	ACRES	FEE
3	DANCING PRAIRIE	MT	680	0.00
7	MULESHOE RANCH	AZ	49120	0.00
40	SOUTH FORK MADISON	MT	121	0.00
14	MCELWAIN-OLSEN	MA	66	0.00
13	TATKON	MA	40	0.00
9	DAVID H. SMITH	MA	830	0.00
11	MIACOMET MOORS	MA	4	0.00
12	MOUNT PLANTAIN	MA	730	0.00
1	COMERTOWN PRAIRIE	MT	1130	0.00
2	PINE BUTTE SWAMP	MT	15000	0.00
10	HOFT FARM	MA	90	0.00

 Copyright Pending 2020

1C. Display all information about any nature preserve that is larger than 1,000 acres.

```
SELECT *
FROM PRESERVE
WHERE ACRES > 1000
```

The result should contain the following rows.

```
PNO PNAME                     STATE ACRES FEE
  7 MULESHOE RANCH            AZ    49120 0.00
  1 COMERTOWN PRAIRIE         MT     1130 0.00
  2 PINE BUTTE SWAMP          MT    15000 0.00
  6 PAPAGONIA-SONOITA CREEK AZ      1200 3.00
```

The following Exercise 1D references another table.

1D. The sample database contains a table called EMPLOYEE. Assume you know nothing about this table except that it is very small. Display all data in this table.

```
SELECT *
FROM EMPLOYEE
```

The result should contain the following rows.

```
ENO  ENAME  SALARY  DNO
1000 MOE    2000.00  20
2000 LARRY  2000.00  10
3000 CURLY  3000.00  20
4000 SHEMP   500.00  40
5000 JOE     400.00  10
6000 GEORGE 9000.00  20
```

 Copyright Pending 2020

1E. Display all information about any nature preserve located in Montana.

```
SELECT *
FROM   PRESERVE
WHERE STATE = 'MT'
```

The result should contain the following rows.

PNO	PNAME	STATE	ACRES	FEE
3	DANCING PRAIRIE	MT	680	0.00
40	SOUTH FORK MADISON	MT	121	0.00
1	COMERTOWN PRAIRIE	MT	1130	0.00
2	PINE BUTTE SWAMP	MT	15000	0.00

1F. Display all information about the Pine Butte Swamp preserve.

```
SELECT *
FROM   PRESERVE
WHERE PNAME = 'PINE BUTTE SWAMP'
```

The result should look like:

PNO	PNAME	STATE	ACRES	FEE
2	PINE BUTTE SWAMP	MT	15000	0.00

 Copyright Pending 2020

1G. Display the preserve number and name, in that left-to-right order, of all nature preserves.

```
SELECT PNO, PNAME
FROM   PRESERVE
```

The result should contain the following rows.:

```
PNO  PNAME
  5  HASSAYAMPA RIVER
  3  DANCING PRAIRIE
  7  MULESHOE RANCH
 40  SOUTH FORK MADISON
 14  MCELWAIN-OLSEN
 13  TATKON
  9  DAVID H. SMITH
 11  MIACOMET MOORS
 12  MOUNT PLANTAIN
 1  COMERTOWN PRAIRIE
 2  PINE BUTTE SWAMP
 80  RAMSEY CANYON
 10  HOFT FARM
  6  PAPAGONIA-SONOITA CREEK
```

 Copyright Pending 2020

1H. Display the state code and preserve name, in that left-to-right order, of all nature preserves.

```
SELECT STATE, PNAME
FROM   PRESERVE
```

The result should contain the following rows.

```
STATE  PNAME
AZ     HASSAYAMPA RIVER
MT     DANCING PRAIRIE
AZ     MULESHOE RANCH
MT     SOUTH FORK MADISON
MA     MCELWAIN-OLSEN
MA     TATKON
MA     DAVID H. SMITH
MA     MIACOMET MOORS
MA     MOUNT PLANTAIN
MT     COMERTOWN PRAIRIE
MT     PINE BUTTE SWAMP
AZ     RAMSEY CANYON
MA     HOFT FARM
AZ     PAPAGONIA-SONOITA CREEK
```

1I. Display the preserve number and name for all nature preserves where the number of acres exceeds 2,000.

```
SELECT PNO, PNAME
FROM   PRESERVE
WHERE ACRES > 2000
```

The result should contain the following rows.:

```
PNO  PNAME
  7  MULESHOE RANCH
  2  PINE BUTTE SWAMP
```

 Copyright Pending 2020

1J. Display the preserve name of all nature preserves located in Massachusetts.

```
SELECT PNAME
FROM   PRESERVE
WHERE STATE  =  'MA'
```

The result should contain the following rows.

```
PNAME
MCELWAIN-OLSEN
TATKON
DAVID H. SMITH
MIACOMET MOORS
MOUNT PLANTAIN
HOFT FARM
```

Optional Exercise:

1K. **Optional (and Unfair) Exercise:** Review the SELECT statement and result table for Sample Query 1.4. This SELECT statement is:

```
SELECT PNAME, ACRES, STATE
FROM PRESERVE
```

This result table does not show any duplicate rows. However, sometime in the future, in a very unusual circumstance, this result table could contain duplicate rows. Why might this happen?

Although it is *highly unlikely*, a new preserve could be assigned the same name, be located in the same state, and have the same number of acres. This is possible because neither PNAME, nor ACRES, nor STATE is defined as a UNIQUE column. In this unusual circumstance, execution of the above SELECT statement would display duplicate rows.

Summary Exercises (Chapter 1)

Exercise 1D asked you to display the EMPLOYEE table.
The result should contain the following rows.

ENO	ENAME	SALARY	DNO
1000	MOE	2000.00	20
2000	LARRY	2000.00	10
3000	CURLY	3000.00	20
4000	SHEMP	500.00	40
5000	JOE	400.00	10
6000	GEORGE	9000.00	20

The following exercises reference the EMPLOYEE table. This table has four columns that are described below.

ENO (Employee Number) Fixed-length character string: CHAR (4)
This column contains unique values.
Note: This "number" is represented by a character-string.

ENAME (Employee Name) Variable-length character string: VARCHAR (25)

SALARY (Employee Salary) Decimal: DECIMAL (7,2)

DNO (Employee's Department Number) Integer: INTEGER

1L. Display all information about any employee whose SALARY value exceeds $1,000.00.

```
SELECT *
FROM EMPLOYEE
WHERE SALARY > 1000.00
```

The result should contain the following rows.

ENO	ENAME	SALARY	DNO
1000	MOE	2000.00	20
2000	LARRY	2000.00	10
3000	CURLY	3000.00	20
6000	GEORGE	9000.00	20

 Copyright Pending 2020

1M. Display all information about Employee 2000 (i.e., ENO value is '2000').

```
SELECT *
FROM EMPLOYEE
WHERE ENO = '2000'
```

The result should look like:

```
ENO  ENAME SALARY   DNO
2000 LARRY 2000.00   10
```

1N. Display the ENAME and DNO values of every employee.

```
SELECT ENAME, DNO
FROM   EMPLOYEE
```

The result should contain the following rows.

```
ENAME      DNO
MOE         20
LARRY       10
CURLY       20
SHEMP       40
JOE         10
GEORGE      20
```

1O. Display the ENAME and SALARY values of every employee whose SALARY value is less than $1,000.00.

```
SELECT ENAME, SALARY
FROM   EMPLOYEE
WHERE SALARY < 1000.00
```

The result should contain the following rows.

```
ENAME  SALARY
SHEMP  500.00
JOE    400.00
```

Chapter-2 – Sorting the Result Table: ORDER BY

2A. Display the entire PRESERVE table. Sort the result by the ACRES column in ascending sequence.

```
SELECT *
FROM  PRESERVE
ORDER BY ACRES
```

The result should look like:

PNO	PNAME	STATE	ACRES	FEE
11	MIACOMET MOORS	MA	4	0.00
13	TATKON	MA	40	0.00
14	MCELWAIN-OLSEN	MA	66	0.00
10	HOFT FARM	MA	90	0.00
40	SOUTH FORK MADISON	MT	121	0.00
80	RAMSEY CANYON	AZ	380	3.00
5	HASSAYAMPA RIVER	AZ	660	3.00
3	DANCING PRAIRIE	MT	680	0.00
12	MOUNT PLANTAIN	MA	730	0.00
9	DAVID H. SMITH	MA	830	0.00
1	COMERTOWN PRAIRIE	MT	1130	0.00
6	PAPAGONIA-SONOITA CREEK	AZ	1200	3.00
2	PINE BUTTE SWAMP	MT	15000	0.00
7	MULESHOE RANCH	AZ	49120	0.00

2B. Display the preserve name and admission fee of every nature preserve located in Montana. Sort the result by preserve name in descending sequence.

```
SELECT PNAME, FEE
FROM   PRESERVE
WHERE STATE = 'MT'
ORDER BY PNAME DESC
```

The result should look like:

PNAME	FEE
SOUTH FORK MADISON	0.00
PINE BUTTE SWAMP	0.00
DANCING PRAIRIE	0.00
COMERTOWN PRAIRIE	0.00

2C. Display the FEE and ACRES columns (in that order) for every row in the PRESERVE table. Sort the displayed rows by ACRES within FEE. (FEE is the major sort field, and ACRES is the minor sort field.)

```
SELECT FEE, ACRES
FROM   PRESERVE
ORDER BY FEE, ACRES
```

The result should look like:

FEE	ACRES
0.00	4
0.00	40
0.00	66
0.00	90
0.00	121
0.00	680
0.00	730
0.00	830
0.00	1130
0.00	15000
0.00	49120
3.00	380
3.00	660
3.00	1200

 Copyright Pending 2020

2D. Display the entire PRESERVE table sorted by the fourth column in descending sequence.

```
SELECT *
FROM  PRESERVE
ORDER BY 4 DESC
```

The result should look like:

PNO	PNAME	STATE	ACRES	FEE
7	MULESHOE RANCH	AZ	49120	0.00
2	PINE BUTTE SWAMP	MT	15000	0.00
6	PAPAGONIA-SONOITA CREEK	AZ	1200	3.00
1	COMERTOWN PRAIRIE	MT	1130	0.00
9	DAVID H. SMITH	MA	830	0.00
12	MOUNT PLANTAIN	MA	730	0.00
3	DANCING PRAIRIE	MT	680	0.00
5	HASSAYAMPA RIVER	AZ	660	3.00
80	RAMSEY CANYON	AZ	380	3.00
40	SOUTH FORK MADISON	MT	121	0.00
10	HOFT FARM	MA	90	0.00
14	MCELWAIN-OLSEN	MA	66	0.00
13	TATKON	MA	40	0.00
11	MIACOMET MOORS	MA	4	0.00

 Copyright Pending 2020

2E. Assume (unrealistically) that PNO values are considered to be confidential. Display the PNAME value for each Arizona nature preserve. Display the result in ascending PNO sequence without displaying the PNO values.

```
SELECT PNAME
FROM   PRESERVE
WHERE STATE = 'AZ'
ORDER BY PNO
```

The result should look like:

```
PNAME
HASSAYAMPA RIVER
PAPAGONIA-SONOITA CREEK
MULESHOE RANCH
RAMSEY CANYON
```

2F. Display the STATE, FEE, and PNO values for any preserve having more than 100 acres. Sort the result table. STATE is the first-level sort field in descending sequence. FEE is the second-level sort field in descending sequence. PNO is the third-level sort field in ascending sequence.

```
SELECT STATE, FEE, PNO
FROM   PRESERVE
WHERE ACRES > 100
ORDER BY STATE DESC, FEE DESC, PNO
```

The result should look like:

STATE	FEE	PNO
MT	0.00	1
MT	0.00	2
MT	0.00	3
MT	0.00	40
MA	0.00	9
MA	0.00	12
AZ	3.00	5
AZ	3.00	6
AZ	3.00	80
AZ	0.00	7

2G. Display all rows where the STATE value is greater than or equal to the letter M.

```
SELECT *
FROM PRESERVE
WHERE STATE >= 'M'
```

The result should contain the following rows.

PNO	PNAME	STATE	ACRES	FEE
3	DANCING PRAIRIE	MT	680	0.00
40	SOUTH FORK MADISON	MT	121	0.00
14	MCELWAIN-OLSEN	MA	66	0.00
13	TATKON	MA	40	0.00
9	DAVID H. SMITH	MA	830	0.00
11	MIACOMET MOORS	MA	4	0.00
12	MOUNT PLANTAIN	MA	730	0.00
1	COMERTOWN PRAIRIE	MT	1130	0.00
2	PINE BUTTE SWAMP	MT	15000	0.00
10	HOFT FARM	MA	90	0.00

 Copyright Pending 2020

2H. Display every row where the preserve name value is less than TATKON.

```
SELECT *
FROM PRESERVE
WHERE PNAME < 'TATKON'
```

The result should contain the following rows.

PNO	PNAME	STATE	ACRES	FEE
5	HASSAYAMPA RIVER	AZ	660	3.00
3	DANCING PRAIRIE	MT	680	0.00
7	MULESHOE RANCH	AZ	49120	0.00
40	SOUTH FORK MADISON	MT	121	0.00
14	MCELWAIN-OLSEN	MA	66	0.00
9	DAVID H. SMITH	MA	830	0.00
11	MIACOMET MOORS	MA	4	0.00
12	MOUNT PLANTAIN	MA	730	0.00
1	COMERTOWN PRAIRIE	MT	1130	0.00
2	PINE BUTTE SWAMP	MT	15000	0.00
80	RAMSEY CANYON	AZ	380	3.00
10	HOFT FARM	MA	90	0.00
6	PAPAGONIA-SONOITA CREEK	AZ	1200	3.00

 Copyright Pending 2020

Summary Exercises (Chapter 2)

The following three exercises pertain to the previously described EMPLOYEE table. Its column-names are ENO, ENAME, SALARY, and DNO.

2I. Display the entire EMPLOYEE table sorted by employee name in ascending sequence.

```
SELECT *
FROM   EMPLOYEE
ORDER BY ENAME
```

The result should look like:

```
ENO  ENAME    SALARY  DNO
3000 CURLY   3000.00   20
6000 GEORGE  9000.00   20
5000 JOE      400.00   10
2000 LARRY   2000.00   10
1000 MOE     2000.00   20
4000 SHEMP    500.00   40
```

2J. Display the name and salary of any employee whose salary is greater than $2,000.00. Sort the result by salary in descending sequence.

```
SELECT ENAME, SALARY
FROM EMPLOYEE
WHERE SALARY > 2000.00
ORDER BY ENAME DESC
```

The result should look like:

```
ENAME    SALARY
GEORGE  9000.00
CURLY   3000.00
```

 Copyright Pending 2020

2K. Display the department number, employee number, and employee name of all employees. Sort the result by employee number (in ascending sequence) within department number (in descending sequence).

```
SELECT DNO, ENO, ENAME
FROM EMPLOYEE
ORDER BY DNO DESC, ENO
```

The result should look like:

```
DNO  ENO   ENAME
 40  4000  SHEMP
 20  1000  MOE
 20  3000  CURLY
 20  6000  GEORGE
 10  2000  LARRY
 10  5000  JOE
```

2L. Do Sample Queries 2.3 - 2.5 return deterministic result tables?

SQ 2.3:
```
SELECT PNO, PNAME, ACRES
FROM   PRESERVE
WHERE STATE = 'AZ'
ORDER BY PNO DESC
```

SQ 2.4:
```
SELECT PNO, ACRES, PNAME
FROM   PRESERVE
WHERE STATE = 'AZ'
ORDER BY 3
```

SQ 2.5:
```
SELECT PNO, PNAME
FROM   PRESERVE
ORDER BY ACRES DESC
```

All result tables are deterministic because each result table displays unique PNO values.

 Copyright Pending 2020

Chapter-3 – Eliminating Duplicate Rows: DISTINCT

3A1. Retrieve every row in PRESERVE. Only display the FEE value for each row. (Do not attempt to remove duplicate rows.) Before you execute the SELECT statement for this exercise, ask yourself the following question. "Can duplicate values possibly appear in this result?"

```
SELECT FEE
FROM  PRESERVE
```

Duplicate FEE values are obvious when you display PRESERVE. More precisely, you have not been told that the FEE column is unique. Therefore, you should assume that duplicate values might be present.

3A2. Retrieve every row in PRESERVE. Only display the ACRES value for each row. (Do not attempt to remove duplicate rows.) Before you execute the SELECT statement for this exercise, ask yourself the following question. "Can duplicate values possibly appear in this result?"

```
SELECT ACRES
FROM  PRESERVE
```

Examination of PRESERVE does not show duplicate ACRES values. And, it is highly unlikely that any two preserves will ever have the exact same number of acres. However, you have not been told that ACRES is unique. Therefore, you should assume that duplicate values can possibly occur sometime in the future.

3B. Display all admission fees in the PRESERVE table. Do not display duplicate values.

```
SELECT DISTINCT FEE
FROM PRESERVE
```

 Copyright Pending 2020

3C. Display the FEE and ACRES values for every row in the PRESERVE table. Before you execute the SELECT statement for this exercise, ask yourself the following question. “Can duplicate rows possibly appear in this result?” What know-your-data insights help you answer this question?

```
SELECT FEE, ACRES
FROM   PRESERVE
```

Currently, there are no duplicate ACRES values in PRESERVE. Hence, there are no duplicate pairs of (FEE, ACRES) values. In this circumstance, executing the above statement would not produce duplicate rows. However, *because neither column is defined as unique,* future updates to PRESERVE could introduce duplicate pairs of (FEE, ACRES) values.

3D. Display the FEE and ACRES values for every row in the PRESERVE table. Do not display duplicate rows in the result table.

```
SELECT DISTINCT FEE, ACRES
FROM PRESERVE
```

3E. Optional Exercise: Remove the ORDER BY clause shown in the above Sample Query 3.4 such that it looks like:

```
SELECT DISTINCT STATE, FEE
FROM PRESERVE
```

Execute this statement. Most likely you will observe that the result table is incidentally sorted. If this sort occurs, the first-level sort could be on either the STATE column or the FEE column.

If FEE is the first-level sort column, the sorted result would look like:

STATE	FEE
AZ	0.00
MA	0.00
MT	0.00
AZ	3.00

If STATE is the first-level sort column, the sorted result would look like:

STATE	FEE
AZ	0.00
AZ	3.00
MA	0.00
MT	0.00

3F. The following two statements return the same rows. Will these rows be in the same row sequence? Answer: Yes, No, or Maybe.

```
SELECT PNO
FROM PRESERVE;

SELECT DISTINCT PNO
FROM PRESERVE;
```

Maybe. These statements return the same rows because PNO is unique. Hence the second SELECT statement does not remove any duplicates.

These rows may or may not be in the same sequence because neither statement specifies an ORDER BY clause. Therefore, if you execute these statements, an incidental sort could appear in either or both of the result tables.

Summary Exercises: (Chapter-3)

The following exercises pertain to the EMPLOYEE table.

3G. Display all DNO values in the EMPLOYEE table. Do not display duplicate values.

```
SELECT DISTINCT DNO
FROM EMPLOYEE
```

```
DNO
 10
 20
 40
```

Rows may appear in any order. The above result is incidental sorted.

 Copyright Pending 2020

3H. Execute each of the following statements. Examine the result tables and make relevant observations.

```
SELECT DNO, SALARY
FROM   EMPLOYEE
```

DNO	SALARY
20	2000.00
10	2000.00
20	3000.00
40	500.00
10	400.00
20	9000.00

Currently, there are no duplicate pairs of (DNO, SALARY) values.
But future execution of this statement could display duplicate rows because future update operations could produce duplicates pairs of (DNO, SALARY) values.
The absence of an ORDER BY clause implies that rows may appear in any sequence.

```
SELECT DISTINCT DNO, SALARY
FROM EMPLOYEE
```

Currently, because there are no duplicate pairs of (DNO, SALARY) values, this statement returns the same rows shown above. However, DISTINCT may cause an incidental sort.

```
SELECT DISTINCT DNO, SALARY
FROM EMPLOYEE
ORDER BY DNO, SALARY
```

DNO	SALARY
10	400.00
10	2000.00
20	2000.00
20	3000.00
20	9000.00
40	500.00

This may be the best way to code this statement.
Duplicate rows cannot appear, and row sequence is explicitly specified.

Chapter-4 Boolean Connectors: AND-OR-NOT

4A. Display all information about any nature preserve in Montana that is smaller than 1,000 acres.

```
SELECT * FROM PRESERVE
WHERE STATE = 'MT' AND ACRES < 1000
```

4B. Display all information about any nature preserve that has an ACRES value between and including 1200 and 15000.

```
SELECT * FROM PRESERVE
WHERE ACRES >= 1200 AND ACRES <= 15000
```

4C. Display all information about any nature preserve that is located in Montana, does not have an admission fee, and is greater than 10,000 acres.

```
SELECT * FROM PRESERVE
WHERE STATE = 'MT' AND FEE = 0.00 AND ACRES > 10000
```

4D. Display all information about nature preserves located in Montana or Massachusetts.

```
SELECT * FROM PRESERVE
WHERE STATE = 'MT' OR STATE = 'MA'
```

4E. Select all information about any nature preserve located in Montana or any preserve that is less than 1,000 acres.

```
SELECT * FROM PRESERVE
WHERE STATE = 'MT' OR ACRES < 1000
```

4F. Select the preserve number and name of any nature preserve having an admission fee that is not equal to zero. Use the keyword NOT in your solution.

```
SELECT PNO, PNAME FROM PRESERVE
WHERE NOT FEE = 0.00
```

4G. Same as the preceding example: Specify a not-equal symbol in your WHERE-condition.

```
SELECT PNO, PNAME FROM PRESERVE
WHERE FEE <> 0.00
```

4H. Display the preserve number and name of those nature preserves that do not have an admission fee of $3.00 and do not have a fee of $10.00.

.

```
SELECT PNO, PNAME FROM PRESERVE
WHERE (NOT FEE =  3.00)   AND     (NOT FEE  = 10.00)

SELECT PNO, PNAME FROM PRESERVE
WHERE FEE <> 3.00  AND   FEE  <> 10.00
```

4I. Display all information about any nature preserve located in Arizona that does not have an admission fee, or any preserve that is smaller than 100 acres (regardless of its STATE and FEE values).

```
SELECT * FROM PRESERVE
WHERE (STATE = 'AZ' AND FEE = 0.00)
OR         ACRES  <  100
```

4J. Display all information about any nature preserve that is smaller than 1,000 acres, and has an admission fee of zero dollars or is located in Arizona.

```
SELECT * FROM PRESERVE
WHERE ACRES < 1000
AND     (FEE = 0.00 OR STATE = 'AZ')
```

4K. Select all information about any nature preserve with an admission fee that is not greater than zero, or any other preserve, regardless of its fee, that is located in Montana and is larger than 1,000 acres.

```
SELECT * FROM PRESERVE
WHERE (NOT FEE > 0.00) OR (STATE = 'MT' AND ACRES > 1000)

SELECT * FROM PRESERVE
WHERE   FEE <= 0.00  OR  (STATE = 'MT' AND ACRES > 1000)
```

4L. Display all information about every nature preserve except those Montana preserves without an admission fee.

```
SELECT * FROM PRESERVE
WHERE NOT (STATE = 'MT' AND FEE = 0.00)
```

4M. Consider the following modified WHERE-clauses (without parentheses) for Sample Queries 4.8, 4.9 and 4.10. Which of the following modified WHERE-clauses will satisfy the specified query objectives?

Sample Query 4.8

The sample query showed:

```
WHERE ACRES > 1000 OR (STATE = 'AZ' AND FEE = 3.00)
```

The modified WHERE-clause is:

```
WHERE ACRES > 1000 OR STATE = 'AZ' AND FEE = 3.00
```

These WHERE-clauses are equivalent because the default hierarchy specifies AND before OR.

Sample Query 4.9

The sample query showed:

```
WHERE (ACRES > 1000 OR STATE = 'AZ') AND FEE = 3.00
```

The modified WHERE-clause is:

```
WHERE ACRES > 1000 OR STATE = 'AZ' AND FEE = 3.00
```

These WHERE-clauses are *not* equivalent because the default hierarchy specifies AND before OR.

Sample Query 4.10

The sample query showed:

```
WHERE NOT (STATE = 'AZ' AND FEE = 3.00)
```

The modified WHERE-clause is:

```
WHERE NOT STATE = 'AZ' AND FEE = 3.00
```

These WHERE-clauses are *not* equivalent. In the first WHERE-clause, the NOT applies to the result of the AND operation. In the second WHERE-clause, the NOT only applies to the STATE = 'AZ' condition.

The Distributed Laws apply to the following exercises.

4N1. Are the following WHERE-clauses logically equivalent?

```
WHERE STATE = 'MA' AND (ACRES > 1000 OR FEE = 0.0)

WHERE  (STATE = 'MA' AND ACRES > 1000)
          OR
         (STATE = 'MA' AND FEE = 0.0)
```

These are equivalent WHERE-clauses.

4N2. Are the following WHERE-clauses logically equivalent?

```
WHERE STATE = 'MA' OR (ACRES > 1000 AND FEE = 0.0)

WHERE (STATE = 'MA' OR ACRES > 1000)
          AND
          (STATE = 'MA' OR FEE = 0.0)
```

These are equivalent WHERE-clauses.

De Morgan's Laws apply to the following exercises.

4O1. Are the following WHERE-clauses logically equivalent??

WHERE NOT (ACRES < 50 AND STATE = 'MA')

WHERE NOT ACRES < 50 **AND** NOT STATE = 'MA'

No.
Applying De Morgan's Laws to the first WHERE-clause would produce:

WHERE NOT ACRES < 50 **OR** NOT STATE = 'MA'

4O2. Are the following WHERE-clauses logically equivalent?

WHERE NOT (ACRES < 50 AND STATE = 'MA')

WHERE ACRES >= 50 OR STATE <> 'MA'

Yes.
Applying De Morgan's Laws to the first WHERE-clause produces:

WHERE NOT ACRES < 50 OR NOT STATE = 'MA'

Then, removing the NOT keywords produce the second WHERE-clause.

4P. Are the following WHERE-clauses logically equivalent?

WHERE NOT (ACRES < 50 OR STATE = 'MA')

WHERE NOT ACRES < 50 **OR** NOT STATE = 'MA'

No.
Applying De Morgan's Laws to the first WHERE-clause would produce:

WHERE NOT ACRES < 50 **AND** NOT STATE = 'MA'

4Q. Are the following WHERE-clauses logically equivalent?

WHERE NOT (ACRES < 50 OR STATE = 'MA')

WHERE NOT ACRES < 50 AND NOT STATE = 'MA'

Yes.

 Copyright Pending 2020

Summary Exercises (Chapter 4)

The following exercises 4R-4T reference the EMPLOYEE table.

4R. Display all information about any employee who works in Department 20 and earns less than $5,000.00.

```
SELECT *
FROM   EMPLOYEE
WHERE DNO = 20
AND    SALARY < 5000.00
```

4S. Display the name and salary of any employee who earns less than $1,000.00 or more than $6,000.00.

```
SELECT ENAME, SALARY
FROM   EMPLOYEE
WHERE SALARY < 1000.00
OR       SALARY > 6000.00
```

4T. Display the name and department number of all employees who do not work for Department 20. Sort the result in ascending sequence by employee name.

```
SELECT ENAME, DNO
FROM EMPLOYEE
WHERE NOT DNO = 20
ORDER BY ENAME;
```

```
SELECT ENAME, DNO
FROM EMPLOYEE
WHERE DNO <> 20
ORDER BY ENAME;
```

 Copyright Pending 2020

The following exercise references the PRESERVE table. It is relatively complex. Yet it should be doable with a little thought.

4U. OR means Inclusive-OR. Code an "Exclusive- OR" for the following query which is a modification of Sample Query 4.5

Display the PNAME, ACRES, and STATE value of any preserve that matches just one (but not both) of the following conditions. (1) The preserve is located in Arizona, or (2) the preserve has more than 1000 acres. The result should look like:

PNAME	ACRES	STATE
HASSAYAMPA RIVER	660	AZ
COMERTOWN PRAIRIE	1130	MT
PINE BUTTE SWAMP	15000	MT
RAMSEY CANYON	380	AZ

Hint: Assume you have two conditions, C1 and C2. The most direct way to think about the Exclusive-OR is:

The first condition (C1) is True or the second condition (C2) is True.
AND
It is not the case that both conditions are True.

Another way to think about the exclusive-OR is:

The first condition is True and the second condition is False.
OR
The first condition is False and the second condition is True.

First hint implies:
```
(C1 OR C2)
AND
NOT (C1 AND C2)
```

Solution-1:
```
SELECT PNAME, ACRES, STATE
FROM   PRESERVE
WHERE   (STATE = 'AZ' OR ACRES > 1000)
AND       NOT (STATE = 'AZ' AND ACRES > 1000)
```

Second hint implies:
```
(C1 AND NOT C2)
OR
(NOT C1 AND C2)
```

Solution-2:
```
SELECT PNAME, ACRES, STATE
FROM   PRESERVE
WHERE  (STATE = 'AZ'  AND NOT ACRES > 1000)
OR         (NOT STATE = 'AZ'   AND ACRES > 1000)
```

= = = = = = =

Two other solutions can be derived from Solution-1 are shown below

Solution-3 (Apply DeMorgan's Laws)

```
SELECT PNAME, ACRES, STATE
FROM   PRESERVE
WHERE  (STATE = 'AZ' OR ACRES > 1000)
AND      (NOT STATE = 'AZ' OR NOT ACRES > 1000)
```

Solution-4

```
SELECT PNAME, ACRES, STATE
FROM   PRESERVE
WHERE          (STATE = 'AZ'    OR ACRES > 1000)
AND              (STATE <> 'AZ' OR ACRES <= 1000)
```

= = = = = = =

Another solution can be derived from Solution-2 is shown below

Solution-5 (Eliminate NOT keyword_

```
SELECT PNAME, ACRES, STATE
FROM   PRESERVE
WHERE          (STATE = 'AZ'    AND ACRES <= 1000)
OR               (STATE <> 'AZ' AND ACRES > 1000)
```

4V. Optional Exercise: Draw a truth table for the Exclusive-OR.

C1	C2	C1 XOR C2
T	T	F
T	F	T
F	T	T
F	F	F

 Copyright Pending 2020

Chapter-5 IN & BETWEEN

5A. Display all information about any nature preserve that has a preserve number in the set {2, 4, 6, 8, 10}.

```
SELECT * FROM PRESERVE
WHERE PNO IN (2, 4, 6, 8, 10)
```

5B. Display all information about the following nature preserves: DANCING PRAIRIE, MULESHOE RANCH, MCELWAIN-OLSEN, and TATKON.

```
SELECT * FROM PRESERVE
WHERE PNAME IN ('DANCING PRAIRIE', 'MULESHOE RANCH',
                'MCELWAIN-OLSEN', 'TATKON')
```

5C. Display all information about all nature preserves except: DANCING PRAIRIE, MULESHOE RANCH, MCELWAIN-OLSEN, and TATKON.

```
SELECT * FROM PRESERVE
WHERE PNAME NOT IN
        ('DANCING PRAIRIE', 'MULESHOE RANCH',
         'MCELWAIN-OLSEN', 'TATKON')
```

5D. Display all information about any nature preserve having PNO value between and including 3 and 10.

```
SELECT * FROM PRESERVE
WHERE PNO BETWEEN 3 AND 10
```

5E. Display all information about any nature preserve having a PNO value that is less than 3 or greater than 10.

```
SELECT * FROM PRESERVE
WHERE PNO NOT BETWEEN 3 AND 10
```

5F. Display the state, preserve number, and size of any nature preserve that is not in Montana and not in Arizona and is less than 50 acres or greater than 800 acres. Sort the result by preserve number in descending sequence.

```
SELECT STATE, PNO, ACRES
FROM PRESERVE
WHERE STATE NOT IN ('MT', 'AZ')
AND ACRES NOT BETWEEN 50 AND 800
ORDER BY PNO DESC
```

 Copyright Pending 2020

Summary Exercises (Chapter 5)

The following exercises pertain to the EMPLOYEE table. Specify the IN and BETWEEN keywords in the SELECT statements for the following exercises.

5G. Display all information about any employee who works in any department with a DNO value in the following list: {10, 40}.

```
SELECT *
FROM EMPLOYEE
WHERE DNO IN (10, 40)
```

5H. Display all information about any employee who works in any department with a DNO value that is not in the following list: {10, 40}.

```
SELECT *
FROM EMPLOYEE
WHERE DNO NOT IN (10, 40)
```

5I. Display all information about any employee whose salary is greater than or equal to $500.00 and less than or equal to $2,000.00.

```
SELECT *
FROM EMPLOYEE
WHERE SALARY BETWEEN 500 AND 2000
```

5J. Display all information about any employee whose salary is less than $500.00 or greater than $2,000.00.

```
SELECT *
FROM EMPLOYEE
WHERE SALARY NOT BETWEEN 500 AND 2000
```

Chapter-6 – Pattern Matching: LIKE

6A. Reference the PRESERVE table. Display the PNAME value of all nature preserves with a name that begins with the letter D.

```
SELECT PNAME FROM PRESERVE
WHERE PNAME LIKE 'D%'
```

6B. Reference the PRESERVE table. Display the name of any nature preserve with TOWN anywhere in its name.

```
SELECT PNAME FROM PRESERVE
WHERE PNAME LIKE '%TOWN%'
```

6C. Reference the PRESERVE table. Display the PNAME value of all nature preserves with a name that ends with PRAIRIE.

```
SELECT PNAME FROM PRESERVE
WHERE PNAME LIKE '%PRAIRIE'
```

6D. Display the name of any nature preserve where the name begins with MULE.

```
SELECT PNAME FROM PRESERVE
WHERE PNAME LIKE 'MULE%'
```

6E. Display the name of any nature preserve having the string ING anywhere in its name.

```
SELECT PNAME FROM PRESERVE
WHERE PNAME LIKE '%ING%'
```

6F. Display the name of any nature preserve where the name ends with the letter E.

```
SELECT PNAME FROM PRESERVE
WHERE PNAME LIKE '%E'
```

6G. Display the name of any nature preserve that has the letter E immediately after the letter M anywhere in its name.

```
SELECT PNAME FROM PRESERVE
WHERE PNAME LIKE '%ME%'
```

6H. Display the name of any nature preserve that has the letter E anywhere after the letter M in its name.

```
SELECT PNAME FROM PRESERVE
WHERE PNAME LIKE '%M%E%'
```

Reference the DEMO1 table for the following exercise.

6I. Display all CHARNAME values in the DEMO1 table where the CHARNAME value ends with D.

(a) SQL Server users can solve this exercise.

```
SELECT CHARNAME FROM DEMO1
WHERE CHARNAME LIKE '%D'
```

```
CHARNAME
DAVID
EUCLID
```

SQL Server, unlike DB2 and ORACLE, produced the correct result because it ignores trailing blanks.

(b) Optionally, DB2 and ORACLE users can solve this exercise if they to jump ahead to Sample Query 6.13 to learn about the RTRIM function.

```
SELECT CHARNAME FROM DEMO1
WHERE RTRIM (CHARNAME)  LIKE '%D'
```

```
CHARNAME
DAVID
EUCLID
```

6J. Reference the PRESERVE table. Display the name of any nature preserve that has the letter A in the second character position.

```
SELECT PNAME FROM PRESERVE
WHERE PNAME LIKE '_A%'
```

```
PNAME
HASSAYAMPA RIVER
DANCING PRAIRIE
TATKON
DAVID H. SMITH
RAMSEY CANYON
PAPAGONIA-SONOITA CREEK
```

6K. Reference the PRESERVE table. Display the name of any nature preserve that has a blank anywhere in its name.

```
SELECT PNAME FROM PRESERVE
WHERE PNAME LIKE '% %'
```

```
PNAME
HASSAYAMPA RIVER
DANCING PRAIRIE
MULESHOE RANCH
SOUTH FORK MADISON
DAVID H. SMITH
MIACOMET MOORS
MOUNT PLANTAIN
COMERTOWN PRAIRIE
PINE BUTTE SWAMP
RAMSEY CANYON
HOFT FARM
PAPAGONIA-SONOITA CREEK
```

6L. Display the name of any nature preserve having FARM or SWAMP or PRAIRIE anywhere in its name.

```
SELECT PNAME FROM PRESERVE
WHERE PNAME LIKE '%FARM%'
OR PNAME LIKE '%SWAMP%'
OR PNAME LIKE '%PRAIRIE%'
```

```
PNAME
DANCING PRAIRIE
COMERTOWN PRAIRIE
PINE BUTTE SWAMP
HOFT FARM
```

6M. Display the name of any nature preserve with a period or a hyphen anywhere in its name.

```
SELECT PNAME FROM PRESERVE
WHERE PNAME LIKE '%.%'
OR      PNAME LIKE '%-%'
```

```
PNAME
MCELWAIN-OLSEN
DAVID H. SMITH
PAPAGONIA-SONOITA CREEK
```

6N. Display the name of any nature preserve that has the letter R in the fifth position and ends with PRAIRIE.

```
SELECT PNAME FROM PRESERVE
WHERE PNAME LIKE '_ _ _ _ R%'
AND     PNAME LIKE '%PRAIRIE'
```

```
PNAME
COMERTOWN PRAIRIE
```

6O. Reference the PRESERVE table. Display the preserve name of any nature preserve that does not end with an E.

Two solutions:

```
SELECT PNAME FROM PRESERVE
WHERE PNAME NOT LIKE '%E'
```

```
SELECT PNAME FROM PRESERVE
WHERE NOT PNAME LIKE '%E'
```

```
PNAME
HASSAYAMPA RIVER
MULESHOE RANCH
SOUTH FORK MADISON
MCELWAIN-OLSEN
TATKON
DAVID H. SMITH
MIACOMET MOORS
MOUNT PLANTAIN
PINE BUTTE SWAMP
RAMSEY CANYON
HOFT FARM
PAPAGONIA-SONOITA CREEK
```

 Copyright Pending 2020

6P. Reference the PRESERVE table. Display the preserve name of any nature preserve that does not end with an E and does not end with an N.

Three solutions:

```
SELECT PNAME FROM PRESERVE
WHERE       PNAME NOT LIKE '%E'
AND         PNAME NOT LIKE '%N'

SELECT PNAME FROM PRESERVE
WHERE       NOT PNAME LIKE '%E'
AND         NOT PNAME LIKE '%N'

SELECT PNAME FROM PRESERVE
WHERE       NOT  (PNAME LIKE '%E'
OR                PNAME LIKE '%N')
```

```
PNAME
HASSAYAMPA RIVER
MULESHOE RANCH
DAVID H. SMITH
MIACOMET MOORS
PINE BUTTE SWAMP
HOFT FARM
PAPAGONIA-SONOITA CREEK
```

 Copyright Pending 2020

Summary Exercises (Chapter 6)

The following exercises pertain to the EMPLOYEE table. The ENAME column has a VARCHAR (25) data-type. ENAME values cannot have any trailing blanks.

6Q. Display the name of any employee whose name of begins with the letter S.

```
SELECT ENAME FROM EMPLOYEE
WHERE ENAME LIKE 'S%'
```

6R. Display the name of any employee whose has the consecutive letters RR anywhere in his name.

```
SELECT ENAME FROM EMPLOYEE
WHERE ENAME LIKE '%RR%'
```

6S. Display the name of any employee whose name of ends with the letter Y.

```
SELECT ENAME FROM EMPLOYEE
WHERE ENAME LIKE '%Y'
```

6T. Display the name of any employee whose name has the letter O in the second position.

```
SELECT ENAME FROM EMPLOYEE
WHERE ENAME LIKE '_O%'
```

 Copyright Pending 2020

Chapter-7 Exercises: Arithmetic Expressions

7A. What would be the size of each nature preserve if its current size were doubled? Display the preserve name, current acreage, and adjusted acreage.

```
SELECT PNAME, ACRES, ACRES*2
FROM PRESERVE
```

7B. What would be the size of each nature preserve if its current size were reduced to one third its current size? Display the preserve name, current acreage, and adjusted acreage.

```
SELECT PNAME, ACRES, ACRES/3.00
FROM PRESERVE
```

7C. For all nature preserves, display the preserve name and its current admission fee. Also display an adjusted fee that is calculated by adding $50.00 to the current fee and then dividing by 2.

```
SELECT PNAME, FEE, (FEE+50.00)/2.00
FROM PRESERVE
```

Summary Exercises (Chapter 7)

The following exercises reference the EMPLOYEE table. Use aliases for calculated columns.

7D. Assume each employee's salary is increased by 10%. Display the employee's number, name, old salary, and new salary. The result table should have four columns named, ENO, ENAME, OLDSALARY, and NEWSALARY. Sort the result by the new salary.

```
SELECT ENO, ENAME,
       SALARY          OLDSALARY,
       SALARY * 1.10   NEWSALARY
FROM  EMPLOYEE
ORDER BY NEWSALARY
```

7E. Modify the previous Exercise 7D. Only display rows for those employees whose new salary exceeds $2,000.00.

```
SELECT ENO, ENAME,
       SALARY          OLDSALARY,
       SALARY * 1.10   NEWSALARY
FROM EMPLOYEE
WHERE SALARY * 1.10 > 2000.00
ORDER BY NEWSALARY
```

7F: Optional Exercise: Commentary for Sample Query 7.3 noted that you cannot reference a column alias in a WHERE-clause. The following WHERE-clause causes an error.

```
SELECT ENO, ENAME, SALARY + 10.00 NEWSALARY
FROM EMPLOYEE
WHERE NEWSALARY > 2000.00  ← Error
```

Can you speculate *why* the system does not allow a WHERE-clause to reference a column alias?

The system assigns the column alias (NEWSLARY) after rows have been selected and the calculation has been done. Row selection, as specified by a WHERE-clause, occurs before this process when the NEWSALARY alias is not yet defined.

PART II

Built-in Functions
&
Null Values

 Copyright Pending 2020

Chapter-8 - Aggregate Functions

8A. Display the average, maximum, and minimum ACRES value of all nature preserves located in Arizona.

```
SELECT AVG (ACRES), MAX (ACRES), MIN (ACRES)
FROM  PRESERVE
WHERE STATE  = 'AZ'
```

8B. Display the first preserve name which appears in alphabetic sequence.

```
SELECT MIN (PNAME)
FROM  PRESERVE
```

8C. Do not consider zero admission fees. How many distinct fees are present in the PRESERVE?

```
SELECT COUNT (DISTINCT FEE)
FROM  PRESERVE
WHERE FEE <> 0
```

8D. Write a SELECT statement to demonstrate that PNAME does not contain any duplicate values.

```
SELECT COUNT (*), COUNT (DISTINCT PNAME)
FROM  PRESERVE
```

If the result shows two equal values (14 in this example), then PNAME contains unique values.

8E. Assume that you intend to establish a new policy for calculating admission fees. Each nature preserve will charge a fee equal to $0.02 per acre. What will be the average admission fee for the Arizona preserves?

```
SELECT AVG (ACRES*0.02)
FROM  PRESERVE
WHERE STATE = 'AZ'
```

Summary Exercises (Chapter 8)

The following exercises pertain to the EMPLOYEE table.

8F. Display the sum, average, maximum, and minimum of all SALARY values.

```
SELECT SUM (SALARY), AVG (SALARY),
            MAX (SALARY), MIN (SALARY)
FROM EMPLOYEE
```

8G. How many employees work in Department 20?

```
SELECT COUNT(*)
FROM  EMPLOYEE
WHERE DNO = 20
```

8H. How many departments have employees?

```
SELECT COUNT (DISTINCT DNO)
FROM  EMPLOYEE
```

 Copyright Pending 2020

Chapter-9 – GROUP BY Clause: Grouping by a Single Column

9A. For each state referenced in the PRESERVE table, display the ACRES value of the smallest nature preserve within the state.

```
SELECT STATE, MIN (ACRES)
FROM  PRESERVE
GROUP BY STATE
```

STATE	MIN(ACRES)
AZ	380
MA	4
MT	121

9B. Do not consider any nature preserve that has more than 10,000 acres. For each state referenced in the PRESERVE table, display the ACRES value of the largest nature preserve within the state.

```
SELECT STATE, MAX (ACRES)
FROM PRESERVE
WHERE ACRES <= 10000
GROUP BY STATE
```

STATE	MAX(ACRES)
AZ	1200
MA	830
MT	1130

9C. Same as preceding Exercise 9B. (Note: Its result was incidentally sorted.) Sort the result by the maximum values in descending sequence.

```
SELECT STATE, MAX (ACRES)
FROM PRESERVE
WHERE ACRES <= 10000
GROUP BY STATE
ORDER BY 2 DESC
```

Alternative: ORDER BY MAX (ACRES) DESC

STATE	MAX(ACRES)
AZ	1200
MA	830
MT	1130

 Copyright Pending 2020

9D. Display the size of the largest ACRES value protected by a nature preserve within each state if that value is less than 25,000 acres.

```
SELECT STATE, MAX (ACRES)
FROM PRESERVE
GROUP BY STATE
HAVING MAX (ACRES) < 25000
```

```
STATE  MAX(ACRES)
MA            830
MT          15000
```

9E. For each state referenced in the PRESERVE table, display the number of acres in the state's smallest preserve if that number is less than 100.

```
SELECT STATE, MIN (ACRES)
FROM PRESERVE
GROUP BY STATE
HAVING MIN (ACRES) < 100
```

```
STATE  MIN(ACRES)
MA               4
```

The above solution is the "most direct" solution that satisfies the query objective. The following equivalent solution is a little more efficient because it produces fewer groups and eliminates the need for the HAVING-clause.

```
SELECT STATE, MIN (ACRES)
FROM  PRESERVE
WHERE ACRES < 100
GROUP BY STATE
```

9F. Only consider nature preserves that have more than 1,000 acres. Display the size of the largest ACRES value protected by a nature preserve within each state if that value is less than 25,000 acres.

```
SELECT STATE, MAX (ACRES)
FROM  PRESERVE
WHERE ACRES > 1000
GROUP BY STATE
HAVING MAX (ACRES) < 25000
```

```
STATE  MAX(ACRES)
MT          15000
```

9G. Display the state and total size of the preserves located in the state if the total size is greater than or equal to 10,000 acres and less than or equal to 50,000 acres.

```
SELECT STATE, SUM (ACRES)
FROM PRESERVE
GROUP BY STATE
HAVING SUM (ACRES) >= 10000 AND SUM (ACRES) <= 50000
```

```
STATE  SUM(ACRES)
MT          16931
```

Alternative Solution:

```
SELECT STATE, SUM (ACRES)
FROM PRESERVE
GROUP BY STATE
HAVING SUM (ACRES) BETWEEN 10000 AND 50000
```

9H: ***If*** your system allows the nesting of aggregate functions, then determine the average number of preserve acres for each state and display the largest of these averages.

```
SELECT MAX (AVG (ACRES))
FROM PRESERVE
GROUP BY STATE
```

```
MAX(AVG(ACRES))
          12840
```

 Copyright Pending 2020

Summary Exercises (Chapter 9)

The following exercises reference the EMPLOYEE table.

9I. For all department DNO values found in the EMPLOYEE table, display the DNO value followed by the average SALARY for that department.

```
SELECT DNO, AVG (SALARY)
FROM  EMPLOYEE
GROUP BY DNO
```

DNO	AVG(SALARY)
10	1200.00
20	4666.66
40	500.00

9J. For all department DNO values found in the EMPLOYEE table, display the DNO value followed by the sum, maximum, and minimum of SALARY values for that department.

```
SELECT DNO, SUM (SALARY), MAX(SALARY), MIN(SALARY)
FROM EMPLOYEE
GROUP BY DNO
```

DNO	SUM(SALARY)	MAX(SALARY)	MIN(SALARY)
10	2400.00	2000.00	400.00
20	14000.00	9000.00	2000.00
40	500.00	500.00	500.00

 Copyright Pending 2020

9K. Consider all departments except for Department 40. For these departments, display their DNO value followed by the number of employees who work in that department.

```
SELECT DNO, COUNT (*)
FROM  EMPLOYEE
WHERE DNO <> 40
GROUP BY DNO
```

DNO	COUNT(*)
10	2
20	3

9L. Assume the SALARY column contains confidential data, and that someone could deduce this confidential data by examining the total of each department's salaries. Display only those departments and their average departmental salary if a department has more than two employees.

```
SELECT DNO, SUM (SALARY)
FROM EMPLOYEE
GROUP BY DNO
HAVING COUNT (*) > 2
```

DNO	SUM(SALARY)
20	14000.00

 Copyright Pending 2020

9M. Outline a poor man's cut-and-paste solution that can be used to produce the following result. (A much better solution will be described later in thus book.)

STATE	SUM(ACRES)
AZ	51360
MA	1760
MT	16931
TOTAL	**70051**

Code two statements

```
SELECT STATE, SUM (ACRES)
FROM  PRESERVE
GROUP BY STATE
```

STATE	SUM(ACRES)
AZ	51360
MA	1760
MT	16931

```
SELECT 'TOTAL', SUM(ACRES)
FROM P RESERVE
```

TOTAL	**70051**

Cut-and-paste the two result tables.

 Copyright Pending 2020

Chapter-9.5 – GROUP BY Clause: Grouping by Multiple Columns

The following exercises group by one column in the PURCHASE table.

9N. Reference the PURCHASE table. For each part, display its part number (PNO) followed by the total cost of all its parts. The result should look like:

PNO	TOTCOST
P1	1450
P2	3200
P3	6150
P4	6700
P5	8400
P6	1800
P7	5400
P8	600

```
SELECT SNO, SUM (COST) TOTCOST
FROM PURCHASE
GROUP BY SNO
ORDER BY SNO
```

9O. Reference the PURCHASE table. For each employee who purchased a part, display his employee number (ENO) followed by the total cost of all parts purchased by the employee. The result should look like:

ENO	TOTCOST
E1	11700
E2	6600
E3	7800
E4	7600

```
SELECT ENO, SUM (COST) TOTCOST
FROM PURCHASE
GROUP BY ENO
ORDER BY ENO
```

 Copyright Pending 2020

The following exercises group by two columns.

9P. Access the PURCHASE table. Calculate the total of COST for each combination of (PJNO, ENO) of values. Display these columns in the (PJNO, ENO) left-to-right column sequence followed by the total cost. Sort the result in ascending sequence by (PJNO, ENO). The result should look like:

PJNO	ENO	TOTCOST
PJ1	E1	9500
PJ1	E3	3900
PJ1	E4	3500
PJ2	E1	1200
PJ2	E2	6000
PJ2	E3	3900
PJ2	E4	4100
PJ3	E1	1000
PJ3	E2	600

```
SELECT PJNO, ENO, SUM (COST) TOTCOST
FROM PURCHASE
GROUP BY PJNO, ENO
ORDER BY PJNO, ENO
```

 Copyright Pending 2020

9Q. Access the PURCHASE table. Calculate the total of COST for each combination of (PNO, SNO) of values. Display these columns in the (PNO, SNO) left-to-right column sequence followed by the total cost. Sort the result in ascending sequence by (PNO, SNO). The result should look like:

PNO	SNO	TOTCOST
P1	S1	1150
P1	S2	300
P2	S1	3200
P3	S1	1550
P3	S3	2400
P3	S4	2200
P4	S1	900
P4	S2	1000
P4	S3	2400
P4	S4	2400
P5	S1	3900
P5	S3	3000
P5	S4	1500
P6	S2	300
P6	S4	1500
P7	S3	2000
P7	S4	3400
P8	S1	500
P8	S2	100

```
SELECT PNO, SNO, SUM (COST) TOTCOST
FROM PURCHASE
GROUP BY PNO, SNO
ORDER BY PNO, SNO
```

 Copyright Pending 2020

The following exercises group by three columns.

9R. Access the PURCHASE table. Calculate the total of COST for each combination of (PJNO, ENO, SNO) values. Display these columns in the (PJNO, ENO, SNO) left-to-right column sequence followed by the total cost. Sort the result in ascending sequence by (PJNO, ENO, SNO). The result should look like:

PJNO	ENO	SNO	TOTCOST
PJ1	E1	S1	6500
PJ1	E1	S3	2000
PJ1	E1	S4	1000
PJ1	E3	S3	3900
PJ1	E4	S4	3500
PJ2	E1	S1	1100
PJ2	E1	S2	100
PJ2	E2	S1	2600
PJ2	E2	S2	1000
PJ2	E2	S4	2400
PJ2	E3	S3	3900
PJ2	E4	S4	4100
PJ3	E1	S1	1000
PJ3	E2	S2	600

```
SELECT PJNO, ENO, SNO, SUM (COST) TOTCOST
FROM PURCHASE
GROUP BY PJNO, ENO, SNO
ORDER BY PJNO, ENO, SNO
```

 Copyright Pending 2020

9S. Access the PURCHASE table. Calculate the total of COST for each combination of (PNO, SNO, ENO) values. Display these columns in the (PNO, SNO, ENO) left-to-right column sequence followed by the total cost. Sort the result in ascending sequence by (PNO, SNO, ENO). The result should look like:

PNO	SNO	ENO	TOTCOST
P1	S1	E1	1150
P1	S2	E2	300
P2	S1	E1	1000
P2	S1	E2	2200
P3	S1	E1	1550
P3	S3	E3	2400
P3	S4	E4	2200
P4	S1	E1	600
P4	S1	E2	300
P4	S2	E2	1000
P4	S3	E3	2400
P4	S4	E4	2400
P5	S1	E1	3900
P5	S3	E3	3000
P5	S4	E4	1500
P6	S2	E2	300
P6	S4	E4	1500
P7	S3	E1	2000
P7	S4	E1	1000
P7	S4	E2	2400
P8	S1	E1	400
P8	S1	E2	100
P8	S2	E1	100

```
SELECT PNO, SNO, ENO, SUM (COST) TOTCOST
FROM PURCHASE
GROUP BY PNO, SNO, ENO
ORDER BY PNO, SNO, ENO
```

 Copyright Pending 2020

9T. Access the PURCHASE table. Exclude from consideration all rows associated with Project PJ2. Display the total of COST for each combination of (PJNO, ENO, SNO) of values (excluding Project PJ2). The result should look like:

PJNO	ENO	SNO	TOTCOST
PJ1	E1	S1	6500
PJ1	E1	S3	2000
PJ1	E1	S4	1000
PJ1	E3	S3	3900
PJ1	E4	S4	3500
PJ3	E1	S1	1000
PJ3	E2	S2	600

```
SELECT PJNO, ENO, SNO, SUM (COST) TOTCOST
FROM PURCHASE
WHERE PJNO <> 'PJ2'
GROUP BY PJNO, ENO, SNO
ORDER BY PJNO, ENO, SNO
```

9U. Access the PURCHASE table. Display the total COST for each combination of (PNO, SNO, ENO) values if that total is greater than or equal to 2000. The result should look like:

PNO	SNO	ENO	TOTCOST
P2	S1	E2	2200
P3	S3	E3	2400
P3	S4	E4	2200
P4	S3	E3	2400
P4	S4	E4	2400
P5	S1	E1	3900
P5	S3	E3	3000
P7	S3	E1	2000
P7	S4	E2	2400

```
SELECT PNO, SNO, ENO, SUM (COST) TOTCOST
FROM PURCHASE
GROUP BY PNO, SNO, ENO
HAVING SUM (COST) >= 2000
ORDER BY PNO, SNO, ENO
```

Summary Exercise (Chapter 9.5)

9V. Reconsider Exercise 9S [Access the PURCHASE table. Calculate the total of COST for each combination of (PNO, SNO, ENO) values. Display these columns in the (PNO, SNO, ENO) left-to-right column sequence followed by the total cost. Sort the result in ascending sequence by (PNO, SNO, ENO)].

This final result table contained 23 rows. Some of these rows corresponded to groups that summarized over just one or two individual PURCHASE rows. To reduce the number rows in the final result, exclude any summary row from the final result if that summary represents a total of just one or two rows. The result should look like:

PNO	SNO	ENO	TOTCOST	GPCT
P1	S1	E1	1150	5
P3	S1	E1	1550	3
P4	S2	E2	1000	3
P5	S1	E1	3900	3
P8	S1	E1	400	4

The GPCT column contains the number of rows in the group.

```
SELECT PNO, SNO, ENO, SUM (COST) TOTCOST, COUNT(*) GPCT
FROM PURCHASE
GROUP BY PNO, SNO, ENO
HAVING COUNT (*) >=3
ORDER BY PNO, SNO, ENO
```

 Copyright Pending 2020

Chapter-10 – Individual Functions

Optional Summary Exercise

VARCHAR columns rarely contain character-strings with leading or trailing blanks. Assume you think that some character-strings with leading or trailing blanks (somehow) found their way into the V1 column in the DEMO2 table. You would like to discover these problematic rows.

Code a SELECT statement to display the V1 value and its length if that value has a blank in its first-character position or a blank in its last-character position.

The current version of DEMO2 does not have any V1 values with leading/trailing blanks. Therefore, to test your SELECT statement, you must execute two INSERT statements to insert two problematic rows. (Do not worry about details of these INSERT statements. INSERT will be covered in Chapter 15.)

After you have tested your SELECT statement, delete the two problematic rows by executing the following DELETE statement. (Again, do not worry about details of this DELTE statement. DELETE will be covered in Chapter 15.)

```
INSERT INTO DEMO2 VALUES (999, 999, '  JOSEPHINE', 'XXX');
INSERT INTO DEMO2 VALUES (888, 888, 'JACQUELINE  ', 'XXX');

SELECT V1, LENGTH (V1)
FROM DEMO2
WHERE SUBSTR (V1,1,1) = ' '
OR    SUBSTR (V1, LENGTH(V1), 1) = ' ';
```

```
V1                  LENGTH(V1)
  JOSEPHINE                 11
JACQUELINE                  12
```

```
DELETE FROM DEMO2 WHERE I1 IN (999, 888);
```

SQL Server: The above SUBSTR function works in DB2 and ORACLE. SQL Server users must replace SUBSTR with SUBSTRING.

Chapter-10.5: Processing DATE Values

10A. Consider the following two statements where the ORDER BY clauses reference character-string columns. One of these statements (somehow) produces a desired result where the rows are sorted in chronological sequence. Which statement? Execute the statements to verify your answer.

```
SELECT MNAME, BDCHAR3
FROM  DEMO3
ORDER BY BDCHAR3
```

MNAME	BDCHAR2
JACQUELINE	January 10, 2019
JOSEPHINE	June 13, 2017
EVAN	June 5, 2017
JESSIE	March 7, 1982
JONHHY	May 10, 2015
JULIE	May 17, 1978
HANNAH	November 25, 2014

```
SELECT MNAME, BDCHAR1
FROM  DEMO3
ORDER BY BDCHAR1
```

MNAME	BDCHAR1
JULIE	1978-05-17
JESSIE	1982-03-07
HANNAH	2014-11-25
JONHHY	2015-05-10
EVAN	2017-06-05
JOSEPHINE	2017-06-13
JACQUELINE	2019-01-10

Both results are in ascending alphanumeric sequence. Sorting by the BDCHAR1 column is interesting because sorting in an YYYY-MM-DD format "just happens to" correspond to a chronological sequence. For this reason, the YYYY-MM-DD format was very popular in ancient history file/database systems that did not support a "real" DATE date-type.

 Copyright Pending 2020

Chapter-11 - Null Values

Display the result table produced by executing:

11A.
```
SELECT A, B, A-B
FROM  NTAB2
```

A	B	A-B
10	-	-
15	10	5
-	30	-
-	10	-
40	40	0
-	-	-

11B.
```
SELECT SUM (A), SUM (B)
FROM  NTAB2
```

SUM (A)	SUM (B)
65	90

11C.
```
SELECT SUM (A+B),  SUM(A) + SUM(B)
FROM  NTAB2
```

SUM (A+B)	SUM (A) + SUM (B)
105	155

11D.
```
SELECT *
FROM NTAB2
WHERE A = B
```

A	B
40	40

11E.
```
SELECT *
FROM NTAB2
WHERE A <> B
```

A	B
15	10

11F.
```
SELECT COUNT(*), COUNT(A), COUNT(B)
FROM NTAB2
```

COUNT (*)	COUNT (A)	COUNT (B)
//////////////////////////////////////	6	3	4

 Copyright Pending 2020

11G.
```
SELECT *
FROM NTAB2
WHERE A <> B OR B < 20
```

A	B
15	10
-	10

11H.
```
SELECT *
FROM NTAB2
WHERE A=B OR A<>B
```

A	B
15	10
40	40

11I.
```
SELECT B
FROM NTAB2
ORDER BY B
```

[DB2 & ORACLE Result]

B
10
10
30
40
-
-

[SQL Server Result]

B
-
-
10
10
30
40

 Copyright Pending 2020

11J. SELECT DISTINCT A
FROM NTAB2

A
10
15
40
-

Here, DISTICT caused an incidental sort, and the null value sorted high.
If SQL Server causes incidental sort, the null value will sort low.

11K. SELECT A, SUM (B)
FROM NTAB2
GROUP BY A

A	SUM (B)
10	-
15	10
40	40
-	40

Here an incidental sort occurred, and the null A value sorted high.
If SQL Server causes incidental sort, the null value will sort low and the result will look like.

A	SUM (B)
-	40
10	-
15	10
40	40

 Copyright Pending 2020

Summary Exercises (Chapter 11)

You are given the following NTAB3 table. Execute the following statements. Sorry! Answers are not show. Determine answer by executing the statements on your system.

A	B
20	20
50	-
-	-
-	30
10	50
10	10
40	50

1. SELECT A, B, A*B FROM NTAB3;
2. SELECT MAX(A), MIN (B) FROM NTAB3;
3. SELECT SUM(A)+SUM(B), SUM(A+B) FROM NTAB3;
4. SELECT COUNT (*), COUNT(A) FROM NTAB3;
5. SELECT * FROM NTAB3 WHERE A = B;
6. SELECT * FROM NTAB3 WHERE A <> B;
7. SELECT COUNT (*) FROM NTAB3 WHERE A = B OR A <> B;
8. SELECT * FROM NTAB3 WHERE A <> 10 AND B < 10
9. SELECT * FROM NTAB3 WHERE A = 10 OR B < 10
10. SELECT * FROM NTAB3 ORDER BY A;
11. SELECT DISTINCT A FROM NTAB3;
12. SELECT A, SUM(B) FROM NTAB3 GROUP BY A;
13. SELECT * FROM NTAB3 WHERE A IS NULL;
14. SELECT * FROM NTAB3 WHERE A IS NOT NULL;
15. SELECT SUM(A)+SUM(B), SUM(A+B) FROM NTAB3
    WHERE A IS NOT NULL AND B IS NOT NULL;
16. SELECT COALESCE (A,25), COALESCE (B,15) FROM NTAB3;

PART III

Data Definition
&
Data Manipulation

 Copyright Pending 2020

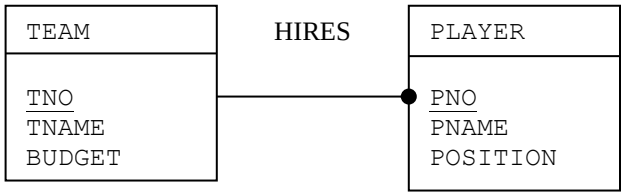
Chapter-12: No Exercises

Chapter-13 - CREATE TABLE: Optional Appendix Exercise

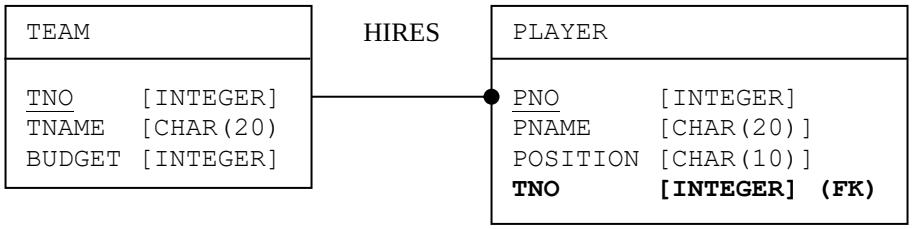
These exercises are optional. (They pertain to database design, a topic that is *not* the primary focus of this book.) For all exercises, you are given a Conceptual Data Model (CDM) that has been produced by database analysis. Transform this model into a Logical Data Model (LDM) and then into a collection of CREATE TABLE statements. Specify foreign-keys. Make reasonable assumptions about data-types. All columns are non-null.

13.1 Professional Sports Team: Each player plays on just one team. Each team has many players.

CDM



LDM

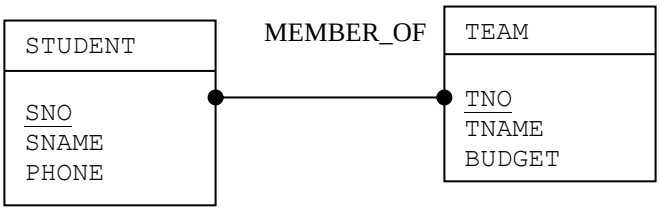


Implementation

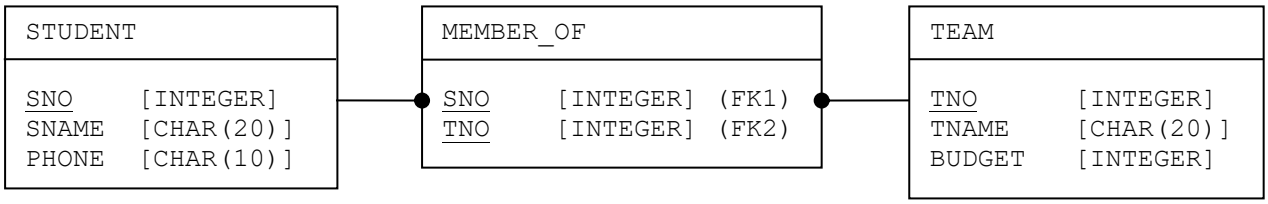
```
DROP TABLE TEAM;
DROP TABLE PLAYER;

CREATE TABLE TEAM
(TNO         INTEGER     NOT NULL,
 TNAME       CHAR (20)   NOT NULL,
 BUDGET      INTEGER     NOT NULL,
 PRIMARY KEY (TNO));

CREATE TABLE PLAYER
(PNO         INTEGER     NOT NULL,
 PNAME       CHAR (20)   NOT NULL,
 POSITION    CHAR (10)   NOT NULL,
 TNO         INTEGER     NOT NULL,
 PRIMARY KEY (PNO),
 FOREIGN KEY (TNO) REFERENCES TEAM);
```

13.2 College Sports Team: A student may play on many teams. Each team has many players.

CDM



LDM

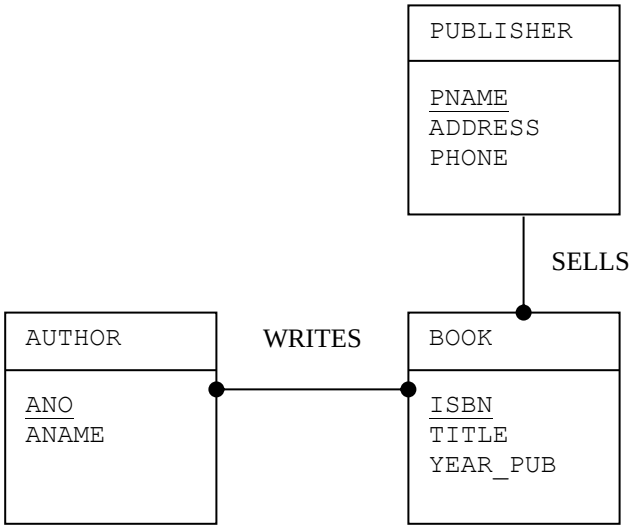


Implementation

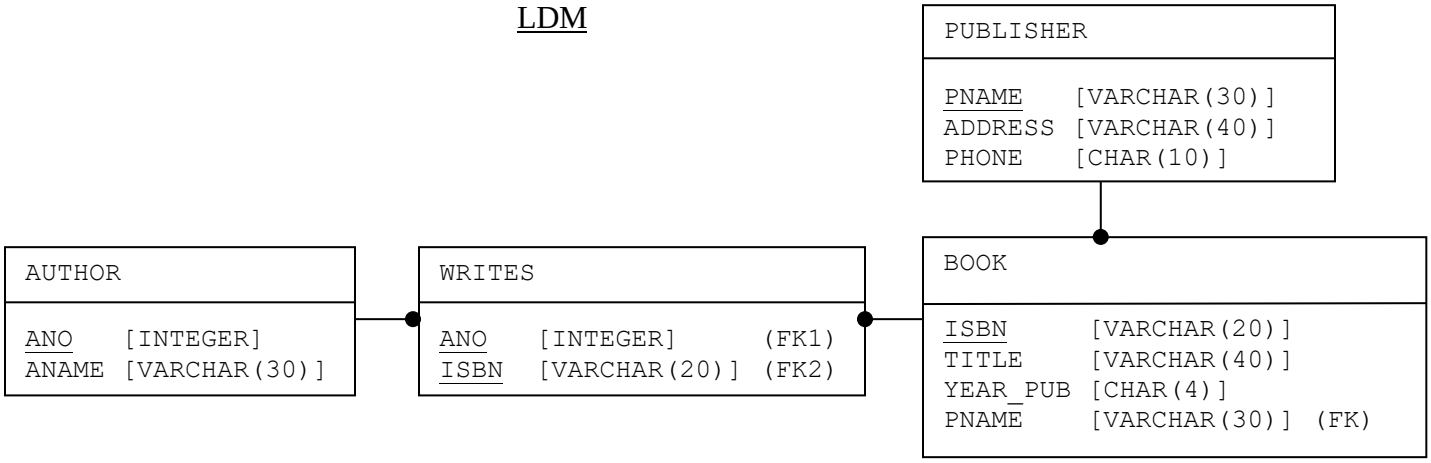
```
DROP TABLE MEMBER_OF;
DROP TABLE STUDENT;
DROP TABLE TEAM;

CREATE TABLE STUDENT
(SNO         INTEGER     NOT NULL,
 SNAME       CHAR (20)   NOT NULL,
 PHONE       CHAR (10)   NOT NULL,
   PRIMARY KEY (SNO));

CREATE TABLE TEAM
(TNO         INTEGER     NOT NULL,
 TNAME       CHAR (20)   NOT NULL,
 BUDGET      INTEGER     NOT NULL,
   PRIMARY KEY (TNO));

CREATE TABLE MEMBER_OF
(SNO         INTEGER     NOT NULL,
 TNO         INTEGER     NOT NULL,
   PRIMARY KEY (SNO, TNO),
   FOREIGN KEY (SNO) REFERENCES STUDENT,
   FOREIGN KEY (TNO) REFERENCES TEAM);
```

13.3 Book Publishing: A publisher sells many books. Each book has one publisher. A book may have multiple coauthors. An author may write many books.

CDM



LDM



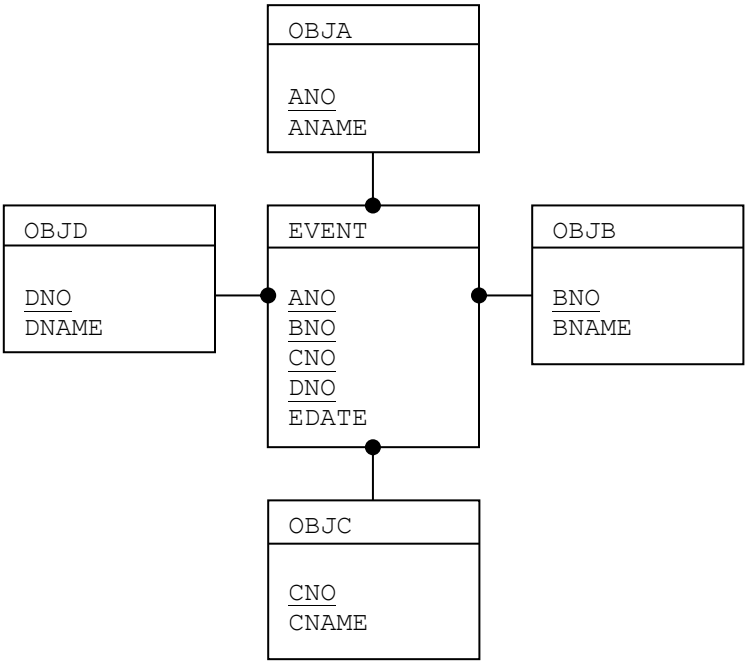
 Copyright Pending 2020

Implementation

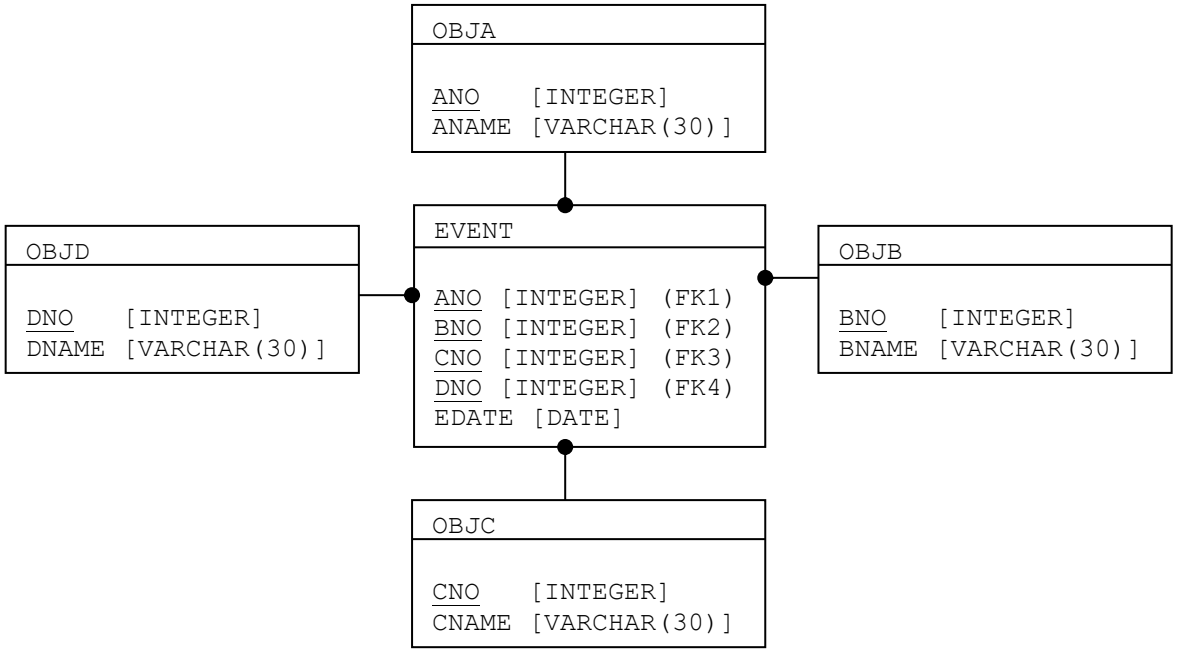
```
DROP TABLE WRITES;
DROP TABLE AUTHOR;
DROP TABLE BOOK;
DROP TABLE PUBLISHER;

CREATE TABLE PUBLISHER
(PNAME      VARCHAR (30)       NOT NULL,
 ADDRESS    VARCHAR (40)       NOT NULL,
 PHONE      CHAR (10),
 PRIMARY KEY (PNAME));

CREATE TABLE BOOK
(ISBN       VARCHAR (20)       NOT NULL,
 TITLE      VARCHAR (40)       NOT NULL,
 YEAR_PUB   CHAR (4)           NOT NULL,
 PNAME      VARCHAR (30)       NOT NULL,
 PRIMARY KEY (ISBN),
 FOREIGN KEY (PNAME) REFERENCES PUBLISHER);

CREATE TABLE AUTHOR
(ANO        INTEGER            NOT NULL,
 ANAME      VARCHAR (30)       NOT NULL,
 PRIMARY KEY (ANO));

CREATE TABLE WRITES
(ANO        INTEGER            NOT NULL,
 ISBN       VARCHAR (20)       NOT NULL,
 PRIMARY KEY (ANO, ISBN),
 FOREIGN KEY (ANO)  REFERENCES AUTHOR,
 FOREIGN KEY (ISBN) REFERENCES BOOK);
```

13.4 Star Design: Sometimes, within a data warehouse application, a designer creates a CDM that looks like a star. The following "star model" shows an EVENT object-type as the center of the star where all other object-types (OBJA, OBJB, OBJC, and OBJD) surround EVENT and have a one-to-many-relationship with EVENT.

CDM



LDM



 Copyright Pending 2020

Implementation

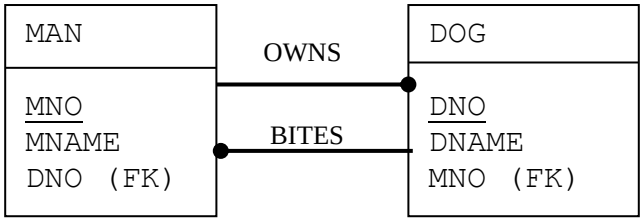
```
DROP TABLE OBJA;
DROP TABLE OBJB;
DROP TABLE OBJC;
DROP TABLE OBJD;
DROP TABLE EVENT;

CREATE TABLE OBJA
(ANO   INTEGER      NOT NULL,
 ANAME VARCHAR (30) NOT NULL,
PRIMARY KEY (ANO));

CREATE TABLE OBJB
(BNO   INTEGER      NOT NULL,
 BNAME VARCHAR (30) NOT NULL,
PRIMARY KEY (BNO));

CREATE TABLE OBJC
(CNO   INTEGER      NOT NULL,
 CNAME VARCHAR (30) NOT NULL,
PRIMARY KEY (CNO));

CREATE TABLE OBJD
(DNO   INTEGER      NOT NULL,
 DNAME VARCHAR (30) NOT NULL,
PRIMARY KEY (DNO));

CREATE TABLE EVENT
(ANO         INTEGER            NOT NULL,
 BNO         INTEGER            NOT NULL,
 CNO         INTEGER            NOT NULL,
 DNO         INTEGER            NOT NULL,
 ENO          DATE,
  PRIMARY KEY (ANO, BNO, CNO, DNO),
  FOREIGN KEY (ANO) REFERENCES OBJA,
  FOREIGN KEY (BNO) REFERENCES OBJB,
  FOREIGN KEY (CNO) REFERENCES OBJC,
  FOREIGN KEY (DNO) REFERENCES OBJD)
```

13.5 Cyclic Design: Sometimes multiple relationships between object-types can form a cycle. Assume we have the MAN and DOG object-types with the following two relationships.

OWNS Relationship: A man can own many dogs; and, each dog must be owned by one man.

BITES Relationship: A dog may bite many men; and each man must be bitten by one dog.

CDM



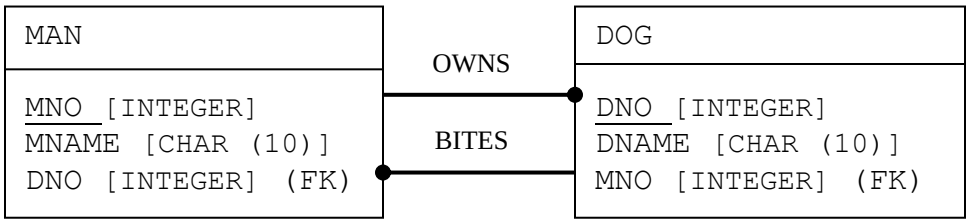
The following two CREATE TABLE statements are "almost correct." The problem involves designating which table to create first. In the following example, which initially attempts to create the MAN table, an error occurs because its foreign-key references DOG, a table that has not yet been created. A similar problem occurs if we attempt to create the DOG table first.

"Almost" Correct (Chicken-Egg Problem)

```
CREATE TABLE MAN
(MNO   INTEGER      NOT NULL,
 MNAME CHAR (10)    NOT NULL,
 DNO   INTEGER      NOT NULL,
     PRIMARY KEY (MNO),
     FOREIGN KEY (DNO) REFERENCES DOG);

CREATE TABLE DOG
(DNO   INTEGER      NOT NULL,
 DNAME CHAR (10)    NOT NULL,
 MNO   INTEGER      NOT NULL,
     PRIMARY KEY (DNO),
     FOREIGN KEY (MNO) REFERENCES MAN);
```

Utilize ALTER TABLE statements, as illustrated in Figure 13.4 to resolve this problem.

LDM



Implementation

```
Correct (Chicken-Egg Problem)

DROP TABLE MAN;
DROP TABLE DOG;

CREATE TABLE MAN
(MNO   INTEGER      NOT NULL,
 MNAME CHAR (10)    NOT NULL,
 DNO   INTEGER      NOT NULL,
               PRIMARY KEY (MNO));

CREATE TABLE DOG
(DNO   INTEGER      NOT NULL,
 DNAME CHAR (10)    NOT NULL,
 MNO   INTEGER      NOT NULL,
               PRIMARY KEY (DNO));

ALTER TABLE MAN
     ADD  CONSTRAINT FK_DOG
          FOREIGN KEY (DNO) REFERENCES DOG;

ALTER TABLE DOG
     ADD CONSTRAINT FK_MAN
          FOREIGN KEY (MNO) REFERENCES MAN;
```

Comment: Having created these tables, we will encounter special considerations regarding the sequence of INSERT statements. See the following page.

 Copyright Pending 2020

Disabling and Enabling Constraints

Assume the MAN and DOG tables have just been created, and they are empty.

Now you want to insert the first row into MAN. For example:

```
INSERT INTO MAN VALUES (123 , 'JOHNNY', 456)
```

Problem: The DNO value 456 must fail because there is no matching row in the DOG table.

Likewise, assume you want to insert the first row into DOG where the MAN table is still empty. For example:

```
INSERT INTO DOG VALUES (456 , 'ROVER',789)
```

Problem: The MNO value 123 must fail because there is no matching row in the MAN table.

To solution to this problem is to temporally disable the enforcement of foreign-key constraints. Some systems use the ALTER TABLE statement to satisfy this objective.

```
ALTER TABLE MAN ALTER FOREIGN KEY FK_DOG NOT ENFORCED;

ALTER TABLE DOG ALTER FOREIGN KEY FK_MAN NOT ENFORCED;

     INSERT INTO MAN VALUES (123 , 'JOHNNY', 456);

     INSERT INTO DOG VALUES (456 , 'ROVER',789);

ALTER TABLE MAN ALTER FOREIGN KEY FK_DOG ENFORCED;

ALTER TABLE DOG ALTER FOREIGN KEY FK_MAN ENFORCED;
```

Again, you will find considerable variation among different systems.

Chapter-14 CREATE INDEX

14.1 Assume that the solution to Exercise 13.1 is coded in the following script.

```
DROP TABLE WRITES;
DROP TABLE AUTHOR;
DROP TABLE BOOK;
DROP TABLE PUBLISHER;

CREATE TABLE PUBLISHER
(PNAME          VARCHAR (30)  NOT NULL,
 ADDRESS        VARCHAR (40)  NOT NULL,
 PHONE          CHAR (10),
 PRIMARY KEY (PNAME));

CREATE TABLE BOOK
(ISBN           VARCHAR (20)  NOT NULL,
 TITLE          VARCHAR (40)  NOT NULL,
 YEAR_PUB       CHAR (4)      NOT NULL,
 PNAME          VARCHAR (30)  NOT NULL,
 PRIMARY KEY (ISBN),
 FOREIGN KEY (PNAME) REFERENCES PUBLISHER);

CREATE TABLE AUTHOR
(ANO            INTEGER              NOT NULL,
 ANAME          VARCHAR (30)         NOT NULL,
 PRIMARY KEY (ANO));

CREATE TABLE WRITES
(ANO            INTEGER              NOT NULL,
 ISBN           VARCHAR (20)         NOT NULL,
 PRIMARY KEY (ANO, ISBN),
 FOREIGN KEY (ANO)  REFERENCES AUTHOR,
 FOREIGN KEY (ISBN) REFERENCES BOOK);
```

a. Create an index on all foreign-keys.

```
CREATE INDEX XPNAME  ON BOOK   (PNAME);
CREATE INDEX XANO    ON WRITES (ANO);
CREATE INDEX XISBN   ON WRITES (ISBN);
```

b. Create a composite index on the TITLE and YEAR_PUB columns (in that order) found in the BOOK table.

```
CREATE INDEX XTB ON BOOK (TITLE, YEAR_PUB);
```

c. The four above indexes on foreign-keys, plus the four automatically created indexes on the four primary-keys, implies a total number of 8 indexes

14.2 Assume that: (i) both the TESTDEPT and TESTEMP tables are very large, (ii) the ENAME column is not unique because two employees may have the same name, and (iii) your organization has an unusual policy of forbidding the assignment of two employees having the same name to the same job. Consider the following query pattern:

You will frequently search on JOBCODE only.

WHERE JOBCODE = _____

You almost never execute a query that searches on ENAME only.

WHERE ENAME = _____

Occasionally you execute a query with a WHERE-clause that looks like:

WHERE ENAME = _____ AND JOBCODE = _____
or
WHERE JOBCODE = _____ AND ENAME = _____

Create one composite index on both the ENAME and JOBCODE columns that could be helpful.
.

```
CREATE UNIQUE INDEX XJOBSAL
ON EMPLOYEE (JOBCODE, ENAME)
```

14.3 This is an unfair exercise. But we invite you to speculate on an answer.

Discussion of Sample Statement 14.2 raised a design decision. If a column will contain unique values, should you declare a UNIQUE constraint or create a UNIQUE index? We stated that declaring a UNIQUE column within the CREATE TABLE statement is usually the preferred approach. Justify this preference.

The fact that a column must contain unique values is *logical*, not a physical, constraint. Hence, this constraint should be declared in the CREATE TABLE statement.

Chapter-15 - INSERT – UPDATE - DELETE

15A. Execute the following statement.

```
CREATE TABLE JUNK1
(C1    INTEGER NOT NULL PRIMARY KEY,
 C2    CHAR (5),
 C3    VARCHAR (10));
```

Insert the following row into JUNK1.

250	HELLO	DOOPY

```
INSERT INTO JUNK1 VALUES (250, 'HELLO', 'DOOPY');
```

Verify the CREATE TABLE and INSERT operations by executing:
SELECT * FROM JUNK1

15B. Insert the following three rows into JUNK1. (The hyphen represents a null value.)

150	-	HAPPY
350	HI	SAD
850	BYE	-

```
INSERT INTO JUNK1 VALUES (150, NULL, 'HAPPY');
INSERT INTO JUNK1 VALUES (350, 'HI',   'SAD');
INSERT INTO JUNK1 VALUES (850, 'BYE', NULL);
```

Verify these insert operations by executing: SELECT * FROM JUNK1

15C. Update the JUNK1 table. Change the row where C1 is 150. Its new C3 value should be set to “GRUMPY”.

```
UPDATE JUNK1
SET C3 = 'GRUMPY'
WHERE C1 = 150
```

Verify this update operation by executing: SELECT * FROM JUNK1

 Copyright Pending 2020

15D. Update the JUNK1 table. Change all rows having a C2 value beginning with the letter H. The new C3 value for each row should be set to CRANKY.

```
UPDATE JUNK1
SET C3 = 'CRANKY'
WHERE C2 LIKE 'H%'
```

Verify this UPDATE operation by executing: SELECT * FROM JUNK1

15E. Delete any row from the JUNK1 table with a C1 value that exceeds 300.

```
DELETE
FROM JUNK1
WHERE C1 > 300
```

Verify this DELETE operation by executing: SELECT * FROM JUNK1

 Copyright Pending 2020

Summary Exercises (Chapter 15)

15F. DELETE all rows from JUNK1.

```
DELETE FROM JUNK1
```

15G. INSERT the following rows into JUNK1.

98	YES1	YES2
95	NO1	NO2

```
INSERT INTO JUNK1 VALUES (98, 'YES1', 'YES2');
INSERT INTO JUNK1 VALUES (95, 'NO1',  'NO2');
```

15H. Update JUNK1 set all C2 values to MAYBE.

```
UPDATE JUNK1
SET C2 = 'MAYBE'
```

15I. DELETE any row where the C1 value is greater than 95.

```
DELETE FROM JUNK1
WHERE C1 > 95
```

15J. Drop the JUNK1 table.

```
DROP TABLE JUNK1
```

 Copyright Pending 2020

PART IV

Join Operations

 Copyright Pending 2020

Chapter-16 - Inner-Join: Getting Started

16A. What result tables are produced by executing the following statements?

a.
```
SELECT ENAME, DNAME
FROM   EMPLOYEE, DEPARTMENT
WHERE EMPLOYEE.DNO = DEPARTMENT.DNO
AND    BUDGET <= 50000.00
ORDER BY ENAME
```

ENAME	DNAME
CURLY	INFO. SYS.
GEORGE	INFO. SYS.
MOE	INFO. SYS.
SHEMP	ENGINEERING

b.
```
SELECT ENAME, DNAME
FROM   EMPLOYEE2, DEPARTMENT
WHERE EMPLOYEE2.DNO = DEPARTMENT.DNO
AND    BUDGET <= 50000.00
ORDER BY ENAME
```

ENAME	DNAME
CURLY	INFO. SYS.
MOE	INFO. SYS.
SHEMP	ENGINEERING

c.
```
SELECT ENAME, DNAME
FROM   EMPLOYEE3, DEPARTMENT
WHERE EMPLOYEE3.DNO = DEPARTMENT.DNO
AND    BUDGET <= 50000.00
ORDER BY ENAME
```

ENAME	DNAME
CURLY	INFO. SYS.
SHEMP	ENGINEERING

 Copyright Pending 2020

16B. What result tables are produced by executing the following statements?

a.
```
SELECT DEPARTMENT.DNO, DNAME, ENO, ENAME
FROM   DEPARTMENT, EMPLOYEE
WHERE EMPLOYEE.DNO = DEPARTMENT.DNO
AND    ENAME NOT LIKE '%O%E%'
ORDER BY DEPARTMENT.DNO, ENAME
```

b.
```
SELECT DEPARTMENT.DNO, DNAME, ENO, ENAME
FROM   DEPARTMENT, EMPLOYEE2
WHERE EMPLOYEE2.DNO = DEPARTMENT.DNO
AND    ENAME NOT LIKE '%O%E%'
ORDER BY DEPARTMENT.DNO, ENAME
```

c.
```
SELECT DEPARTMENT.DNO, DNAME, ENO, ENAME
FROM   DEPARTMENT, EMPLOYEE3
WHERE EMPLOYEE3.DNO = DEPARTMENT.DNO
AND    ENAME NOT LIKE '%O%E%'
ORDER BY DEPARTMENT.DNO, ENAME
```

Same result for all three queries:

DNO	DNAME	ENO	ENAME
10	ACCOUNTING	2000	LARRY
20	INFO. SYS.	3000	CURLY
40	ENGINEERING	4000	SHEMP

 Copyright Pending 2020

The next three exercises reference Design-1 (DEPARTMENT and EMPLOYEE tables).

16C. Display every employee's name, salary, and the name of the department he works in. Sort the result table by employee name.

```
SELECT ENAME, SALARY, DNAME
FROM   EMPLOYEE, DEPARTMENT
WHERE  EMPLOYEE.DNO = DEPARTMENT.DNO
ORDER BY ENAME
```

```
ENAME    SALARY DNAME
CURLY   3000.00 INFO. SYS.
GEORGE  9000.00 INFO. SYS.
JOE      400.00 ACCOUNTING
LARRY   2000.00 ACCOUNTING
MOE     2000.00 INFO. SYS.
SHEMP    500.00 ENGINEERING
```

16D. Display the employee number and name of any employee who works for a department having a budget that is greater than $24,000.00. Sort the result table by employee number.

```
SELECT ENO, ENAME
FROM   EMPLOYEE, DEPARTMENT
WHERE EMPLOYEE.DNO = DEPARTMENT.DNO
AND    BUDGET > 24000.00
ORDER BY ENO
```

```
ENO ENAME
2000 LARRY
4000 SHEMP
5000 JOE
```

 Copyright Pending 2020

16E. Display the department numbers and names of all departments that have at least one employee earning a salary that is greater than $1,000.00. Sort the result table by department number.

```
SELECT DEPARTMENT.DNO, DNAME
FROM   EMPLOYEE, DEPARTMENT
WHERE EMPLOYEE.DNO = DEPARTMENT.DNO
AND    SALARY > 1000.00
```

```
DNO  DNAME
10   ACCOUNTING
20   INFO. SYS.
20   INFO. SYS.
20   INFO. SYS.
```

This statement only displays columns from the parent-table (DEPARTMENT). Hence, duplicate rows can occur. You can specify DISTINCT to remove the duplicate rows.

16F. Re-write the following statement using:
(a) Alias D for DEPARTMENT and alias E for EMPLOYEE
(b) The JOIN-ON syntax without table aliases
(c) The JOIN-ON syntax with table aliases

```
SELECT ENO, ENAME, SALARY,
          DEPARTMENT.DNO, DNAME, BUDGET
FROM   EMPLOYEE, DEPARTMENT
WHERE EMPLOYEE.DNO = DEPARTMENT.DNO
```

a.
```
SELECT E.ENO, E.ENAME, E.SALARY,
          D.DNO, DNAME,D.BUDGET
FROM   EMPLOYEE E, DEPARTMENT D
WHERE  E.DNO = D.DNO
```

b.
```
SELECT  ENO, ENAME, SALARY,
          DEPARTMENT.DNO, DNAME, BUDGET
FROM   EMPLOYEE INNER JOIN DEPARTMENT
ON       EMPLOYEE.DNO = DEPARTMENT.DNO
```

c.
```
SELECT E.ENO, E.ENAME, E.SALARY,
          D.DNO, D.DNAME, D.BUDGET
FROM   EMPLOYEE E INNER JOIN DEPARTMENT D
ON       E.DNO = D.DNO
```

 Copyright Pending 2020

16G. Re-write the following statement using:
(a) Alias DP for DEPARTMENT and alias EMP for EMPLOYEE
(b) The JOIN-ON syntax without table aliases
(c) The JOIN-ON syntax with table aliases

```
SELECT DEPARTMENT.DNO, ENAME
FROM    DEPARTMENT, EMPLOYEE
WHERE DEPARTMENT.DNO = EMPLOYEE.DNO
AND     BUDGET  > 21000
ORDER BY DEPARTMENT.DNO, ENAME
```

a.
```
SELECT DP.DNO, EMP.ENAME
FROM    DEPARTMENT DP, EMPLOYEE EMP
WHERE DP.DNO = EMP.DNO
AND     DP. BUDGET  > 21000
ORDER BY DP.DNO, EMP.ENAME
```

b.
```
SELECT DEPARTMENT.DNO, EMPLOYEE.ENAME
FROM   DEPARTMENT INNER JOIN EMPLOYEE
ON       DEPARTMENT.DNO = EMPLOYEE.DNO
AND    DEPARTMENT.BUDGET  > 21000
ORDER BY DEPARTMENT.DNO, EMPLOYEE.ENAME
```

c.
```
SELECT DP.DNO, EMP.ENAME
FROM  DEPARTMENT DP INNER JOIN EMPLOYEE EMP
ON       DP.DNO = EMP.DNO
AND    DP.BUDGET  > 21000
ORDER BY DP.DNO, EMP.ENAME
```

 Copyright Pending 2020

Summary Exercises (Chapter 16)

These exercises reference Design-1 (DEPARTMENT and EMPLOYEE tables). Do not display duplicate rows in any result table. Produce two solutions using the FROM-WHERE syntax and the JOIN-ON syntax. Both solutions should specify table aliases.

16H. Display every employee's number, department number, and the name of the department he works in. Sort the result table by employee number. The result should look like:

```
ENO     DNO  DNAME
1000    20   INFO. SYS.
2000    10   ACCOUNTING
3000    20   INFO. SYS.
4000    40   ENGINEERING
5000    10   ACCOUNTING
6000    20   INFO. SYS.
```

```
SELECT E.ENO, D.DNO, D.DNAME
FROM    EMPLOYEE E, DEPARTMENT D
WHERE  E.DNO = D.DNO
ORDER BY E.ENO
```

```
SELECT E.ENO, D.DNO, D.DNAME
FROM    EMPLOYEE E INNER JOIN DEPARTMENT D
ON        E.DNO = D.DNO
ORDER BY E.ENO
```

 Copyright Pending 2020

16I. Display the employee name and salary of any employee who works for a department having a budget that is less than $25,000.00. Sort the result table by employee name. The result should look like:

ENAME	SALARY
CURLY	3000.00
GEORGE	9000.00
MOE	2000.00

```
SELECT E.ENAME, E.SALARY
FROM   EMPLOYEE E, DEPARTMENT D
WHERE E.DNO = D.DNO
AND     D.BUDGET  < 25000.00
ORDER BY E.ENAME
```

```
SELECT E.ENAME, E.SALARY
FROM   EMPLOYEE E INNER JOIN DEPARTMENT D
ON        E.DNO = D.DNO
WHERE  D.BUDGET  < 25000.00
ORDER BY E.ENAME
```

The SELECT statement only displays columns from the child-table (EMPLOYEE). Hence, including DISTINCT is unnecessary.

 Copyright Pending 2020

16J. Display the department numbers and budgets of all departments that have at least one employee earning a salary that is greater than $1,000.00. Sort the result table by department number. The result should look like:

DNO	DNAME	BUDGET
10	ACCOUNTING	75000.00
40	ENGINEERING	25000.00

```
SELECT  DISTINCT D.DNO,  D.DNAME, D.BUDGET
FROM    EMPLOYEE E, DEPARTMENT D
WHERE   E.DNO = D.DNO
AND     E.SALARY < 1000.00
ORDER BY D.DNO
```

```
SELECT  DISTINCT D.DNO,  D.DNAME, D.BUDGET
FROM    EMPLOYEE E INNER JOIN DEPARTMENT D
ON      E.DNO = D.DNO
AND     E.SALARY < 1000.00
ORDER BY D.DNO
```

DISTINCT is necessary because SELECT only displays columns from the parent-table.

Chapter-17 Exercises: More About Inner-Join

17A. Modify Sample Query 17.2 to express each ratio as a percentage.

```
SELECT E.ENO, E.SALARY,
       D.DNO, D.BUDGET, (E.SALARY/D.BUDGET)*100
FROM   DEPARTMENT D, EMPLOYEE E
WHERE D.DNO = E.DNO
```

ENO	SALARY	DNO	BUDGET	(SALARY/BUDGET)*100
1000	2000.00	20	20000.00	10.00
2000	2000.00	10	75000.00	2.66
3000	3000.00	20	20000.00	15.00
4000	500.00	40	25000.00	2.00
5000	400.00	10	75000.00	0.53
6000	9000.00	20	20000.00	45.00

17B. Display the average salary of employees who work for a department with a budget that is greater than $20,000.00.

```
SELECT AVG (SALARY)
FROM   DEPARTMENT D, EMPLOYEE E
WHERE  D.DNO = E.DNO
AND    D.BUDGET > 20000.00
```

AVG (SALARY)
966.66

 Copyright Pending 2020

17Ca. Reference the DEPARTMENT and EMPLOYEE tables. For each department that has employees, display the department name along with the maximum departmental salary for employees who work in the department.

```
SELECT D.DNAME, MAX (E.SALARY)
FROM   DEPARTMENT D, EMPLOYEE E
WHERE D.DNO = E.DNO
GROUP BY D.DNAME
```

DNAME	MAX(SALARY)
ACCOUNTING	2000.00
ENGINEERING	500.00
INFO. SYS.	9000.00

17Cb. Reference the DEPARTMENT and EMPLOYEE tables. For each department with at least one employee, display the department number, department name, department budget, maximum salary, and minimum salary of all employees who work in the department.

```
SELECT D.DNO, D.DNAME, D.BUDGET,
        MAX (E.SALARY), MIN (E.SALARY)
FROM   DEPARTMENT D, EMPLOYEE E
WHERE D.DNO = E.DNO
GROUP BY D.DNO, D.DNAME, D.BUDGET
```

DNO	DNAME	BUDGET	MAX(SALARY)	MIN(SALARY)
10	ACCOUNTING	75000.00	2000.00	400.00
20	INFO. SYS.	20000.00	9000.00	2000.00
40	ENGINEERING	25000.00	500.00	500.00

 Copyright Pending 2020

17D. Assume every employee is given a $20,000.00 raise. Under this circumstance, does any employee have a new salary that exceeds the budget of *his own* department? If yes, display the employee's name, old salary, and new salary.

```
SELECT E.ENAME, E.SALARY, E.SALARY + 20000.00
FROM   EMPLOYEE E, DEPARTMENT D
WHERE E.DNO = D.DNO
AND    (E.SALARY + 20000.00) > D.BUDGET
```

ENAME	SALARY	SALARY+20000.00
MOE	2000.00	22000.00
CURLY	3000.00	23000.00
GEORGE	9000.00	29000.00

 Copyright Pending 2020

Summary Exercises (Chapter 17)

17E. Assume that every employee is given a raise equal to 5% of the employee's departmental budget. Display every employee's name, old salary, and new salary.

```
SELECT E.ENAME, SALARY, SALARY+ (0.05 * BUDGET) NEWSAL
FROM    DEPARTMENT D, EMPLOYEE E
WHERE D.DNO = E.DNO
```

ENAME	SALARY	NEWSAL
MOE	2000.00	3000.0000
LARRY	2000.00	5750.0000
CURLY	3000.00	4000.0000
SHEMP	500.00	1750.0000
JOE	400.00	4150.0000
GEORGE	9000.00	10000.0000

17F. Only consider departments that have employees. How many of these departments have a budget that exceeds $20.000.00? And, what is the total number of employees hired by these departments?

```
SELECT COUNT (DISTINCT D.DNO) DCT, COUNT (E.ENO) EMPCT
FROM    DEPARTMENT D, EMPLOYEE E
WHERE D.DNO = E.DNO
AND      D.BUDGET > 20000.00
```

DCT	EMPCT
2	3

We need to specify DISTINCT in "COUNT (DISTINCT D.DNO)" because duplicate D.DNO values can appear in the join result. We did not specify DISTINCT for COUNT (E.ENO) because we know that the join result cannot contain duplicate E.ENO values.

 Copyright Pending 2020

17G. Extend the previous exercise. Calculate a third column by dividing the second column (number of employees) by the first value (number of departments) to determine the overall average of employees per department.

```
SELECT COUNT (DISTINCT D.DNO) DCT, COUNT (E.ENO) EMPCT,
       (COUNT (E.ENO) *1.00) /COUNT (DISTINCT D.DNO) OAVG
FROM   DEPARTMENT D, EMPLOYEE E
WHERE  D.DNO = E.DNO
AND    D.BUDGET > 20000.00
```

```
DCT  EMPCT       OAVG
  2      3       1.50
```

COUNT returns integer values. Hence, the calculation for the third column divides an integer by an integer. We multiplied by 1.00 to produce get a decimal result.

17H. Only consider departments that have employees. Display the department name and the average departmental salary for each department.

```
SELECT D.DNAME, AVG (E.SALARY) AVGSAL
FROM   DEPARTMENT D, EMPLOYEE E
WHERE D.DNO = E.DNO
GROUP BY D.DNAME
```

```
DNAME            AVGSAL
ACCOUNTING      1200.00
ENGINEERING      500.00
INFO. SYS.      4666.66
```

 Copyright Pending 2020

17I. Modify the previous exercise. Display the department name and the average departmental salary if that average is less than $1,000.00

```
SELECT D.DNAME, AVG (E.SALARY) AVGSAL
FROM   DEPARTMENT D, EMPLOYEE E
WHERE D.DNO = E.DNO
GROUP BY D.DNAME
HAVING AVG (E.SALARY) < 1000.00
```

DNAME	AVGSAL
ENGINEERING	500.00

17J. Only consider departments that have employees. For each such department, display the department name and the minimum salary paid to an employee who works in the department.

```
SELECT D.DNAME, MIN (E.SALARY) MINSAL
FROM    DEPARTMENT D, EMPLOYEE E
WHERE D.DNO = E.DNO
GROUP BY D.DNAME
```

DNAME	MINSAL
ACCOUNTING	400.00
ENGINEERING	500.00
INFO. SYS.	2000.00

 Copyright Pending 2020

17K. Modify the previous exercise. We want to display the department name and the smallest salary paid to some employee who works in the department if that minimum salary value is less than $1,000.00.

```
SELECT D.DNAME, MIN (E.SALARY) MINSAL
FROM   DEPARTMENT D, EMPLOYEE E
WHERE D.DNO = E.DNO
GROUP BY D.DNAME
HAVING MIN (E.SALARY) < 1000.00
```

```
DNAME          MINSAL
ACCOUNTING     400.00
ENGINEERING    500.00
```

The logic of this query allows for an alternative solution (that could be more efficient). We use the WHERE-clause to only select rows with a SALARY this is less than $1,000.00.

```
SELECT D.DNAME, MIN (E.SALARY) MINSAL
FROM   DEPARTMENT D, EMPLOYEE E
WHERE D.DNO = E.DNO
AND E.SALARY < 1000.00
GROUP BY D.DNAME
```

 Copyright Pending 2020

Chapter-18 Exercises: Multi-Table Inner-Join

18A. Display the name of every customer, followed by the name of the state and the name of the region where the customer is located. Display the result in ascending sequence by customer name.

```
SELECT C.CNAME, ST.STNAME, R.RNAME
FROM REGION R,
        STATE ST,
        CUSTOMER C
WHERE R.RNO = ST.RNO
AND     ST.STCODE = C.STCODE
ORDER BY C.CNAME
```

CNAME	STNAME	RNAME
BOLYAI	MASSACHUSETTS	NORTHEAST
BOOLE	FLORIDA	SOUTHEAST
CANTOR	FLORIDA	SOUTHEAST
CHURCH	NEW MEXICO	SOUTHWEST
DECARTES	WASHINGTON	NORTHWEST
EUCLID	MASSACHUSETTS	NORTHEAST
GODEL	GEORGIA	SOUTHEAST
HILBERT	MASSACHUSETTS	NORTHEAST
HYPATIA	MASSACHUSETTS	NORTHEAST
LEIBNIZ	OREGON	NORTHWEST
MANDELBROT	ARIZONA	SOUTHWEST
NEWTON	OREGON	NORTHWEST
PASCAL	WASHINGTON	NORTHWEST
PYTHAGORAS	MASSACHUSETTS	NORTHEAST
RUSSELL	GEORGIA	SOUTHEAST
TURING	ARIZONA	SOUTHWEST
VON NEUMANN	NEW MEXICO	SOUTHWEST
ZENO	MASSACHUSETTS	NORTHEAST

 Copyright Pending 2020

18B. Display the name of every supplier, followed by the names of the region and state where the supplier is located. Display the result in ascending sequence by supplier name.

```
SELECT S.SNAME, R.RNAME, ST.STNAME
FROM  REGION R,
        STATE ST,
        SUPPLIER S
WHERE R.RNO = ST.RNO
AND     ST.STCODE = S.STCODE
ORDER BY S.SNAME
```

SNAME	RNAME	STNAME
SUPPLIER1	NORTHEAST	MASSACHUSETTS
SUPPLIER2	NORTHEAST	MASSACHUSETTS
SUPPLIER3	NORTHEAST	CONNECTICUT
SUPPLIER4	SOUTHEAST	FLORIDA
SUPPLIER5	SOUTHEAST	GEORGIA
SUPPLIER6	NORTHWEST	WASHINGTON
SUPPLIER7	NORTHWEST	OREGON
SUPPLIER8	NORTHWEST	OREGON

 Copyright Pending 2020

18C. For every supplier who can sell your organization some part, display the supplier number and name, the part number and name, and the price you will pay to the supplier for the part. Display the columns in the following left-to-right sequence: SNO, SNAME, PNO, PNAME, and PSPRICE. Sort the result by SNO, PNO.

```
SELECT S.SNO, S.SNAME, P.PNO, P.PNAME, PS.PSPRICE
FROM  PARTSUPP PS,
      PART P,
      SUPPLIER S
WHERE PS.SNO = S.SNO
AND   PS.PNO = P.PNO
ORDER BY S.SNO, P.PNO
```

SNO	SNAME	PNO	PNAME	PSPRICE
S1	SUPPLIER1	P5	PART5	10.00
S2	SUPPLIER2	P1	PART1	10.50
S2	SUPPLIER2	P5	PART5	10.00
S2	SUPPLIER2	P7	PART7	2.00
S3	SUPPLIER3	P3	PART3	12.00
S4	SUPPLIER4	P1	PART1	11.00
S4	SUPPLIER4	P3	PART3	12.50
S4	SUPPLIER4	P4	PART4	12.00
S4	SUPPLIER4	P5	PART5	11.00
S4	SUPPLIER4	P6	PART6	4.00
S4	SUPPLIER4	P7	PART7	3.00
S4	SUPPLIER4	P8	PART8	5.00
S5	SUPPLIER5	P7	PART7	3.50
S6	SUPPLIER6	P6	PART6	4.00
S6	SUPPLIER6	P7	PART7	3.50
S6	SUPPLIER6	P8	PART8	4.00
S8	SUPPLIER8	P6	PART6	4.00
S8	SUPPLIER8	P8	PART8	3.00

18D. We are only interested in customers who have one or more purchase orders. Display the customer's number and name, followed by the name of the state and the name of the region where the customer is located, followed by the date of the purchase order. Display the result in ascending sequence by purchase order date within customer number.

```
SELECT C.CNO, C.CNAME, ST.STNAME, R.RNAME, PO.PODATE
FROM REGION R,
     STATE ST,
     CUSTOMER C,
     PUR_ORDER PO
WHERE R.RNO = ST.RNO
AND   ST.STCODE = C.STCODE
AND   C.CNO = PO.CNO
ORDER BY C.CNO, PO.PODATE
```

CNO	CNAME	STNAME	RNAME	PODATE
100	PYTHAGORAS	MASSACHUSETTS	NORTHEAST	1
100	PYTHAGORAS	MASSACHUSETTS	NORTHEAST	3
110	EUCLID	MASSACHUSETTS	NORTHEAST	47
110	EUCLID	MASSACHUSETTS	NORTHEAST	49
200	HYPATIA	MASSACHUSETTS	NORTHEAST	20
200	HYPATIA	MASSACHUSETTS	NORTHEAST	21
220	ZENO	MASSACHUSETTS	NORTHEAST	5
220	ZENO	MASSACHUSETTS	NORTHEAST	22
220	ZENO	MASSACHUSETTS	NORTHEAST	23
230	BOLYAI	MASSACHUSETTS	NORTHEAST	6
300	NEWTON	OREGON	NORTHWEST	7
300	NEWTON	OREGON	NORTHWEST	8
330	LEIBNIZ	OREGON	NORTHWEST	9
330	LEIBNIZ	OREGON	NORTHWEST	61
400	DECARTES	WASHINGTON	NORTHWEST	62
400	DECARTES	WASHINGTON	NORTHWEST	63
440	PASCAL	WASHINGTON	NORTHWEST	64
440	PASCAL	WASHINGTON	NORTHWEST	65
440	PASCAL	WASHINGTON	NORTHWEST	71
500	HILBERT	MASSACHUSETTS	NORTHEAST	72
600	BOOLE	FLORIDA	SOUTHEAST	73
600	BOOLE	FLORIDA	SOUTHEAST	74
660	CANTOR	FLORIDA	SOUTHEAST	1
660	CANTOR	FLORIDA	SOUTHEAST	75
700	RUSSELL	GEORGIA	SOUTHEAST	1
770	GODEL	GEORGIA	SOUTHEAST	3
800	VON NEUMANN	NEW MEXICO	SOUTHWEST	3
880	TURING	ARIZONA	SOUTHWEST	3
880	TURING	ARIZONA	SOUTHWEST	4
880	TURING	ARIZONA	SOUTHWEST	10
880	TURING	ARIZONA	SOUTHWEST	10

 Copyright Pending 2020

18E. For every part that you can purchase from some supplier, display the part number and name, followed by the supplier number and name, followed by the name of the state where the supplier is located, followed by the price you will pay (PSPRICE) to the supplier for the part. Sort the result by PNO, SNO.

```
SELECT P.PNO, P.PNAME, S.SNO, S.SNAME, ST.STNAME, PS.PSPRICE
FROM  PART P,
      PARTSUPP PS,
      SUPPLIER S,
      STATE ST
WHERE P.PNO = PS.PNO
AND   PS.SNO = S.SNO
AND   S.STCODE = ST.STCODE
ORDER BY P.PNO, S.SNO
```

PNO	PNAME	SNO	SNAME	STNAME	PSPRICE
P1	PART1	S2	SUPPLIER2	MASSACHUSETTS	10.50
P1	PART1	S4	SUPPLIER4	FLORIDA	11.00
P3	PART3	S3	SUPPLIER3	CONNECTICUT	12.00
P3	PART3	S4	SUPPLIER4	FLORIDA	12.50
P4	PART4	S4	SUPPLIER4	FLORIDA	12.00
P5	PART5	S1	SUPPLIER1	MASSACHUSETTS	10.00
P5	PART5	S2	SUPPLIER2	MASSACHUSETTS	10.00
P5	PART5	S4	SUPPLIER4	FLORIDA	11.00
P6	PART6	S4	SUPPLIER4	FLORIDA	4.00
P6	PART6	S6	SUPPLIER6	WASHINGTON	4.00
P6	PART6	S8	SUPPLIER8	OREGON	4.00
P7	PART7	S2	SUPPLIER2	MASSACHUSETTS	2.00
P7	PART7	S4	SUPPLIER4	FLORIDA	3.00
P7	PART7	S5	SUPPLIER5	GEORGIA	3.50
P7	PART7	S6	SUPPLIER6	WASHINGTON	3.50
P8	PART8	S4	SUPPLIER4	FLORIDA	5.00
P8	PART8	S6	SUPPLIER6	WASHINGTON	4.00
P8	PART8	S8	SUPPLIER8	OREGON	3.00

 Copyright Pending 2020

18F. We are only interested in customers who have purchased parts. (These are customers who have completed a purchase order with line items. Recall that some purchase orders may not have any line items.) Display the customer's name, followed by the name of the state and the name of the region where the customer is located, followed by the date of the purchase order, followed by the part number of the purchased part. Display the result in ascending sequence by CNAME, PODATE, PNO.

```
SELECT C.CNAME, ST.STNAME, R.RNAME, PO.PODATE, LI.PNO
FROM REGION R,
        STATE ST,
        CUSTOMER C,
        PUR_ORDER PO,
        LINEITEM LI
WHERE R.RNO = ST.RNO
AND     ST.STCODE = C.STCODE
AND     C.CNO = PO.CNO
AND     PO.PONO = LI.PONO
ORDER BY C.CNAME, PO.PODATE, LI.PNO
```

CNAME	STNAME	RNAME	PODATE	PNO
BOLYAI	MASSACHUSETTS	NORTHEAST	6	P4
BOLYAI	MASSACHUSETTS	NORTHEAST	6	P5
BOOLE	FLORIDA	SOUTHEAST	73	P5
BOOLE	FLORIDA	SOUTHEAST	73	P7
BOOLE	FLORIDA	SOUTHEAST	73	P8
BOOLE	FLORIDA	SOUTHEAST	74	P8
CANTOR	FLORIDA	SOUTHEAST	1	P8
CANTOR	FLORIDA	SOUTHEAST	75	P1
CANTOR	FLORIDA	SOUTHEAST	75	P3
CANTOR	FLORIDA	SOUTHEAST	75	P4
CANTOR	FLORIDA	SOUTHEAST	75	P5
DECARTES	WASHINGTON	NORTHWEST	62	P6
DECARTES	WASHINGTON	NORTHWEST	62	P7
DECARTES	WASHINGTON	NORTHWEST	63	P7
DECARTES	WASHINGTON	NORTHWEST	63	P8
. . . . .				
ZENO	MASSACHUSETTS	NORTHEAST	5	P1
ZENO	MASSACHUSETTS	NORTHEAST	5	P3
ZENO	MASSACHUSETTS	NORTHEAST	22	P4
ZENO	MASSACHUSETTS	NORTHEAST	22	P5
ZENO	MASSACHUSETTS	NORTHEAST	23	P6
ZENO	MASSACHUSETTS	NORTHEAST	23	P7

Total of 62 rows

18G. We are only interested in parts that you can purchase from some supplier. For these parts, display the part number and name, followed by the price you will pay of the part, followed by the number and name of the supplier who will sell you this part at this price. Also include the names of the state and region where the supplier is located. Display the result in ascending sequence by price with part number.

```
SELECT P.PNO, P.PNAME, PS.PSPRICE,
       S.SNO, S.SNAME, ST.STNAME, R.RNAME
FROM PART P,
     PARTSUPP PS,
     SUPPLIER S,
     STATE ST,
     REGION R
WHERE P.PNO = PS.PNO
AND   PS.SNO = S.SNO
AND   S.STCODE = ST.STCODE
AND   ST.RNO = R.RNO
ORDER BY P.PNO, PS.PSPRICE
```

PNO	PNAME	PSPRICE	SNO	SNAME	STNAME	RNAME
P1	PART1	10.50	S2	SUPPLIER2	MASSACHUSETTS	NORTHEAST
P1	PART1	11.00	S4	SUPPLIER4	FLORIDA	SOUTHEAST
P3	PART3	12.00	S3	SUPPLIER3	CONNECTICUT	NORTHEAST
P3	PART3	12.50	S4	SUPPLIER4	FLORIDA	SOUTHEAST
P4	PART4	12.00	S4	SUPPLIER4	FLORIDA	SOUTHEAST
P5	PART5	10.00	S1	SUPPLIER1	MASSACHUSETTS	NORTHEAST
P5	PART5	10.00	S2	SUPPLIER2	MASSACHUSETTS	NORTHEAST
P5	PART5	11.00	S4	SUPPLIER4	FLORIDA	SOUTHEAST
P6	PART6	4.00	S4	SUPPLIER4	FLORIDA	SOUTHEAST
P6	PART6	4.00	S6	SUPPLIER6	WASHINGTON	NORTHWEST
P6	PART6	4.00	S8	SUPPLIER8	OREGON	NORTHWEST
P7	PART7	2.00	S2	SUPPLIER2	MASSACHUSETTS	NORTHEAST
P7	PART7	3.00	S4	SUPPLIER4	FLORIDA	SOUTHEAST
P7	PART7	3.50	S5	SUPPLIER5	GEORGIA	SOUTHEAST
P7	PART7	3.50	S6	SUPPLIER6	WASHINGTON	NORTHWEST
P8	PART8	3.00	S8	SUPPLIER8	OREGON	NORTHWEST
P8	PART8	4.00	S6	SUPPLIER6	WASHINGTON	NORTHWEST
P8	PART8	5.00	S4	SUPPLIER4	FLORIDA	SOUTHEAST

18H. We are only interested in customers who have purchased parts. (These are customers who have completed a purchase order with line items. Recall that some purchase orders may not have any line items.) Display the customer's name, followed by the names of the state and region where the customer is located, followed by the purchase order number, followed by the part number, line-item price (LIPRICE) and purchase price (PSPRICE) of the part. Display the result in ascending sequence by CNAME, PONO, PNO.

```
SELECT C.CNAME, ST.STNAME, R.RNAME,
          PO.PONO, LI.PNO, LI.LIPRICE, PS.PSPRICE
FROM  REGION R,
        STATE ST,
        CUSTOMER C,
        PUR_ORDER PO,
        LINEITEM LI,
        PARTSUPP PS
WHERE R.RNO = ST.RNO
AND     ST.STCODE = C.STCODE
AND     C.CNO = PO.CNO
AND     PO.PONO = LI.PONO
AND     LI.PNO = PS.PNO AND LI.SNO = PS.SNO
ORDER BY C.CNAME, PO.PONO, LI.PNO
```

CNAME	STNAME	RNAME	PONO	PNO	LIPRICE	PSPRICE
BOLYAI	MASSACHUSETTS	NORTHEAST	11124	P4	13.00	12.00
BOLYAI	MASSACHUSETTS	NORTHEAST	11124	P5	11.00	10.00
BOOLE	FLORIDA	SOUTHEAST	11152	P5	12.00	11.00
BOOLE	FLORIDA	SOUTHEAST	11152	P7	3.00	2.00
BOOLE	FLORIDA	SOUTHEAST	11152	P8	4.00	3.00
BOOLE	FLORIDA	SOUTHEAST	11153	P8	4.00	3.00
CANTOR	FLORIDA	SOUTHEAST	11154	P1	11.50	10.50
CANTOR	FLORIDA	SOUTHEAST	11154	P3	14.50	12.50
CANTOR	FLORIDA	SOUTHEAST	11154	P4	13.00	12.00
CANTOR	FLORIDA	SOUTHEAST	11154	P5	11.00	10.00
CANTOR	FLORIDA	SOUTHEAST	11155	P8	4.50	3.00
DECARTES	WASHINGTON	NORTHWEST	11142	P6	5.00	4.00
DECARTES	WASHINGTON	NORTHWEST	11142	P7	3.00	2.00
DECARTES	WASHINGTON	NORTHWEST	11144	P7	4.00	3.00
DECARTES	WASHINGTON	NORTHWEST	11144	P8	5.00	4.00
. . . . .						
VON NEUMANN	NEW MEXICO	SOUTHWEST	11158	P1	11.50	10.50
VON NEUMANN	NEW MEXICO	SOUTHWEST	11158	P3	13.50	12.50
ZENO	MASSACHUSETTS	NORTHEAST	11120	P4	13.00	12.00
ZENO	MASSACHUSETTS	NORTHEAST	11120	P5	11.00	10.00
ZENO	MASSACHUSETTS	NORTHEAST	11121	P6	5.00	4.00
ZENO	MASSACHUSETTS	NORTHEAST	11121	P7	4.00	3.00
ZENO	MASSACHUSETTS	NORTHEAST	11122	P1	11.50	10.50
ZENO	MASSACHUSETTS	NORTHEAST	11122	P3	13.00	12.00

Total of 62 rows

18I. This example extends the previous Exercise 18H. We are only interested in customers who have purchased parts. (These are customers who have completed a purchase order with line items. Recall that some purchase orders may not have any line items.) Display the customer's name, followed by the names of the state and region where the customer is located, followed by the purchase order number, followed by the part number *and name*, followed by the line-item price (LIPRICE) and purchase price (PSPRICE) of the part. Display the result in ascending sequence by CNAME, PONO, PNO.

```
SELECT C.CNAME, ST.STNAME, R.RNAME, PO.PONO,
         LI.PNO, P.PNAME, LI.LIPRICE, PS.PSPRICE
FROM  REGION R,
        STATE ST,
        CUSTOMER C,
        PUR_ORDER PO,
        LINEITEM LI,
        PARTSUPP PS,
        PART P
WHERE R.RNO = ST.RNO
AND     ST.STCODE = C.STCODE
AND     C.CNO = PO.CNO
AND     PO.PONO = LI.PONO
AND     LI.PNO = PS.PNO AND LI.SNO = PS.SNO
AND     PS.PNO = P.PNO
ORDER BY C.CNAME, PO.PONO, LI.PNO
```

CNAME	STNAME	RNAME	PONO	PNO	PNAME	LIPRICE	PSPRICE
BOLYAI	MASSACHUSETTS	NORTHEAST	11124	P4	PART4	13.00	12.00
BOLYAI	MASSACHUSETTS	NORTHEAST	11124	P5	PART5	11.00	10.00
BOOLE	FLORIDA	SOUTHEAST	11152	P5	PART5	12.00	11.00
BOOLE	FLORIDA	SOUTHEAST	11152	P7	PART7	3.00	2.00
BOOLE	FLORIDA	SOUTHEAST	11152	P8	PART8	4.00	3.00
BOOLE	FLORIDA	SOUTHEAST	11153	P8	PART8	4.00	3.00
. . . . .							
ZENO	MASSACHUSETTS	NORTHEAST	11120	P4	PART4	13.00	12.00
ZENO	MASSACHUSETTS	NORTHEAST	11120	P5	PART5	11.00	10.00
ZENO	MASSACHUSETTS	NORTHEAST	11121	P6	PART6	5.00	4.00
ZENO	MASSACHUSETTS	NORTHEAST	11121	P7	PART7	4.00	3.00
ZENO	MASSACHUSETTS	NORTHEAST	11122	P1	PART1	11.50	10.50
ZENO	MASSACHUSETTS	NORTHEAST	11122	P3	PART3	13.00	12.00

Total of 62 rows

18J. We are only interested in customers having names that begin with the letter “B”. Display the customers’ name, followed by the name of the state and the name of the region where the customer is located. Display the result in ascending sequence by customer name.

```
SELECT C.CNAME, ST.STNAME, R.RNAME
FROM REGION R,
        STATE ST,
        CUSTOMER C
WHERE R.RNO = ST.RNO
AND     ST.STCODE = C.STCODE
AND     CNAME LIKE 'B%'
ORDER BY C.CNAME
```

CNAME	STNAME	RNAME
BOLYAI	MASSACHUSETTS	NORTHEAST
BOOLE	FLORIDA	SOUTHEAST

18K. We are only interested in suppliers who are located in Florida (STCODE = ‘FL’) who sell parts having a weight (PWT) that is less than 20 pounds. Display each supplier’s number and name, followed by the part number, name, and weight, followed by the price you will pay to the supplier for the part. Sort the result by SNO, PNO.

```
SELECT S.SNO, S.SNAME, P.PNO, P.PNAME, P.PWT, PS.PSPRICE
FROM  PARTSUPP PS,
        PART P,
        SUPPLIER S
WHERE PS.SNO = S.SNO
AND     PS.PNO = P.PNO
AND     S.STCODE = 'FL'
AND     P.PWT < 20
ORDER BY S.SNO, P.PNO
```

SNO	SNAME	PNO	PNAME	PWT	PSPRICE
S4	SUPPLIER4	P4	PART4	10	12.00
S4	SUPPLIER4	P6	PART6	12	4.00
S4	SUPPLIER4	P8	PART8	15	5.00

 Copyright Pending 2020

18L. The basic objective is to determine total number of parts each customer has purchased. This amount is equal to sum of the LINEITEM.QTY values for each customer. Display the customer's number followed by the total number of parts purchased. Sort the result in ascending sequence by customer number.

```
SELECT C.CNO, SUM (LI.QTY) SUMQTY
FROM  CUSTOMER C,
        PUR_ORDER PO,
        LINEITEM LI
WHERE C.CNO = PO.CNO
AND     PO.PONO = LI.PONO
GROUP BY C.CNO
ORDER BY C.CNO
```

CNO	SUMQTY
100	60
110	70
200	50
220	100
230	20
300	30
330	60
400	60
440	120
500	30
600	95
660	90
700	40
770	30
800	30
880	60

18M. This example is a minor modification to the previous exercise (18L). Along with the customer number, we also want to display the customer name and state code. (This exercise is really a review of grouping.)

```
SELECT C.CNO, C.CNAME, C.STCODE, SUM (LI.QTY) SUMQTY
FROM  CUSTOMER C,
        PUR_ORDER PO,
        LINEITEM LI
WHERE C.CNO = PO.CNO
AND     PO.PONO = LI.PONO
GROUP BY C.CNO, C.CNAME, C.STCODE
ORDER BY C.CNO
```

CNO	CNAME	STCODE	SUMQTY
100	PYTHAGORAS	MA	60
110	EUCLID	MA	70
200	HYPATIA	MA	50
220	ZENO	MA	100
230	BOLYAI	MA	20
300	NEWTON	OR	30
330	LEIBNIZ	OR	60
400	DECARTES	WA	60
440	PASCAL	WA	120
500	HILBERT	MA	30
600	BOOLE	FL	95
660	CANTOR	FL	90
700	RUSSELL	GE	40
770	GODEL	GE	30
800	VON NEUMANN	NM	30
880	TURING	AZ	60

 Copyright Pending 2020

18N. This example extends the previous exercise (18M). We only want to display information about those customers who have purchased a total of more than 100 parts.

```
SELECT C.CNO, C.CNAME, C.STCODE, SUM (LI.QTY) SUMQTY
FROM  CUSTOMER C,
        PUR_ORDER PO,
        LINEITEM LI
WHERE C.CNO = PO.CNO
AND     PO.PONO = LI.PONO
GROUP BY C.CNO, C.CNAME, C.STCODE
HAVING SUM (LI.QTY) > 100
ORDER BY C.CNO
```

```
CNO  CNAME              STCODE   SUMQTY
440  PASCAL             WA          120
```

 Copyright Pending 2020

18O. Re-code and execute Sample Queries 18.9 and 18.10 using the JOIN-ON syntax.

Sample Query 18.9:

```
SELECT R.RNAME, ST.STNAME, S.SNAME, P.PNAME, LI.PONO
FROM REGION R
        INNER JOIN STATE ST       ON R.RNO = ST.RNO
        INNER JOIN SUPPLIER S     ON ST.STCODE = S.STCODE
        INNER JOIN PARTSUPP PS ON S.SNO = PS.SNO
        INNER JOIN PART P           ON PS.PNO = P.PNO
        INNER JOIN LINEITEM LI  ON PS.PNO = LI.PNO AND PS.SNO = LI.SNO
ORDER BY R.RNAME, ST.STNAME, S.SNAME, P.PNAME, LI.PONO
```

Sample Query 18.10:

```
SELECT R.RNAME, ST.STNAME, S.SNAME, P.PNAME, LI.PONO, PO.POSTATUS
FROM REGION R
        INNER JOIN STATE ST          ON R.RNO = ST.RNO
        INNER JOIN SUPPLIER S        ON ST.STCODE = S.STCODE
        INNER JOIN PARTSUPP PS     ON S.SNO = PS.SNO
        INNER JOIN PART P              ON PS.PNO = P.PNO
        INNER JOIN LINEITEM LI       ON PS.PNO = LI.PNO AND PS.SNO = LI.SNO
        INNER JOIN PUR_ORDER PO    ON LI.PONO = PO.PONO
ORDER BY R.RNAME, ST.STNAME, S.SNAME, P.PNAME, LI.PONO
```

 Copyright Pending 2020

Summary Exercises (Chapter 18)

18P. Consider all customers. Display each customer's location (region name and state name) followed by the customer's name. Display the result in ascending sequence by customer name within region name.

```
SELECT R.RNAME, ST.STNAME, C.CNAME
FROM REGION R,
        STATE ST,
        CUSTOMER C
WHERE R.RNO = ST.RNO
AND     ST.STCODE = C.STCODE
ORDER BY R.RNAME, C.CNAME
```

RNAME	STNAME	CNAME
NORTHEAST	MASSACHUSETTS	BOLYAI
NORTHEAST	MASSACHUSETTS	EUCLID
NORTHEAST	MASSACHUSETTS	HILBERT
NORTHEAST	MASSACHUSETTS	HYPATIA
NORTHEAST	MASSACHUSETTS	PYTHAGORAS
NORTHEAST	MASSACHUSETTS	ZENO
NORTHWEST	WASHINGTON	DECARTES
NORTHWEST	OREGON	LEIBNIZ
NORTHWEST	OREGON	NEWTON
NORTHWEST	WASHINGTON	PASCAL
SOUTHEAST	FLORIDA	BOOLE
SOUTHEAST	FLORIDA	CANTOR
SOUTHEAST	GEORGIA	GODEL
SOUTHEAST	GEORGIA	RUSSELL
SOUTHWEST	NEW MEXICO	CHURCH
SOUTHWEST	ARIZONA	MANDELBROT
SOUTHWEST	ARIZONA	TURING
SOUTHWEST	NEW MEXICO	VON NEUMANN

 Copyright Pending 2020

18Q. Only consider regions that have suppliers. Display the name of the region name followed by the name of the supplier. Display the result in ascending sequence by supplier name within region name.

```
SELECT R.RNAME, S.SNAME
FROM REGION R,
        STATE ST,
        SUPPLIER S
WHERE R.RNO = ST.RNO
AND     ST.STCODE = S.STCODE
ORDER BY R.RNAME, S.SNAME
```

RNAME	SNAME
NORTHEAST	SUPPLIER1
NORTHEAST	SUPPLIER2
NORTHEAST	SUPPLIER3
NORTHWEST	SUPPLIER6
NORTHWEST	SUPPLIER7
NORTHWEST	SUPPLIER8
SOUTHEAST	SUPPLIER4
SOUTHEAST	SUPPLIER5

18R. Only consider Massachusetts (STCODE=‘MA’) customers that have completed a purchase order. For each such customer, display the customer’s name, and the number of purchase orders the customer has completed. (This only requires a two-table join. This exercise sets the stage for the next two exercises.)

```
SELECT      C.CNAME, COUNT (*)  POCT
FROM        CUSTOMER  C,
            PUR_ORDER PO
WHERE       C.CNO = PO.CNO
AND         C.STCODE = 'MA'
GROUP BY C.CNAME
ORDER BY C.CNAME’
```

CNAME	POCT
BOLYAI	1
EUCLID	2
HILBERT	1
HYPATIA	2
PYTHAGORAS	2
ZENO	3

 Copyright Pending 2020

18S. Only consider customers located in Region3 (RNO=3) that have completed a purchase order. For each such customer, display the customer's name, and the number of purchase orders the customer has completed.

```
SELECT      C.CNAME, COUNT (*) POCT
FROM        STATE ST,
            CUSTOMER  C,
            PUR_ORDER PO
WHERE       ST.STCODE = C.STCODE
AND         C.CNO = PO.CNO
AND         ST.RNO = 3
GROUP BY C.CNAME
ORDER BY C.CNAME
```

```
CNAME              POCT
BOOLE                 2
CANTOR                2
GODEL                 1
RUSSELL               1
```

18T. Reconsider the preceding exercise. You realize that customer names (CNAME values) are not necessarily unique. Revise the query objective to state: Only consider customers located in Region 3 (RNO=3) who have completed a purchase order. For each such customer, display the customer's number and name, followed by the number of purchase orders the customer has completed.

```
SELECT      C.CNO, C.CNAME, COUNT (*) POCT
FROM        STATE       ST,
            CUSTOMER  C,
            PUR_ORDER PO
WHERE       ST.STCODE = C.STCODE
AND         C.CNO = PO.CNO
AND         ST.RNO = 3
GROUP BY C.CNO, C.CNAME
ORDER BY C.CNO, C.CNAME
```

```
CNO  CNAME          POCT
600  BOOLE             2
660  CANTOR            2
700  RUSSELL           1
770  GODEL             1
```

 Copyright Pending 2020

18U. How many parts were sold in states that are located in the Northeast or Southeast regions? Display the region name, followed by the state name, followed by the total quantity of parts sold in the state. Sort the result in ascending sequence by state name within region name.

```
SELECT R.RNAME, ST.STNAME, SUM (LI.QTY) SUMQTY
FROM REGION      R,
     STATE       ST,
     CUSTOMER    C,
     PUR_ORDER   PO,
     LINEITEM    LI
WHERE R.RNO = ST.RNO
AND   ST.STCODE = C.STCODE
AND   C.CNO = PO.CNO
AND   PO.PONO = LI.PONO
AND   R.RNAME IN ('NORTHEAST', 'SOUTHEAST')
GROUP BY R.RNAME, ST.STNAME
ORDER BY R.RNAME, ST.STNAME
```

RNAME	STNAME	SUMQTY
NORTHEAST	MASSACHUSETTS	330
SOUTHEAST	FLORIDA	185
SOUTHEAST	GEORGIA	70

 Copyright Pending 2020

18V. Display the region name and state name, followed by the total quantity of parts sold in the state if that quantity exceeds 100. Sort the result in ascending sequence by state name within region name.

```
SELECT R.RNAME, ST.STNAME, SUM (LI.QTY) PARTQTY
FROM REGION      R,
     STATE       ST,
     CUSTOMER    C,
     PUR_ORDER   PO,
     LINEITEM    LI
WHERE R.RNO = ST.RNO
AND   ST.STCODE = C.STCODE
AND   C.CNO = PO.CNO
AND   PO.PONO = LI.PONO
GROUP BY R.RNAME, ST.STNAME
HAVING SUM (LI.QTY) > 100
ORDER BY R.RNAME, ST.STNAME
```

RNAME	STNAME	PARTQTY
NORTHEAST	MASSACHUSETTS	330
NORTHWEST	WASHINGTON	180
SOUTHEAST	FLORIDA	185

 Copyright Pending 2020

Chapter-19 - Outer-Join: Getting Started

19A. Reference the REGION and STATE tables in the MTPCH database. Designate REGION as the left-table. Execute a full outer-join.

```
SELECT *
FROM REGION R FULL OUTER JOIN STATE ST
ON  R.RNO = ST.RNO
```

RNO	RNAME	CLIMATE	STCODE	STNAME	POPULATION	RNO
1	NORTHEAST	Cold	CT	CONNECTICUT	3502000	1
1	NORTHEAST	Cold	MA	MASSACHUSETTS	6450000	1
2	NORTHWEST	Cold	OR	OREGON	3747000	2
2	NORTHWEST	Cold	WA	WASHINGTON	6468000	2
3	SOUTHEAST	Hot	FL	FLORIDA	18251000	3
3	SOUTHEAST	Hot	GE	GEORGIA	9545000	3
4	SOUTHWEST	Hot	NM	NEW MEXICO	1970000	4
4	SOUTHWEST	Hot	AZ	ARIZONA	6339000	4
5	MIDWEST	Empty	-	-	-	-

19B. Reference the REGION and STATE tables. Execute a left outer-join. Designate REGION as the left-table. (Observe that the result is the same as the previous exercise.)

```
SELECT *
FROM REGION R LEFT OUTER JOIN STATE ST
ON  R.RNO = ST.RNO
```

Result: Same as previous exercise. But row sequence may be different because neither statement specifies an ORDER BY clause/

 Copyright Pending 2020

19C. Reference the REGION and STATE tables. Execute a right outer-join. Designate STATE as the right-table. (Observe that the result is the same as that produced by an inner-join.)

```
SELECT *
FROM REGION R RIGHT OUTER JOIN STATE ST
ON  R.RNO = ST.RNO
```

RNO	RNAME	CLIMATE	STCODE	STNAME	POPULATION	RNO
1	NORTHEAST	Cold	CT	CONNECTICUT	3502000	1
1	NORTHEAST	Cold	MA	MASSACHUSETTS	6450000	1
2	NORTHWEST	Cold	OR	OREGON	3747000	2
2	NORTHWEST	Cold	WA	WASHINGTON	6468000	2
3	SOUTHEAST	Hot	FL	FLORIDA	18251000	3
3	SOUTHEAST	Hot	GE	GEORGIA	9545000	3
4	SOUTHWEST	Hot	NM	NEW MEXICO	1970000	4
4	SOUTHWEST	Hot	AZ	ARIZONA	6339000	4

19D. We generally discourage use of the right outer-join. But, for tutorial purposes only, we ask you to use the right outer-join to satisfy the query objective for Sample Query 19.2: Reference the DEPARTMENT and EMPLOYEE3 tables. Display all information about *all* departments along with all information about employees who work in those departments. (Display information about every department, even if the department does not have any employees.) Sort the result by DNO, ENO.

```
SELECT D.DNO, D.DNAME, D.BUDGET,
          E.ENO, E.ENAME, E.SALARY, E.DNO
FROM EMPLOYEE3 E RIGHT OUTER JOIN DEPARTMENT D
ON   E.DNO = D.DNO
ORDER BY D.DNO, E. ENO
```

DNO	DNAME	BUDGET	ENO	ENAME	SALARY	DNO
10	ACCOUNTING	75000.00	2000	LARRY	2000.00	10
10	ACCOUNTING	75000.00	5000	JOE	400.00	10
20	INFO. SYS.	20000.00	3000	CURLY	3000.00	20
30	PRODUCTION	7000.00	-	-	-	-
40	ENGINEERING	25000.00	4000	SHEMP	500.00	40

Exercises 19E AND 19F reference the following versions of the MAN and DOG tables. These tables are related via a PK-FK relationship (DOG.MNO references MAN.MNO). These are "paper and pencil" exercises. The MAN and DOG tables were not created in the CREATE-ALL-TABLES scripts for the sample tables.

MAN

MNO	MNAME
77	MOE
88	LARRY
99	CURLY

DOG

DNO	DNAME	MNO
1000	SPOT	99
3000	ROVER	77
2000	WALLY	99
4000	SPIKE	99

19E. What are the result tables produced by the following left outer-join operations?

a. SELECT *
FROM MAN LEFT OUTER JOIN DOG
ON MAN.MNO = DOG.MNO
WHERE MAN.MNAME LIKE '%R%'

MNO	MNAME	DNO	DNAME	MNO1
88	LARRY	-	-	-
99	CURLY	1000	SPOT	99
99	CURLY	2000	WALLY	99
99	CURLY	4000	SPIKE	99

Note: WHERE-clause applied after outer-join.

b. SELECT *
FROM MAN LEFT OUTER JOIN DOG
ON MAN.MNO = DOG.MNO **AND** MAN.MNAME LIKE '%R%'

MNO	MNAME	DNO	DNAME	MNO1
77	MOE	-	-	-
88	LARRY	-	-	-
99	CURLY	1000	SPOT	99
99	CURLY	2000	WALLY	99
99	CURLY	4000	SPIKE	99

Note: AND-clause applied during outer-join.

Observation: MOE's dog (ROVER) information is not displayed

 Copyright Pending 2020

19F. What are the result tables produced by the following left outer-join operations?

a. SELECT *
FROM MAN LEFT OUTER JOIN DOG
ON MAN.MNO = DOG.MNO
WHERE DOG.DNAME LIKE 'S%'

MNO	MNAME	DNO	DNAME	MNO1
99	CURLY	1000	SPOT	99
99	CURLY	4000	SPIKE	99

This WHERE-clause is applied after the outer-join. Note that it references a column (DNAME) in the right-table (the child-table). Hence, it will never select any of the non-matching parent rows. You should consider coding an inner-join from this statement.

SELECT *
FROM MAN **INNER** JOIN DOG
ON MAN.MNO = DOG.MNO
WHERE DOG.DNAME LIKE 'S%'

b. SELECT *
FROM MAN LEFT OUTER JOIN DOG
ON MAN.MNO = DOG.MNO **AND** DOG.DNAME LIKE 'S%'

MNO	MNAME	DNO	DNAME	MNO1
77	MOE	-	-	-
88	LARRY	-	-	-
99	CURLY	1000	SPOT	99
99	CURLY	4000	SPIKE	99

 Copyright Pending 2020

Summary Exercises (Chapter 19)

Exercises 19G-19J reference the DEPARTMENT and EMPLOYEE tables.

19G. Display the name and budget for *all* departments. Also display the name and salary of any employee who works in a department having a budget that exceeds $50,000.00. The result should look like:

DNAME	BUDGET	ENAME	SALARY
ACCOUNTING	75000.00	LARRY	2000.00
ACCOUNTING	75000.00	JOE	400.00
INFO. SYS.	20000.00	-	-
PRODUCTION	7000.00	-	-
ENGINEERING	25000.00	-	-

```
SELECT DNAME, BUDGET, ENAME, SALARY
FROM DEPARTMENT D LEFT OUTER JOIN EMPLOYEE E
ON      D.DNO = E.DNO AND D.BUDGET > 50000.00
```

19H. Display the name and budget of those departments having a budget that is less than $50,000.00. If any such department has employees, also display name and salary of the employees. The result should look like:

DNAME	BUDGET	ENAME	SALARY
INFO. SYS.	20000.00	MOE	2000.00
INFO. SYS.	20000.00	GEORGE	9000.00
INFO. SYS.	20000.00	CURLY	3000.00
PRODUCTION	7000.00	-	-
ENGINEERING	25000.00	SHEMP	500.00

```
SELECT DNAME, BUDGET, ENAME, SALARY
FROM DEPARTMENT D LEFT OUTER JOIN EMPLOYEE E
ON   D.DNO = E.DNO
WHERE D.BUDGET < 50000.00
```

 Copyright Pending 2020

19I. Display the name and salary of any employee who earns more than $1,000.00, along with the employee's departmental name and budget. The result should look like:

ENAME	SALARY	DNAME	BUDGET
LARRY	2000.00	ACCOUNTING	75000.00
MOE	2000.00	INFO. SYS.	20000.00
GEORGE	9000.00	INFO. SYS.	20000.00
CURLY	3000.00	INFO. SYS.	20000.00

Following left outer-join will work.

```
SELECT ENAME, SALARY, DNAME, BUDGET
FROM DEPARTMENT D LEFT OUTER JOIN EMPLOYEE E
ON   D.DNO = E.DNO
WHERE E.SALARY > 1000.00
```

It is better to code an inner join. Observe that the WHERE-clause is applied to a column from the EMPLOYEE table (the right-table). This would eliminate any non-matching EMPLOYEE rows containing null values produced by the left outer-join. (I.e., The WHERE-clause effectively "undoes" the left-outer-join)

```
SELECT ENAME, SALARY, DNAME, BUDGET
FROM DEPARTMENT D INNER JOIN EMPLOYEE E
ON   D.DNO = E.DNO
WHERE E.SALARY > 1000.00
```

19J. Display the name and budget for *all* departments. Also, display the name and salary of any employee who works in each department and earns more than $1,000.00. The result should look like:

DNAME	BUDGET	ENAME	SALARY
ACCOUNTING	75000.00	LARRY	2000.00
INFO. SYS.	20000.00	MOE	2000.00
INFO. SYS.	20000.00	GEORGE	9000.00
INFO. SYS.	20000.00	CURLY	3000.00
PRODUCTION	7000.00	-	-
ENGINEERING	25000.00	-	-

```
SELECT DNAME, BUDGET, ENAME, SALARY
FROM DEPARTMENT D LEFT OUTER JOIN EMPLOYEE E
ON   D.DNO = E.DNO AND E.SALARY > 1000.00
```

19K. Consider Sample Query 19.10 shown below.

```
SELECT *
FROM DEPARTMENT D LEFT OUTER JOIN EMPLOYEE E
ON   D.DNO = E.DNO AND D.BUDGET < 24000.00
WHERE D.DNO <> 40
ORDER BY D.DNO, E.ENO
```

Assume you replaced the keyword WHERE with the keyword AND to formulate the following statement.

```
SELECT *
FROM DEPARTMENT D LEFT OUTER JOIN EMPLOYEE E
ON  D.DNO = E.DNO AND D.BUDGET < 24000.00
AND D.DNO <> 40
ORDER BY D.DNO
```

Is this statement equivalent to the result shown for Sample Query 19.10? (Does it produce the same correct result?)

Answer: Different results.

In Sample Query 19.10, the DEPARTMENT 40 row does not match the join-condition, but it appears in the outer-join intermediate result because DEPARTMENT is the left table. Then the WHERE D.DNO <> 40 condition, executed after of outer-join, removes the DEPARTMENT 40 row from the final result.

In modified statement, AND D.DNO <> 40 is part of outer-join and the row for DEPARTMENT 40 does not match the outer join-condition. However, it appears in the outer-join result because DEPARTMENT is the left table. Hence, the DEPARTMENT 40 row appears in the final result.

Chapter-20 - Multi-Table Left Outer-Joins

20A. Display the number and name of *every* region, the number and name of *every* state in each region, and the number and name of *every* supplier in each state. Display the columns in the following left-to-right sequence: RNO, RNAME, STCODE, STNAME, SNO, and SNAME. Sort the result by SNO within STCODE within RNO. (Hint: Follow the REGION-STATE-SUPPLIER hierarchical path.)

```
SELECT R.RNO, R.RNAME, ST.STCODE, ST.STNAME, S.SNO, S.SNAME
FROM REGION R
        LEFT OUTER JOIN STATE ST    ON R.RNO = ST.RNO
        LEFT OUTER JOIN SUPPLIER S ON ST.STCODE = S.STCODE
ORDER BY R.RNO, ST.STCODE, S.SNO
```

RNO	RNAME	STCODE	STNAME	SNO	SNAME
1	NORTHEAST	CT	CONNECTICUT	S3	SUPPLIER3
1	NORTHEAST	MA	MASSACHUSETTS	S1	SUPPLIER1
1	NORTHEAST	MA	MASSACHUSETTS	S2	SUPPLIER2
2	NORTHWEST	OR	OREGON	S7	SUPPLIER7
2	NORTHWEST	OR	OREGON	S8	SUPPLIER8
2	NORTHWEST	WA	WASHINGTON	S6	SUPPLIER6
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4
3	SOUTHEAST	GE	GEORGIA	S5	SUPPLIER5
4	SOUTHWEST	AZ	ARIZONA	-	-
4	SOUTHWEST	NM	NEW MEXICO	-	-
5	MIDWEST	-	-	-	-

 Copyright Pending 2020

20B. Display the number and name of *every* region, the code and name of *every* state in each region, the number and name of *every* supplier in each state, and the part number of every part that you can purchase from these suppliers. Display the columns in the following left-to-right sequence: RNO, RNAME, STCODE, STNAME, SNO, SNAME, PNO. Sort the result by PNO within SNO within STCODE within RNO. (Hint: Follow the REGION-STATE-SUPPLIER-PARTSUPP hierarchical path.)

```
SELECT R.RNO, R.RNAME, ST.STCODE,ST.STNAME,
        S.SNO, S.SNAME, PS.PNO
FROM REGION R
      LEFT OUTER JOIN STATE ST       ON R.RNO = ST.RNO
      LEFT OUTER JOIN SUPPLIER S     ON ST.STCODE = S.STCODE
      LEFT OUTER JOIN PARTSUPP  PS ON  S.SNO = PS.SNO
ORDER BY R.RNO, ST.STCODE, S.SNO, PS.PNO
```

RNO	RNAME	STCODE	STNAME	SNO	SNAME	PNO
1	NORTHEAST	CT	CONNECTICUT	S3	SUPPLIER3	P3
1	NORTHEAST	MA	MASSACHUSETTS	S1	SUPPLIER1	P5
1	NORTHEAST	MA	MASSACHUSETTS	S2	SUPPLIER2	P1
1	NORTHEAST	MA	MASSACHUSETTS	S2	SUPPLIER2	P5
1	NORTHEAST	MA	MASSACHUSETTS	S2	SUPPLIER2	P7
2	NORTHWEST	OR	OREGON	S7	SUPPLIER7	-
2	NORTHWEST	OR	OREGON	S8	SUPPLIER8	P6
2	NORTHWEST	OR	OREGON	S8	SUPPLIER8	P8
2	NORTHWEST	WA	WASHINGTON	S6	SUPPLIER6	P6
2	NORTHWEST	WA	WASHINGTON	S6	SUPPLIER6	P7
2	NORTHWEST	WA	WASHINGTON	S6	SUPPLIER6	P8
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P1
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P3
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P4
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P5
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P6
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P7
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P8
3	SOUTHEAST	GE	GEORGIA	S5	SUPPLIER5	P7
4	SOUTHWEST	AZ	ARIZONA	-	-	-
4	SOUTHWEST	NM	NEW MEXICO	-	-	-
5	MIDWEST	-	-	-	-	-

 Copyright Pending 2020

20C. Display the number and name of every region, the code of every state in each region, the number and name of every supplier in each state, and the purchase-order number and part-number that appeared in every line-item for each supplier. Sort the result by PNO within PONO within SNO within STCODE within RNO. (Hints: Follow the REGION-STATE-SUPPLIER-PARTSUPP-LINEITEM hierarchy. Note that no data from the PARTSUPP table is displayed. This table serves as a link-table between the SUPPLIER and LINEITEM tables.)

```
SELECT R.RNO, R.RNAME, ST.STCODE, S.SNO, S.SNAME,
       LI.PONO, LI.PNO
FROM REGION R
     LEFT OUTER JOIN STATE ST       ON R.RNO = ST.RNO
     LEFT OUTER JOIN SUPPLIER S     ON ST.STCODE = S.STCODE
     LEFT OUTER JOIN PARTSUPP  PS ON  S.SNO = PS.SNO
     LEFT OUTER JOIN LINEITEM  LI
                    ON  PS.SNO = LI.SNO AND PS.PNO = LI.PNO
ORDER BY R.RNO, ST.STCODE,S.SNO, LI.PONO,LI.PNO
```

RNO	RNAME	STCODE	SNO	SNAME	PONO	PNO
1	NORTHEAST	CT	S3	SUPPLIER3	11101	P3
1	NORTHEAST	CT	S3	SUPPLIER3	11102	P3
1	NORTHEAST	CT	S3	SUPPLIER3	11122	P3
1	NORTHEAST	MA	S1	SUPPLIER1	11108	P5
.	.	.	.	.	.	.
.	.	.	.	.	.	.
1	NORTHEAST	MA	S2	SUPPLIER2	11160	P7
2	NORTHWEST	OR	S7	SUPPLIER7	-	-
2	NORTHWEST	OR	S8	SUPPLIER8	11142	P6
2	NORTHWEST	OR	S8	SUPPLIER8	11149	P8
2	NORTHWEST	OR	S8	SUPPLIER8	11152	P8
2	NORTHWEST	OR	S8	SUPPLIER8	11153	P8
2	NORTHWEST	OR	S8	SUPPLIER8	11155	P8
2	NORTHWEST	WA	S6	SUPPLIER6	11121	P6
2	NORTHWEST	WA	S6	SUPPLIER6	11144	P8
2	NORTHWEST	WA	S6	SUPPLIER6	11146	P8
2	NORTHWEST	WA	S6	SUPPLIER6	-	-
3	SOUTHEAST	FL	S4	SUPPLIER4	11102	P4
.	.	.	.	.	.	.
.	.	.	.	.	.	.
3	SOUTHEAST	GE	S5	SUPPLIER5	11149	P7
4	SOUTHWEST	AZ	-	-	-	-
4	SOUTHWEST	NM	-	-	-	-
5	MIDWEST	-	-	-	-	-

Result table has 67 rows

 Copyright Pending 2020

20D. Display the number and name of *every* part, the supplier number of every supplier who can sell the part, and the line-item price for each sale of the part by the supplier. Display the columns in the following left-to-right sequence: PNO, PNAME, SNO, and LIPRICE. Sort the result by SNO within PNO. (Hint: Follow the PART-PARTSUPP-LINEITEM hierarchy.)

```
SELECT P.PNO, P.PNAME, PS.SNO, LI.LIPRICE
FROM  PART P
LEFT   OUTER JOIN PARTSUPP PS ON P.PNO = PS.PNO
LEFT   OUTER JOIN LINEITEM LI   ON PS.PNO = LI.PNO
                                AND PS.SNO = LI.SNO
ORDER BY P.PNO, PS.SNO
```

PNO	PNAME	SNO	LIPRICE
P1	PART1	S2	11.50
.	.	.	.
.	.	.	.
P1	PART1	S4	12.00
P2	PART2	-	-
P3	PART3	S3	12.00
.	.	.	.
.	.	.	.
P7	PART7	S2	3.00
P7	PART7	S2	3.00
P7	PART7	S2	3.00
P7	PART7	S2	3.00
P7	PART7	S2	3.00
P7	PART7	S2	3.00
P7	PART7	S4	4.00
P7	PART7	S4	4.00
P7	PART7	S5	4.50
P7	PART7	S5	4.50
P7	PART7	S6	-
P8	PART8	S4	6.00
P8	PART8	S4	6.00
P8	PART8	S4	6.00
P8	PART8	S6	5.00
P8	PART8	S6	5.00
P8	PART8	S8	4.00
P8	PART8	S8	4.00
P8	PART8	S8	4.00
P8	PART8	S8	4.50

Result table has 64 rows

 Copyright Pending 2020

20E. Display the number and name of *every* region followed by the minimum and maximal PSPRICE values of parts sold by suppliers in each region. Display the columns in the following left-to-right sequence: RNO, RNAME, and MINPSPRICE and MAXPSPRICE values (column headings for the min and max prices of parts sold by suppliers in each region). Sort the result by RNO. Display zero for null values. (Hint: Follow the REGION-STATE-SUPPLIER-PARTSUPP hierarchy.)

```
SELECT R.RNO, R.RNAME,
       COALESCE (MIN (PS.PSPRICE), 0) MINPSPRICE,
       COALESCE (MAX (PS.PSPRICE), 0) MAXPSPRICE
FROM REGION R
       LEFT OUTER JOIN STATE ST       ON R.RNO = ST.RNO
       LEFT OUTER JOIN SUPPLIER S     ON ST.STCODE = S.STCODE
       LEFT OUTER JOIN PARTSUPP PS  ON S.SNO = PS.SNO
GROUP BY R.RNO, R.RNAME
ORDER BY R.RNO
```

RNO	RNAME	MINPSPRICE	MAXPSPRICE
1	NORTHEAST	2.00	12.00
2	NORTHWEST	3.00	4.00
3	SOUTHEAST	3.00	12.50
4	SOUTHWEST	0.00	0.00
5	MIDWEST	0.00	0.00

20F. Display the number and name of *every* Western region. (The RNAME value ends with 'WEST'.) Also, display the code and name of *every* state in each Western region, and the number and name of *every* supplier in each Western region. Sort the result by SNO within STCODE within RNO. (Hint: Follow the REGION-STATE-SUPPLIER hierarchical path.)]

```
SELECT R.RNO, R.RNAME, ST.STCODE, ST.STNAME, S.SNO, S.SNAME
FROM REGION R
        LEFT OUTER JOIN STATE ST    ON R.RNO = ST.RNO
        LEFT OUTER JOIN SUPPLIER S ON ST.STCODE = S.STCODE
WHERE RTRIM (RNAME) LIKE '%WEST'
ORDER BY R.RNO, ST.STCODE, S.SNO
```

RNO	RNAME	STCODE	STNAME	SNO	SNAME
2	NORTHWEST	OR	OREGON	S7	SUPPLIER7
2	NORTHWEST	OR	OREGON	S8	SUPPLIER8
2	NORTHWEST	WA	WASHINGTON	S6	SUPPLIER6
4	SOUTHWEST	AZ	ARIZONA	-	-
4	SOUTHWEST	NM	NEW MEXICO	-	-
5	MIDWEST	-	-	-	-

20G. Display the number and name of *every* region, the code, name. and population of every state with a population over 4 million people, and the number and name of every supplier in these states. Sort the result by SNO within STCODE within RNO. (Hint: Follow the REGION-STATE-SUPPLIER hierarchical path.)]

```
SELECT R.RNO, R.RNAME, ST.STCODE, ST.STNAME, ST.POPULATION,
S.SNO, S.SNAME
FROM REGION R
        LEFT OUTER JOIN STATE ST
                ON R.RNO = ST.RNO AND ST.POPULATION > 4000000
        LEFT OUTER JOIN SUPPLIER S ON ST.STCODE = S.STCODE
ORDER BY R.RNO, ST.STCODE, S.SNO
```

RNO	RNAME	STCODE	STNAME	POPULATION	SNO	SNAME
1	NORTHEAST	MA	MASSACHUSETTS	6450000	S1	SUPPLIER1
1	NORTHEAST	MA	MASSACHUSETTS	6450000	S2	SUPPLIER2
2	NORTHWEST	WA	WASHINGTON	6468000	S6	SUPPLIER6
3	SOUTHEAST	FL	FLORIDA	18251000	S4	SUPPLIER4
3	SOUTHEAST	GE	GEORGIA	9545000	S5	SUPPLIER5
4	SOUTHWEST	AZ	ARIZONA	6339000	-	-
5	MIDWEST	-	-	-	-	-

 Copyright Pending 2020

20H Work backwards. Describe the join-sequence for the following FROM-clause. Then, transform this sequence into pseudo-code.

```
FROM REGION R
        LEFT OUTER JOIN STATE ST
                LEFT OUTER JOIN CUSTOMER C
                        LEFT OUTER JOIN PUR_ORDER PO
                                LEFT OUTER JOIN LINEITEM LI
                                ON PO.PONO = LI.PONO            ←1
                        ON C.CNO = PO.CNO                       ←2
                ON ST.STCODE = C.STCODE                         ←3
        ON R.RNO = ST.RNO                                       ←4
```

The join-sequence follows the ON-clauses from top to bottom:

1. ON PO.PONO = LI.PONO → Join PUR_ORDER and LINEITEM
2. ON C.CNO = PO.CNO → Join CUSTOMER and PUR_ORDER
3. ON ST.STCODE = C.STCODE → Join STATE and CUSTOMER
4. ON R.RNO = ST.RNO → Join REGION and STATE

```
REGION LOJ (STATE LOJ (CUSTOMER LOJ (PUR_ORDER LOJ LINEITEM)))
   4          3              2                  1
```

 Copyright Pending 2020

20Ia. Transform the following pseudo-code into a FROM-clause.

REGION LOJ ((STATE LOJ CUSTOMER) LOJ PUR_ORDER)

Assign sequence-numbers.

```
REGION LOJ ((STATE LOJ CUSTOMER) LOJ PUR_ORDER)
             3                1                    2
```

Corresponding sequence of ON-clauses

```
ON ST.STCODE = C.STCODE
ON C.CNO = PO.CNO
ON R.RNO = ST.RNO
```

LEFT OUTER JOIN for first ON-clause.

```
(STATE ST LEFT OUTER JOIN CUSTOMER C
    ON ST.STCODE = C.STCODE)
```

LEFT OUTER JOIN for second ON-clause.

```
((STATE ST LEFT OUTER JOIN CUSTOMER C
    ON ST.STCODE = C.STCODE)
LEFT OUTER JOIN PUR_ORDER PO
    ON C.CNO = PO.CNO)
```

LEFT OUTER JOIN for third ON-clause, with FROM-clause

```
FROM
REGION R LEFT OUTER JOIN
    ((STATE ST LEFT OUTER JOIN CUSTOMER C
        ON ST.STCODE = C.STCODE)
    LEFT OUTER JOIN PUR_ORDER PO
        ON C.CNO = PO.CNO)
ON R.RNO = ST.RNO
```

 Copyright Pending 2020

20Ib. Transform the following pseudo-code into a FROM-clause.

(REGION LOJ STATE) LOJ (CUSTOMER LOJ PUR_ORDER)

Assign sequence-numbers.

```
(REGION LOJ STATE) LOJ (CUSTOMER LOJ PUR_ORDER)
                1              3                       2
```

Corresponding sequence of ON-clauses

```
ON R.RNO = ST.RNO
ON C.CNO = PO.CNO
ON ST.STCODE = C.STCODE
```

LEFT OUTER JOIN for first ON-clause.

```
(REGION R LEFT OUTER JOIN STATE ST
 ON R.RNO = ST.RNO)
```

LEFT OUTER JOIN for second ON-clause.

```
(CUSTOMER C LEFT OUTER JOIN PUR_ORDER PO
 ON C.CNO = PO.CNO)
```

LEFT OUTER JOIN for third ON-clause, with FROM-clause

```
FROM
   (REGION R LEFT OUTER JOIN STATE ST
    ON R.RNO = ST.RNO)
LEFT OUTER JOIN
   (CUSTOMER C LEFT OUTER JOIN PUR_ORDER PO
    ON C.CNO = PO.CNO)
ON ST.STCODE = C.STCODE
```

 Copyright Pending 2020

Summary Exercise (Chapter 20)

20J. Optional Exercise: Modify the SELECT statement for Sample Query 20.5. Change the FROM-clause. Specify an INNER JOIN operation to join the LINEITEM and PARTSUPP tables as shown below.

```
SELECT R.RNO, R.RNAME, ST.STCODE, C.CNO,
          PO.PONO, LI.PNO, LI.SNO, LI.LIPRICE, PS.PSPRICE
FROM REGION R
       LEFT OUTER JOIN STATE ST          ON R.RNO = ST.RNO
       LEFT OUTER JOIN CUSTOMER C     ON ST.STCODE = C.STCODE
       LEFT OUTER JOIN PUR_ORDER PO ON C.CNO = PO.CNO
       LEFT OUTER JOIN LINEITEM LI      ON PO.PONO = LI.PONO
       INNER JOIN PARTSUPP PS
                    ON LI.PNO = PS.PNO AND LI.SNO = PS.SNO
ORDER BY R.RNO, ST.STCODE, C.CNO, PO.PONO, LI.PNO
```

Execute this statement and examine the result. All LEFT OUTER JOIN operations *appear* to behave like INNER JOIN operations. Why did this happen?

This happened because the four LEFT OUTER JOIN operations produce a join-result with null values in the PNO and SNO columns for the five non-matching rows. (See result table for Sample Query 20.4) These five rows with null values will not be preserved by the INNER JOIN operation because no value can match on a null value. The next chapter will address this topic.

Chapter 20.5 - Mixing Inner-Joins and Left Outer-Joins

20K. Display the number and name of every region that contains at least one state, the code and name of *every* state (including states without any suppliers), and the number and name of *every* supplier in each state. Display the columns in the following left-to-right sequence: RNO, RNAME, STCODE, STNAME, SNO, and SNAME. Sort the result by SNO within STCODE within RNO. (Hint: Follow the REGION-STATE-SUPPLIER hierarchy.)

Pseudo-code: (REGION IJ STATE) LOJ SUPPLIER

```
SELECT R.RNO, R.RNAME, ST.STCODE, ST.STNAME,
         S.SNO, S.SNAME
FROM   REGION R
       INNER JOIN STATE ST          ON R.RNO = ST.RNO
       LEFT OUTER JOIN SUPPLIER S  ON ST.STCODE = S.STCODE
ORDER BY R.RNO, ST.STCODE, S.SNO
```

RNO	RNAME	STCODE	STNAME	SNO	SNAME
1	NORTHEAST	CT	CONNECTICUT	S3	SUPPLIER3
1	NORTHEAST	MA	MASSACHUSETTS	S1	SUPPLIER1
1	NORTHEAST	MA	MASSACHUSETTS	S2	SUPPLIER2
2	NORTHWEST	OR	OREGON	S7	SUPPLIER7
2	NORTHWEST	OR	OREGON	S8	SUPPLIER8
2	NORTHWEST	WA	WASHINGTON	S6	SUPPLIER6
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4
3	SOUTHEAST	GE	GEORGIA	S5	SUPPLIER5
4	SOUTHWEST	AZ	ARIZONA	-	-
4	SOUTHWEST	NM	NEW MEXICO	-	-

 Copyright Pending 2020

20L. Display the number and name of every region, the code and name of those states that have at least one supplier, and the number and name of these suppliers. Display the columns in the following left-to-right sequence: RNO, RNAME, STCODE, STNAME, SNO, and SNAME. Sort the result by SNO within STCODE within RNO. (Hint: Follow the REGION-STATE-SUPPLIER hierarchy.)

Pseudo-code: REGION LOJ (STATE IJ SUPPLIER)

```
SELECT R.RNO, R.RNAME, ST.STCODE, ST.STNAME,
          S.SNO, S.SNAME
FROM REGION R
  LEFT OUTER JOIN
       STATE ST INNER JOIN SUPPLIER S ON ST.STCODE = S.STCODE
  ON R.RNO = ST.RNO
ORDER BY R.RNO, ST.STCODE, S.SNO
```

RNO	RNAME	STCODE	STNAME	SNO	SNAME
1	NORTHEAST	CT	CONNECTICUT	S3	SUPPLIER3
1	NORTHEAST	MA	MASSACHUSETTS	S1	SUPPLIER1
1	NORTHEAST	MA	MASSACHUSETTS	S2	SUPPLIER2
2	NORTHWEST	OR	OREGON	S7	SUPPLIER7
2	NORTHWEST	OR	OREGON	S8	SUPPLIER8
2	NORTHWEST	WA	WASHINGTON	S6	SUPPLIER6
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4
3	SOUTHEAST	GE	GEORGIA	S5	SUPPLIER5
4	SOUTHWEST	-	-	-	-
5	MIDWEST	-	-	-	-

Two regions have null STCODE and STNAME values.

Region 4 (SOUTHWEST) has two states (AZ and NM), but these states do not have any suppliers.

Region 5 (MIDWEST) has no states

 Copyright Pending 2020

20M. Display the number and name of every region with at least one state, the code and name of every state with at least one supplier, the number and name of every supplier (including suppliers who do not sell any parts), and the part numbers of parts that can be purchased from these suppliers. Display the columns in the following left-to-right sequence: RNO, RNAME, STCODE, STNAME, SNO, SNAME, and PNO. Sort the result by PNO within SNO within STCODE within RNO. (Hint: Traverse REGION-STATE-SUPPLIER-PARTSUPP hierarchy.)

Join-sequence: ((REGION IJ STATE) IJ SUPPLIER) LOJ PARTSUPP

```
SELECT R.RNO, R.RNAME, ST.STCODE, ST.STNAME,
       S.SNO, S.SNAME, PS.PNO
FROM REGION R
     INNER JOIN STATE ST          ON R.RNO = ST.RNO
     INNER JOIN SUPPLIER S        ON ST.STCODE = S.STCODE
     LEFT OUTER JOIN PARTSUPP PS ON S.SNO = PS.SNO
ORDER BY R.RNO, S.STCODE, S.SNO, PS.PNO
```

RNO	RNAME	STCODE	STNAME	SNO	SNAME	PNO
1	NORTHEAST	CT	CONNECTICUT	S3	SUPPLIER3	P3
1	NORTHEAST	MA	MASSACHUSETTS	S1	SUPPLIER1	P5
1	NORTHEAST	MA	MASSACHUSETTS	S2	SUPPLIER2	P1
1	NORTHEAST	MA	MASSACHUSETTS	S2	SUPPLIER2	P5
1	NORTHEAST	MA	MASSACHUSETTS	S2	SUPPLIER2	P7
2	NORTHWEST	OR	OREGON	S7	SUPPLIER7	-
2	NORTHWEST	OR	OREGON	S8	SUPPLIER8	P6
2	NORTHWEST	OR	OREGON	S8	SUPPLIER8	P8
2	NORTHWEST	WA	WASHINGTON	S6	SUPPLIER6	P6
2	NORTHWEST	WA	WASHINGTON	S6	SUPPLIER6	P7
2	NORTHWEST	WA	WASHINGTON	S6	SUPPLIER6	P8
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P1
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P3
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P4
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P5
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P6
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P7
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P8
3	SOUTHEAST	GE	GEORGIA	S5	SUPPLIER5	P7

20N. Display the number and name of every region, the code and name of every state with at least one supplier, the number and name of every supplier that sells at least one part, and the part number and PSPRICE of these parts. Display the columns in the following left-to-right sequence: RNO, RNAME, STCODE, STNAME, SNO, SNAME, PNO, and PSPRICE. Sort the result by PNO within SNO within STCODE within RNO.

Join-sequence is:

REGION LOJ ((STATE IJ SUPPLIER) IJ PARTSUPP)

```
SELECT R.RNO, R.RNAME, ST.STCODE, ST.STNAME,
          S.SNO, S.SNAME, PS.PNO, PS.PSPRICE
FROM REGION R
   LEFT OUTER JOIN
   STATE ST INNER JOIN SUPPLIER S    ON ST.STCODE = S.STCODE
                INNER JOIN PARTSUPP PS ON S.SNO = PS.SNO
   ON R.RNO = ST.RNO
ORDER BY R.RNO, ST.STCODE, S.SNO, PS.PNO
```

RNO	RNAME	STCODE	STNAME	SNO	SNAME	PNO	PSPRICE
1	NORTHEAST	CT	CONNECTICUT	S3	SUPPLIER3	P3	12.00
1	NORTHEAST	MA	MASSACHUSETTS	S1	SUPPLIER1	P5	10.00
1	NORTHEAST	MA	MASSACHUSETTS	S2	SUPPLIER2	P7	2.00
1	NORTHEAST	MA	MASSACHUSETTS	S2	SUPPLIER2	P5	10.00
1	NORTHEAST	MA	MASSACHUSETTS	S2	SUPPLIER2	P1	10.50
2	NORTHWEST	OR	OREGON	S8	SUPPLIER8	P8	3.00
2	NORTHWEST	OR	OREGON	S8	SUPPLIER8	P6	4.00
2	NORTHWEST	WA	WASHINGTON	S6	SUPPLIER6	P7	3.50
2	NORTHWEST	WA	WASHINGTON	S6	SUPPLIER6	P6	4.00
2	NORTHWEST	WA	WASHINGTON	S6	SUPPLIER6	P8	4.00
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P7	3.00
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P6	4.00
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P8	5.00
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P1	11.00
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P5	11.00
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P4	12.00
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P3	12.50
3	SOUTHEAST	GE	GEORGIA	S5	SUPPLIER5	P7	3.50
4	SOUTHWEST	-	-	-	-	-	-
5	MIDWEST	-	-	-	-	-	-

 Copyright Pending 2020

20O. Display the number and name of every region, the code and name of every state, the number and name of every supplier that sells at least one part, and the part number and PSPRICE value of these parts. Display the columns in the following left-to-right sequence: RNO, RNAME, STCODE, STNAME, SNO, SNAME, PNO, and PSPRICE. Sort the result by PNO within SNO within STCODE within RNO.

Join-sequence is: (REGION LOJ STATE) LOJ (SUPPLIER IJ PARTSUPP)

```
SELECT R.RNO, R.RNAME, ST.STCODE, ST.STNAME,
                    S.SNO, S.SNAME, PS.PNO, PS.PSPRICE
FROM REGION R
        LEFT OUTER JOIN STATE ST ON R.RNO = ST.RNO
        LEFT OUTER JOIN
           SUPPLIER S INNER JOIN PARTSUPP PS ON S.SNO=PS.SNO
        ON ST.STCODE = S.STCODE
ORDER BY R.RNO, ST.STCODE, S.SNO, PS.PNO
```

RNO	RNAME	STCODE	STNAME	SNO	SNAME	PNO	PSPRICE
1	NORTHEAST	CT	CONNECTICUT	S3	SUPPLIER3	P3	12.00
1	NORTHEAST	MA	MASSACHUSETTS	S1	SUPPLIER1	P5	10.00
1	NORTHEAST	MA	MASSACHUSETTS	S2	SUPPLIER2	P1	10.50
1	NORTHEAST	MA	MASSACHUSETTS	S2	SUPPLIER2	P5	10.00
1	NORTHEAST	MA	MASSACHUSETTS	S2	SUPPLIER2	P7	2.00
2	NORTHWEST	OR	OREGON	S8	SUPPLIER8	P6	4.00
2	NORTHWEST	OR	OREGON	S8	SUPPLIER8	P8	3.00
2	NORTHWEST	WA	WASHINGTON	S6	SUPPLIER6	P6	4.00
2	NORTHWEST	WA	WASHINGTON	S6	SUPPLIER6	P7	3.50
2	NORTHWEST	WA	WASHINGTON	S6	SUPPLIER6	P8	4.00
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P1	11.00
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P3	12.50
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P4	12.00
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P5	11.00
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P6	4.00
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P7	3.00
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P8	5.00
3	SOUTHEAST	GE	GEORGIA	S5	SUPPLIER5	P7	3.50
4	SOUTHWEST	AZ	ARIZONA	-	-	-	-
4	SOUTHWEST	NM	NEW MEXICO	-	-	-	-
5	MIDWEST	-	-	-	-	-	-

20P. Display the following information about regions, states, suppliers, and the parts that each supplier is allowed to sell, and the parts the supplier has already sold.

- Display the number and name of all regions.
- Display the code and name for all states.
- Display the supplier numbers and names for those suppliers who are allowed to sell at least one part.
- Display the part numbers of these parts.
- Display the LIPRICE value of those parts these suppliers have already sold.

Join-sequence is:
((REGION LOJ STATE) LOJ (SUPPLIER IJ PARTSUPP)) LOJ LINEITEM

```
SELECT R.RNO, R.RNAME, ST.STCODE, ST.STNAME,  S.SNO, S.SNAME,
          PS.PNO, LI.LIPRICE
FROM REGION R
       LEFT OUTER JOIN STATE ST ON R.RNO = ST.RNO
       LEFT OUTER JOIN
             SUPPLIER S INNER JOIN PARTSUPP PS ON S.SNO = PS.SNO
        ON ST.STCODE = S.STCODE
       LEFT OUTER JOIN LINEITEM LI
       ON PS.PNO = LI.PNO AND PS.SNO = LI.SNO
ORDER BY R.RNO, ST.STCODE, S.SNO, PS.PNO, LI.PNO
```

Show some of result table's 66 rows.

Observe that Supplier S6 is allowed to sell Part P7, but has not yet sold this part.

RNO	RNAME	STNAME	SNO	SNAME	PNO	LIPRICE	
1	NORTHEAST	CONNECTICUT	S3	SUPPLIER3	P3	12.00	
1	NORTHEAST	CONNECTICUT	S3	SUPPLIER3	P3	13.00	
1	NORTHEAST	CONNECTICUT	S3	SUPPLIER3	P3	13.00	
. . . . .							
. . . . .							
2	NORTHWEST	WASHINGTON	S6	SUPPLIER6	P6	5.00	
2	NORTHWEST	WASHINGTON	S6	SUPPLIER6	P7	-	←
2	NORTHWEST	WASHINGTON	S6	SUPPLIER6	P8	5.00	
2	NORTHWEST	WASHINGTON	S6	SUPPLIER6	P8	5.00	
. . . . .							
. . . . .							
3	SOUTHEAST	GEORGIA	S5	SUPPLIER5	P7	4.50	
3	SOUTHEAST	GEORGIA	S5	SUPPLIER5	P7	4.50	
4	SOUTHWEST	ARIZONA	-	-	-	-	
4	SOUTHWEST	NEW MEXICO	-	-	-	-	
5	MIDWEST	-	-	-	-	-	

 Copyright Pending 2020

20Q. Display the following information about regions, states, suppliers, and parts.

- Display the number and name of any region that has at least one state.
- Display the code and name of any state that has at least one supplier.
- Display the number and name of all suppliers, including those suppliers who are not yet allowed to sell any parts.
- Display the part number and LIPRICE of each part the supplier has sold.

Join-sequence is:
((REGION IJ STATE) IJ SUPPLIER) LOJ (PARTSUPP IJ LINEITEM)

```
SELECT R.RNO, R.RNAME, ST.STCODE, ST.STNAME, S.SNO, S.SNAME,
       LI.PNO, LI.LIPRICE
FROM
((REGION R INNER JOIN STATE ST   ON R.RNO = ST.RNO)
            INNER JOIN SUPPLIER S ON ST.STCODE = S.STCODE)
  LEFT OUTER JOIN
      (PARTSUPP PS INNER JOIN LINEITEM LI
       ON PS.PNO = LI.PNO AND PS.SNO = LI.SNO)
  ON S.SNO = PS.SNO
ORDER BY R.RNO, ST.STCODE, S.SNO, LI.PNO
```

Show some of result table's 63 rows.
Observe Supplier S7 is not allowed to sell any parts.
Also, every supplier has sold at least one part that the supplier is allowed to sell.

RNO	RNAME	STCODE	STNAME	SNO	SNAME	PNO	LIPRICE
1	NORTHEAST	CT	CONNECTICUT	S3	SUPPLIER3	P3	12.00
1	NORTHEAST	CT	CONNECTICUT	S3	SUPPLIER3	P3	13.00
1	NORTHEAST	CT	CONNECTICUT	S3	SUPPLIER3	P3	13.00
1	NORTHEAST	MA	MASSACHUSETTS	S1	SUPPLIER1	P5	11.00
.	.	.	.	.	.	.	.
1	NORTHEAST	MA	MASSACHUSETTS	S2	SUPPLIER2	P7	3.00
1	NORTHEAST	MA	MASSACHUSETTS	S2	SUPPLIER2	P7	3.00
2	NORTHWEST	OR	OREGON	S7	SUPPLIER7	-	-
2	NORTHWEST	OR	OREGON	S8	SUPPLIER8	P6	5.00
2	NORTHWEST	OR	OREGON	S8	SUPPLIER8	P8	4.00
2	NORTHWEST	OR	OREGON	S8	SUPPLIER8	P8	4.00
2	NORTHWEST	OR	OREGON	S8	SUPPLIER8	P8	4.00
2	NORTHWEST	OR	OREGON	S8	SUPPLIER8	P8	4.50
2	NORTHWEST	WA	WASHINGTON	S6	SUPPLIER6	P6	5.00
2	NORTHWEST	WA	WASHINGTON	S6	SUPPLIER6	P8	5.00
2	NORTHWEST	WA	WASHINGTON	S6	SUPPLIER6	P8	5.00
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P1	12.00
.	.	.	.	.	.	.	.
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P8	6.00
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4	P8	6.00
3	SOUTHEAST	GE	GEORGIA	S5	SUPPLIER5	P7	4.50
3	SOUTHEAST	GE	GEORGIA	S5	SUPPLIER5	P7	4.50

 Copyright Pending 2020

Summary Exercises (Chapter 20.5)

Exercises 20R1, 20R2, 20S1, and 20S2 are optional exercises.

20R1. Work backwards. Transform the following FROM-clauses into equivalent pseudo-code.

```
FROM REGION R
  LEFT OUTER JOIN STATE ST ON R.RNO = ST.RNO
  LEFT OUTER JOIN
        CUSTOMER C INNER JOIN PUR_ORDER PO
        ON C.CNO=PO.CNO
  ON ST.STCODE = C.STCODE
```

1. ON R.RNO = ST.RNO → Join REGION and STATE
2. ON C.CNO=PO.CNO → Join CUSTOMER and PUR_ORDER
3. ON ST.STCODE = C.STCODE → Join STATE and CUSTOMER

```
(REGION LOJ STATE) LOJ (CUSTOMER IJ PUR_ORDER)
          1              3              2
```

20R2. Work backwards. Transform the following FROM-clauses into equivalent pseudo-code.

```
FROM REGION R
  LEFT OUTER JOIN STATE ST ON R.RNO = ST.RNO
  LEFT OUTER JOIN
        CUSTOMER C INNER JOIN PUR_ORDER PO
        ON C.CNO = PO.CNO
  ON ST.STCODE = C.STCODE
  LEFT OUTER JOIN LINEITEM LI ON PO.PONO = LI.PONO
```

1. ON R.RNO = ST.RNO → Join REGION and STATE
2. ON C.CNO = PO.CNO → Join CUSTOMER and PUR_ORDER
3. ON ST.STCODE = C.STCODE → Join STATE and CUSTOMER
4. ON PO.PONO = LI.PONO → Join PUR_ORDER and LINEITEM

```
(REGION LOJ STATE) LOJ (CUSTOMER IJ PUR_ORDER) LOJ LINEITEM
          1              3              2              4
```

 Copyright Pending 2020

20S1. Convert the following pseudo-code expression into a FROM-clause.

```
R LOJ (ST IJ (C LOJ PO))
```

```
FROM REGION R
        LEFT OUTER JOIN
          (STATE ST INNER JOIN
              (CUSTOMER C LEFT OUTER JOIN PUR_ORDER PO
                ON C.CNO=PO.CNO )
            ON ST.STCODE = C.STCODE)
        ON R.RNO = ST.RNO
```

20S2. Convert the following pseudo-code expression into a FROM-clause.

```
(R IJ ST) IJ ((C LOJ PO) LOJ LI))
```

```
FROM (REGION R INNER JOIN STATE ST ON R.RNO = ST.RNO)
        INNER JOIN
          ((CUSTOMER C LEFT OUTER JOIN PUR_ORDER PO
                                        ON C.CNO=PO.CNO)
            LEFT OUTER JOIN LINEITEM LI ON PO.PONO = LI.PONO)
        ON C.STCODE = ST.STCODE
```

Some of the following exercises will have multiple solutions.

Suggestion: Represent query objective in pseudo-code and then transform pseudo-code to a FROM-clause.

20T. Display the number and name of every region, the code and name of every state with at least one supplier, and the number and name of every supplier in these states. Display the columns in the following left-to-right sequence: RNO, RNAME, STCODE, STNAME, SNO, and SNAME. Sort the result by SNO within STCODE within RNO.

Pseudo-code: REGION LOJ (STATE IJ SUPPLIER)

```
SELECT  R.RNO, R.RNAME, ST.STCODE, ST.STNAME,
        S.SNO, S.SNAME
FROM REGION R LEFT OUTER JOIN
        (STATE ST INNER JOIN SUPPLIER S
                ON ST.STCODE = S.STCODE)
    ON R.RNO = ST.RNO
ORDER BY R.RNO, ST.STCODE, S.SNO
```

RNO	RNAME	STCODE	STNAME	SNO	SNAME
1	NORTHEAST	CT	CONNECTICUT	S3	SUPPLIER3
1	NORTHEAST	MA	MASSACHUSETTS	S1	SUPPLIER1
1	NORTHEAST	MA	MASSACHUSETTS	S2	SUPPLIER2
2	NORTHWEST	OR	OREGON	S7	SUPPLIER7
2	NORTHWEST	OR	OREGON	S8	SUPPLIER8
2	NORTHWEST	WA	WASHINGTON	S6	SUPPLIER6
3	SOUTHEAST	FL	FLORIDA	S4	SUPPLIER4
3	SOUTHEAST	GE	GEORGIA	S5	SUPPLIER5
4	SOUTHWEST	-	-	-	-
5	MIDWEST	-	-	-	-

 Copyright Pending 2020

20U. Display the part number every part, the supplier number of every supplier who has sold this part, and the purchase-order number and line-item price for each sale of the part by the supplier. Display the columns in the following left-to-right sequence: PNO, SNO, PONO, and LIPRICE. Sort the result by PONO, SNO within PNO. (Hint: Follow the PART-PARTSUPP-LINEITEM hierarchy.)

Pseudo-code: PART LOJ (PARTSUPP IJ LINEITEM)

```
SELECT P.PNO, PS.SNO, LI.PONO, LI.LIPRICE
FROM   PART P LEFT OUTER JOIN
                (PARTSUPP PS INNER JOIN LINEITEM LI
                   ON PS.PNO = LI.PNO AND PS.SNO = LI.SNO)
           ON P.PNO = PS.PNO
ORDER BY P.PNO, PS.SNO, LI.PONO
```

Display some of result table's 63 rows.

Observe that no supplier has sold Part P2.

PNO	SNO	PONO	LIPRICE
P1	S2	11101	11.50
P1	S2	11109	11.50
P1	S2	11122	11.50
P1	S2	11148	11.50
P1	S2	11154	11.50
P1	S2	11156	11.50
P1	S2	11158	11.50
P1	S2	11160	12.50
P1	S4	11111	12.00
P1	S4	11133	12.00
P2	-	-	-
P3	S3	11101	12.00
P3	S3	11102	13.00
P3	S3	11122	13.00
.	.	.	.
.	.	.	.
P8	S4	11109	6.00
P8	S4	11110	6.00
P8	S4	11148	6.00
P8	S6	11144	5.00
P8	S6	11146	5.00
P8	S8	11149	4.00
P8	S8	11152	4.00
P8	S8	11153	4.00
P8	S8	11155	4.50

20V. Display the number and name of any region that contains at least one state, the code and name of every state (including states without customers), the number and name of every customer in each state (including customers without purchase-orders), and the date of every purchase-order completed by these customers. Display the columns in the following left-to-right sequence: RNO, RNAME, STCODE, STNAME, CNO, CNAME, and PODATE. Sort the result by PODATE within CNO within STCOE within RNO.

((REGION IJ STATE) LOJ CUSTOMER) LOJ PUR_ORDER

```
SELECT R.RNO, R.RNAME, ST.STCODE, ST.STNAME,
          C.CNO, C.CNAME, PO.PODATE
FROM ((REGION R INNER JOIN STATE ST  ON R.RNO = ST.RNO)
          LEFT OUTER JOIN CUSTOMER C
                ON ST.STCODE = C.STCODE)
          LEFT OUTER JOIN PUR_ORDER PO ON C.CNO = PO.CNO
ORDER BY R.RNO, ST.STCODE, C.CNO, PO.PODATE
```

Display some of result table's 34 rows.

RNO	RNAME	STCODE	STNAME	CNO	CNAME	PODATE
1	NORTHEAST	CT	CONNECTICUT	-	-	-
1	NORTHEAST	MA	MASSACHUSETTS	100	PYTHAGORAS	1
.	.	.	.	.	.	.
.	.	.	.	.	.	.
4	SOUTHWEST	AZ	ARIZONA	880	TURING	3
4	SOUTHWEST	AZ	ARIZONA	880	TURING	4
4	SOUTHWEST	AZ	ARIZONA	880	TURING	10
4	SOUTHWEST	AZ	ARIZONA	880	TURING	10
4	SOUTHWEST	AZ	ARIZONA	890	MANDELBROT	-
4	SOUTHWEST	NM	NEW MEXICO	780	CHURCH	-
4	SOUTHWEST	NM	NEW MEXICO	800	VON NEUMANN	3

Alternative Solution to Exercise 20V.

(REGION IJ STATE) LOJ (CUSTOMER LOJ PUR_ORDER)

```
SELECT R.RNO, R.RNAME, ST.STCODE, ST.STNAME,
          C.CNO, C.CNAME, PO.PODATE

FROM (REGION R INNER JOIN STATE ST  ON R.RNO = ST.RNO)
         LEFT OUTER JOIN
        (CUSTOMER C LEFT OUTER JOIN PUR_ORDER PO ON C.CNO = PO.CNO)
        ON ST.STCODE = C.STCODE

ORDER BY R.RNO, ST.STCODE, C.CNO, PO.PODATE
```

20W. Reference the STATE-CUSTOMER-PUR_ORDER-LINEITEM hierarchy.

- Display the RNO value of any region that has at least one state.
- Display the STCODE value of any state that has at least one customer.
- For each such state, display the CNO and CNAME values of its customers.
- For each customer with at least one purchase-order, display the customer's purchase-order numbers.
- For each purchase-order, display its LINE and corresponding PNO values even if the purchase order does not have any line items.

[Note: There is no need to reference the REGION table.]

Pseudo-code:

((STATE IJ CUSTOMER) IJ PUR_ORDER) **LOJ** LINEITEM

```
SELECT ST.RNO, ST.STCODE, C.CNO, C.CNAME,
          PO.PONO, LI.LINE, LI.PNO
FROM ((STATE ST INNER JOIN CUSTOMER C
                    ON ST.STCODE = C.STCODE)
          INNER JOIN PUR_ORDER PO ON C.CNO = PO.CNO)
          LEFT OUTER JOIN LINEITEM LI ON PO.PONO = LI.PONO
ORDER BY ST.RNO, ST.STCODE, C.CNO, PO.PONO, LI.LINE
```

Show some of result table's 63 rows.

RNO	STCODE	CNO	CNAME	PONO	LINE	PNO
1	MA	100	PYTHAGORAS	11101	1	P1
1	MA	100	PYTHAGORAS	11101	2	P3
1	MA	100	PYTHAGORAS	11102	1	P3
1	MA	100	PYTHAGORAS	11102	2	P4
1	MA	110	EUCLID	11108	1	P5
1	MA	110	EUCLID	11108	2	P6
1	MA	110	EUCLID	11109	1	P1
1	MA	110	EUCLID	11109	2	P7
1	MA	110	EUCLID	11109	3	P8
.	.	.	.	.	.	.
4	AZ	880	TURING	11159	1	P6
4	AZ	880	TURING	11159	2	P7
4	AZ	880	TURING	11160	1	P1
4	AZ	880	TURING	11160	2	P7
4	AZ	880	TURING	11170	1	P3
4	AZ	880	TURING	11170	2	P4
4	AZ	880	TURING	11198	-	-
4	NM	800	VON NEUMANN	11158	1	P1
4	NM	800	VON NEUMANN	11158	2	P3

Alternative Solution to Exercise 20W.

Pseudo-code:
    (STATE IJ (CUSTOMER IJ PUR_ORDER)) **LOJ** LINEITEM

```
SELECT ST.RNO, ST.STCODE, C.CNO, C.CNAME,
           PO.PONO, LI.LINE, LI.PNO
FROM
  (STATE ST INNER JOIN
    (CUSTOMER C  INNER JOIN PUR_ORDER PO ON C.CNO = PO.CNO)
   ON ST.STCODE = C.STCODE)
  LEFT OUTER JOIN LINEITEM LI ON PO.PONO = LI.PONO
ORDER BY ST.RNO, ST.STCODE, C.CNO, PO.PONO, LI.LINE
```

20X. Reference the STATE-CUSTOMER-PUR_ORDER-LINEITEM hierarchy.

- Display the STCODE and RNO values of any state that has at least one customer.
- For each such state, display the CNO and CNAME values of its customers, even if those customers do not have any purchase-orders.
- For each customer with at least one purchase-order that has at least one line-item, display the customer's purchase-order numbers.
- For those purchase-orders, display each line-item's LINE and PNO values.

Pseudo-code:
STATE IJ (CUSTOMER LOJ (PUR-ORDER **IJ** LINEITEM))

```
SELECT ST.RNO, ST.STCODE, C.CNO, C.CNAME,
         PO.PONO, LI.LINE, LI.PNO
FROM   STATE ST  INNER JOIN
          (CUSTOMER C LEFT OUTER JOIN
               (PUR_ORDER PO INNER JOIN LINEITEM LI
                                 ON PO.PONO = LI.PONO)
                             ON C.CNO = PO.CNO)
        ON ST.STCODE = C.STCODE
ORDER BY ST.RNO, ST.STCODE, C.CNO, PO.PONO, LI.LINE
```

Display some of this result table's 64 rows.

RNO	STCODE	CNO	CNAME	PONO	LINE	PNO	
1	MA	100	PYTHAGORAS	11101	1	P1	
1	MA	100	PYTHAGORAS	11101	2	P3	
1	MA	100	PYTHAGORAS	11102	1	P3	
1	MA	100	PYTHAGORAS	11102	2	P4	
.	.	.	.	.	.	.	
.	.	.	.	.	.	.	
.	.	.	.	.	.	.	
4	AZ	880	TURING	11170	2	P4	
4	AZ	890	MANDELBROT	-	-	-	← C-No-PO
4	NM	780	CHURCH	-	-	-	← C-No-P
4	NM	800	VON NEUMANN	11158	1	P1	
4	NM	800	VON NEUMANN	11158	2	P3	

 Copyright Pending 2020

Alternative Solution to Exercise 20X.

Pseudo-code:
(STATE IJ CUSTOMER) LOJ (PUR_ORDER **IJ** LINEITEM))

```
SELECT ST.RNO, ST.STCODE, C.CNO, C.CNAME,
            PO.PONO, LI.LINE, LI.PNO
FROM
  (STATE ST  INNER JOIN CUSTOMER C ON ST.STCODE = C.STCODE)
  LEFT OUTER JOIN
  (PUR_ORDER PO INNER JOIN LINEITEM LI ON PO.PONO = LI.PONO)
  ON C.CNO = PO.CNO
ORDER BY ST.RNO, ST.STCODE, C.CNO, PO.PONO, LI.LINE
```

PART V

Set Operations
&
CASE Expressions

 Copyright Pending 2020

Chapter-21 - Set Operations

Exercises 21A-21E reference the following PROJ1PARTS and PROJ2PARTS tables. Recall that some parts (e.g., P4 and P5) can be used in both projects.

PROJ1PARTS

PNO	PNAME	PCOLOR	QTY
P1	PART1	RED	16
P2	PART2	BLUE	16
P4	PART4	YELLOW	17
P5	PART5	RED	15

PROJ2PARTS

PNO	PNAME	PWT
P3	PART3	20
P4	PART4	10
P5	PART5	20
P6	PART6	12

21A. Display the part number and name of all parts used by either Project1 or Project2.

```
SELECT PNO, PNAME FROM PROJ1PARTS
UNION
SELECT PNO, PNAME FROM PROJ2PARTS
```

PNO	PNAME
P1	PART1
P2	PART2
P3	PART3
P4	PART4
P5	PART5
P6	PART6

21B. Display the part number and name of any part that is used by both Project1 and Project2.

```
SELECT PNO, PNAME FROM PROJ1PARTS
INTERSECT
SELECT PNO, PNAME FROM PROJ2PARTS
```

PNO	PNAME
P4	PART4
P5	PART5

 Copyright Pending 2020

21C. (i) Display the part number and name of any part that is used by Project1 but not used by Project2.

```
SELECT PNO, PNAME FROM PROJ1PARTS
EXCEPT
SELECT PNO, PNAME FROM PROJ2PARTS
```

PNO	PNAME
P1	PART1
P2	PART2

(ii) Display the part number and name of any part that is used by Project2 but not used by Project1.

```
SELECT PNO, PNAME FROM PROJ2PARTS
EXCEPT
SELECT PNO, PNAME FROM PROJ1PARTS
```

PNO	PNAME
P3	PART3
P6	PART6

 Copyright Pending 2020

21D. The following statement produces a potentially confusing result. Why?

```
SELECT PNO, PNAME, QTY
FROM PROJ1PARTS
UNION
SELECT PNO, PNAME, PWT
FROM PROJ2PARTS
ORDER BY 1
```

The result looks like:

PNO	PNAME	QTY
P1	PART1	16
P2	PART2	16
P3	PART3	20
P4	PART4	10
P4	PART4	17
P5	PART5	15
P5	PART5	20
P6	PART6	12

The third column is problematic because it "mixes apples and oranges." Although its header is QTY, the column contains both part-quantity (QTY) values and part-weight (PWT) values. The system allows this behavior because both of the QTY and PWT columns have the same data type (INTEGER).

Also, given the current values in the QTY and PWT columns, no QTY value matches any PWT value. But this could occur in the future. Assume that:

PROJ1PARTS contained a row that looked like: P9 PART9 88
and
PROJ2PARTS contained a row that looked like: P9 PART9 88

In this circumstance, the UNION operation would eliminate one of the duplicate rows, thereby producing an ambiguous result. Therefore, this example should encourage you to attach labels to each row in the result.

21E. Modify the above statement to display a label to distinguish QTY values from PWT values.

Solution1 (Four columns in result table)

```
SELECT PNO, PNAME, 'THE QUANITY IS: ' MYLABEL, QTY
FROM PROJ1PARTS
UNION
SELECT PNO, PNAME,   'THE WEIGHT IS: ' MYLABEL, PWT
FROM PROJ2PARTS
ORDER BY 1
```

Solution2 (Three columns in result table):

For SQL Server:

```
SELECT PNO, PNAME, 'THE QUANITY IS: ' + CAST (QTY AS CHAR(5))
FROM PROJ1PARTS
UNION
SELECT PNO, PNAME, 'THE WEIGHT IS: ' + CAST (PWT AS CHAR(5))
FROM PROJ2PARTS
ORDER BY 1
```

For DB2 and ORACLE:

```
SELECT PNO, PNAME, 'THE QUANITY IS: ' || CAST (QTY AS CHAR(5))
FROM PROJ1PARTS
UNION
SELECT PNO, PNAME, 'THE WEIGHT IS: ' || CAST (PWT AS CHAR(5))
FROM PROJ2PARTS
ORDER BY 1
```

21F. Code an alternative solution to Sample Query 21.3 using a join-operation instead of specifying the INTERSECT operation.

```
SELECT E.ENO, E.ENAME
FROM    EMPLOYEE E, PROJMGR P
WHERE E.ENO = P.ENO
```

Reference the PROJ1PARTS and PROJ3PARTS tables. Recall that Project1 and Project3 can never use the same part.

21G. Display the part number and name of all parts used by either Project1 or Project3.

PNO	PNAME
P1	PART1
P2	PART2
P3	PART3
P4	PART4
P5	PART5
P6	PART6
P7	PART7
P8	PART8

Two solutions:

```
SELECT PNO, PNAME FROM PROJ1PARTS
UNION
SELECT PNO, PNAME FROM PROJ3PARTS
```

This works because PNO values in Project1 and Project3 are disjoint.
This result may be probably be incidentally sorted.

```
SELECT PNO, PNAME FROM PROJ1PARTS
UNION ALL
SELECT PNO, PNAME FROM PROJ3PARTS
```

This result will probably not be incidentally sorted.

 Copyright Pending 2020

21H. Reference the PROJ1PARTS1 table. Produce a result that displays every part number and name, followed by a character-string indicating if the QTY column contains a value that is less than, equal to, or greater than 16. Sort the result by PNO. The result should look like:

```
PNO   PNAME      COMMENTARY
P1    PART1      QTY EQUAL TO 16
P2    PART2      QTY EQUAL TO 16
P4    PART4      QTY GREATER THAN 16
P5    PART5      QTY LESS THAN 16
```

```
SELECT PNO, PNAME, 'QTY LESS THAN 16'  COMMENTARY
FROM   PROJ1PARTS
WHERE QTY < 16
 UNION ALL
SELECT PNO, PNAME, 'QTY EQUAL TO 16'
FROM   PROJ1PARTS
WHERE QTY = 16
 UNION ALL
SELECT PNO, PNAME, 'QTY GREATER THAN 16'
FROM   PROJ1PARTS
WHERE QTY > 16
ORDER BY 1
```

 Copyright Pending 2020

Summary Exercises (Chapter 21)

21I. Reference the EMPLOYEE and PROJMGR tables. Display the employee number and name of any person who works in or manages projects for Department 20.

```
SELECT ENO, ENAME
FROM   EMPLOYEE
WHERE DNO = 20
  UNION
SELECT ENO, PMNAME
FROM   PROJMGR
WHERE DNO = 20
ORDER BY 1
```

```
ENO  ENAME
1000 MOE
3000 CURLY
6000 GEORGE
```

21J. Reference the EMPLOYEE and PROJMGR tables. Modify the previous exercise. Display "EMPLOYEE" or "PROJECT MANAGER" in the third column to indicate that the person is an employee or a project manager. (Two rows will be displayed for any person who is both an employee and a project manager.)

```
SELECT ENO, ENAME, 'EMPLOYEE'
FROM   EMPLOYEE
WHERE DNO = 20
UNION
SELECT ENO, PMNAME, 'PROJECT MANAGER'
FROM   PROJMGR
WHERE DNO = 20
ORDER BY 1
```

```
ENO  ENAME
1000 MOE    EMPLOYEE
1000 MOE    PROJECT MANAGER
3000 CURLY  EMPLOYEE
6000 GEORGE EMPLOYEE
6000 GEORGE PROJECT MANAGER
```

21K. Reference the EMPLOYEE and PROJMGR tables. Display the employee number and name of any person who is both an employee and project manager in Department 20. Sort the result by the first column.

```
SELECT ENO, ENAME
FROM   EMPLOYEE
WHERE DNO = 20
INTERSECT
SELECT ENO, PMNAME
FROM   PROJMGR
WHERE DNO = 20
ORDER BY 1
```

```
ENO   ENAME
1000 MOE
6000 GEORGE
```

21L. Reference the EMPLOYEE and PROJMGR tables. Display the employee number and name of any project manager who is not an employee.

```
SELECT ENO, PMNAME
FROM   PROJMGR
EXCEPT
SELECT ENO, ENAME
FROM   EMPLOYEE
ORDER BY 1
```

```
ENO   PMNAME
2500 DICK
4500 DON
```

21M. Reference the PROJ2PARTS table. Display every part number and name and a character-string indicating if the PWT column contains a value that is less than, equal to, or greater than 12. Sort the result by the first column. The result should look like:

```
PNO   PNAME   COMMENTARY
P3    PART3   WEIGHT IS GREATER THAN 12
P4    PART4   WEIGHT IS LESS THAN 12
P5    PART5   WEIGHT IS GREATER THAN 12
P6    PART6   WEIGHT IS EQUAL TO 12
```

```
SELECT PNO, PNAME, 'WEIGHT IS LESS THAN 12' COMMENTARY
FROM   PROJ2PARTS
WHERE PWT < 12
 UNION ALL
SELECT PNO, PNAME, 'WEIGHT IS EQUAL TO 12'
FROM   PROJ2PARTS
WHERE PWT = 12
 UNION ALL
SELECT PNO, PNAME, 'WEIGHT IS GREATER THAN 12'
FROM   PROJ2PARTS
WHERE PWT > 12
ORDER BY 1
```

21N. Reference the EMPLOYEE table. Display the department number and the total salary for each department. Also, display the final total of all salaries. Your SELECT statement should specify UNION ALL. The result should look like:

```
DNO        SUMSALARY
10           2400.00
20          14000.00
40            500.00
FINAL       16900.00
```

```
SELECT CAST (DNO AS CHAR (5)) DNO, SUM (SALARY) SUMSALARY
FROM    EMPLOYEE
GROUP BY DNO
UNION ALL
SELECT 'FINAL', SUM (SALARY)
FROM    EMPLOYEE
ORDER BY 1
```

Notice that the different data types in the first column in each Sub-SELECT inhibit union-compatibility. The DNO column is an integer and “FINAL” is a character string. For this reason, the DNO column was converted to a character-string by specifying the CAST function.

Comment: The optional Chapter 9.5 (Sample Query 9.21) described a better method using the ROLLUP option with the GROUP BY clause.

```
SELECT  DNO, SUM (SALARY) SUMSALARY
FROM    EMPLOYEE
GROUP BY ROLLUP (DNO)
ORDER BY DNO
```

21O. Consider the following SELECT statements. Produce two results for each statement. (1) Assume that INTERSECT has precedence over UNION. (2) Assume there is no precedence among the set operations. Sometimes, both assumptions will produce the same result

Statement-1:

```
(SELECT PNO, PNAME FROM PROJ2PARTS
UNION
SELECT PNO, PNAME FROM PROJ3PARTS)
INTERSECT
SELECT PNO, PNAME FROM PROJ1PARTS
```

Same result under both assumptions. Under both assumptions, parentheses dictate that UNION is executed first, and the final result is:

PNO	PNAME
P4	PART4
P5	PART5

Statement-2:

```
SELECT PNO, PNAME FROM PROJ2PARTS
UNION
SELECT PNO, PNAME FROM PROJ3PARTS
INTERSECT
SELECT PNO, PNAME FROM PROJ1PARTS
```

Assume INTERSECT has precedence over UNION.

PNO	PNAME
P3	PART3
P4	PART4
P5	PART5
P6	PART6

Assume no precedence among the set operations.

PNO	PNAME
P4	PART4
P5	PART5

 Copyright Pending 2020

Statement-3:
```
SELECT PNO, PNAME FROM PROJ2PARTS
        INTERSECT
(SELECT PNO, PNAME FROM PROJ3PARTS
 UNION
SELECT PNO, PNAME FROM PROJ1PARTS)
```

Same result under both assumptions. Under both assumptions, parentheses dictate that UNION is executed first, and the final result is:

PNO	PNAME
P3	PART3
P4	PART4
P5	PART5
P6	PART6

21P. Display the part numbers and names of any part this used in all three projects. (Trick question!)

Solution-1: Inferior solution

```
(SELECT PNO, PNAME FROM PROJ1PARTS
  INTERSECT
SELECT PNO, PNAME FROM PROJ2PARTS)
  INTERSECT
SELECT PNO, PNAME FROM PROJ3PARTS
```

Result: "No rows returned"

Solution-2: Better solution - Know-your-data

Don't bother executing any statement because we know that Project-1 and Project-3 cannot have any parts in common.

Chapter-22 - CASE

22A. For every row in the DEPARTMENT table, display a character-string that is derived from the DNO value according to the following rule.

- If DNO = 10, then display DEPARTMENT-10
- If DNO = 20, then display DEPARTMENT-20
- If DNO = 30, then display DEPARTMENT-30
- If DNO = 40, then display DEPARTMENT-40
- Otherwise, display “Some other department”

Also, display each department’s BUDGET value. Specify DEPTNO as a column-alias for the first column generated by the CASE-expression. Code two SELECT statements using both variations of CASE.

```
DEPTNO           BUDGET
DEPARTMENT-10 75000.00
DEPARTMENT-20 20000.00
DEPARTMENT-30  7000.00
DEPARTMENT-40 25000.00
```

Simple-CASE

```
SELECT
        CASE DNO
                WHEN 10 THEN 'DEPARTMENT-10'
                WHEN 20 THEN 'DEPARTMENT-20'
                WHEN 30 THEN 'DEPARTMENT-30'
                WHEN 40 THEN 'DEPARTMENT-40'
                ELSE 'Some other department'
        END DEPTNO,
        BUDGET
FROM DEPARTMENT;
```

Searched-CASE

```
SELECT
        CASE
                WHEN DNO = 10 THEN 'DEPARTMENT-10'
                WHEN DNO = 20 THEN 'DEPARTMENT-20'
                WHEN DNO = 30 THEN 'DEPARTMENT-30'
                WHEN DNO = 40 THEN 'DEPARTMENT-40'
                ELSE 'Some other department'
        END DEPTNO,
        BUDGET
FROM DEPARTMENT;
```

22B. Reference the REGION table. For each row, display a two-character code for the RNO value followed by the value of the CLIMATE column. Character codes for the RNO values are: 1 = NE, 2 = NW, 3 = SE, 4 = SW, and 5 = MW. Specify "RCODE" as a column-alias for the first column generated by the CASE-expression Code two SELECT statements using both variations of CASE.

```
RCODE  CLIMATE
NE     Cold
NW     Cold
SE     Hot
SW     Hot
MW     Empty
```

Solution-1 (Simple-CASE)

```
SELECT CASE RNO
            WHEN 1 THEN 'NE'
            WHEN 2 THEN 'NW'
            WHEN 3 THEN 'SE'
            WHEN 4 THEN 'SW'
             WHEN 5 THEN 'MW'
             ELSE 'SOME OTHER CODE'
       END RCODE,
      CLIMATE
FROM REGION;
```

Solution-2 (Searched-CASE)

```
SELECT  CASE
            WHEN RNO = 1 THEN 'NE'
            WHEN RNO = 2 THEN 'NW'
            WHEN RNO = 3 THEN 'SE'
            WHEN RNO = 4 THEN 'SW'
            WHEN RNO = 5 THEN 'MW'
            ELSE 'SOME OTHER CODE'
       END RCODE,
      CLIMATE
 FROM REGION;
```

22C. Reference the NTAB table. Only consider rows where both the A and B columns contain non-null values. For each such row, display the A and B values followed by:

- "EQUAL VALUES" if A is equal to B
- "NON-EQUAL VALUES" if A is not equal to B

Specify NOTNULL as a column alias for the result which should look like:

```
A  B NOTNULL
5  5 EQUAL VALUES
5 10 NON-EQUAL VALUES
```

```
SELECT A, B,
       CASE  WHEN A = B THEN 'EQUAL VALUES'
             ELSE 'NON-EQUAL VALUES'
        END  NOTNULL
FROM NTAB
WHERE A IS NOT NULL AND B IS NOT NULL
```

22D. Reference the EMPLOYEE table. Consider the total of all SALARY values in this table. If this total is less than 10,000, display "SMALL TOTAL SALARY". If this total exceeds 20,000, display "LARGE TOTAL SALARY". Otherwise, display "OK SALARY". The result should look like:

```
TEXTMSG
OK SALARY
```

```
SELECT   CASE
              WHEN SUM (SALARY) < 10000
                     THEN 'SMALL TOTAL SALARY'
              WHEN SUM (SALARY) > 20000
                     THEN 'LARGE TOTAL SALARY'
              ELSE 'OK SALARY'
         END TEXTMSG
FROM EMPLOYEE
```

 Copyright Pending 2020

22E. Make the following substitution and then calculate the total of all SALARY values in the EMPLOYEE table. For each SALARY value that is less than 1,000, substitute 1,000 for that value. The result should look like:

```
ADJUSTEDSALARY
--------------
      18000.00
```

```
SELECT SUM (CASE
                WHEN SALARY < 1000 THEN 1000
                ELSE SALARY
            END) ADJUSTEDSALARY
FROM EMPLOYEE
```

22F. Reference the EMPLOYEE table. Assume that all ENAME values are unique. Display three summary totals:

(i) The total of all employee salaries.

(ii) The total of all employee salaries assuming that MOE has been fired. (MOE's SALARY value is zero).

(iii) The total of all employee salaries assuming that both LARRY and CURLY have been fired. (Both SALARY values are zero.)

The result should look like:

```
 ALLEMPLOYEES      NOMOE    NOLARRYCURLY
--------------  ----------  ------------
     16900.00   14900.00        11900.00
```

```
SELECT SUM (SALARY) ALLEMPLOYEES,
       SUM (CASE WHEN ENAME = 'MOE'   THEN 0
                 ELSE SALARY END) NOMOE,
       SUM (CASE WHEN ENAME IN ('LARRY', 'CURLY') THEN 0
                 ELSE SALARY END) NOLARRYCURLY
FROM EMPLOYEE
```

22G. Reference the PRESERVE table. Display the state code and total acreage for all preserves in any state having a total acreage that exceeds 15,000 acres. If a state has less than or equal to 15,000 acres, display the state code followed by a character-string stating “LESS THAN OR EQUAL TO 15000 ACRES”. The result should look like:

```
STATE      TOTACRES
AZ         51360
MA         LESS THAN OR EQUAL TO 15000 ACRES
MT         16931
```

```
SELECT STATE,
    CASE
        WHEN SUM (ACRES) > 15000 THEN CHAR (SUM (ACRES))
        ELSE   'LESS THAN OR EQUAL TO 15000 ACRES'
    END TOTACRES
FROM  PRESERVE
GROUP BY STATE
```

Summary Exercises (Chapter 22)

Specify CASE-Expressions to satisfy the following query objectives.

22H. This exercise has the same query objective as Exercise 21H. Reference the PROJ1PARTS1 table. Produce a result that displays every part number and name, followed by a character-string indicating if the QTY column contains a value that is less than, equal to, or greater than 16. Sort the result by PNO. The result should look like:

```
PNO   PNAME      SIZE
P1    PART1      EQUAL TO 16
P2    PART2      EQUAL TO 16
P4    PART4      GREATER THAN 16
P5    PART5      LESS THAN 16
```

```
SELECT PNO, PNAME,
            CASE WHEN QTY < 16 THEN 'LESS THAN 16'
                 WHEN QTY = 16 THEN 'EQUAL TO 16'
                 ELSE 'GREATER THAN 16'
            END SIZE
FROM PROJ1PARTS
ORDER BY PNO
```

22I. This exercise has the same query objective as Sample Query 11.13b. Reference the NTAB table. Calculate the grand total of all values using the two cross-tabulation patterns. (1) Summarize the subtotals of column values. (2) Summarize the subtotals of row values. Substitute 6 for any null value in column A, and substitute 9 for any null value in column B. The result should look like:

```
GRANDTOTAL1     GRANDTOTAL2
          70              70
```

```
SELECT
    SUM (CASE WHEN A IS NULL THEN 6 ELSE A END) +
    SUM (CASE WHEN B IS NULL THEN 9 ELSE B END) GRANDTOTAL1,
    SUM ((CASE WHEN A IS NULL THEN 6 ELSE A END) +
         (CASE WHEN B IS NULL THEN 9 ELSE B END)) GRANDTOTAL2
FROM NTAB
```

 Copyright Pending 2020

22J. This exercise is a variation on Exercise 22F. For each department referenced in the EMPLOYEE table, display the department number followed by three summary totals: (i) The total of each departmental salary assuming that MOE will be fired. (ii) The total of each departmental salary assuming that LARRY will be fired. (iii) The total of each departmental salary assuming that CURLY will be fired. The result should look like:

DNO	SUMWITHOUTMOE	SUMWITHOUTLARRY	SUMWITHOUTCURLY
10	2400.00	400.00	2400.00
20	12000.00	14000.00	11000.00
40	500.00	500.00	500.00

```
SELECT DNO,
  SUM (CASE WHEN ENAME = 'MOE'   THEN 0
                 ELSE SALARY END) SUMWITHOUTMOE,
  SUM (CASE WHEN ENAME = 'LARRY' THEN 0
                 ELSE SALARY END) SUMWITHOUTLARRY,
  SUM (CASE WHEN ENAME = 'CURLY' THEN 0
                 ELSE SALARY END) SUMWITHOUTCURLY
FROM EMPLOYEE
GROUP BY DNO;
```

 Copyright Pending 2020

22K. This exercise extends the preceding Exercise 22J. Display a final row in the result table that contains the grand totals of all salaries. The result should look like:

```
DNO  WITHOUTMOE  WITHOUTLARRY   WITHOUTCURLY
10     2400.00        400.00        2400.00
20    12000.00      14000.00       11000.00
40      500.00        500.00         500.00
TOTAL 14900.00       14900.00       13900.00
```

Hint: Consider the UNION ALL operation. Also, regarding the first column, note that DNO contains integer values, but "TOTAL" is a character string.

```
SELECT CAST (DNO AS CHAR(5)),
            SUM (CASE WHEN ENAME = 'MOE'   THEN 0
                          ELSE SALARY END) SUMWITHOUTMOE,
            SUM (CASE WHEN ENAME = 'LARRY' THEN 0
                          ELSE SALARY END) SUMWITHOUTLARRY,
             SUM (CASE WHEN ENAME = 'CURLY' THEN 0
                           ELSE SALARY END) SUMWITHOUTCURLY
FROM EMPLOYEE
GROUP BY DNO
   UNION ALL
SELECT 'TOTAL',
            SUM (CASE WHEN ENAME = 'MOE'   THEN 0
                          ELSE SALARY END),
            SUM (CASE WHEN ENAME = 'LARRY' THEN 0
                           ELSE SALARY END),
            SUM (CASE WHEN ENAME = 'CURLY' THEN 0
                           ELSE SALARY END)
FROM EMPLOYEE
ORDER BY 1
```

 Copyright Pending 2020

22L. Reference the EMPLOYEE table. For each department that has at least one employee, display the department number and its average salary followed by a comment that indicates if this departmental average is less than, equal to, or greater than the overall average salary of all employees. Sort the result by department numbers. The result should look like:

```
DNO    AVGSAL     COMMENTARY
 10   1200.00     LESS THAN OVERALL DEPARTMENTAL AVERAGE
 20   4666.66     GREATER THAN OVERALL DEPARTMENTAL AVERAGE
 40    500.00     LESS THAN OVERALL DEPARTMENTAL AVERAGE
```

```
SELECT DNO, AVG (SALARY) AVGSAL,
        CASE
        WHEN AVG (SALARY) < (SELECT AVG (SALARY) FROM EMPLOYEE)
                THEN 'LESS THAN OVERALL DEPARTMENTAL AVERAGE'
        WHEN AVG (SALARY) = (SELECT AVG (SALARY) FROM EMPLOYEE)
                THEN 'EQUAL TO OVERALL DEPARTMENTAL AVERAGE'
        ELSE  'GREATER THAN OVERALL DEPARTMENTAL AVERAGE'
        END  COMMENTARY
FROM EMPLOYEE
GROUP BY DNO
ORDER BY DNO
```

22M. This exercise is a variation of the preceding Exercise 22L. Address the circumstance where a department may have only one or two employees, allowing for the deduction of confidential individual salaries. For each department that has at least one employee, display the department number and a count of the number of employees who work in the department. If the department has more than two employees, display a comment indicating if the departmental average is less than, equal to, or greater than the overall average salary of all employees. Otherwise, if the department only has one or two employees, the comment should state "CONFIDENTIAL". The result should look like:

```
DNO   EMPCT     COMMENTARY
10         2     CONFIDENTIAL
20         3     GREATER THAN OVERALL DEPARTMENTAL AVERAGE
40         1     CONFIDENTIAL
```

```
SELECT DNO, COUNT(*) EMPCT,
        CASE
                WHEN COUNT (*) < 3 THEN 'CONFIDENTIAL'
                WHEN AVG (SALARY) < (SELECT AVG (SALARY) FROM EMPLOYEE)
                        THEN 'LESS THAN OVERALL DEPARTMENTAL AVERAGE'
                WHEN AVG (SALARY) = (SELECT AVG (SALARY) FROM EMPLOYEE)
                        THEN 'EQUAL TO OVERALL DEPARTMENTAL AVERAGE'
                ELSE    'GREATER THAN OVERALL DEPARTMENTAL AVERAGE'
        END COMMENTARY
FROM EMPLOYEE
GROUP BY DNO
ORDER BY DNO
```

For the sake of illustration, we present another solution that illustrates the nesting of a CASE-expression within another CASE-expression.

```
SELECT DNO, COUNT(*) EMPCT,
        CASE
          WHEN COUNT(*) < 3 THEN 'CONFIDENTIAL'
          ELSE CASE
                WHEN AVG (SALARY) < (SELECT AVG (SALARY) FROM EMPLOYEE)
                        THEN 'LESS THAN OVERALL DEPARTMENTAL AVERAGE'
                WHEN AVG (SALARY) = (SELECT AVG (SALARY) FROM EMPLOYEE)
                        THEN 'EQUAL TO OVERALL DEPARTMENTAL AVERAGE'
                ELSE  'GREATER THAN OVERALL DEPARTMENTAL AVERAGE'
          END
        END COMMENTARY
FROM EMPLOYEE
GROUP BY DNO
ORDER BY DNO
```

22N. Pivot a table: This is an optional and very challenging exercise. This exercise asks you to use CASE to "pivot" (or "rotate") tabular data into a spreadsheet format. Again, we recommend using your front-end tool for this kind of report formatting. Also, some database vendors provide special purpose built-in functions (e.g., PIVOT) that can pivot tabular data. [These functions are not covered in this book. They may be presented in a future edition.]

Query Objective: Represent the following PARTSUPP table in a spreadsheet format as illustrated below. Assume you know that supplier numbers range from S1 to S8.

PARTSUPP Table

PNO	SNO	PSPRICE
P5	S1	10.00
P1	S2	10.50
P5	S2	10.00
P7	S2	2.00
P3	S3	12.00
P1	S4	11.00
P3	S4	12.50
P4	S4	12.00
P5	S4	11.00
P6	S4	4.00
P7	S4	3.00
P8	S4	5.00
P7	S5	3.50
P6	S6	4.00
P7	S6	3.50
P8	S6	4.00
P6	S8	4.00
P8	S8	3.00

Spreadsheet Format

	S1	S2	S3	S4	S5	S6	S7	S8
P1	0.00	10.50	0.00	11.00	0.00	0.00	0.00	0.00
P3	0.00	0.00	12.00	12.50	0.00	0.00	0.00	0.00
P4	0.00	0.00	0.00	12.00	0.00	0.00	0.00	0.00
P5	10.00	10.00	0.00	11.00	0.00	0.00	0.00	0.00
P6	0.00	0.00	0.00	4.00	0.00	4.00	0.00	4.00
P7	0.00	2.00	0.00	3.00	3.50	3.50	0.00	0.00
P8	0.00	0.00	0.00	5.00	0.00	4.00	0.00	3.00

Hint: Form groups of PNO values. Display PNO followed by eight SUM functions, one for each SNO value. Each SUM function should be similar to that shown below.

```
SUM (CASE WHEN SNO = 'S1' THEN PSPRICE ELSE 0 END) S1
```

We develop the SELECT statement solution on the following pages.

The final solution is:

```
SELECT PNO,
        SUM (CASE WHEN SNO = 'S1' THEN PSPRICE ELSE 0 END) S1,
        SUM (CASE WHEN SNO = 'S2' THEN PSPRICE ELSE 0 END) S2,
        SUM (CASE WHEN SNO = 'S3' THEN PSPRICE ELSE 0 END) S3,
        SUM (CASE WHEN SNO = 'S4' THEN PSPRICE ELSE 0 END) S4,
        SUM (CASE WHEN SNO = 'S5' THEN PSPRICE ELSE 0 END) S5,
        SUM (CASE WHEN SNO = 'S6' THEN PSPRICE ELSE 0 END) S6,
        SUM (CASE WHEN SNO = 'S7' THEN PSPRICE ELSE 0 END) S7,
        SUM (CASE WHEN SNO = 'S8' THEN PSPRICE ELSE 0 END) S8
FROM PARTSUPP
GROUP BY PNO
ORDER BY PNO
```

Below we show the “components” of this statement.

After the grouping operation is applied, the intermediate result looks like:

PNO	SNO	PSPRICE
P1	S2	10.50
P1	S4	11.00
P3	S3	12.00
P3	S4	12.50
P4	S4	12.00
P5	S1	10.00
P5	S2	10.00
P5	S4	11.00
P6	S4	4.00
P6	S6	4.00
P6	S8	4.00
P7	S2	2.00
P7	S4	3.00
P7	S5	3.50
P7	S6	3.50
P8	S4	5.00
P8	S6	4.00
P8	S8	3.00

Note: There is no group for Part P2 because no suppliers currently supply this part.

Consider first group for Part P1

```
P1  S2    10.50
P1  S4    11.00
```

Looking at first row in desired result table, we want to ("somehow") have these two rows appear in the result table as:

```
      S1    S2    S3    S4    S5    S6    S7    S8
P1  0.00 10.50  0.00 11.00 0.00 0.00 0.00 0.00
```

Likewise, for all PNO values.

The first step in this "somehow" is to specify a CASE statement for each SNO values in the SELECT-clause.

```
CASE WHEN SNO = 'S1' THEN PSPRICE ELSE 0 END S1,
CASE WHEN SNO = 'S2' THEN PSPRICE ELSE 0 END S2,
CASE WHEN SNO = 'S3' THEN PSPRICE ELSE 0 END S3,
CASE WHEN SNO = 'S4' THEN PSPRICE ELSE 0 END S4,
CASE WHEN SNO = 'S5' THEN PSPRICE ELSE 0 END S5,
CASE WHEN SNO = 'S6' THEN PSPRICE ELSE 0 END S6,
CASE WHEN SNO = 'S7' THEN PSPRICE ELSE 0 END S7,
CASE WHEN SNO = 'S8' THEN PSPRICE ELSE 0 END S8
```

As an experiment, temporally remove the GROUP BY clause and SUM functions from the final solution, and execute the following statement.

```
SELECT PNO,
        CASE WHEN SNO = 'S1' THEN PSPRICE ELSE 0 END S1,
        CASE WHEN SNO = 'S2' THEN PSPRICE ELSE 0 END S2,
        CASE WHEN SNO = 'S3' THEN PSPRICE ELSE 0 END S3,
        CASE WHEN SNO = 'S4' THEN PSPRICE ELSE 0 END S4,
        CASE WHEN SNO = 'S5' THEN PSPRICE ELSE 0 END S5,
        CASE WHEN SNO = 'S6' THEN PSPRICE ELSE 0 END S6,
        CASE WHEN SNO = 'S7' THEN PSPRICE ELSE 0 END S7,
        CASE WHEN SNO = 'S8' THEN PSPRICE ELSE 0 END S8
FROM PARTSUPP
ORDER BY PNO
```

The temporary result looks like:

PNO	S1	S2	S3	S4	S5	S6	S7	S8
P1	0.00	10.50	0.00	0.00	0.00	0.00	0.00	0.00
P1	0.00	0.00	0.00	11.00	0.00	0.00	0.00	0.00
P3	0.00	0.00	12.00	0.00	0.00	0.00	0.00	0.00
P3	0.00	0.00	0.00	12.50	0.00	0.00	0.00	0.00
P4	0.00	0.00	0.00	12.00	0.00	0.00	0.00	0.00
P5	10.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
P5	0.00	10.00	0.00	0.00	0.00	0.00	0.00	0.00
P5	0.00	0.00	0.00	11.00	0.00	0.00	0.00	0.00
P6	0.00	0.00	0.00	4.00	0.00	0.00	0.00	0.00
P6	0.00	0.00	0.00	0.00	0.00	4.00	0.00	0.00
P6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.00
P7	0.00	2.00	0.00	0.00	0.00	0.00	0.00	0.00
P7	0.00	0.00	0.00	3.00	0.00	0.00	0.00	0.00
P7	0.00	0.00	0.00	0.00	3.50	0.00	0.00	0.00
P7	0.00	0.00	0.00	0.00	0.00	3.50	0.00	0.00

Now consider first two rows for Part P1

PNO	S1	S2	S3	S4	S5	S6	S7	S8
P1	0.00	10.50	0.00	0.00	0.00	0.00	0.00	0.00
P1	0.00	0.00	0.00	11.00	0.00	0.00	0.00	0.00

We want to “merge” the information in these two rows into one row. The “trick” is to use the GROUP by clause.

Again, temporally, enhance the preceding statement by *only* specifying a GROUP BY clause.

```
SELECT PNO,
        CASE WHEN SNO = 'S1' THEN PSPRICE ELSE 0 END S1,
        CASE WHEN SNO = 'S2' THEN PSPRICE ELSE 0 END S2,
        CASE WHEN SNO = 'S3' THEN PSPRICE ELSE 0 END S3,
        CASE WHEN SNO = 'S4' THEN PSPRICE ELSE 0 END S4,
        CASE WHEN SNO = 'S5' THEN PSPRICE ELSE 0 END S5,
        CASE WHEN SNO = 'S6' THEN PSPRICE ELSE 0 END S6,
        CASE WHEN SNO = 'S7' THEN PSPRICE ELSE 0 END S7,
        CASE WHEN SNO = 'S8' THEN PSPRICE ELSE 0 END S8
GROUP BY PNO  ←
FROM PARTSUPP
ORDER BY PNO
```

This statement will not execute because the SELECT-clause fails the basic grouping syntax rule. An aggregate function must be specified for every result column except the PNO column. This leads to the specification of the SUM functions.

```
SELECT PNO,
        SUM (CASE WHEN SNO = 'S1' THEN PSPRICE ELSE 0 END) S1,
        SUM (CASE WHEN SNO = 'S2' THEN PSPRICE ELSE 0 END) S2,
        SUM (CASE WHEN SNO = 'S3' THEN PSPRICE ELSE 0 END) S3,
        SUM (CASE WHEN SNO = 'S4' THEN PSPRICE ELSE 0 END) S4,
        SUM (CASE WHEN SNO = 'S5' THEN PSPRICE ELSE 0 END) S5,
        SUM (CASE WHEN SNO = 'S6' THEN PSPRICE ELSE 0 END) S6,
        SUM (CASE WHEN SNO = 'S7' THEN PSPRICE ELSE 0 END) S7,
        SUM (CASE WHEN SNO = 'S8' THEN PSPRICE ELSE 0 END) S8
FROM PARTSUPP
GROUP BY PNO
ORDER BY PNO
```

Executing this statement produces the desired result. Amen!

Again, consider first two P1-rows in previous temporary result.

PNO	S1	S2	S3	S4	S5	S6	S7	S8
P1	0.00	10.50	0.00	0.00	0.00	0.00	0.00	0.00
P1	0.00	0.00	0.00	11.00	0.00	0.00	0.00	0.00

Observe that you could have specified MAX instead of SUM and still get the desired result.

.

Author Comment: I did not derive this solution starting from scratch. I found the basic structure in multiple locations on the web. The moral of this story is simple. If you have a complex query objective that conforms to some generic pattern (e.g., table-to-spreadsheet), take the time to explore the web.

There is another lesson: I think it would have taken me a long time (perhaps forever) to satisfy this query objective by starting from scratch. But, some very smart person did solve it. I may know as much SQL as this person, but she “put the SQL pieces together” in a creative manner. The moral: Learning SQL is just a starting point.

PART VI

Sub-SELECTs

Chapter-23 - Regular Sub-SELECTs

23A. Display all information about any employee who earns the largest salary.

```
SELECT * FROM EMPLOYEE
WHERE SALARY = (SELECT MAX (SALARY) FROM EMPLOYEE)
```

```
ENO   ENAME   SALARY   DNO
6000 GEORGE 9000.00   20
```

23B. Display all information about any employee whose salary exceeds the overall average salary.

```
SELECT * FROM EMPLOYEE
WHERE SALARY > (SELECT AVG (SALARY) FROM EMPLOYEE)
```

```
ENO   ENAME   SALARY   DNO
6000 GEORGE 9000.00   20
```

23C. Be careful with your logic. Note that the above Sample Query 23.3 specified the same WHERE-clause in the Sub-SELECT and the Outer-SELECT. Is this an unnecessary redundancy?

(a) What if "WHERE DNO=10" is *only* specified in the Sub-SELECT:

```
SELECT  * FROM EMPLOYEE
WHERE SALARY = (SELECT MAX (SALARY) FROM EMPLOYEE
                      WHERE DNO = 10)
```

```
ENO   ENAME      SALARY     DNO
1000 MOE        2000.00    20
2000 LARRY      2000.00    10
```

(b) What if "WHERE DNO=10" is *only* specified in the Outer-SELECT:

```
SELECT  * FROM EMPLOYEE
WHERE DNO = 10
AND SALARY = (SELECT MAX (SALARY) FROM EMPLOYEE)
```

Result is "no row returned"

Both of the above results are wrong. (Both results differ from the correct result shown for Sample Query 23.2.) The logic requires the "DNO = 10" condition to be specified in both the Outer-SELECT and the Sub-SELECT.

 Copyright Pending 2020

23D. Display the name and salary of any employee who has a salary that exceeds the smallest BUDGET value in the DEPARTMENT table.

```
SELECT ENAME, SALARY FROM EMPLOYEE
WHERE SALARY > (SELECT MIN (BUDGET) FROM DEPARTMENT)
```

```
ENAME    SALARY
GEORGE 9000.00
```

23E. Reference the REGION and STATE tables in the MTPCH database. Display the name of every REGION that is related to some row in the STATE table. Specify a Sub-SELECT in your solution.

```
SELECT RNAME FROM REGION
WHERE RNO IN (SELECT RNO FROM STATE)
```

```
RNAME
NORTHEAST
NORTHWEST
SOUTHEAST
SOUTHWEST
```

23F. Reference the REGION and STATE tables. Display the name of every state that is located in the NORTHEAST region. Specify a Sub-SELECT in your solution.

```
SELECT STNAME
FROM   STATE
WHERE RNO IN (SELECT RNO FROM REGION
                    WHERE RNAME = 'NORTHEAST')
```

```
STNAME
CONNECTICUT
MASSACHUSETTS
```

23G. Review: Use join operations to solve the exercises 23E and 23F.

```
[23E]    SELECT DISTINCT R.RNAME
         FROM    REGION R, STATE ST
         WHERE R.RNO = ST.RNO

[23F]    SELECT ST.STNAME
         FROM    REGION R, STATE ST
         WHERE R.RNO = ST.RNO
         AND      R.RNAME = 'NORTHEAST'
```

23H. Reference the REGION and STATE tables. Display the name of any region with a CLIMATE of "Hot" and is related to some state. Code two solutions using (i) a Sub-SELECT and (ii) a join operation.

```
SELECT RNAME
FROM REGION
WHERE CLIMATE = 'Hot'
AND RNO IN (SELECT RNO FROM STATE)
```

```
SELECT DISTINCT RNAME
FROM REGION R, STATE ST
WHERE R.RNO = ST.RNO
AND R.CLIMATE = 'Hot'
```

```
RNAME
SOUTHEAST
SOUTHWEST
```

23I. Important Exercise: In the commentary for Sample Query 17.3.2, we considered the following query objective and concluded that it could not be satisfied by coding a join-operation:

> Reference the DEPARTMENT and EMPLOYEE tables. Display the overall total budget of those departments which have at least one employee.

Satisfy this query objective by coding a Sub-SELECT.

```
SELECT SUM (BUDGET) TOTBUDGET
FROM DEPARTMENT
WHERE DNO IN (SELECT DNO FROM EMPLOYEE)
```

```
TOTBUDGET
120000.00
```

23J. Reference the REGION and STATE tables. Display the name of any region that is not associated with a state.

```
SELECT RNAME
FROM  REGION
WHERE RNO NOT IN (SELECT RNO FROM STATE)
```

```
RNAME
MIDWEST
```

 Copyright Pending 2020

23K. Reference the EMPLOYEE table. You are asked to display all information about any employee who is not assigned to some department. The following statement produces the correct result.

```
SELECT * FROM EMPLOYEE
WHERE DNO NOT IN (SELECT DNO FROM DEPARTMENT)
```

However, why does this statement constitute a "silly" solution?

"Ask a silly question – Get a silly answer"

Executing the above statement must return an empty table. This solution is silly because the query objective is silly. *There is no need to execute any statement because we know that every employee is assigned to some department.* We know this because EMPLOYEE.DNO is a *non-null* foreign-key that references DEPARTMENT.DNO.

23L. Reference the STATE and CUSTOMER tables in the MTPCH database. Display the name of any state that does not have at least one customer.

```
SELECT STNAME FROM STATE
WHERE STCODE NOT IN (SELECT STCODE FROM CUSTOMER)
```

```
STNAME
CONNECTICUT
```

23M. Reference the CUSTOMER and PUR_ORDER tables in the MTPCH database. Display the number and name of any customer who has not purchased any parts (i.e., is not related to any purchase orders).

```
SELECT CNO, CNAME FROM CUSTOMER
WHERE CNO NOT IN (SELECT CNO FROM PUR_ORDER)
```

```
CNO  CNAME
890  MANDELBROT
780  CHURCH
```

 Copyright Pending 2020

23N. Reference the STATE, CUSTOMER, and PUR-ORDER tables in the MTPCH database. Display the name of any state that has a customer who has not purchased any parts.

Two Solutions:

```
SELECT STNAME FROM STATE
WHERE STCODE IN
        (SELECT STCODE FROM CUSTOMER
         WHERE CNO NOT IN
                    (SELECT CNO FROM PUR_ORDER));
```

```
SELECT ST.STNAME
FROM    STATE ST, CUSTOMER C
WHERE ST.STCODE = C.STCODE
AND    C.CNO NOT IN (SELECT CNO FROM PUR_ORDER);
```

```
STNAME
ARIZONA
NEW MEXICO
```

 Copyright Pending 2020

23O. Reference the PART, SUPPLIER, and PARTSUPP tables in the MTPCH database. Display the supplier number and name of any supplier who can sell you PART5 (i.e., PNAME value is PART5). Code four solutions.

(a) Code a Sub-SELECT where the Sub-SELECT specifies a two-table join. (Similar to Sample Query 23.10)

(b) Code a Sub-SELECT where the Outer-SELECT specifies a two-table join. (Similar to Sample Query 23.11)

(c) Code a Sub-SELECT nested within another Sub-SELECT. (Similar to Sample Query 23.12)

(d) For review purposes, code a three-table join.

```
SNO  SNAME
S1   SUPPLIER1
S2   SUPPLIER2
S4   SUPPLIER4
```

(a)
```
SELECT S.SNO, S.SNAME
FROM   SUPPLIER S
WHERE S.SNO IN
       (SELECT PS.SNO
       FROM   PART P, PARTSUPP PS
       WHERE P.PNO = PS.PNO
       AND      P.PNAME = 'PART5')
```

(b)
```
SELECT S.SNO, S.SNAME
FROM   SUPPLIER S, PARTSUPP PS
WHERE S.SNO = PS.SNO
AND   PS.PNO IN
       (SELECT PNO
       FROM    PART P
       WHERE PNAME = 'PART5')
```

(c)
```
SELECT SNO, SNAME
FROM   SUPPLIER
WHERE SNO IN
       (SELECT SNO
       FROM    PARTSUPP
       WHERE  PNO IN
              (SELECT PNO
              FROM    PART P
              WHERE  PNAME = 'PART5') )
```

(d)
```
SELECT      S.SNO, S.SNAME
FROM        SUPPLIER S, PARTSUPP PS, PART P
WHERE       S.SNO = PS.SNO
AND         P.PNO = PS.PNO
AND         P.PNAME = 'PART5'
```

23P. Reference the PART, SUPPLIER, and PARTSUPP tables in the MTPCH database. Display the supplier number and name of any supplier who can sell you PART8. (i.e., PNAME value is PART8.) Also display the price (PSPRICE) the supplier charges for this part. Code two solutions.

(a) Code a Sub-SELECT where the Outer-SELECT specifies a two-table join. (Similar to Sample Query 23.11)

(b) For review purposes, code a three-table join.

```
SNO  SNAME       PSPRICE
S4   SUPPLIER4      5.00
S6   SUPPLIER6      4.00
S8   SUPPLIER8      3.00
```

(a)
```
SELECT S.SNO, S.SNAME, PS.PSPRICE
FROM   SUPPLIER S, PARTSUPP PS
WHERE S.SNO = PS.SNO
AND     PS.PNO IN (SELECT PNO
                   FROM   PART
                   WHERE  PNAME = 'PART8')
```

(b)
```
SELECT S.SNO, S.SNAME, PS.PSPRICE
FROM   SUPPLIER S, PARTSUPP PS, PART P
WHERE  S.SNO = PS.SNO
AND      PS.PNO = P.PNO
AND      PNAME = 'PART8'
```

23Q: Display all information about the lowest paid employee in every department that has at least one employee.

```
SELECT *
FROM   EMPLOYEE
WHERE (DNO, SALARY)  IN   (  SELECT DNO, MIN (SALARY)
                             FROM EMPLOYEE
                             GROUP BY DNO)
```

```
ENO   ENAME  SALARY   DNO
1000  MOE    2000.00   20
4000  SHEMP   500.00   40
5000  JOE     400.00   10
```

 Copyright Pending 2020

23R. Reference the EMPLOYEE table. Only consider employees who earn less than $5,000.00. Display the DNO and minimum employee salary in those departments having a minimal employee salary that exceeds the overall average salary for all employees under consideration.

```
SELECT DNO, MIN (SALARY) MINSALARY
FROM   EMPLOYEE
WHERE SALARY < 5000.00
GROUP BY DNO
HAVING MIN (SALARY)   >      ( SELECT AVG (SALARY)
                               FROM   EMPLOYEE
                               WHERE SALARY < 5000.00 )
```

```
DNO  MINSALARY
 20    2000.00
```

23S. Consider changing each department's BUDGET value to a value that is equal to the largest BUDGET value minus 10% of the department's current BUDGET value. Display each department's number, name, current budget, and the adjusted budget.

```
SELECT DNO, DNAME, BUDGET,
  (SELECT MAX (BUDGET) FROM DEPARTMENT) - (.10 * BUDGET)
                                                ADJBUDGET
FROM DEPARTMENT
```

```
DNO  DNAME          BUDGET   ADJBUDGET
 10  ACCOUNTING   75000.00  67500.0000
 20  INFO. SYS.   20000.00  73000.0000
 30  PRODUCTION    7000.00  74300.0000
 40  ENGINEERING  25000.00  72500.0000
```

 Copyright Pending 2020

23T. For each department that has at least one employee, display its department number and maximum departmental salary followed textual comment indicating that departmental maximum value is less than or equal to the overall maximum salary.

```
SELECT DNO, MAX (SALARY) MAXSALARY,
 CASE
  WHEN MAX (SALARY) < (SELECT MAX (SALARY) FROM EMPLOYEE)
              THEN 'LESS THAN OVERALL MAX SALARY'
  WHEN MAX (SALARY) = (SELECT MAX(SALARY) FROM EMPLOYEE)
              THEN 'EQUAL TO OVERALL MAX SALARY'
 ELSE         'SOMETHING STRANGE'
END TEXTCOMMENT
FROM EMPLOYEE
GROUP BY DNO
```

```
DNO MAXSALARY TEXTCOMMENT
 10   2000.00 LESS THAN OVERALL MAX SALARY
 20   9000.00 EQUAL TO OVERALL MAX SALARY
 40    500.00 LESS THAN OVERALL MAX SALARY
```

23U. a. Rewrite the following join-operation using a Sub-SELECT.

```
SELECT E3.ENAME, E3.SALARY
FROM   EMPLOYEE3 E3, DEPARTMENT D
WHERE  E3.DNO = D.DNO
```

Sub-SELECT Solution:

```
SELECT ENAME, SALARY
FROM  EMPLOYEE3
WHERE DNO IN (SELECT DNO FROM DEPARTMENT)
```

b. Is the following statement equivalent to the above statement?

```
SELECT ENAME, SALARY FROM EMPLOYEE3
```

No. Observe that EMPLOEE3.DNO is not a foreign-key.

SELECT ENAME, SALARY FROM EMPLOYEE3 will display rows describing MOE and GEORGE. The other two statements do not.

Summary Exercises (Chapter 23)

Code Sub-SELECTs for the following Exercises 23V – 23Ze which reference tables in the MTPCH sample database.

23V. Display all information about the state with the largest population.

```
SELECT * FROM STATE
WHERE POPULATION = (SELECT MAX (POPULATION) FROM STATE)
```

```
STCODE STNAME              POPULATION RNO
    FL FLORIDA               18251000   3
```

23W. Display all information about any state having a population that is less than the overall average population.

```
SELECT * FROM STATE
WHERE POPULATION < (SELECT AVG (POPULATION) FROM STATE)
```

```
STCODE STNAME             POPULATION RNO
CT     CONNECTICUT          3502000   1
MA     MASSACHUSETTS        6450000   1
OR     OREGON               3747000   2
WA     WASHINGTON           6468000   2
NM     NEW MEXICO           1970000   4
AZ     ARIZONA              6339000   4
```

 Copyright Pending 2020

23X. (i) Display the number and name of every supplier who sells part P6.

```
SELECT SNO, SNAME FROM SUPPLIER
WHERE SNO IN
      (SELECT SNO FROM PARTSUPP WHERE PNO = 'P6')
```

```
SNO  SNAME
S4   SUPPLIER4
S6   SUPPLIER6
S8   SUPPLIER8
```

(ii) Display the number and name of every supplier who does not sell part P6.

```
SELECT SNO, SNAME FROM SUPPLIER
WHERE SNO NOT IN
      (SELECT SNO FROM PARTSUPP WHERE PNO = 'P6')
```

```
SNO  SNAME
S1   SUPPLIER1
S2   SUPPLIER2
S3   SUPPLIER3
S5   SUPPLIER5
S7   SUPPLIER7
```

 Copyright Pending 2020

23Y. (i) Display the number and name of every supplier who sells at least one pink part.

```
SELECT SNO, SNAME FROM SUPPLIER
WHERE SNO IN
        (SELECT SNO FROM PARTSUPP WHERE PNO IN
          (SELECT PNO FROM PART WHERE PCOLOR = 'PINK'))
```

```
SNO  SNAME
S2   SUPPLIER2
S3   SUPPLIER3
S4   SUPPLIER4
S5   SUPPLIER5
S6   SUPPLIER6
S8   SUPPLIER8
```

(ii) Display the number and name of every supplier who does not sell any pink parts.

```
SELECT SNO, SNAME FROM SUPPLIER
WHERE SNO NOT IN
        (SELECT SNO FROM PARTSUPP WHERE PNO IN
          (SELECT PNO FROM PART WHERE PCOLOR = 'PINK'))
```

```
SNO  SNAME
S1   SUPPLIER1
S7   SUPPLIER7
```

 Copyright Pending 2020

23Za: For each region with at least one state, display all information about the state with the lowest population in the region.

RNO	STCODE	STNAME	POPULATION
1	CT	CONNECTICUT	3502000
2	OR	OREGON	3747000
3	GE	GEORGIA	9545000
4	NM	NEW MEXICO	1970000

```
SELECT RNO, STCODE, STNAME, POPULATION
FROM   STATE
WHERE (RNO, POPULATION) IN
                    (SELECT RNO, MIN (POPULATION)
                    FROM STATE
                    GROUP BY RNO)
```

If this Sub-SELECT does not work on your system, the following equivalent statement specifies a dynamic view (to be described in Chapter 26) that will satisfy the query objective.

```
SELECT ST.RNO, ST.STCODE, ST.STNAME, ST.POPULATION
FROM STATE ST,
                (SELECT RNO, MIN (POPULATION) MINPOPREG
                 FROM STATE
                 GROUP BY RNO) TEMP
WHERE ST.RNO = TEMP.RNO
AND ST.POPULATION = TEMP.MINPOPREG
```

23Zb. Consider the state with the smallest population in each region that has at least one state. Display the region number and its smallest state population if that population value is less than the overall average population for all states.

```
SELECT RNO, MIN (POPULATION) MINPOP
FROM   STATE
GROUP BY RNO
HAVING MIN (POPULATION)  <   (SELECT AVG (POPULATION)
                               FROM STATE)
```

RNO	MINPOP
1	3502000
2	3747000
4	1970000

23Zc. Reference the PARTSUPP table. Determine the overall average PSPRICE value. For each row, display its SNO, PNO, and PSPRICE values, followed by the difference between the PSPRICE and the overall average PSPRICE value. Sort the result by the fourth column. Observe that the fourth column will contain negative values for PSPRICE values that are less than the average. (Hint: Consider specifying a Sub-SELECT in the Main-SELECT-clause.)

```
SELECT SNO, PNO, PSPRICE,
        PSPRICE - (SELECT AVG (PSPRICE) FROM PARTSUPP) PRDIFF
FROM PARTSUPP
ORDER BY 4
```

SNO	PNO	PSPRICE	PRDIFF
S2	P7	2.00	-4.94
S4	P7	3.00	-3.94
S8	P8	3.00	-3.94
S5	P7	3.50	-3.44
S6	P7	3.50	-3.44
S4	P6	4.00	-2.94
S6	P6	4.00	-2.94
S8	P6	4.00	-2.94
S6	P8	4.00	-2.94
S4	P8	5.00	-1.94
S1	P5	10.00	3.05
S2	P5	10.00	3.05
S2	P1	10.50	3.55
S4	P1	11.00	4.05
S4	P5	11.00	4.05
S3	P3	12.00	5.05
S4	P4	12.00	5.05
S4	P3	12.50	5.55

 Copyright Pending 2020

23Zd. Modify the above Exercise 23Zc. Only display rows for where the PSPRICE exceeds the average PSPRICE. (Hint: Consider specifying the same Sub-SELECT in the Main-SELECT-clause and the WHERE-clause.)

```
SELECT SNO, PNO, PSPRICE,
  PSPRICE - (SELECT AVG (PSPRICE) FROM PARTSUPP)  PRDIFF
FROM PARTSUPP
WHERE PSPRICE > (SELECT AVG (PSPRICE) FROM PARTSUPP)
```

Specifying the same Sub-SELECT twice is reduntant.. Chapter 27 introduces the WITH-clause that will offer a better solution to this problem.

SNO	PNO	PSPRICE	PRDIFF
S2	P1	10.50	3.55
S4	P1	11.00	4.05
S3	P3	12.00	5.05
S4	P3	12.50	5.55
S4	P4	12.00	5.05
S1	P5	10.00	3.05
S2	P5	10.00	3.05
S4	P5	11.00	4.05

 Copyright Pending 2020

23Ze. Reference the DEPARTMENT and EMPLOYEE tables. Revisit Sample Query 17.3.2 where you were asked to summarize a numeric parent-column for the parent-table participating in a parent-child join operation: Only consider those departments that have employees and have a budget that is less than or equal to $50,000.00. Display the total budget for these departments.

```
SUM (DISTINCT D.BUDGET)
                45000.00
```

Sample Query 17.3.2 considered following the following "almost correct" (i.e., wrong) "solution."

```
SELECT SUM (DISTINCT D.BUDGET)
FROM   DEPARTMENT D, EMPLOYEE E
WHERE D.DNO = E.DNO
AND     D.BUDGET <= 50000.00
```

This solution "got lucky" because no two DEPARTMENT rows happened to have the same BUDGET value. Code a SELECT statement that constitutes a correct solution.

```
SELECT SUM (BUDGET)  TOTBUDGET
FROM   DEPARTMENT
WHERE DNO IN
        ( SELECT D.DNO
          FROM    DEPARTMENT D, EMPLOYEE E
          WHERE D.DNO = E.DNO
          AND     D.BUDGET <= 50000.00)
```

The Sub-SELECT finds the DNO values of those departments that have employees and have a budget that is less than or equal to $50,000.00. The outer-SELECT summarizes the BUDGET values of these departments, including any duplicate BUDGET values corresponding to different departments that happen to have the same budget.

 Copyright Pending 2020

The following exercises are presented for review purposes.

23Zf. Review Exercise: Satisfy Sample Queries 23.10 and 23.11 using join-operations.

Sample Query 23.10: Reference the PART, SUPPLIER, and PARTSUPP tables in the MTPCH database. Display the part number and name of any part that you can purchase from SUPPLIER2 (i.e., SNAME value is "SUPPLIER2".)

```
SELECT P.PNO, P.PNAME
FROM   PART P, SUPPLIER S, PARTSUPP PS
WHERE  S.SNO = PS.SNO
AND    P.PNO = PS.PNO
AND    S.SNAME = 'SUPPLIER2'
```

Sample Query 23.11**:** Reference the PART, SUPPLIER, and PARTSUPP tables in the MTPCH database. Display the part number, name, and price for any part that you can purchase from SUPPLIER2.

```
SELECT  P.PNO, P.PNAME, PS.PSPRICE
FROM   PART P, SUPPLIER S, PARTSUPP PS
WHERE  S.SNO = PS.SNO
AND    P.PNO = PS.PNO
AND    S.SNAME = 'SUPPLIER2'
```

23Zg. Optional Review Exercise: This is a strange tutorial exercise. Assume you simply did not want to write a statement that contains NOT IN. You are invited you to code a very inconvenient, rather roundabout (and obviously inefficient) solution to Sample Query 23.8 (Display the DNO, DNAME and BUDGET values for any department that does not have any employees.) Generate two intermediate results. The first has the DNO, DNAME and BUDGET values of all departments. The second has the same values for those departments that have employees. Then use EXCEPT to "subtract" the second intermediate result from the first.

```
SELECT DNO, DNAME, BUDGET
FROM   DEPARTMENT
EXCEPT
SELECT D.DNO, D.DNAME, D.BUDGET
FROM   DEPARTMENT D, EMPLOYEE E
WHERE D.DNO = E.DNO
```

```
DNO  DNAME       BUDGET
 30  PRODUCTION 7000.00
```

The following exercises address some previously described SQL challenges.

23Zh. Review Sample Query 8.3 which described a common error shown below.

```
SELECT *
FROM PRESERVE
WHERE FEE > AVG (FEE)
```

Code a correct SELECT statement to satisfy this query objective.

```
SELECT *
FROM PRESERVE
WHERE FEE > (SELECT AVG (FEE) FROM PRESERVE)
```

23Zi. Review the page after Sample Query 7.6 which discussed a potential problem of dividing-by-zero in a calculated condition. There we described two potentially problematic statements.

Statement-A:
```
SELECT PNAME, ACRES/FEE
FROM    PRESERVE
WHERE FEE <> 0 AND ACRES/FEE > 200.00
```

*S*tatement-B:
```
SELECT PNAME, ACRES/FEE
FROM   PRESERVE
WHERE ACRES/FEE > 200.00 AND FEE <> 0
```

Code an alternative equivalent SELECT statement that satisfies this query objective where you are asked to display the PNAME and ratio ACRES/FEE for all preserves where this ratio exceeds 200.00.

```
SELECT PNAME, ACRES/FEE RATIO
FROM    PRESERVE
WHERE PNO IN       (SELECT PNO
                    FROM PRESERVE
                    WHERE FEE <> 0)
AND   ACRES/FEE > 200.00
```

 Copyright Pending 2020

Chapter-24 --Sub-SELECTs in DML

24A. Delete all rows from the MYEMP table.

```
DELETE FROM MYEMP
```

24B. Copy the ENAME, SALARY and DNO values from EMPLOYEE into MYEMP. Only copy rows for employees having a salary that is less $8,000.00.

```
INSERT INTO MYEMP
      SELECT ENAME, SALARY, DNO
      FROM   EMPLOYEE
      WHERE SALARY < 8000.00
```

MYEMP now looks like:

MYENAME	MYSALARY	MYDNO
MOE	2000.00	20
LARRY	2000.00	10
CURLY	3000.00	20
SHEMP	500.00	40
JOE	400.00	10

24C. Update the MYEMP table. Change the MYENAME values of all rows having an MYDNO value of 10. All modified MYENAME values should be the same as the name of the MYEMP employee having the largest salary.

```
UPDATE MYEMP
SET  MYENAME =
       (SELECT MYENAME
        FROM    MYEMP
        WHERE MYSALARY  =   ( SELECT MAX (MYSALARY)
                              FROM     MYEMP))
WHERE MYDNO = 10
```

MYEMP now looks like:

MYENAME	MYSALARY	MYDNO
MOE	2000.00	20
CURLY	2000.00	10
CURLY	3000.00	20
SHEMP	500.00	40
CURLY	400.00	10

 Copyright Pending 2020

24D. Delete MYEMP rows corresponding to employees who have the same name as the highest paid employee. Assume that multiple employees can have the same largest salary.

```
DELETE FROM MYEMP
WHERE MYENAME IN (SELECT MYENAME
                  FROM    MYEMP
                  WHERE MYSALARY =
                      (SELECT MAX (MYSALARY)
                      FROM     MYEMP))
```

MYEMP now looks like:

MYENAME	MYSALARY	MYDNO
MOE	2000.00	20
SHEMP	500.00	40

24E. Drop the MYEMP table.

```
DROP TABLE MYEMP
```

Chapter-25 - Correlated Sub-SELECTs

25A. Display all information about the lowest paid employee in each department.

```
SELECT *
FROM   EMPLOYEE EX
WHERE SALARY = (  SELECT MIN (SALARY)
                  FROM   EMPLOYEE
                  WHERE DNO = EX.DNO)
```

```
ENO   ENAME  SALARY   DNO
1000  MOE    2000.00   20
4000  SHEMP   500.00   40
5000  JOE     400.00   10
```

25B. Display the name and salary of any employee whose salary is less than the average employee salary for his department.

```
SELECT ENAME, SALARY
FROM   EMPLOYEE EX
WHERE SALARY < (SELECT AVG (SALARY)
                FROM   EMPLOYEE
                WHERE DNO = EX.DNO)
```

```
ENAME   SALARY
MOE    2000.00
CURLY  3000.00
JOE     400.00
```

25C. Code two alternative solutions for this Sample Query 25.4. The first solution should specify IN. The second solution should specify a join-operation.

```
SELECT *
FROM DEPARTMENT
WHERE DNO IN (SELECT DNO FROM EMPLOYEE);
```

```
SELECT DISTINCT D.DNO, DNAME, BUDGET
FROM    DEPARTMENT D, EMPLOYEE E
WHERE D.DNO = E.DNO;
```

25D. Reference the STATE and CUSTOMER tables in the MTPCH database. Write three different statements to display all information about every state that has at least one customer. The first statement should specify EXISTS; the second statement should specify IN; the third statement should specify a join-operation.

STCODE	STNAME	POPULATION	RNO
MA	MASSACHUSETTS	6450000	1
OR	OREGON	3747000	2
WA	WASHINGTON	6468000	2
FL	FLORIDA	18251000	3
GE	GEORGIA	9545000	3
NM	NEW MEXICO	1970000	4
AZ	ARIZONA	6339000	4

```
SELECT *
FROM   STATE ST
WHERE EXISTS (SELECT 'X' FROM CUSTOMER
                    WHERE STCODE = ST.STCODE)
```

```
SELECT *
FROM   STATE
WHERE STCODE IN (SELECT STCODE FROM CUSTOMER)
```

```
SELECT DISTINCT ST.STCODE, STNAME, POPULATION, RNO
FROM   STATE ST, CUSTOMER C
WHERE ST.STCODE = C.STCODE
```

25E. Reference the STATE and CUSTOMER tables in the MTPCH database. Write two different statements to display all information about every state that does not have any customers. The first statement should specify NOT EXISTS, and the second statement should specify NOT IN.

STCODE	STNAME	POPULATION	RNO
CT	CONNECTICUT	3502000	1

```
SELECT *
FROM   STATE ST
WHERE NOT EXISTS (SELECT 'X' FROM CUSTOMER
                        WHERE STCODE = ST.STCODE)
```

```
SELECT *
FROM   STATE
WHERE STCODE NOT IN (SELECT STCODE FROM CUSTOMER)
```

 Copyright Pending 2020

25F. Optional Exercise: Write the ancient history solution for a full outer-join of the DEPARTMENT and EMPLOYEE3 tables.

```
SELECT D.DNO, DNAME, BUDGET, ENO, ENAME, SALARY, E.DNO
FROM   DEPARTMENT D, EMPLOYEE3 E
WHERE D.DNO = E.DNO
  UNION ALL
SELECT DNO, DNAME, BUDGET, '0', 'No Emp',  0, 0
FROM   DEPARTMENT DX
WHERE NOT EXISTS (SELECT 'X'
                  FROM EMPLOYEE3
                  WHERE DNO = DX.DNO)
  UNION ALL
SELECT 0, 'No Dept', 0, ENO, ENAME, SALARY, DNO
FROM   EMPLOYEE3 EX
WHERE NOT EXISTS (SELECT 'X'
                  FROM DEPARTMENT
                  WHERE DNO = EX.DNO)
ORDER BY 1
```

DNO	DNAME	BUDGET	ENO	ENAME	SALARY	DNO
0	No Dept	0.00	1000	MOE	2000.00	99
0	No Dept	0.00	6000	GEORGE	9000.00	-
10	ACCOUNTING	75000.00	2000	LARRY	2000.00	10
10	ACCOUNTING	75000.00	5000	JOE	400.00	10
20	INFO. SYS.	20000.00	3000	CURLY	3000.00	20
30	PRODUCTION	7000.00	0	No Emp	0.00	0
40	ENGINEERING	25000.00	4000	SHEMP	500.00	40

Summary Exercises (Chapter 25)

Code solutions that specify correlated Sub-SELECTs unless directed otherwise.

25G. Reference the PRESERVE table. Determine the largest preserve (greatest number of acres) in each state. Display the state code followed the preserve number, name, and acreage.

```
SELECT STATE, PNO, PNAME, ACRES
FROM   PRESERVE P
WHERE ACRES = ( SELECT MAX (ACRES)
                FROM   PRESERVE
                WHERE STATE = P.STATE)
```

```
STATE  PNO  PNAME             ACRES
AZ       7  MULESHOE RANCH    49120
MA       9  DAVID H. SMITH      830
MT       2  PINE BUTTE SWAMP  15000
```

25H. Code an alternative solution to the preceding Exercise 25G. Specify a regular Sub-SELECT that returns multiple columns. Hint: Review Exercise 23.14. [Skip this exercise if your system does not allow regular Sub-SELECTs to return multiple columns.]

```
SELECT STATE, PNO, PNAME, ACRES
FROM PRESERVE
WHERE (STATE, ACRES)  IN     ( SELECT STATE, MAX (ACRES)
                               FROM PRESERVE
                               GROUP BY STATE)
```

```
STATE  PNO  PNAME             ACRES
AZ       7  MULESHOE RANCH    49120
MA       9  DAVID H. SMITH      830
MT       2  PINE BUTTE SWAMP  15000
```

 Copyright Pending 2020

25I. Reference the PARTSUPP table in the MTPCH database. The basic objective is to display information about each part having a price that is less than the average price for the part. Specifically, for every part that you can purchase from some supplier, display the PNO, SNO, and PSPRICE values for any part having a price that is less than the average price for the part. Sort the result by SNO within PNO.

```
SELECT PNO, SNO, PSPRICE
FROM   PARTSUPP PSX
WHERE PSPRICE < (SELECT AVG (PSPRICE)
                        FROM PARTSUPP
                        WHERE PNO = PSX.PNO)
ORDER BY PNO, SNO
```

PNO	SNO	PSPRICE
P1	S2	10.50
P3	S3	12.00
P5	S1	10.00
P5	S2	10.00
P7	S2	2.00
P8	S8	3.00

25J. Reference the PARTSUPP and SUPPLIER tables in the MTPCH database. Modify the above Exercise 25I to include the name of the supplier.

```
SELECT PSX.PNO, PSX.SNO, S.SNAME, PSX.PSPRICE
FROM   PARTSUPP PSX, SUPPLIER S
WHERE PSX.SNO = S.SNO
AND    PSX.PSPRICE < (SELECT AVG (PSPRICE)
                             FROM PARTSUPP
                             WHERE PNO = PSX.PNO)
ORDER BY PSX.PNO, PSX.SNO
```

PNO	SNO	SNAME	PSPRICE
P1	S2	SUPPLIER2	10.50
P3	S3	SUPPLIER3	12.00
P5	S1	SUPPLIER1	10.00
P5	S2	SUPPLIER2	10.00
P7	S2	SUPPLIER2	2.00
P8	S8	SUPPLIER8	3.00

 Copyright Pending 2020

25K. Sample Query 21.3 specified an INTERSECT operation (shown below) to display the employee numbers and names of all persons who are described in both the EMPLOYEE and PROJMGR tables.

```
SELECT ENO, ENAME
FROM   EMPLOYEE
INTERSECT
SELECT ENO, PMNAME
FROM   PROJMGR
ORDER BY 1
```

a. Code an alternative solution using EXISTS.

```
SELECT ENO, ENAME
FROM    EMPLOYEE E
WHERE EXISTS
                (SELECT 'X' FROM PROJMGR WHERE ENO = E.ENO)
ORDER BY 1
```

b. Code another alternative solution using IN.

```
SELECT ENO, ENAME
FROM    EMPLOYEE
WHERE  ENO IN (SELECT ENO FROM PROJMGR)
ORDER BY 1
```

25L. Sample Query 21.4 specified an EXCEPT operation (shown below) to display the employee number and name of every employee who is not a project manager.

```
SELECT ENO, ENAME
FROM   EMPLOYEE
EXCEPT
SELECT ENO, PMNAME
FROM   PROJMGR
ORDER BY 1
```

a. Code an alternative solution using NOT EXISTS.

```
SELECT ENO, ENAME
FROM   EMPLOYEE E
WHERE NOT EXISTS
        (SELECT 'X' FROM PROJMGR WHERE ENO = E.ENO)
ORDER BY 1
```

b. Code another alternative solution using NOT IN.

```
SELECT ENO, ENAME
FROM   EMPLOYEE
WHERE ENO NOT IN (SELECT ENO FROM PROJMGR)
ORDER BY 1
```

c. Important Question: How do you know that, in this circumstance, the NOT EXISTS and NOT IN solutions are equivalent to each other?

For the NOT IN solution, the Sub-SELECT cannot return a null value because ENO is a primary-key that cannot contain null values.

25M1. Reference the DEPARTMENT and EMPLOYEE tables. Assume that management is considering adjusting each department's budget. Each new departmental budget might be changed to twice the total salary of all employees who work in the department. Before implementing this change, management asks you to produce a report that displays each department's number, name, current budget, and the proposed new budget. If a department does not have any employees, then display a null value for the proposed new budget. The result should look like:

DNO	DNAME	BUDGET	NEWBUDGET
10	ACCOUNTING	75000.00	4800.00
20	INFO. SYS.	20000.00	28000.00
30	PRODUCTION	7000.00	-
40	ENGINEERING	25000.00	1000.00

Your solution should specify a correlated Sub-SELECT within the SELECT-clause as shown in Sample Query 25.8. (The following Exercise 25M2 suggests an alternative solution.)

```
SELECT D.DNO, D.DNAME, D.BUDGET,
                      (SELECT 2.00 * SUM (SALARY)
                       FROM EMPLOYEE
                       WHERE DNO=D.DNO)  NEWBUDGET
FROM DEPARTMENT D
```

25M2. This is an optional exercise. Code an alternative solution for the preceding Exercise 25M1. Instead of coding a Sub-SELECT, your solution should specify a left outer-join operation and group by the DNO, DNAME, and BUDGET columns.

```
SELECT D.DNO, DNAME, BUDGET, 2.00 * SUM (SALARY) NEWBUDGET
FROM    DEPARTMENT D LEFT OUTER JOIN EMPLOYEE E
ON      D.DNO = E.DNO
GROUP BY D.DNO, DNAME, BUDGET
```

 Copyright Pending 2020

25N. Exercise 23I asked you to code a regular Sub-SELECT to satisfy the query objective: Reference the DEPARTMENT and EMPLLOYEE tables. Display the overall total budget of those departments which have at least one employee. Code another solution using a correlated Sub-SELECT. The result should look like:

```
TOTBUDGET
120000.00
```

```
SELECT SUM (BUDGET) TOTBUDGET
FROM   DEPARTMENT D
WHERE EXISTS       (SELECT 'X' FROM EMPLOYEE
                     WHERE DNO = D.DNO)
```

 Copyright Pending 2020

25O. This exercise modifies Exercise 25M1. The user does not want to see any null values in the report. Therefore, if a department does not have any employees, the new budget should be the same as the current budget. The result should look like:

```
DNO   DNAME          BUDGET       NEWBUDGET
 10   ACCOUNTING   75000.00         4800.00
 20   INFO. SYS.   20000.00        28000.00
 30   PRODUCTION    7000.00         7000.00 ←
 40   ENGINEERING  25000.00         1000.00
```

Code two solutions, each having the same basic structure as the solution for Exercise 25M1.

The first should use the COALESCE function to substitute the current BUDGET value for a null value in the NEWBUDGET column. The basic structure of the SELECT-clause is:

```
SELECT . . . COALESCE ((correlated Sub-SELECT...),
                                BUDGET) NEWBUDGET
```

Solition-1

```
SELECT DNO, DNAME, BUDGET,
COALESCE ((SELECT 2.00 * SUM (SALARY)
              FROM EMPLOYEE
               WHERE DNO=D.DNO), BUDGET ) NEWBUDGET
FROM DEPARTMENT D
```

The second solution should specify a CASE-expression to substitute the current BUDGET value for a null value in the NEWBUDGET column. The basic structure of the CASE-expression is:

```
CASE (SELECT COUNT(*) FROM EMPLOYEE
                  WHERE DNO=D.DNO)
      WHEN 0 THEN . . .
      ELSE (correlated Sub-SELECT . . .)
END NEWBUDGET
```

Solition-2

```
SELECT D.DNO,D. DNAME, D.BUDGET,
   CASE (SELECT COUNT (*) FROM EMPLOYEE WHERE DNO=D.DNO)
      WHEN 0 THEN BUDGET
      ELSE  (SELECT 2.00 * SUM (SALARY) FROM EMPLOYEE
            WHERE DNO=D.DNO)
   END NEWBUDGET
FROM DEPARTMENT D
```

25P. Review Exercise: Same as for Sample Query 25.7. Reference the EMPLOYEE table. Display all information about any employee whose salary is unique. This means that no other employee earns the same salary.

Do not specify a correlated Sub-SELECT. Code a regular Sub-SELECT that joins the EMPLOYEE table with itself to return ENO values of any employee who has the same salary as another employee.

ENO	ENAME	SALARY	DNO
3000	CURLY	3000.00	20
4000	SHEMP	500.00	40
5000	JOE	400.00	10
6000	GEORGE	9000.00	20

```
SELECT *
FROM     EMPLOYEE
WHERE   ENO NOT IN (
                SELECT E1.ENO
                FROM     EMPLOYEE E1, EMPLOYEE E2
                WHERE E1.SALARY = E2.SALARY
                AND      E1.ENO <> E2.ENO )
```

25Q. Review Exercise: Same as for Sample Query 25.6. Reference the DEPARTMENT and EMPLOYEE3 tables. Display all information about any employee assigned to a department that is not represented in the DEPARTMENT table. (This includes any employee with a null DNO value.)

Code a roundabout solution that specifies NOT IN and UNION ALL.

ENO	ENAME	SALARY	DNO
6000	GEORGE	9000.00	-
1000	MOE	2000.00	99

```
SELECT *
FROM   EMPLOYEE3
WHERE DNO NOT IN  (SELECT DNO FROM DEPARTMENT)
   UNION ALL
SELECT *
FROM   EMPLOYEE3
WHERE DNO IS NULL
```

Chapter-26 – Inline Views

Solve the following exercises by coding inline views. These exercises reference the EMPLOYEE table.

26A. Determine the total salary for each department. Then display the largest of these totals. The result should look like:

```
LARGESTTOTAL
------------
    14000.00
```

```
SELECT MAX (TOTSAL) LARGESTTOTAL
FROM  (SELECT DNO, SUM (SALARY) TOTSAL
       FROM    EMPLOYEE
       GROUP BY DNO) AS TOTALS
```

26B. Determine the average salary for each department. Then display the smallest of these averages. The result should look like:

```
SMALLESTAVG
-----------
     500.00
```

```
SELECT MIN (AVGSAL) SMALLESTAVG
FROM  (SELECT DNO, AVG (SALARY) AVGSAL
       FROM    EMPLOYEE
       GROUP BY DNO) AS AVERAGES
```

26C. Display all information about the lowest paid employee in each department. The result should look like:

```
ENO  ENAME      SALARY   DNO
---- ---------- -------- ---
1000 MOE        2000.00   20
4000 SHEMP       500.00   40
5000 JOE         400.00   10
```

```
SELECT E.ENO, E.ENAME, E.SALARY, E.DNO
FROM EMPLOYEE E,
      (SELECT DNO, MIN (SALARY) AS MINSAL
      FROM    EMPLOYEE
      GROUP BY DNO) AS TMINS
WHERE  E.DNO    = TMINS.DNO
AND    E.SALARY = TMINS.MINSAL
```

 Copyright Pending 2020

26D. For each department, display all information about every departmental employee who has a salary that is greater than or equal to the average salary for the department. The result should look like:

ENO	ENAME	SALARY	DNO
2000	LARRY	2000.00	10
4000	SHEMP	500.00	40
6000	GEORGE	9000.00	20

```
SELECT E.ENO, E.ENAME, E.SALARY, E.DNO
FROM    EMPLOYEE E,
              (SELECT DNO, AVG (SALARY) AS AVGSAL
               FROM EMPLOYEE
              GROUP BY DNO) AS TAVGS
WHERE  E.DNO  = TAVGS.DNO
AND      E.SALARY >= TAVGS.AVGSAL
```

26E1. Will this SELECT statement (for Sample Query 26.4) work if two departments have the same minimal budget? Will it work if two departmental employees have the same maximum salary?

```
SELECT TMAXES.DNO MAXSALDEPT,  TMAXES.MAXSAL,
     DMIN.DNO MINBUDGETDEPT, DMIN.BUDGET MINBUDGET
FROM (SELECT DNO, MAX (SALARY) MAXSAL
         FROM EMPLOYEE
        GROUP BY DNO) AS TMAXES,
         (SELECT DNO, BUDGET
         FROM  DEPARTMENT
         WHERE BUDGET = (SELECT MIN (BUDGET)
                                 FROM DEPARTMENT)) AS DMIN
WHERE TMAXES.MAXSAL > DMIN.BUDGET
```

Yes and Yes.

 Copyright Pending 2020

26E2. Reference the PARTSUPP and LINEITEM tables. For each part sold, the actual selling price (LIPRICE) is always greater than or equal to the part's purchase price (PSPRICE). Hence, a part's average selling price is always greater than or equal to its average purchase price. Display information about any part where the difference between these averages is less than 75 cents. For any such part, display its part number followed by its average purchase price and average selling price. The result should look like:

PNO	AVGPS	AVGLI
P7	3.00	3.50

```
SELECT PS.PNO, PS.AVGPS, LI.AVGLI
FROM    (SELECT PNO, AVG (PSPRICE) AVGPS
              FROM PARTSUPP
              GROUP BY PNO) PS,
          (SELECT PNO, AVG (LIPRICE) AVGLI
              FROM LINEITEM
              GROUP BY PNO) LI
WHERE PS.PNO = LI.PNO
AND     LI.AVGLI - PS.AVGPS < 0.75
```

 Copyright Pending 2020

26F. Reference the EMPLOYEE table. Display the department number and total salary of the department having the largest total salary. The result should look like:

```
DNO    LARGESTTOTAL
 20         14000.00
```

Observe that the following solution specifies the same Sub-SELECT in two locations. This redundancy is not desirable. Exercise 27F will illustrate a better solution using the WITH-clause.

```
SELECT DSUMS1.DNO, DSUMS1.DSUM LARGESTTOTAL
FROM        (SELECT DNO, SUM (SALARY) DSUM
             FROM    EMPLOYEE
             GROUP BY DNO) AS DSUMS1
WHERE DSUMS1.DSUM =
             (SELECT MAX (DSUM)
              FROM (SELECT DNO, SUM (SALARY) DSUM
                    FROM    EMPLOYEE
                    GROUP BY DNO) AS DSUMS2)
```

 Copyright Pending 2020

Summary Exercises: (Chapter 26)

26G. Reference the EMPLOYEE table. Determine the average salary in each department. Then display the largest of these averages. The result should look like:

```
MAXAVGSAL
  4666.66
```

```
SELECT MAX (TAVGS.AVGSAL)  MAXAVGSAL
FROM  (SELECT  DNO, AVG (SALARY) AVGSAL
         FROM     EMPLOYEE
         GROUP BY DNO) AS TAVGS
```

26H. Reference the PRESERVE table. For each state, display the state code and preserve's number, name, acreage for every preserve that is larger than the average preserve acreage for the state. The result should look like:

```
STATE  PNO  PNAME                ACRES
AZ       7  MULESHOE RANCH       49120
MA       9  DAVID H. SMITH         830
MA      12  MOUNT PLANTAIN         730
MT       2  PINE BUTTE SWAMP     15000
```

```
SELECT P.STATE, P.PNO, P.PNAME, P.ACRES
FROM PRESERVE P,
            (SELECT STATE, AVG (ACRES) AS AVGACRES
             FROM   PRESERVE
            GROUP BY STATE) AS AVGS
WHERE P.STATE = AVGS.STATE
AND     P.ACRES > AVGS.AVGACRES
```

 Copyright Pending 2020

26I. This exercise has the same query objective as Exercise 25M1. Your solution should specify an inline view.

Reference the DEPARTMENT and EMPLOYEE tables. Assume that management is considering adjusting each department's budget. Each new departmental budget might be changed to twice the total salary of all employees who work in the department. Before implementing this change, management asks you to produce a report that displays each department's number, name, current budget, and the proposed new budget. If a department does not have any employees, then display a null value for the proposed new budget. The result should look like:

DNO	DNAME	BUDGET	NEWDBUDGET
10	ACCOUNTING	75000.00	4800.00
20	INFO. SYS.	20000.00	28000.00
30	PRODUCTION	7000.00	-
40	ENGINEERING	25000.00	1000.00

```
SELECT D.DNO, D.DNAME, D.BUDGET,
          2.00 * TEMP.DTOTSAL NEWBUDGET
FROM   DEPARTMENT D LEFT OUTER JOIN
                (SELECT DNO, SUM (SALARY) DTOTSAL
                 FROM EMPLOYEE
                GROUP BY DNO) TEMP
ON D.DNO = TEMP.DNO
```

 Copyright Pending 2020

26J. This exercise modifies the preceding Exercise 26I. (It also has same query objective as Exercise 25N.) The user does not want to see any null values in the report. Therefore, if a department does not have any employees, the new budget should be the same as the current budget. The result should look like:

DNO	DNAME	BUDGET	NEWDBUDGET
10	ACCOUNTING	75000.00	4800.00
20	INFO. SYS.	20000.00	28000.00
30	PRODUCTION	7000.00	**7000.00**
40	ENGINEERING	25000.00	1000.00

Code two solutions which specify inline views.

The first solution should use the COALESCE function to substitute the current BUDGET value for a null value in the NEWBUDGET column.

Solition-1

```
SELECT D.DNO, DNAME, BUDGET,
        COALESCE ((2.00 * DTOTSAL), BUDGET) NEWBUDGET
FROM DEPARTMENT D LEFT OUTER JOIN
                (SELECT DNO, SUM (SALARY) DTOTSAL
                 FROM EMPLOYEE
                 GROUP BY DNO) TEMP
ON D.DNO = TEMP.DNO
```

The second solution should specify a CASE-expression to substitute the current BUDGET value for a null value in the NEWBUDGET column.

Solution-2

```
SELECT D.DNO, DNAME, BUDGET,
       CASE
            WHEN DTOTSAL IS NULL THEN BUDGET
            ELSE 2.00 * DTOTSAL
       END NEWBUDGET
FROM DEPARTMENT D LEFT OUTER JOIN
                (SELECT DNO, SUM (SALARY) DTOTSAL
                 FROM EMPLOYEE
                 GROUP BY DNO) TEMP
ON D.DNO = TEMP.DNO
```

26K. Extend Sample Query 26.3. (Display all information about the highest paid employee in each department that has employees.) Also display the department name along with the department number. The result should look like:

```
ENO   ENAME          SALARY  DNO   DNAME
2000  LARRY         2000.00   10   ACCOUNTING
4000  SHEMP          500.00   40   ENGINEERING
6000  GEORGE        9000.00   20   INFO. SYS.
```

```
SELECT E.ENO, E.ENAME, E.SALARY,     E.DNO, D.DNAME
FROM   EMPLOYEE E, DEPARTMENT D,
                (SELECT DNO, MAX (SALARY) AS MAXSAL
                 FROM EMPLOYEE
                 GROUP BY DNO) AS TMAXES
WHERE E.DNO       = TMAXES.DNO
AND     E.SALARY = TMAXES.MAXSAL
AND     E.DNO       = D.DNO
```

26L. Same query objective as Exercise 23S. Consider changing each DEPARTMENT.BUDGET value to a value that is equal to the largest BUDGET value minus 10% of the department's current BUDGET value. Display each department number, name, current budget, and the adjusted budget. (Hint: Review Sample Query 26.5.) The result should look like:

```
DNO   DNAME              BUDGET   ADJBUDGET
10    ACCOUNTING      75000.00    67500.00
20    INFO. SYS.      20000.00    73000.00
30    PRODUCTION       7000.00    74300.00
40    ENGINEERING     25000.00    72500.00
```

```
SELECT D.DNO, D.DNAME, D.BUDGET,
          TEMP.MAXBUD - (.10*D.BUDGET) ADJBUDGET
FROM DEPARTMENT D,
                (SELECT MAX (BUDGET) MAXBUD
                 FROM DEPARTMENT) AS TEMP
```

26M. Reference the PARTSUPP and LINEITEM tables. For each part, display its part number, its largest purchase price, and its lowest selling price, if this largest purchase price is greater than its lowest selling price. The result should look like:

PNO	MAXPAID	MINSOLD
P3	12.50	12.00
P7	3.50	3.00
P8	5.00	4.00

Hint: Specify two dynamic views that look like:

BOUGHT

PNO	MAXPS
P1	11.00
P3	12.50
P4	12.00
P5	11.00
P6	4.00
P7	3.50
P8	5.00

SOLD

PNO	MINLI
P1	11.50
P3	12.00
P4	13.00
P5	11.00
P6	5.00
P7	3.00
P8	4.00

```
SELECT BOUGHT.PNO, BOUGHT.MAXPS  MAXPAID,
       SOLD.MINLI   MINSOLD
FROM        (SELECT PNO, MAX (PSPRICE) MAXPS
              FROM PARTSUPP
             GROUP BY PNO) BOUGHT,
            (SELECT PNO, MIN (LIPRICE) MINLI
              FROM LINEITEM
              GROUP BY PNO) SOLD
WHERE BOUGHT.PNO = SOLD.PNO
AND     BOUGHT.MAXPS > SOLD.MINLI
```

 Copyright Pending 2020

26N. Specify a dynamic view to satisfy Sample Query 25.8. Reference the EMPLOYEE table. Consider adjusting each employee's salary to a value that is equal to the employee's *departmental* average salary plus 5% of the employee's current salary. Display each employee number, name, and current salary, followed by the adjusted salary. The result should look like:

ENO	ENAME	SALARY	ADJUSTEDSALARY
1000	MOE	2000.00	4766.66
2000	LARRY	2000.00	1300.00
3000	CURLY	3000.00	4816.66
4000	SHEMP	500.00	525.00
5000	JOE	400.00	1220.00
6000	GEORGE	9000.00	5116.66

```
SELECT E.ENO, E.ENAME, E.SALARY,
          TEMP.AVGSAL + (.05* E.SALARY) ADJUSTEDSALARY
FROM EMPLOYEE E,
                (SELECT DNO, AVG (SALARY) AVGSAL
                 FROM EMPLOYEE
                 GROUP BY DNO) TEMP
WHERE E.DNO = TEMP.DNO
```

 Copyright Pending 2020

26O. Use an inline view to enhance Sample Query 23.16. Reference the EMPLOYEE table. Consider the impact of adjusting each employee's salary to a value that is equal to the overall average of all current salaries plus 5% of the employee's current salary. Display each employee number, name, current salary, and adjusted salary. Also, display a narrative label "SALARY INCREASED" or "SALARY DECREASED" or "NO CHANGE" in the last column in result table. The result should look like:

ENO	ENAME	SALARY	ADJSAL	NARRATIVE
1000	MOE	2000.00	2916.66	SALARY INCREASED
2000	LARRY	2000.00	2916.66	SALARY INCREASED
3000	CURLY	3000.00	2966.66	SALARY DECREASED
4000	SHEMP	500.00	2841.66	SALARY INCREASED
5000	JOE	400.00	2836.66	SALARY INCREASED
6000	GEORGE	9000.00	3266.66	SALARY DECREASED

```
SELECT E.ENO, E.ENAME, E.SALARY,
        TEMP.AVGSAL+ (.05*SALARY) ADJSAL,
        CASE
            WHEN SALARY < TEMP.AVGSAL+ (.05*SALARY)
                THEN 'SALARY INCREASED'
            WHEN SALARY > TEMP.AVGSAL+ (.05*SALARY)
                THEN 'SALARY DECREASED'
            ELSE   'NO CHANGE'
      END NARRATIVE
FROM EMPLOYEE E,
      (SELECT AVG (SALARY) AVGSAL FROM EMPLOYEE) AS TEMP
```

26P. Reference the PRESERVE table. For each row, if its FEE value is not zero, calculate the ratio of ACRES divided by FEE. Display the preserve name and ratio if its ratio is greater than 200. The result should look like:

PNAME	RATIO
HASSAYAMPA RIVER	220.00
PAPAGONIA-SONOITA CREEK	400.00

Review the page after Sample Query 7.6 and Exercise 23Zi.Your solution should specify a dynamic view.

```
SELECT NOZERO.PNAME, NOZERO.RATIO
FROM        (SELECT PNAME, ACRES/FEE RATIO
            FROM PRESERVE
            WHERE FEE <>0) NOZERO
WHERE NOZERO.RATIO > 200.00
```

26Q. Code an alternative solution for Sample Query 23.11. Do not display information about any employee with a SALARY value of 2000.00. For other employees, display the ENO, ENAME, SALARY, and ratio of SALARY/(SALARY–2000.00) if this ratio is greater than or equal to 2.00. (Notice that, when a SALARY value equals 2000.00, we have a divide-by-zero problem.)

ENO	ENAME	SALARY	RATIO
3000	CURLY	3000.00	3.00

```
SELECT E.ENO, E.ENAME, E.SALARY,
       E.SALARY/(E.SALARY - 2000.00) RATIO
FROM (SELECT ENO, ENAME, SALARY
      FROM EMPLOYEE
      WHERE SALARY <> 2000.00) E
WHERE E.SALARY/(E.SALARY - 2000.00) >= 2.00
```

 Copyright Pending 2020

Chapter-27 - WITH-Clause: Common Table Expressions

The following exercises have the same query objectives as Exercises 26A-26D. Solve by coding WITH-clauses. These exercises reference the EMPLOYEE table.

27A. Determine the total salary for each department. Then display the largest of these totals. The result should look like:

```
LARGESTTOTAL
    14000.00
```

```
WITH TOTALS
AS
  (SELECT DNO, SUM (SALARY) TOTSAL
   FROM   EMPLOYEE
  GROUP BY DNO)
SELECT MAX (TOTSAL) LARGSETTOTAL
FROM    TOTALS
```

Alternative: Specify column-names in WITH-Clause:

```
WITH TOTALS  (DNO, TOTSAL)
AS
  (SELECT DNO, SUM (SALARY)
   FROM   EMPLOYEE
  GROUP BY DNO)
SELECT MAX (TOTSAL) LARGSETTOTAL
FROM    TOTALS
```

27B. Determine the average salary for each department. Then display the smallest of these averages. The result should look like:

```
SMALLESTAVG
     500.00
```

```
WITH AVERAGES
AS
  (SELECT   DNO, AVG (SALARY) AVGSAL
   FROM     EMPLOYEE
   GROUP BY DNO)
SELECT MIN (AVGSAL)  SMALLESTAVG
FROM   AVERAGES
```

 Copyright Pending 2020

27C. Display all information about the lowest paid employee in each department. The result should look like:

ENO	ENAME	SALARY	DNO
1000	MOE	2000.00	20
4000	SHEMP	500.00	40
5000	JOE	400.00	10

```
WITH TMINS
AS
      (SELECT DNO, MIN (SALARY) AS MINSAL
      FROM EMPLOYEE
      GROUP BY DNO)
SELECT E.ENO,E.ENAME, E.SALARY, E.DNO
FROM   EMPLOYEE E, TMINS
WHERE  E.DNO      = TMINS.DNO
AND    E.SALARY = TMINS.MINSAL
```

27D. For each department, display all information about every departmental employee who has a salary that is greater than or equal to the average salary for the department. The result should look like:

ENO	ENAME	SALARY	DNO
2000	LARRY	2000.00	10
4000	SHEMP	500.00	40
6000	GEORGE	9000.00	20

```
WITH TAVGS
AS
      (SELECT DNO, AVG (SALARY) AVGSAL
      FROM EMPLOYEE
      GROUP BY DNO)
SELECT E.ENO, E.ENAME, E.SALARY, E.DNO
FROM   EMPLOYEE E, TAVGS
WHERE E.DNO       = TAVGS.DNO
AND    E.SALARY >= TAVGS.AVGSAL
```

 Copyright Pending 2020

27E. Same as Exercise 26E2. Reference the PARTSUPP and LINEITEM tables in the MTPC database. For each part sold, the selling price (LIPRICE) is always greater than or equal to the part's purchase price (PSPRICE). Hence, a part's average selling price will always be greater than or equal to its average purchase price. We want to display information about any part where the difference between these averages is less than 75 cents. For any such part, display the part number followed by its average purchase price and average selling price .

```
PNO  AVGPS  AVGLI
P7     3.00   3.50
```

```
WITH
PS (PNO, AVGPS) AS
   (SELECT PNO, AVG (PSPRICE)
    FROM PARTSUPP
    GROUP BY PNO),
LI (PNO, AVGLI) AS
    (SELECT PNO, AVG (LIPRICE)
     FROM LINEITEM
     GROUP BY PNO)
SELECT PS.PNO, PS.AVGPS, LI.AVGLI
FROM PS,LI
WHERE PS.PNO = LI.PNO
AND   LI.AVGLI - PS.AVGPS < 0.75
```

27F. Same as Exercise 26F: Display the department number and total salary of the department having the largest total salary. The result should look like:

```
DNO    LARGESTTOTAL
20          14000.00
```

```
WITH DSUMS (DNO, DEPTSUM)
AS
        (SELECT DNO, SUM (SALARY)
         FROM   EMPLOYEE
         GROUP BY DNO)
SELECT DNO, DEPTSUM LARGESTTOTAL
FROM    DSUMS
WHERE DEPTSUM = (SELECT MAX (DEPTSUM) FROM DSUMS)
```

 Copyright Pending 2020

Summary Exercises (Chapter 27)

The following Exercises 27G-27O have the same query objectives as Exercises 26G-26O. Utilize the WITH-clause to satisfy these query objectives.

27G. Reference the EMPLOYEE table. Determine the average salary in each department. Then display the largest of these averages. The result should look like:

```
MAXAVGSAL
 4666.66
```

```
WITH TAVGS AS
        (SELECT DNO, AVG (SALARY) AVGSAL
         FROM   EMPLOYEE
         GROUP BY DNO)
SELECT MAX (AVGSAL) MAXAVGSAL
FROM TAVGS
```

27H. Reference the PRESERVE table. For each state, display the state code and preserve's number, name, acreage for every preserve that is larger than the average preserve acreage for the state. The result should look like:

STATE	PNO	PNAME	ACRES
AZ	7	MULESHOE RANCH	49120
MA	9	DAVID H. SMITH	830
MA	12	MOUNT PLANTAIN	730
MT	2	PINE BUTTE SWAMP	15000

```
WITH AVGS AS
        (SELECT STATE, AVG (ACRES) AS AVGACRES
        FROM PRESERVE
         GROUP BY STATE)
SELECT P.STATE, P.PNO, P.PNAME, P.ACRES
FROM   PRESERVE P, AVGS
WHERE P.STATE = AVGS.STATE
AND     P.ACRES > AVGS.AVGACRES
```

27I. This exercise has the same query objective as Exercises 25M1 and 26I. Reference the DEPARTMENT and EMPLOYEE tables. Assume that management is considering adjusting each department's budget. Each new departmental budget might be changed to twice the total salary of all employees who work in the department. Before implementing this change, management asks you to produce a report that displays each department's number, name, current budget, and the proposed new budget. If a department does not have any employees, then display a null value for the proposed new budget. The result should look like:

DNO	DNAME	BUDGET	NEWBUDGET
10	ACCOUNTING	75000.00	4800.00
20	INFO. SYS.	20000.00	28000.00
40	ENGINEERING	25000.00	1000.00
30	PRODUCTION	7000.00	-

```
WITH TEMP AS
        (SELECT DNO, SUM (SALARY) DTOTSAL
         FROM EMPLOYEE
         GROUP BY DNO)
SELECT D.DNO, D.DNAME, D.BUDGET, 2.00 * DTOTSAL NEWBUDGET
FROM   DEPARTMENT D LEFT OUTER JOIN TEMP
ON       D.DNO = TEMP.DNO
```

 Copyright Pending 2020

27J. This exercise modifies the preceding Exercise 27I. (It also has same query objective as Exercise 25N.) The user does not want to see any null values in the report. Therefore, if a department does not have any employees, the new budget should be the same as the current budget. The result should look like:

```
DNO  DNAME          BUDGET   NEWBUDGET
 10  ACCOUNTING   75000.00     4800.00
 20  INFO. SYS.   20000.00    28000.00
 40  ENGINEERING  25000.00     1000.00
 30  PRODUCTION    7000.00     7000.00
```

Code two solutions which specify WITH-clauses.

The first solution should use the COALESCE function to substitute the current BUDGET value for a null value in the NEWBUDGET column.

Solition-1

```
WITH TEMP AS
        (SELECT DNO, SUM (SALARY) DTOTSAL
           FROM EMPLOYEE
           GROUP BY DNO)
SELECT D.DNO, D.DNAME, D.BUDGET,
        COALESCE ((2.00 * DTOTSAL), D.BUDGET) NEWBUDGET
FROM DEPARTMENT D LEFT OUTER JOIN TEMP
ON      D.DNO = TEMP.DNO
```

The second solution should specify a CASE-expression to substitute the current BUDGET value for a null value in the NEWBUDGET column.

Solution-2

```
WITH TEMP AS
        (SELECT DNO, SUM (SALARY) DTOTSAL
         FROM EMPLOYEE
         GROUP BY DNO)
SELECT D.DNO, D.DNAME, D.BUDGET,
        CASE
                WHEN DTOTSAL IS NULL THEN BUDGET
                ELSE    DTOTSAL * 2.00
        END NEWBUDGET
FROM DEPARTMENT D LEFT OUTER JOIN TEMP
ON      D.DNO = TEMP.DNO
```

 Copyright Pending 2020

27K. Extend Sample Query 27.3. (Display all information about the highest paid employee in each department that has at least one employee.) Also display the department name along with the department number. The result should look like:

ENO	ENAME	SALARY	DNO	DNAME
2000	LARRY	2000.00	10	ACCOUNTING
4000	SHEMP	500.00	40	ENGINEERING
6000	GEORGE	9000.00	20	INFO. SYS.

```
WITH TMAXES AS
        (SELECT DNO, MAX (SALARY) AS MAXSAL
         FROM EMPLOYEE
         GROUP BY DNO)
SELECT E.ENO, E.ENAME, E.SALARY, E.DNO, D.DNAME
FROM   EMPLOYEE E,  DEPARTMENT D,  TMAXES
WHERE E.DNO = TMAXES.DNO
AND    E.SALARY = TMAXES.MAXSAL
AND    E.DNO = D.DNO
```

27L. Consider changing each DEPARTMENT.BUDGET value to a value that is equal to the largest BUDGET value minus 10% of the department's current BUDGET value. Display each department number, name, current budget, and the adjusted budget. (Hint: Review Sample Query 27.5) The result should look like:

DNO	DNAME	BUDGET	ADJBUDGET
10	ACCOUNTING	75000.00	67500.00
20	INFO. SYS.	20000.00	73000.00
30	PRODUCTION	7000.00	74300.00
40	ENGINEERING	25000.00	72500.00

```
WITH TEMP (MAXBUD) AS
        (SELECT MAX (BUDGET) FROM DEPARTMENT)
SELECT D.DNO, D.DNAME, D.BUDGET,
        TEMP.MAXBUD - (.10*D.BUDGET) ADJBUDGET
FROM DEPARTMENT D, TEMP
```

27M. Reference the PARTSUPP and LINEITEM tables. For each part, display its part number, its largest purchase price, and its lowest selling price, if this largest purchase price is greater than its lowest selling price. The result should look like:

PNO	MAXPAID	MINSOLD
P3	12.50	12.00
P7	3.50	3.00
P8	5.00	4.00

Hint: Specify two common table expressions for the following two tables that look like:

BOUGHT	
PNO	MAXPS
P1	11.00
P3	12.50
P4	12.00
P5	11.00
P6	4.00
P7	3.50
P8	5.00

SOLD	
PNO	MINLI
P1	11.50
P3	12.00
P4	13.00
P5	11.00
P6	5.00
P7	3.00
P8	4.00

```
WITH
BOUGHT AS
        (SELECT PNO, MAX (PSPRICE) MAXPS
        FROM PARTSUPP
        GROUP BY PNO),
SOLD AS
        (SELECT PNO, MIN(LIPRICE) MINLI
        FROM LINEITEM
        GROUP BY PNO)
SELECT B.PNO, B.MAXPS MAXPAID, S.MINLI MINSOLD
FROM   BOUGHT B, SOLD S
WHERE B.PNO = S.PNO
AND     B.MAXPS > S.MINLI
```

 Copyright Pending 2020

27N. Specify a WITH-clause to satisfy Sample Query 25.8. Reference the EMPLOYEE table. Consider adjusting each employee's salary to a value that is equal to the employee's *departmental* average salary plus 5% of the employee's current salary. Display each employee number, name, and current salary, followed by the adjusted salary. The result should look like:

```
ENO   ENAME       SALARY ADJUSTEDSALARY
1000 MOE        2000.00         4766.66
2000 LARRY      2000.00         1300.00
3000 CURLY      3000.00         4816.66
4000 SHEMP       500.00          525.00
5000 JOE         400.00         1220.00
6000 GEORGE     9000.00         5116.66
```

```
WITH TEMP (DNO, AVGSAL) AS
        (SELECT DNO, AVG (SALARY)
         FROM EMPLOYEE
        GROUP BY DNO)
SELECT E.ENO, E.ENAME, E.SALARY,
          TEMP.AVGSAL + (.05* E.SALARY) ADJUSTEDSALARY
FROM    EMPLOYEE E, TEMP
WHERE E.DNO = TEMP.DNO
ORDER BY E.ENO
```

 Copyright Pending 2020

27O. Reference the EMPLOYEE table. Consider the impact of adjusting each employee's salary to a value that is equal to the overall average of all current salaries plus 5% of the employee's current salary. Display each employee number, name, current salary, and adjusted salary. Also, display a narrative label "SALARY INCREASED" or "SALARY DECREASED" or "NO CHANGE" in the last column in result table. The result should look like:

```
ENO   ENAME      SALARY     ADJSAL     NARRATIVE
1000  MOE        2000.00    2916.66    SALARY INCREASED
2000  LARRY      2000.00    2916.66    SALARY INCREASED
3000  CURLY      3000.00    2966.66    SALARY DECREASED
4000  SHEMP       500.00    2841.66    SALARY INCREASED
5000  JOE         400.00    2836.66    SALARY INCREASED
6000  GEORGE     9000.00    3266.66    SALARY DECREASED
```

```
WITH TEMP (AVGSAL) AS
        (SELECT AVG (SALARY) AVGSAL FROM EMPLOYEE)
SELECT E.ENO, E.ENAME, E.SALARY,
        TEMP.AVGSAL + (.05*E.SALARY) ADJSAL,
        CASE
                WHEN E.SALARY < TEMP.AVGSAL+ (.05*E.SALARY)
                        THEN 'SALARY INCREASED'
                WHEN E.SALARY > TEMP.AVGSAL+ (.05*E.SALARY)
                        THEN 'SALARY DECREASED'
                ELSE   'NO CHANGE'
        END NARRATIVE
FROM EMPLOYEE E, TEMP
```

27P. Reference the PRESERVE table. For each row, if its FEE value is not zero, calculate the ratio of ACRES divided by FEE. Display the preserve name and ratio if its ratio is greater than 200. The result should look like:

```
PNAME                        RATIO
HASSAYAMPA RIVER            220.00
PAPAGONIA-SONOITA CREEK     400.00
```

Review the page after Sample Query 7.6, Exercise 23Zi, and Exercise26P.Your solution should specify a common table expression.

.

```
WITH NOZERO
AS      (SELECT PNAME, ACRES/FEE RATIO
        FROM PRESERVE
        WHERE FEE <>0)
SELECT PNAME, RATIO
FROM   NOZERO
WHERE RATIO > 200.00
```

27Q. Code an alternative solution for Sample Query 23.11 (and Exercise 26Q). Do not display information about any employee with a SALARY value of 2000.00. For other employees, display the ENO, ENAME, SALARY, and ratio of SALARY/(SALARY–2000.00) if this ratio is greater than or equal to 2.00. (Notice that, when a SALARY value equals 2000.00, we have a divide-by-zero problem.)

ENO	ENAME	SALARY	RATIO
3000	CURLY	3000.00	3.00

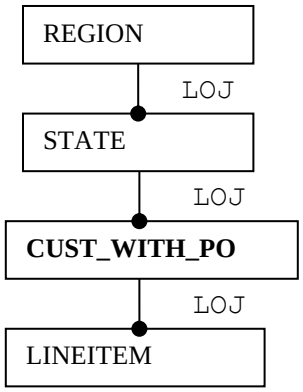
```
WITH NOZERO (ENO, ENAME, SALARY)
AS
        (SELECT ENO, ENAME, SALARY
         FROM    EMPLOYEE
         WHERE SALARY <> 2000.00)
SELECT ENO, ENAME, SALARY,
          SALARY/(SALARY - 2000.00) RATIO
FROM NOZERO
WHERE SALARY/(SALARY - 2000.00) >= 2.00
```

27R. Same as Sample Query 20.15: Display the following information about regions, states, customers, purchase-orders, and line-items.

- Display the region number and name of all regions, including regions without any states.
- Display the code and name for all states, including states without any customers.
- Display customer number and name for those customers that have at least one purchase-order.
- Display each customer's purchase-order numbers, including numbers for purchase-orders that do not have any line-items.
- Display each line-item's line-number and part-number values.

Specify a CTE called CUST_WITH_PO which executes an INNER JOIN to join the CUSTOMER and PUR_ORDER tables. Then the following code would represent a sequence of LEFT OUTER JOIN operations that traverse a four-level hierarchy.

```
FROM REGION R
    LEFT OUTER JOIN STATE ST
                ON R.RNO = ST.RNO
    LEFT OUTER JOIN CUST_WITH_PO CWPO
                ON ST.STCODE = CWPO.STCODE
    LEFT OUTER JOIN LINEITEM LI
                ON PO.PONO = LI.PONO
```

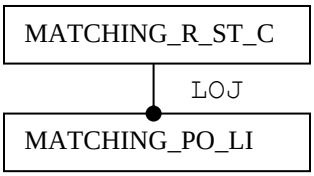



```
WITH CUST_WITH_PO (CNO, CNAME, STCODE, PONO)
AS
(SELECT C.CNO, C.CNAME, C.STCODE, PO.PONO
FROM CUSTOMER C INNER JOIN PUR_ORDER PO ON C.CNO = PO.CNO)

SELECT   R.RNO, R.RNAME, ST.STCODE, ST.STNAME,
         CWPO.CNO, CWPO.CNAME, CWPO.PONO, LI.LINE, LI.PNO
FROM REGION R
    LEFT OUTER JOIN STATE ST
          ON R.RNO = ST.RNO
    LEFT OUTER JOIN CUST_WITH_PO CWPO
              ON ST.STCODE = CWPO.STCODE
     LEFT OUTER JOIN LINEITEM LI
          ON CWPO.PONO = LI.PONO
ORDER BY R.RNO, ST.STCODE, CWPO.CNO, CWPO.PONO, LI.LINE
```

27S. Same as Sample Query 20.16: Display the following information about regions, states, customers, purchase-orders, and line-items.

- Display the region number of any region that has at least one state.
- Display the code of any state that has at least one customer.
- Display the number and name of all customers, including customers without purchase- orders.
- Display each customer's purchase-order numbers if the purchase-order has at least one line-item.
- Display the line-number and corresponding part-number of each line-item.

Think of the INNER JOIN operations forming two intermediate join-results in tables called MATCHING_R_ST_C and MATCHING_PO_LI. Then the following code would represent a sequence of LEFT OUTER JOIN operations that traverse a four-level hierarchy.




```
 FROM MATCHING_R_ST_C RSTC
            LEFT OUTER JOIN MATCHING_PO_LI POLI

            ON RSTC.CNO = POLI.CNO
```

```
WITH MATCHING_R_ST_C (RNO, RNAME, STCODE, STNAME, CNO, CNAME)
AS
(SELECT R.RNO, R.RNAME, ST.STCODE, ST.STNAME,  C.CNO, C.CNAME
  FROM REGION R INNER JOIN STATE ST
                              ON R.RNO = ST.RNO
                    INNER JOIN CUSTOMER C
                              ON ST.STCODE = C.STCODE),

  MATCHING_PO_LI (CNO, PONO, LINE, PNO)
  AS
  (SELECT PO.CNO, PO.PONO, LI.LINE, LI.PNO
  FROM   PUR_ORDER PO INNER JOIN LINEITEM LI
                              ON PO.PONO = LI.PONO)

SELECT RSTC.RNO, RSTC.RNAME, RSTC.STCODE, RSTC.STNAME,
     RSTC.CNO, RSTC.CNAME, POLI.PONO, POLI.LINE, POLI.PNO
FROM MATCHING_R_ST_C RSTC LEFT OUTER JOIN MATCHING_PO_LI POLI
                                        ON RSTC.CNO = POLI.CNO

ORDER BY RSTC.RNO, RSTC.STCODE, RSTC.CNO, POLI.PONO, POLI.LINE
```

Chapter-28 - CREATE VIEW Statement

Summary Exercises

Exercises 28A – 28F assume that the DEPTSTATSV and EMPDEPTV tables (views) already exist because Sample Statements 28.4 and 28.8 have been executed.

28A. Same query objective as Exercise 27A. Reference the DEPTSTATSV table. Determine the total salary for each department. Then display the largest of these totals. The result should look like:

```
LARGESTTOTAL
    14000.00
```

```
SELECT MAX (TOTALSAL) LARGESTTOTAL
FROM DEPTSTATSV
```

28B. Same query objective as Exercise 27C. Reference the EMPLOYEE and DEPTSTATSV tables. Display all information about the lowest paid employee in each department. The result should look like:

```
ENO   ENAME      SALARY   DNO
1000  MOE        2000.00   20
4000  SHEMP       500.00   40
5000  JOE         400.00   10
```

```
SELECT ENO, ENAME, SALARY, E.DNO
FROM   EMPLOYEE E, DEPTSTATSV  DV
WHERE  E.DNO = DV.DNO
AND    E.SALARY = DV.MINSAL
ORDER BY ENO
```

28C. Same query objective as Exercise 27E. Reference the DEPTSTATSV table. Display the department number and total salary of the department having the largest total salary. The result should look like:

```
DNO    LARGESTTOTAL
 20        14000.00
```

```
SELECT DNO, TOTALSAL LARGESTTOTAL
FROM   DEPTSTATSV
WHERE TOTALSAL =
      (SELECT MAX (TOTALSAL) FROM DEPTSTATSV)
```

 Copyright Pending 2020

28D. Expand upon the previous Exercise 28C. Reference the DEPARTMENT and DEPTSTATSV tables. Display the department name along with the department number. The result should look like:

DNO	DNAME	LARGESTTOTAL
20	INFO. SYS.	14000.00

```
SELECT DS.DNO, DNAME, TOTALSAL LARGESTTOTAL
FROM   DEPTSTATSV DS, DEPARTMENT D
WHERE DS.DNO = D.DNO
AND    TOTALSAL =
                    (SELECT MAX (TOTALSAL) FROM DEPTSTATSV)
```

28E. Reference the EMPDEPTV table. Display the employee number and name of any employee whose salary exceeds $2,000.00 and works in the Accounting Department. The result should look like:

ENO	ENAME
2000	LARRY

```
SELECT ENO, ENAME
FROM EMPDEPTV
WHERE SALARY > 1000.00 AND DNAME = 'ACCOUNTING'
```

28F. Reference the EMPDEPTV and DEPTSTATSV tables. For any department where the difference between the largest and smallest employee salaries exceeds $3,000.00, display the department name, followed by the name and salary of each of its employees. The result should look like:

DNAME	ENAME	SALARY
INFO. SYS.	MOE	2000.00
INFO. SYS.	CURLY	3000.00
INFO. SYS.	GEORGE	9000.00

```
SELECT E.DNAME, E.ENAME, E.SALARY
FROM   EMPDEPTV E, DEPTSTATSV D
WHERE E.DNO = D.DNO
AND     D.MAXSAL - D.MINSAL > 3000.00
```

 Copyright Pending 2020

28G. (a) The DEPTSTATSV view does not contain the average salary for each department. Create another view called DEPTSTATSV2 that contains the same data as DEPTSTATSV plus another column called AVGSAL that contains the average salary for each department.

```
CREATE VIEW DEPTSTATSV2
        (DNO, MAXSAL, MINSAL, TOTALSAL, AVGSAL)
AS
SELECT DNO, MAX (SALARY), MIN (SALARY),
                SUM (SALARY),  AVG (SALARY)
FROM EMPLOYEE
GROUP BY DNO
```

(c) Reference the above DEPTSTATSV2. Display the smallest of average departmental salary. The result should look like:

```
MINAVG
500.00
```

```
SELECT MIN (AVGSAL) MINAVG FROM DEPTSTATSV2
```

(c) Reference the EMPLOYEE and DEPTSTATSV2 tables. For each department, display all information about every departmental employee who has a salary that is greater than or equal to the average salary for the department. The result should look like:

```
ENO  ENAME   SALARY   DNO
2000 LARRY   2000.00   10
4000 SHEMP    500.00   40
6000 GEORGE  9000.00   20
```

```
SELECT E.ENO, E.ENAME, E.SALARY, E.DNO
FROM    EMPLOYEE E, DEPTSTATSV2 D
WHERE E.DNO =  D.DNO
AND      E.SALARY >= D.AVGSAL
```

(d) Drop the DEPTSTATSV2 view.

```
DROP VIEW DEPTSTATSV2
```

 Copyright Pending 2020

28H. This exercise is a variation of Exercise 27M.

Reference the PARTSUPP and LINEITEM tables to create a view called BOUGHT_SOLD_STATS. This view contains each part number, its largest purchase price (MAXPAID), and its lowest selling price (MINSOLD). Hint: Modify the solution to Exercise 27M. Data for the BOUGHT_SOLD_STATS should look like:

PNO	MAXPAID	MINSOLD
P1	11.00	11.50
P3	12.50	12.00
P4	12.00	13.00
P5	11.00	11.00
P6	4.00	5.00
P7	3.50	3.00
P8	5.00	4.00

```
CREATE VIEW BOUGHT_SOLD_STATS
    AS
    WITH
        BOUGHT AS
            (SELECT PNO, MAX (PSPRICE) MAXPS
            FROM PARTSUPP
            GROUP BY PNO),
        SOLD AS
            (SELECT PNO, MIN(LIPRICE) MINLI
            FROM LINEITEM
            GROUP BY PNO)
        SELECT B.PNO, B.MAXPS MAXPAID, S.MINLI MINSOLD
        FROM   BOUGHT B, SOLD S
        WHERE B.PNO = S.PNO
```

Display any row in BOUGHT_SOLD_STATS where this largest purchase price is greater than its lowest selling price. The result should look like:

PNO	MAXPAID	MINSOLD
P3	12.50	12.00
P7	3.50	3.00
P8	5.00	4.00

```
SELECT *
FROM   BOUGHT_SOLD_STATS
WHERE MAXPAID > MINSOLD
```

Drop the BOUGHT_SOLD_STATS view

```
DROP VIEW BOUGHT_SOLD_STATS
```

PART VII

Special Topics

Chapter-29 – Transaction Processing

No Exercises

Chapter-30 - Recursive Queries

Exercises for Section A. Recursive One-to-Many Recursive Relationships

Although we have only presented one relatively simple recursive sample query, you should be able to utilize this example to code solutions for the following exercises. Do not specify an ORDER BY clause for any of these exercises. (Optionally, you are invited to detect a special kind of row sequence in the result tables. This topic will be discussed later in this section.)

30A1. Reference the REMPLOYEE table**.** Display ENO, ENAME, and SENO values for Employee 8000 and all employees who directly or indirectly work for this employee. I.e., Display data about Employee 8000 and all his descendants. The result table should look like:

ENO	ENAME	SENO
8000	JOE	1000
8500	GEORGE	8000
8600	DICK	8500
8700	HANK	8500

Hint: This exercise only requires one trivial modification to the solution for Sample Query 30.1.

Specify 8000 instead of 2000 in Sample Query 30.1

```
WITH DESCENDANTS (ENO, ENAME, SENO)
AS
(SELECT       ENO, ENAME,  SENO
 FROM         REMPLOYEE
 WHERE        ENO = '8000'
  UNION ALL
 SELECT      R.ENO, R.ENAME, R.SENO
 FROM         DESCENDANTS D, REMPLOYEE R
 WHERE        D.ENO  = R.SENO
)
 SELECT * FROM DESCENDANTS
```

30A2. Enhance the previous exercise 30A1 to also display SALARY values. The result table should look like:

```
ENO   ENAME   SALARY   SENO
8000  JOE     8000.00  1000
8500  GEORGE  7000.00  8000
8600  DICK    6000.00  8500
8700  HANK    6000.00  8500
```

Specify SALARY in first the two Sub-SELECTs.

```
WITH DESCENDANTS (ENO, ENAME, SALARY, SENO)
AS
(SELECT      ENO, ENAME, SALARY, SENO
 FROM        REMPLOYEE
 WHERE       ENO = '8000'
  UNION ALL
 SELECT      R.ENO, R.ENAME, R.SALARY, R.SENO
 FROM        DESCENDANTS D, REMPLOYEE R
 WHERE       D.ENO =  R.SENO
)
 SELECT * FROM DESCENDANTS
```

 Copyright Pending 2020

30B1. Reference the REMPLOYEE table**.** Traverse its tree from top to bottom. Start with the row for Employee 1000 (root node). Display all data about this employee and all employees who directly or indirectly work for this employee. (I.e., Display the entire tree.)

```
WITH DESCENDANTS (ENO, ENAME, SALARY, SENO)
        AS
        (SELECT      ENO, ENAME, SALARY, SENO
         FROM        REMPLOYEE
         WHERE       ENO = '1000'
          UNION ALL
         SELECT      R.ENO, R.ENAME, R.SALARY, R.SENO
         FROM        DESCENDANTS D, REMPLOYEE R
         WHERE       D.ENO  = R.SENO
        )
 SELECT * FROM DESCENDANTS
```

Result looks like:

ENO	ENAME	SALARY	SENO
1000	MOE	2000.00	-
2000	JANET	2000.00	1000
3000	LARRY	3000.00	1000
8000	JOE	8000.00	1000
4000	JULIE	500.00	2000
5000	JESSIE	400.00	2000
6000	FRANK	9000.00	2000
6500	CURLY	8000.00	3000
8500	GEORGE	7000.00	8000
4500	JOHNNY	2000.00	4000
4600	ELEANOR	3000.00	4000
5500	HANNAH	4000.00	5000
7500	SHEMP	9000.00	6500
8600	DICK	6000.00	8500
8700	HANK	6000.00	8500
4700	ANDY	2000.00	4600
4800	MATT	3000.00	4600

 Copyright Pending 2020

30B2. The solution for the previous Exercise 30B1 assumes you know that the root node has an ENO value of 1000. Assume you do not have this knowledge. Code an alternative solution that satisfies the same query objective. Start at the root node (without knowing its ENO value) and display all information about all its descendants.

```
WITH DESCENDANTS (ENO, ENAME, SALARY, SENO)
        AS
        (SELECT      ENO, ENAME, SALARY, SENO
        FROM         REMPLOYEE
        WHERE        SENO IS NULL
         UNION ALL
         SELECT      R.ENO, R.ENAME, R.SALARY, R.SENO
         FROM        DESCENDANTS D, REMPLOYEE R
         WHERE       D.ENO  = R.SENO
        )
 SELECT * FROM DESCENDANTS
```

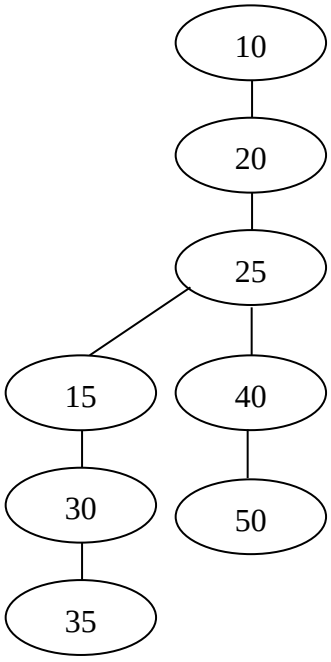
[Same result as preceding exercise.]

30C. Consider the recursive RDEMO1 table shown below in Figure 30.2. All columns contain integer values. PKEY is the primary-key. FKEY is a foreign-key that references PKEY. (The rows happen to be displayed in PKEY sequence.)

PKEY	CODE	FKEY
10	0	-
15	1000	25
20	0	10
25	0	20
30	0	15
35	0	30
40	0	25
50	0	40

Figure 30.2: RDEMO1 Table

Tree Diagram



 Copyright Pending 2020

Display the PKEY, CODE, and FKEY values for the rows with a PKEY value of 25 and all its descendant rows. The result should contain the following rows (without regard to sequence).

PKEY	CODE	FKEY
25	0	20
15	1000	25
40	0	25
30	0	15
50	0	40
35	0	30

Hints: Code a CTE called DESCENDANTS.

- The initialization Sub-SELECT should retrieve the row with a PKEY value of 25.
- The recursive Sub-SELECT should join DESCENDANTS with RDEMO1 by matching DESCENDANTS's primary-key with RDEMO1's foreign-key.

  DESCENDANTS.PKEY = RDEMO1.FKEY

- The Main-SELECT should display all information in DESCENDANTS.

```
WITH DESCENDANTS (PKEY, CODE, FKEY)
        AS
        (SELECT      PKEY, CODE, FKEY
         FROM        RDEMO1
         WHERE       PKEY = 25
          UNION ALL
         SELECT      R.PKEY, R.CODE, R.FKEY
         FROM        DESCENDANTS D, RDEMO1 R
         WHERE       D.PKEY = R.FKEY
        )
 SELECT * FROM DESCENDANTS
```

 Copyright Pending 2020

30D1: Reconsider the rows in the RDEMO1 table shown in Figure 30.2. Using pencil and paper, display all these rows in Breadth-First Hierarchical Sequence.

PKEY	CODE	FKEY
10	0	-
20	0	10
25	0	20
15	1000	25
40	0	25
30	0	15
50	0	40
35	0	30

30D2: Append ORDER BY ENO to the Main-SELECT in Sample Query 30.1. Execute the statement. Observe that the rows are no longer displayed in breadth-first hierarchical sequence.

```
WITH DESCENDANTS (ENO, ENAME, SENO)
AS
(SELECT      ENO, ENAME, SENO
 FROM        REMPLOYEE
 WHERE       ENO = '2000'
  UNION ALL
 SELECT   R.ENO, R.ENAME, R.SENO
 FROM        DESCENDANTS D, REMPLOYEE R
 WHERE       D.ENO  = R.SENO
)
 SELECT * FROM DESCENDANTS
 ORDER BY ENO
```

ENO	ENAME	SENO
2000	JANET	1000
4000	JULIE	2000
4500	JOHNNY	4000
4600	ELEANOR	4000
4700	ANDY	4600
4800	MATT	4600
5000	JESSIE	2000
5500	HANNAH	5000
6000	FRANK	2000

 Copyright Pending 2020

30D3. Reconsider the RDEMO1 table shown in Figure 30.2. Using pencil and paper, display its rows in depth-first hierarchical sequence.

PKEY	CODE	FKEY
10	0	-
20	0	10
25	0	20
15	1000	25
30	0	15
35	0	30
40	0	25
50	0	40

30E1. Consider the RDEMO1 table shown in Figure 30.2. What is the result of executing the following statement? Execute the statement to verify your answer.

```
WITH DESCENDANTS (PKEY, CODE, FKEY)
AS
(SELECT    PKEY, CODE, FKEY
 FROM      RDEMO1
 WHERE     PKEY IN (15, 40)
  UNION ALL
 SELECT    R.PKEY, R.CODE, R.FKEY
 FROM      RDEMO1 R, DESCENDANTS D
 WHERE     R.FKEY = D.PKEY
)
 SELECT * FROM DESCENDANTS
```

PKEY	CODE	FKEY
15	1000	25
40	0	25
30	0	15
50	0	40
35	0	30

30E2. Again, consider the RDEMO1 table. What is the result of executing the following statement? Observe and explain the presence of duplicate rows in the result. Execute the statement to verify your answer.

```
WITH DESCENDANTS (PKEY, CODE, FKEY)
AS
(SELECT    PKEY, CODE, FKEY
 FROM      RDEMO1
 WHERE     PKEY IN (25, 40)
  UNION ALL
 SELECT   R.PKEY, R.CODE, R.FKEY
 FROM      RDEMO1 R, DESCENDANTS D
 WHERE     R.FKEY = D.PKEY
)
 SELECT * FROM DESCENDANTS
```

PKEY	CODE	FKEY	
25	0	20	
40	0	25	←
15	1000	25	
40	0	25	←
50	0	40	←←
30	0	15	
50	0	40	←←
35	0	30	

Duplicate rows appear for PKEY values 40 and 50 because their corresponding nodes lie on the downward path starting at node 25 and on the downward path starting at node 40.

 Copyright Pending 2020

30F. This exercise focuses on the significant difference between specifying a restriction in the recursive Sub-SELECT versus the Main-SELECT. Consider the following statements which reference the RDEMO1 table shown in Figure 30.2. What is the result of executing each statement? Execute each statement to verify your answers.

```
WITH DESCENDANTS (PKEY, CODE, FKEY)
AS
(SELECT    PKEY, CODE, FKEY
 FROM      RDEMO1
 WHERE     PKEY = 20
  UNION ALL
 SELECT    R.PKEY, R.CODE, R.FKEY
 FROM      RDEMO1 R, DESCENDANTS D
 WHERE     R.FKEY = D.PKEY
)
 SELECT * FROM DESCENDANTS
 WHERE     CODE = 0
```

PKEY	CODE	FKEY
20	0	10
25	0	20
40	0	25
30	0	15
50	0	40
35	0	30

Specifying the CODE = 0 condition in the third Sub-SELECT eliminates the row with PKEY value of 15. However, note that its descendants (PKEY values 30 and 35) appear in the result.

```
WITH DESCENDANTS (PKEY, CODE, FKEY)
AS
(SELECT    PKEY, CODE, FKEY
 FROM      RDEMO1
 WHERE     PKEY = 20
  UNION ALL
 SELECT    R.PKEY, R.CODE, R.FKEY
 FROM      RDEMO1 R, DESCENDANTS D
 WHERE     R.FKEY = D.PKEY
 AND       R.CODE = 0
)
 SELECT * FROM DESCENDANTS
```

PKEY	CODE	FKEY
20	0	10
25	0	20
40	0	25
50	0	40

Specifying the CODE = 0 condition in the second Sub-SELECT eliminates the row with PKEY value of 15 *and all its descendants* (PKEY values 30 and 35).

 Copyright Pending 2020

30G. Modify Exercise 30A which displayed the ENO, ENAME, SALARY, and SENO values for Employee 8000 and all employees who directly or indirectly work for this employee. This time only display information about an employee who directly or indirectly works for Employee 8000 if the employee's salary exceeds $6500.00. The result should look like:

```
ENO   ENAME      SALARY     SENO
8000  JOE        8000.00    1000
8500  GEORGE     7000.00    8000
```

```
WITH DESCENTANTS (ENO, ENAME, SALARY, SENO)
AS
(SELECT      ENO, ENAME, SALARY, SENO
 FROM        REMPLOYEE
 WHERE       ENO = '8000'
          UNION ALL
 SELECT   R.ENO, R.ENAME, R.SALARY, R.SENO
 FROM        DESCENTANTS D, REMPLOYEE R
 WHERE       D.ENO = R.SENO
)
SELECT * FROM DESCENTANTS
WHERE SALARY > 6500.00
```

 Copyright Pending 2020

30H. Display the ENO, ENAME, and SENO values for Employee 2000 and all her descendants. However, exclude the row describing Employee 4000 and all descendants of this employee. The result should look like:

ENO	ENAME	SENO
2000	JANET	1000
5000	JESSIE	2000
6000	FRANK	2000
5500	HANNAH	5000

Code two solutions.

Solution-1 should specify the same restriction in both the recursive Sub-SELECT and the Main-SELECT. The recursive Sub-SELECT stores a row for Employee 4000 into DESCENTANTS but eliminates all its descendants. The Main-SELECT eliminates the row for Employee 4000.

Solution-2 specifies just one restriction in the recursive Sub-SELECT which prevents the row for EMPLOYEE 4000 from being placed into DESCENDANTS. Hence, none of its descendants will be placed into DESCENDANTS.

Solution-1

```
WITH DESCENTANTS (ENO, ENAME, SENO)
AS
(SELECT      ENO, ENAME, SENO
 FROM        REMPLOYEE
 WHERE       ENO = '2000'
   UNION ALL
 SELECT       R.ENO, R.ENAME, R.SENO
 FROM        DESCENTANTS D, REMPLOYEE R
 WHERE       D.ENO  = R.SENO
 AND         D.ENO <> '4000'
)
 SELECT * FROM DESCENTANTS
 WHERE       ENO <> '4000'
```

 Copyright Pending 2020

Solution-2

```
WITH DESCENTANTS (ENO, ENAME, SENO)
AS
(SELECT      ENO, ENAME, SENO
 FROM        REMPLOYEE
 WHERE       ENO = '2000'
  UNION ALL
 SELECT      R.ENO, R.ENAME, R.SENO
 FROM        DESCENTANTS D, REMPLOYEE R
 WHERE       D.ENO  = R.SENO
 AND         R.ENO <> '4000'
)
 SELECT * FROM DESCENTANTS
```

Notice the difference between Solution-1 and Solution-2. Focus on the second (recursive) Sub-SELECT.

Solution-1 specifies: **D**.ENO <> 4000

Solution-2 specifies: **R**.ENO <> 4000

The Solution-1 condition allows the Employee 4000 row to be placed into DESCENDENTS, but it prevents its descendent rows from being placed into DESCENDENTS. (This is why Solution-1 must specify the ENO <> '4000' condition in the third Sub-SELECT.)

The Solution-2 condition prevents the Employee 4000 row and its descendent rows from being placed into DESCENDENTS.

Suggestion: Perform a paper-and-pencil step-by-step walkthrough for each solution to confirm your understanding.

30I. Display the ENO, ENAME, SALARY, and BENO values of Employee 2000. Also display these values for any employee who directly or indirectly works for this employee with the following exception. Do not display information about an employee *and his dependents* if the employee earns less than $1000.00. The result should look like:

```
ENO   ENAME   SALARY   BENO
2000  JANET   2000.00  1000
6000  FRANK   9000.00  2000
```

Code two solutions similar to the two solutions for the preceding Exercise 30H.

Solution-1

```
WITH DESCENTANTS (ENO, ENAME, SALARY, SENO)
AS
(SELECT      ENO, ENAME, SALARY, SENO
 FROM        REMPLOYEE
 WHERE       ENO = '2000'
  UNION ALL
 SELECT   R.ENO, R.ENAME, R.SALARY, R.SENO
 FROM        DESCENTANTS D, REMPLOYEE R
 WHERE       D.ENO  = R.SENO
 AND         D.SALARY > 1000.00
)
 SELECT * FROM DESCENTANTS
 WHERE    SALARY > 1000.00
```

Solution-2

```
WITH DESCENTANTS (ENO, ENAME, SALARY, SENO)
AS
(SELECT      ENO, ENAME, SALARY, SENO
 FROM        REMPLOYEE
 WHERE       ENO = '2000'
  UNION ALL
 SELECT   R.ENO, R.ENAME, R.SALARY, R.SENO
 FROM        DESCENTANTS D, REMPLOYEE R
 WHERE       D.ENO  = R.SENO
 AND         R.SALARY > 1000.00
)
 SELECT * FROM DESCENTANTS
```

 Copyright Pending 2020

30J. What is total salary of all employees who report to Employee 8000? The result should look like:

```
   TOTSAL
---------
19000.00
```

Hint: You only need to modify the third Sub-SELECT in the solution for Exercise 30A2.

```
WITH DESCENTANTS (ENO, ENAME, SALARY, SENO)
AS
(SELECT      ENO, ENAME, SALARY, SENO
 FROM        REMPLOYEE
 WHERE       ENO = '8000'
  UNION ALL
 SELECT   R.ENO, R.ENAME, R.SALARY, R.SENO
 FROM        DESCENTANTS D, REMPLOYEE R
 WHERE       D.ENO  = R.SENO
)
 SELECT SUM (SALARY) TOTSAL FROM DESCENTANTS
 WHERE  ENO <> '8000'
```

 Copyright Pending 2020

The following two exercises do not require you to code recursive SQL.

30K1. Display the ENO, ENAME, SALARY values for all supervisors. Sort the result by ENO values. The result should look like:

```
ENO  ENAME     SALARY
1000 MOE     2000.00
2000 JANET   2000.00
3000 LARRY   3000.00
4000 JULIE    500.00
4600 ELEANOR 3000.00
5000 JESSIE   400.00
6500 CURLY   8000.00
8000 JOE     8000.00
8500 GEORGE  7000.00
```

Note: This result is not in hierarchical sequence.

```
SELECT ENO, ENAME, SALARY
FROM   REMPLOYEE
WHERE ENO IN (SELECT SENO FROM REMPLOYEE)
ORDER BY ENO
```

30K2. Display the ENO, ENAME, SALARY values for all non-supervisors. Sort the result by ENO values. The result should look like:

```
ENO  ENAME    SALARY
4500 JOHNNY 2000.00
4700 ANDY   2000.00
4800 MATT   3000.00
5500 HANNAH 4000.00
6000 FRANK  9000.00
7500 SHEMP  9000.00
8600 DICK   6000.00
8700 HANK   6000.00
```

Note: This result is not in hierarchical sequence.

```
SELECT ENO, ENAME, SALARY
FROM REMPLOYEE
WHERE ENO NOT IN
        (SELECT SENO FROM REMPLOYEE  WHERE SENO IS NOT NULL)
ORDER BY ENO
```

Interesting Observation: Execute this statement after removing the "WHERE SENO IS NOT NULL" clause.

30L. Display the ENO, ENAME, SALARY, and SENO values for Employee 4000 and all her direct and indirect supervisees. Also, for each employee, display a running total of the employee's salary plus the total salary of all her direct and indirect supervisors. The result should look like:

```
ENO  ENAME     SALARY TOTPATH SENO
4000 JULIE     500.00  500.00 2000
4500 JOHNNY   2000.00 2500.00 4000
4600 ELEANOR  3000.00 3500.00 4000
4700 ANDY     2000.00 5500.00 4600
4800 MATT     3000.00 6500.00 4600
```

Hint: This exercise is similar to Sample Query 30.3.

```
WITH DESCENTANTS (ENO, ENAME, SALARY, TOTPATH, SENO)
AS
(SELECT      ENO, ENAME, SALARY, SALARY, SENO
 FROM        REMPLOYEE
 WHERE       ENO = '4000'
  UNION ALL
 SELECT      R.ENO, R.ENAME,
             R.SALARY, R.SALARY + D.TOTPATH, R.SENO
 FROM        DESCENTANTS D, REMPLOYEE R
 WHERE       D.ENO  = R.SENO
)
 SELECT * FROM DESCENTANTS
```

 Copyright Pending 2020

30M. Start with Employee 5500. Display all data about this employee and all data about her direct or indirect managers. The result should look like:

```
ENO  ENAME      SALARY   BENO
5500 HANNAH   4000.00   5000
5000 JESSIE    400.00   2000
2000 JANET    2000.00   1000
1000 MOE      2000.00   -
```

```
WITH ANCESTORS (ENO, ENAME, SALARY, BENO)
AS
(SELECT ENO, ENAME, SALARY, BENO
 FROM   REMPLOYEE
 WHERE  ENO ='5500'
 UNION ALL
 SELECT R.ENO, R.ENAME, R.SALARY, R.BENO
 FROM   REMPLOYEE R, ANCESTORS A
 WHERE  R.ENO = A.BENO
 )
 SELECT * FROM ANCESTORS;
```

 Copyright Pending 2020

30N. Reference the RDEMO1 table described in Exercise 30C. Within the context of upward tree traversal, this exercise focuses on the specification a restriction in the recursive Sub-SELECT and Main-SELECT. Consider the following three statements. What is the result of executing each statement? Execute each statement to verify your answer.

```
Statement 37N-1
WITH ANCESTORS (PKEY, CODE, FKEY)
AS
(SELECT    PKEY, CODE, FKEY
 FROM      RDEMO1
 WHERE     PKEY = 35
  UNION ALL
 SELECT    R.PKEY, R.CODE, R.FKEY
 FROM      ANCESTORS A, RDEMO1 R
 WHERE     A.FKEY = R.PKEY
)
 SELECT * FROM ANCESTORS
 WHERE    CODE = 0          ←
```

PKEY	CODE	FKEY
35	0	30
30	0	15
25	0	20
20	0	10
10	0	-

```
Statement 37N-2
WITH ANCESTORS (PKEY, CODE, FKEY)
AS
(SELECT    PKEY, CODE, FKEY
 FROM      RDEMO1
 WHERE     PKEY = 35
  UNION ALL
 SELECT    R.PKEY, R.CODE, R.FKEY
 FROM      ANCESTORS A, RDEMO1 R
 WHERE     A.FKEY = R.PKEY
 AND       A.CODE = 0        ←
)
 SELECT * FROM ANCESTORS
```

PKEY	CODE	FKEY
35	0	30
30	0	15
15	1000	25

```
Statement 37N-3
WITH ANCESTORS (PKEY, CODE, FKEY)
AS
(SELECT    PKEY, CODE, FKEY
 FROM      RDEMO1
 WHERE     PKEY = 35
  UNION ALL
 SELECT R.PKEY, R.CODE, R.FKEY
 FROM      ANCESTORS A, RDEMO1 R
 WHERE     A.FKEY = R.PKEY
 AND       R.CODE = 0        ←
)
 SELECT * FROM ANCESTORS
```

PKEY	CODE	FKEY
35	0	30
30	0	15

30O1. Reference the RDEMO1 table. Display the PKEY, CODE, and FKEY values for the row with a PKEY value of 25 and all its descendants. Also display the level number for each row. The result should look like:

```
LVL  PKEY  CODE  FKEY
  1    25     0    20
  2    15  1000    25
  2    40     0    25
  3    30     0    15
  3    50     0    40
  4    35     0    30
```

```
WITH DESCENDANTS (LVL, PKEY, CODE, FKEY)
AS
(SELECT      1, PKEY, CODE, FKEY
 FROM        RDEMO1
 WHERE       PKEY = 25
  UNION ALL
 SELECT      D.LVL+1, R.PKEY, R.CODE, R.FKEY
 FROM        DESCENDANTS D, RDEMO1 R
 WHERE       D.PKEY = R.FKEY
)
SELECT * FROM DESCENDANTS
```

30O2. Modify the previous query objective such that downward traversal is restricted to three levels. The result should look like:

```
LVL  PKEY  CODE  FKEY
  1    25     0    20
  2    15  1000    25
  2    40     0    25
  3    30     0    15
  3    50     0    40
```

```
WITH DESCENDANTS (LVL, PKEY, CODE, FKEY)
AS
(SELECT      1, PKEY, CODE, FKEY
 FROM        RDEMO1
 WHERE       PKEY = 25
  UNION ALL
 SELECT      D.LVL+1, R.PKEY, R.CODE, R.FKEY
 FROM        DESCENDANTS D, RDEMO1 R
 WHERE       D.PKEY = R.FKEY
 AND         D.LVL+1 <= 3
)
   SELECT * FROM DESCENDANTS
```

30P1. Reference the RDEMO1 table. Display the PKEY, CODE, and FKEY values for the row with a PKEY value of 40 and all its ancestors. Also display the level number for each row. The result should look like:

```
LVL  PKEY  CODE  FKEY
  1    40     0    25
  2    25     0    20
  3    20     0    10
  4    10     0     -
```

```
WITH ANCESTORS (LVL, PKEY, CODE, FKEY)
AS
(SELECT      1, PKEY, CODE, FKEY
 FROM        RDEMO1
 WHERE       PKEY = 40
  UNION ALL
 SELECT      A.LVL+1, R.PKEY, R.CODE, R.FKEY
 FROM        ANCESTORS A, RDEMO1 R
 WHERE       A.FKEY = R.PKEY
)
 SELECT * FROM ANCESTORS
```

30P2. Modify this query objective such that upward traversal is restricted to three levels. The result should look like:

```
LVL  PKEY  CODE  FKEY
  1    40     0    25
  2    25     0    20
  3    20     0    10
```

```
WITH ANCESTORS (LVL, PKEY, CODE, FKEY)
AS
(SELECT      1, PKEY, CODE, FKEY
 FROM        RDEMO1
 WHERE       PKEY = 40
  UNION ALL
 SELECT      A.LVL+1, R.PKEY, R.CODE, R.FKEY
 FROM        ANCESTORS A, RDEMO1 R
 WHERE       A.FKEY = R.PKEY
 AND         A.LVL+1 <= 3
)
 SELECT * FROM ANCESTORS
```

30Q: Modify the Main-SELECT in Sample Query 30.8a such that the result looks like Result-3. Then specify an ORDER BY clause to produce Result-4.

Start with Result-1 and generate Result-3:

LVL	ENO	ENAME	SENO
1	4600	ELEANOR	4000
2	4000	JULIE	2000
3	2000	JANET	1000
4	1000	MOE	-

Result-1

LVL	ENO	ENAME	SENO
4	4600	ELEANOR	4000
3	4000	JULIE	2000
2	2000	JANET	1000
1	1000	MOE	-

Result-3

Assume you know a query result will display four levels of rows as in Result-1. You could produce this Result-3 by changing the third Sub-SELECT to:

```
SELECT 4 - (LVL-1) LVL, ENO, ENAME, SENO
FROM ANCESTORS
```

However, you may not have foreknowledge that 4 is the highest level in the result. Hence, you will need to formulate some expression that will generate this value. The expression is: (SELECT MAX (LVL) FROM ANCESTORS)
Then substitute this expression for the 4 as shown below.

```
WITH ANCESTORS (LVL, ENO, ENAME, SENO)
AS
(SELECT 1, ENO, ENAME, SENO
        FROM   REMPLOYEE
WHERE  ENO ='4600'
  UNION ALL
SELECT LVL+1, R.ENO, R.ENAME, R.SENO
FROM   ANCESTORS A, REMPLOYEE R
WHERE  A.SENO = R.ENO
)
 SELECT      (SELECT MAX (LVL) FROM ANCESTORS) - (LVL-1) LVL,
             ENO, ENAME, SENO
FROM ANCESTORS
 - - - - - - - - - - - - - - - - - - - - - - - - -
```

Next, generate Result-4 by appending ORDER BY LVL to the above statement

LVL	ENO	ENAME	SENO
4	4600	ELEANOR	4000
3	4000	JULIE	2000
2	2000	JANET	1000
1	1000	MOE	-

Result-3

LVL	ENO	ENAME	SENO
1	1000	MOE	-
2	2000	JANET	1000
3	4000	JULIE	2000
4	4600	ELEANOR	4000

Result-4

 Copyright Pending 2020

30R. Optional Exercise: Assume that many users would like the REMPLOYEE_V2 table. For this reason, you decide to create a view called REMPLOYEE_V2 that looks like the REMPLOYEE_V2 table. Create this view, and then execute SELECT * FROM REMPLOYEE_V2 to display its contents.

```
CREATE VIEW REMPLOYEE_V2 (ENO, ENAME, SALARY, SENO)
AS
SELECT  R1.ENO, R1.ENAME, R1.SALARY, R2.ENO
FROM    REMPLOYEE R1 LEFT OUTER JOIN REMPLOYEE R2
            ON R1.ENO = R2.SENO;

SELECT * FROM VREMPLOYEE_V2
ORDER BY ENO, SENO;
```

 Copyright Pending 2020

Exercises for Section B. Recursive Many-to-Many Recursive Relationships

30S1. Reference the REPORTS_TO and REMPLOYEE2 tables. Apply the Soultion-1 code-pattern to display the ENO, ENAME, SALARY, and SENO values for Employee 5000 and all employees who directly or indirectly work for her. The result should look like:

```
ENO  ENAME     SALARY SENO
5000 JESSIE    400.00 2000
4600 ELEANOR 3000.00 5000
5500 HANNAH  4000.00 5000
4700 ANDY    2000.00 4600
4800 MATT    3000.00 4600
4800 MATT    3000.00 5500
```

```
WITH DESCENDANTS (ENO, ENAME, SALARY, SENO)
AS
(SELECT     R.ENO, R2.ENAME, R2.SALARY, R.SENO
 FROM       REPORTS_TO R, REMPLOYEE2 R2
 WHERE      R.ENO = R2.ENO
 AND        R.ENO = '5000'
  UNION ALL
 SELECT     R.ENO, R2.ENAME, R2.SALARY, R.SENO
 FROM       DESCENDANTS D, REPORTS_TO R, REMPLOYEE2 R2
 WHERE      R.ENO = R2.ENO
 AND        D.ENO = R.SENO
 )
 SELECT * FROM DESCENDANTS
```

 Copyright Pending 2020

30S2. Apply the Solution-2 code-pattern to code an equivalent solution for Exercise 30S1.

```
WITH FULLTAB (ENO, ENAME, SALARY, SENO)
 AS
 (SELECT RT.ENO, R2.ENAME, R2.SALARY, RT.SENO
  FROM REPORTS_TO RT, REMPLOYEE2 R2
  WHERE RT.ENO = R2.ENO),

  DESCENDANTS (ENO, ENAME, SALARY, SENO)
  AS
  (SELECT ENO, ENAME, SALARY, SENO
   FROM FULLTAB
   WHERE ENO = '5000'
   UNION ALL
   SELECT     F.ENO, F.ENAME, F.SALARY, F.SENO
   FROM       DESCENDANTS D, FULLTAB F
   WHERE      D.ENO = F.SENO
  )
  SELECT * FROM DESCENDANTS
```

30S3. Apply the Solution-3 code-pattern to code another equivalent solution for Exercise 30S1.

```
WITH DESCENDANTS (ENO, SENO)
AS
(SELECT      ENO, SENO
 FROM        REPORTS_TO
 WHERE       ENO = '5000'
  UNION ALL
 SELECT      R.ENO, R.SENO
 FROM        DESCENDANTS D, REPORTS_TO R
 WHERE       D.ENO = R.SENO
 )
 SELECT      D.ENO, E.ENAME, E.SALARY, D.SENO
 FROM        DESCENDANTS D, REMPLOYEE2 E
 WHERE       D.ENO = E.ENO
```

30T. Code three SELECT-statements to satisfy the following query objective. Each statement should be similar in structure to those statements presented in Solution-1, Solution-2, and Solution-3 for Sample Query 30.11.

Reference the RDEMO2 and RDEMO2MM tables. Display the CHILDKEY, AMT, and PARENTKEY values for CHILDKEY 10 and all its descendants. The result should look like:

```
CHILDKEY  AMT  PARENTKEY
      10  200         40
      50  200         10
      60  500         50
      70  600         60
```

Solution-1

```
WITH DESCENDANTS (CHILDKEY, AMT, PARENTKEY)
AS
(SELECT MM.CHILDKEY, R2.AMT, MM.PARENTKEY
 FROM   RDEMO2MM MM, RDEMO2 R2
 WHERE  MM.CHILDKEY = R2.KEY
 AND      MM.CHILDKEY = 10
  UNION ALL
 SELECT MM.CHILDKEY, R2.AMT, MM.PARENTKEY
 FROM   DESCENDANTS D,  RDEMO2MM MM, RDEMO2 R2
 WHERE  D.CHILDKEY = MM.PARENTKEY
 AND    MM.CHILDKEY = R2.KEY
 )
 SELECT * FROM DESCENDANTS
```

Solution-2

```
WITH FULLTAB (CHILDKEY, AMT, PARENTKEY)
 AS
 (SELECT      MM.CHILDKEY, R2.AMT, MM.PARENTKEY
  FROM        RDEMO2MM MM, RDEMO2 R2
  WHERE       MM.CHILDKEY = R2.KEY
    ),
  DESCENDANTS (CHILDKEY, AMT, PARENTKEY)
  AS
  (SELECT *   FROM FULLTAB
   WHERE       CHILDKEY = 10
   UNION ALL
   SELECT      F.CHILDKEY, F.AMT, F.PARENTKEY
   FROM        DESCENDANTS D, FULLTAB F
   WHERE       D.CHILDKEY = F.PARENTKEY
    )
  SELECT * FROM DESCENDANTS
```

Solution-3

```
WITH DESCENTANTS (CHILDKEY, PARENTKEY)
AS
(SELECT CHILDKEY, PARENTKEY
 FROM   RDEMO2MM
 WHERE CHILDKEY = 10
  UNION ALL
 SELECT MM.CHILDKEY, MM.PARENTKEY
 FROM   DESCENTANTS D, RDEMO2MM MM
 WHERE D.CHILDKEY = MM.PARENTKEY
)
SELECT D.CHILDKEY, R2.AMT, D.PARENTKEY
FROM   DESCENTANTS D, RDEMO2 R2
WHERE D.CHILDKEY = R2.KEY
```

30U. Modify Sample Query 30.11 to remove duplicate rows from the result by grouping and counting the number of duplicate rows. Show this count value in the CNT column. The result will contain the following rows, but these rows might appear in a different sequence.

ENO	ENAME	SALARY	SENO	CNT
2000	JANET	2000.00	1000	1
5000	JESSIE	400.00	2000	1
4000	JULIE	500.00	2000	1
6000	FRANK	9000.00	2000	1
4500	JOHNNY	2000.00	4000	1
4600	ELEANOR	3000.00	4000	1
4700	ANDY	2000.00	4600	2
4800	MATT	3000.00	4600	2
4600	ELEANOR	3000.00	5000	1
5500	HANNAH	4000.00	5000	1
4800	MATT	3000.00	5500	1
4800	MATT	3000.00	6000	1

Early-join of REMPLOYEE2

```
WITH DESCENDANTS (ENO, ENAME, SALARY, SENO)
AS
(SELECT     R.ENO, R2.ENAME, R2.SALARY, R.SENO
 FROM       REPORTS_TO R, REMPLOYEE R2
 WHERE      R.ENO = R2.ENO
 AND        R.ENO = '2000'
  UNION ALL
 SELECT     R.ENO, R2.ENAME, R2.SALARY, R.SENO
 FROM       DESCENDANTS D, REPORTS_TO R, REMPLOYEE2 R2
 WHERE      R.ENO = R2.ENO
 AND        D.ENO = R.SENO
)
 SELECT     ENO, ENAME, SALARY, SENO,  COUNT (*) CNT
 FROM       DESCENDANTS
 GROUP BY ENO, ENAME, SALARY, SENO
```

Late-join of REMPLOYEE2

```
WITH DESCENTANTS (ENO, SENO)
AS
(SELECT       ENO, SENO
 FROM         REPORTS_TO
 WHERE        ENO = '2000'
  UNION ALL
 SELECT       R.ENO, R.SENO
 FROM         DESCENTANTS D, REPORTS_TO R
 WHERE        D.ENO = R.SENO
)
 SELECT   D.ENO, E.ENAME, E.SALARY, D.SENO,  COUNT (*) CNT
 FROM     DESCENTANTS D, REMPLOYEE2 E
 WHERE    D.ENO = E.ENO
 GROUP BY D.ENO, E.ENAME, E.SALARY, D.SENO
```

30V. Reference RDEMO2 and RDEMO2MM. Display the CHILDKEY, AMT, and PARENTKEY values for Node-30 and its descendants. However, if a descendant's AMT is greater than or equal to 600, then exclude all descendants of this descendant. The result should look like:

CHILDKEY	AMT	PARENTKEY
30	500	40
50	200	30
60	500	50

Solution-1 code-pattern

```
WITH DESCENDANTS (CHILDKEY, AMT, PARENTKEY)
AS
(SELECT      MM.CHILDKEY, R2.AMT, MM.PARENTKEY
 FROM        RDEMO2MM MM, RDEMO2 R2
 WHERE       MM.CHILDKEY = R2.KEY
 AND         MM.CHILDKEY = 30
  UNION ALL
 SELECT      MM.CHILDKEY, R2.AMT, MM.PARENTKEY
 FROM         DESCENDANTS D,  RDEMO2MM MM, RDEMO2 R2
 WHERE       D.CHILDKEY = MM.PARENTKEY
 AND         MM.CHILDKEY = R2.KEY
 AND         R2.AMT <  600
 )
 SELECT * FROM DESCENDANTS;
```

Solution-2 code-pattern

```
WITH
 FULLTAB (CHILDKEY, AMT, PARENTKEY )AS
 (SELECT    MM.CHILDKEY, R2.AMT, MM.PARENTKEY
 FROM        RDEMO2MM MM, RDEMO2 R2
 WHERE       MM.CHILDKEY = R2.KEY),

 DESCENDANTS (CHILDKEY, AMT, PARENTKEY) AS
 (SELECT     CHILDKEY, AMT, PARENTKEY
 FROM        FULLTAB
 WHERE       CHILDKEY = 30
  UNION ALL
 SELECT      F.CHILDKEY, F.AMT, F.PARENTKEY
 FROM          DESCENDANTS D, FULLTAB F
 WHERE       D.CHILDKEY = F.PARENTKEY
 AND         F.AMT <  600
 )
 SELECT * FROM DESCENDANTS
```

 Copyright Pending 2020

30W. Reference the RDEMO2 and RDEMO2MM tables. Start with Node-60. Display the CHILDKEY, AMT, and PARENTKEY values for this node and all of its direct and indirect ancestors. The result should look like:

CHILDKEY	AMT	PARENTKEY
60	500	50
50	200	10
50	200	20
50	200	30
10	200	40
20	700	40
30	500	40
40	100	0
40	100	0
40	100	0

Solution-3 code-pattern

```
WITH ANCESTORS (CHILDKEY, PARENTKEY)
AS
(SELECT       CHILDKEY, PARENTKEY
 FROM          RDEMO2MM
 WHERE        CHILDKEY = 60
   UNION ALL
 SELECT       R2MM.CHILDKEY, R2MM.PARENTKEY
 FROM         ANCESTORS A, RDEMO2MM R2MM
 WHERE        A.PARENTKEY  = R2MM.CHILDKEY
)
 SELECT       A.CHILDKEY, R2.AMT, A.PARENTKEY
 FROM         ANCESTORS A, RDEMO2 R2
 WHERE        A.CHILDKEY = R2.KEY
```

Solution-2 code-pattern

```
WITH
 FULLTAB (CHILDKEY, AMT, PARENTKEY )AS
 (SELECT   MM.CHILDKEY, R2.AMT, MM.PARENTKEY
 FROM         RDEMO2MM MM, RDEMO2 R2
 WHERE        MM.CHILDKEY = R2.KEY),

 DESCENDANTS (CHILDKEY, AMT, PARENTKEY) AS
 (SELECT      CHILDKEY, AMT, PARENTKEY
 FROM         FULLTAB
 WHERE        CHILDKEY = 60
  UNION ALL
 SELECT       F.CHILDKEY, F.AMT, F.PARENTKEY
 FROM           DESCENDANTS D, FULLTAB F
 WHERE        F.CHILDKEY = D.PARENTKEY
 )
 SELECT * FROM DESCENDANTS
```

 Copyright Pending 2020

30X. Reference the REPORTS_TO and REMPLOYEE2 tables. Display the ENO, ENAME, and SENO values for Employee 4000 and all his direct or indirect supervisors. The result should look like:

```
ENO   ENAME      SENO
4000  JULIE      2000
2000  JANET      1000
1000  MOE        0000
```

Solution-3 code-pattern

```
WITH ANCESTORS (ENO, SENO)
AS
(SELECT       ENO, SENO
 FROM          REPORTS_TO
 WHERE        ENO ='4000'
  UNION ALL
 SELECT       R.ENO, R.SENO
 FROM         ANCESTORS A, REPORTS_TO R
 WHERE        A.SENO  = R.ENO
)
 SELECT       A.ENO, R2.ENAME, A.SENO
 FROM         ANCESTORS A, REMPLOYEE2 R2
 WHERE        A.ENO = R2.ENO
```

Solution-2 code-pattern

```
WITH
FULLTAB (ENO, ENAME, SENO)  AS
          (SELECT RT.ENO, R2.ENAME, RT.SENO
           FROM REPORTS_TO RT, REMPLOYEE2 R2
           WHERE RT.ENO = R2.ENO),

ANCESTORS (ENO, ENAME, SENO) AS
         (SELECT ENO, ENAME, SENO
         FROM FULLTAB
          WHERE ENO = '4000'
            UNION ALL
          SELECT       F.ENO, F.ENAME, F.SENO
          FROM         ANCESTORS A, FULLTAB F
          WHERE        A.SENO = F.ENO
 )
SELECT * FROM ANCESTORS
```

30Y1. Modify Sample Query 30.14 to limit the downward traversal to three levels. (Hint: Review Sample Query 30.7b.)

```
WITH DESCENDANTS (LVL, ENO, ENAME, SALARY, SENO)
AS
(SELECT     1, R.ENO, R2.ENAME, R2.SALARY, R.SENO
 FROM       REPORTS_TO R, REMPLOYEE2 R2
 WHERE      R.ENO = R2.ENO
 AND        R.ENO = '2000'
  UNION ALL
 SELECT     LVL+1, R.ENO, R2.ENAME, R2.SALARY, R.SENO
 FROM       DESCENDANTS D, REPORTS_TO R, REMPLOYEE2 R2
 WHERE      R.ENO = R2.ENO AND D.ENO = R.SENO
AND         D.LVL+1 <= 3
 )
SELECT * FROM DESCENDANTS
```

LVL	ENO	ENAME	SALARY	SENO
1	2000	JANET	2000.00	1000
2	4000	JULIE	500.00	2000
2	5000	JESSIE	400.00	2000
2	6000	FRANK	9000.00	2000
3	4500	JOHNNY	2000.00	4000
3	4600	ELEANOR	3000.00	4000
3	4600	ELEANOR	3000.00	5000
3	5500	HANNAH	4000.00	5000
3	4800	MATT	3000.00	6000

 Copyright Pending 2020

30Y2. Modify the above Sample Query 30.15 to (i) limit the upward traversal to three levels, (ii) remove duplicate rows from the result table, and (iii) modify the level numbers such that the result looks like:

```
LVL   ENO  ENAME    SALARY   SENO
  2   2000 JANET   2000.00 1000
  3   5000 JESSIE   400.00 2000
  3   4000 JULIE    500.00 2000
  4   4600 ELEANOR 3000.00 4000
  4   4600 ELEANOR 3000.00 5000
```

Hint: Review Exercise 30Q.

```
WITH ANCESTORS (LVL, ENO, ENAME, SALARY, SENO)
 AS
(SELECT      1, R.ENO, R2.ENAME, R2.SALARY, R.SENO
 FROM        REPORTS_TO R, REMPLOYEE2 R2
 WHERE       R.ENO = R2.ENO
 AND         R.ENO ='4600'
   UNION ALL
 SELECT      LVL+1, R.ENO, R2.ENAME, R2.SALARY, R.SENO
 FROM        ANCESTORS A, REPORTS_TO R, REMPLOYEE2 R2
 WHERE       A.SENO = R.ENO
 AND         R.ENO = R2.ENO
 AND         LVL+1 <=3
)
 SELECT DISTINCT
        (SELECT MAX (LVL) FROM ANCESTORS) - (LVL-2)  LVL,
          ENO, ENAME, SALARY, SENO
 FROM ANCESTORS
 ORDER BY LVL;
```

 Copyright Pending 2020

Exercises for Section C. Self-Joins for Recursive Queries

30Z1. Consider Employees 3000 and 8600. Display the number, name, and salary for these employees. Also, if either of these employees is a supervisor, display the number, name, and salary of each immediate supervisee. The result should look like:

```
BOSSENO BOSSENAME BOSSSALARY ENO  ENAME  SALARY
3000    LARRY         3000.00 6500 CURLY 8000.00
8600    DICK          6000.00 -    -       -
```

```
SELECT PARENT.ENO BOSSENO, PARENT.ENAME BOSSENAME,
          PARENT.SALARY BOSSSALARY,
          CHILD.ENO, CHILD.ENAME, CHILD.SALARY
FROM  REMPLOYEE PARENT LEFT OUTER JOIN REMPLOYEE CHILD
         ON  PARENT.ENO = CHILD.SENO
WHERE      PARENT.ENO IN ('3000', '8600')
ORDER BY PARENT.ENO, CHILD.ENO
```

30Z2. Consider Employees 3000 and 8600. Display each employee's number, name, and salary followed by the number, name and salary of the employee's immediate supervisor. The result should look like:

```
ENO  ENAME SALARY  BOSSENO BOSSENAME BOSSSALARY
3000 LARRY 3000.00 1000    MOE          2000.00
8600 DICK  6000.00 8500    GEORGE       7000.00
```

```
SELECT CHILD.ENO, CHILD.ENAME, CHILD.SALARY,
          PARENT.ENO BOSSENO, PARENT.ENAME BOSSENAME,
          PARENT.SALARY BOSSSALARY
FROM   REMPLOYEE CHILD LEFT OUTER JOIN REMPLOYEE PARENT
          ON  CHILD.SENO = PARENT.ENO
WHERE  CHILD.ENO IN ('3000',  '8600')
ORDER BY CHILD.ENO
```

 Copyright Pending 2020

30Z3. Display the numbers and names of Employees 3000, 5000, and 8000. If any of these employees is a supervisor, display each supervisee's number and name; and, if any of these supervisees is also a supervisor, display each of these supervisee's number and name. Sort the result by the supervisor's (the parent's) ENO value. The result should look like:

PARENTENO	PARENTNAME	CHILDENO	CHILDNAME	GRANDCHILDENO	GRANDCHILDNAME
3000	LARRY	6500	CURLY	7500	SHEMP
5000	JESSIE	5500	HANNAH	-	-
8000	JOE	8500	GEORGE	8600	DICK
8000	JOE	8500	GEORGE	8700	HANK

Query-Pattern-6: Three-Level, Parent-Oriented, Non-Matching

```
SELECT PARENT.ENO              PARENTENO,
       PARENT.ENAME            PARENTNAME,
       CHILD.ENO               CHILDENO,
        CHILD.ENAME            CHILDNAME,
        GRANDCHILD.ENO         GRANDCHILDENO,
       GRANDCHILD.ENAME        GRANDCHILDNAME
 FROM  REMPLOYEE PARENT
            LEFT OUTER JOIN REMPLOYEE CHILD
                    ON PARENT.ENO = CHILD.SENO
            LEFT OUTER JOIN REMPLOYEE GRANDCHILD
                    ON CHILD.ENO = GRANDCHILD.SENO
WHERE  PARENT.ENO IN ('3000', '5000', '8000')
ORDER BY PARENT.ENO
```

30Z4. Consider Employees 3000, 6000, and 8500. If any of these employees is supervised by a supervisor who is also supervised by a supervisor, then display the number and name of all such employees. The result should look like:

GRANDCHILDENO	GRANDCHILDNAME	CHILDENO	CHILDNAME	PARENTENO	PARENTNAME
6000	FRANK	2000	JANET	1000	MOE
8500	GEORGE	8000	JOE	1000	MOE

Query-Pattern-7: Three-Level, Parent-Oriented, Matching

```
SELECT    GRANDCHILD.ENO GRANDCHILDENO,
          GRANDCHILD.ENAME GRANDCHILDNAME,
          CHILD.ENO CHILDENO,
          CHILD.ENAME CHILDNAME,
          PARENT.ENO PARENTENO,
          PARENT.ENAME PARENTENAME
FROM      REMPLOYEE GRANDCHILD,
          REMPLOYEE CHILD,
          REMPLOYEE PARENT
WHERE     GRANDCHILD.SENO = CHILD.ENO
AND       CHILD.SENO = PARENT.ENO
AND       GRANDCHILD.ENO IN ('3000', '6000', '8500')
```

 Copyright Pending 2020

30Z5. Reference the REROrTS_TO and the REMPLOYEE2 tables (representing a recursive many-to-many relationship). Display the numbers and names of Employees 3000, 5000, and 8500. If any of these employees is a supervisor, display each supervisee's number and name; and, if any of these supervisees is also a supervisor, display each of these supervisee's number and name. Sort the result by the supervisor's (the parent's) ENO value. The result should look like:

PARENTENO	PARENTNAME	CHILDENO	CHILDNAME	GRANDCHILDENO	GRANDCHILDNAME
3000	LARRY	6500	CURLY	7500	SHEMP
5000	JESSIE	4600	ELEANOR	4700	ANDY
5000	JESSIE	4600	ELEANOR	4800	MATT
5000	JESSIE	5500	HANNAH	4800	MATT
8500	GEORGE	8700	HANK	-	-
8500	GEORGE	8600	DICK	-	-

Query-Pattern-6: Three-Level, Parent-Oriented, Non-Matching

```
WITH REMPLOYEE (ENO, ENAME, SALARY, SENO)
AS
(SELECT RT.ENO, R2.ENAME, R2.SALARY, RT.SENO
FROM REPORTS_TO RT, REMPLOYEE2 R2
WHERE RT.ENO = R2.ENO)

SELECT PARENT.ENO              PARENTENO,
       PARENT.ENAME            PARENTNAME,
       CHILD.ENO               CHILDENO,
       CHILD.ENAME             CHILDNAME,
       GRANDCHILD.ENO          GRANDCHILDENO,
       GRANDCHILD.ENAME        GRANDCHILDNAME
FROM REMPLOYEE PARENT
     LEFT OUTER JOIN REMPLOYEE CHILD
          ON PARENT.ENO = CHILD.SENO
     LEFT OUTER JOIN REMPLOYEE GRANDCHILD
          ON CHILD.ENO = GRANDCHILD.SENO
WHERE  PARENT.ENO IN ('3000', '5000', '8500')
ORDER BY PARENT.ENO
```