IBM i DB2 and SQL School

(Course code OD47)

Instructor Exercises Guide

ERC 7.0

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

AS/400®	DB™	DB2 Universal Database™
DB2®	Domino®	DRDA®
Integrated Language Environment®	iSeries®	i5/OS™
Language Environment®	Notes®	OmniFind®
OS/400®	Power Systems™	Power Systems Software™
Power®	Rational®	Redbooks®
RPG/400®	Solid®	System i®
System p®	System x®	VisualAge®
400®		

Adobe is either a registered trademark or a trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Pentium is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other product and service names might be trademarks of IBM or other companies.

May 2013 edition

Contents

Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

AS/400®	DB™	DB2 Universal Database™
DB2®	Domino®	DRDA®
Integrated Language Environment®	iSeries®	i5/OS™
Language Environment®	Notes®	OmniFind®
OS/400®	Power Systems™	Power Systems Software™
Power®	Rational®	Redbooks®
RPG/400®	Solid®	System i®
System p®	System x®	VisualAge®
400®		

Adobe is either a registered trademark or a trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Pentium is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other product and service names might be trademarks of IBM or other companies.

Instructor exercises overview

The objective of the lab exercises is to have the students apply their newly gained knowledge in an application development environment.

Solutions: Sample solutions to the exercises can be found following each exercise of the instructor and student exercise guides.

The students will be asked to create a solution for each problem from scratch.

The user ID OD4700 with password OD47INST is reserved for instructor use.

Note: It is the responsibility of the instructor to be familiar with each lab and its solution in order to be able to explain the lab problem to the students at the beginning of each lab and in order to be able to help the students resolve any problems that they encounter in the lab exercises.

Depending on how your classroom was set up, and whether icons were placed on the desktop for System i Navigator and for the i, it might be necessary for you to lead short tutorials on opening System i Navigator and starting a 5250 session.

Copies of SAVFs for students

Many students like to take away a copy of their schemas so that they can use them back in the office. I encourage you to do this but it is the responsibility of the instructor to explain the process of saving to a save file and then using FTP to get the data onto their PC so that they can then attach to an e-mail and send them to themselves.

One approach follows:

Ask the students who want a copy of the class libraries to save their schemas, FTP the save file to the student PC and then e-mail it to themselves. Specifically the instructions are:

1. Create a savf named OD47xx (where xx is your team number) in the QTEMP library.
2. Save your team library for your schema `OD47xxCOL` (where xx is your user number) to this save file. One parameter to pay attention to is: **Level of IBM i** - Remind the students to save to their target IBM i level if it is lower than the classroom system.
3. To save the student library enter the command:

   ```
   SAVLIB LIB(od47xx) DEV(*SAVF) SAVF(QTEMP/OD47xx)
   ```

4. On your PC open a DOS session then get to the root by entering the command `cd\`
5. Next you will start an FTP session to the i by entering the following command: `FTP xx.xx.xx.xx` (where x refers to the IP address of the server)
6. Next you will be prompted for your user ID and password to sign on to the system. This user ID must have FTP capability to the server.
7. Once at the DOS prompt type in:

   a. `BIN`

   b. `QUOTE SITE NAMEFMT 1`

   c. `GET /qsys.lib/qtemp.lib/od47xx.savf c:\od47xx.sav`

      Note: Be sure to specify a fully qualified PC path which you will need to specify when you are send the file to yourself.

   d. `Quit`
8. It is recommended that the student should compress the file before they send it to themselves.
9. Open your e-mail software and send this file to yourselves.

Exercises configuration

IBM i version

The labs were created on Version 7 Release 1 of IBM i and have only been tested on V7R1. The class exercises savf was saved at V7R1.

Exercises description

Each student will be assigned a team or student ID by the instructor that will be used in the exercises. Student profiles are:

```
USRPRF(OD47xx)
PASSWRD(OD47pwd)
```

where the password is set to expire at first sign-on. Choose a new password that you can remember easily.

Each student profile (**OD47xx**) has authority to its own objects but does not have authority to other students' libraries.

Students are prompted to type STUDENT NAME when first signing on the IBM i. This information is stored in the user profile TEXT field with the date that the profile was first used. At each sign-on, the user's name in the user profile is concatenated with the user ID and inserted in the job's PRTTXT environment to help identify printed reports for the students.

Authority and solving lab problems

Each student has authority to copy any objects from the class schema OD47xxCOL (where xx is the version of the class; for example V1) that may be used to refresh lab objects and file members when required.

It is suggested that the first thing you do with each lab step is to read the problem from beginning to the end. This will give you an overall understanding of the problem to be solved. Once you have this understanding, then you should begin to compose your solution.

All of the objects that you require to solve the problems will be copied into your student schema, which you will create in the lab exercises. Should any table or view object be accidentally deleted or corrupted, you may refresh the appropriate object by copying the original master from the class schema **OD47xxCOL**. For the tables used in most of the exercises, there are members in QSQLSRC that you can use to recreate them. Instructions are included in the exercises.

Use all materials and resources available to you in the classroom. Some of the problems might require you to do some research beyond the lecture that was covered in the class. The IBM i Information Center Web site should be available in the classroom to assist you.

If you have any problem that you cannot solve on your own, please ask your instructor for assistance.

Exercise instructions are included in each exercise to explain the objectives that the student is to accomplish. Students are given the

opportunity to work through the lab exercises given what they learned in the unit presentation, utilizing the Student Notebook, their past experience and maybe a little intuition. Some exercises in the guide have skeleton programs to limit the student's coding to new functions covered in lecture. Other exercises do not have skeleton programs and the student is required to develop a complete solution.

Optional exercises are additional exercises to perform relating to certain units of lecture. They should be performed after you have completed the required exercises. The required exercises reinforce the most pertinent knowledge provided in the lecture material.

Refresh data: You might need to refresh tables as a result of making changes during the lab exercises. In case you need to perform this function, Exercise 2 documents the process and explains what to do.

Exercise 1. Getting to know SQL

Estimated time

00:30

What this exercise is about

In this exercise, you will use familiar IBM i objects (such as libraries and source files) and tools (such as PDM and SEU) to code SQL scripts. You will use these scripts to create and manipulate an SQL table.

What you should be able to do

At the end of the lab, you should be able to:

- Use basic SQL DDL statements to create SQL tables
- Use CL commands to investigate SQL objects

Introduction

In this exercise, you will create a simple SQL table using familiar IBM i features.

The table (STULST) will be designed to hold information about students attending a training course.

In subsequent exercises you will use a different approach, relying solely on SQL. In this exercise, you will see how traditional IBM i methods can be mixed with SQL.

Instructor exercise overview

The students will have the opportunity to use some of the SQL statements discussed in Unit 1, even though they are not yet familiar with the interactive SQL interface using the **STRSQL** command.

Before the students start this exercise, be sure you have assigned each student their user ID OL37*nn* (where *nn* is your team number) and the password OL37PWD. The password is set to expire at first sign-on so students will have to change the default password.

The students with experience using PDM will notice that the instructions are written to a basic level. This was done to accommodate all users who attend the class.

Exercise instructions

To perform this exercise, you will need to sign on to a 5250 session using your user ID, OL37nn which was assigned by the instructor.

The password, OL37PWD, is set to expire at your first sign-on. Choose a password that you can easily remember. Do not share this password with anyone else in class.

Setting up your environment

__ 1. Create a new IBM i library OL37nnLIB by entering the following command:

`CRTLIB OL37nnLIB` (where *nn* is your team number).

__ 2. Change your current library to OL37nnLIB by entering the following command:

```
CHGCURLIB OL37nnLIB
```

__ 3. Create a new source file QSQLSRC in your library, by entering the command:

```
CRTSRCPF QSQLSRC
```

This file will be used to contain various SQL scripts in separate source members.

Define an SQL table

__ 4. Navigate PDM to show the Work With Members display for your new source file:

__ a. Type **`STRPDM`**.

__ b. Select menu option **3.**

__ c. Specify file `QSQLSRC` in library `*CURLIB`.

__ 5. Press the **F6** = Create function key to create a new member in your source file. Fill in the information as follows:

```
Source member   CRTSTULST
Member Type     TXT
Text            Create STULST table
```

__ 6. In the SEU Edit session, type an SQL statement to create a table called STULST. The columns are defined as specified in the following table:

Column	Type	Size	Null?
FIRST_NAME	CHAR	15	no
MID_INIT	CHAR	1	yes
SURNAME	CHAR	25	no
CUST_NAME	VARCHAR	50	no
CUST_NO	CHAR	10	yes
TEAM_NO	NUMERIC	2,0	no
CRSE_CODE	CHAR	5	no
CRSE_DATE	DATE		no

Do not deviate from the naming, size, or type of data columns as listed above.

Your code should look similar to the following:

```
CREATE TABLE OL37nnLIB/STULST
   (FIRST_NAME CHAR(15) NOT NULL,
   MID_INIT CHAR(1),
         .
         .
         .

TEAM_NO NUMERIC(2,0) NOT NULL,
CRSE_CODE CHAR(5) NOT NULL,
CRSE_DATE DATE NOT NULL);
```

Note

For the purpose of the exercise, you must terminate our SQL statement with a semicolon (;). The above example is incomplete as it does not list all of the data columns. If you are not sure how to code this complete entry in the `QSQLSRC` file, or if you would like to check what you have already entered against the answer, review the information listed in the Answers section of the appendix. Remember that SQL is a free-format language, so that your source code can be arranged for readability and its lines can be repeated (copied) for ease of editing.

__ 7. Exit SEU and save your source member (press **F3** and then press **Enter**).

Create the STULST table

__ 8. Run the SQL script in member CRTSTULST from the command line by using the following **RUNSQLSTM** command:

```
RUNSQLSTM SRCFILE(*CURLIB/QSQLSRC) SRCMBR(CRTSTULST)
COMMIT(*NONE) NAMING(*SYS)
```

Remember that the **RUNSQLSTM** command allows you to run a series of (multiple) SQL statements.

__ 9. Review the results of the previous step. A log is produced from RUNSQLSTM. Review your printout using the **WRKSPLF** command.

Were there any errors? Look carefully at any messages.

_____

Was the table created successfully? If not, correct any mistakes in the CRTSTULST source member and rerun the **RUNSQLSTM** command until you are successful.

_____

__ 10. You should now have a table (STULST) in your current library. Explore this table using the **DSPFD** and **DSPFFD** CL commands:

```
DSPFD STULST
DSPFFD STULST
```

__ 11. Use DFU to add a few records of your choice to your STULST table. Be sure to enter at least one record with a blank (versus nothing or a letter) for the middle initial (MID_INIT) data column. Enter the following command to add some data into your file:

```
UPDDTA STULST
```

__ 12. Review the results of adding your data by using either (or both) of the following commands:

```
DSPPFM    STULST
RUNQRY *N STULST
```

End of exercise

Instructor exercises review/wrap-up

This exercise demonstrated a number of points that you should review. You might let student's ask questions or simply review the following points:

- SQL tables are IBM i physical files.
- Since SQL can access and manipulate data in SQL tables, it can also access and manipulate data in IBM i PFs.
- **`RUNSQLSTM`** is a powerful command that is included with the IBM i operating system. Thus, SQL scripts in a source member can be run on any IBM i, even if the SQL Development Kit is not installed.
- When viewing data with the **`RUNQRY`** command, NULL columns that contain null values are identified with a hyphen, '-' versus actual data such as a letter or blank.

Exercise 2. How SELECT works

Estimated time

00:30

What this exercise is about

This exercise introduces you to the power of the SELECT statement as a data extraction tool. You also practice coding the SELECT statement clauses in the correct order.

What you should be able to do

At the end of the lab, you should be able to:

- Code a simple SELECT statement
- Explain the order in which the SQL SELECT statement must be coded

Introduction

The EMPLOYEE table of the Corporate Database is used in this exercise as well as in subsequent exercises. While you are performing this exercise, it might help to have details of the EMPLOYEE table available. Refer to Appendix A.

Instructor introduction

This lab includes some setup work using SQL scripts that must be completed by the students. These scripts must be customized by the student, which requires reasonable SEU skills. Some students will possess little or no programming background, and perhaps no expertise in SEU at all. You should lead the class through the process as a group. Follow all the steps, including the ones that use System i Navigator to view the tables' contents and properties.

This should alleviate any problems that the students might have.

Exercise instructions

Set up the exercise environment

In this exercise, you begin to use the query facilities of SQL. However, before you start, we want you to perform some basic tasks that will set up your own schema and some tables necessary for all subsequent exercises.

Your instructor will lead you through this activity, so wait until you are told to start the exercise.

__ 1. To perform this exercise, you will need to sign on to a 5250 session using your user ID, OL37nn.

__ 2. Enter the command **STRSQL** to start interactive SQL (ISQL). You should see a display that looks similar to the IBM i command entry screen. There will be different information at the top of the display. You can enter any SQL command on these lines and then execute each command.

__ 3. Create your schema (if not already done), by typing the following command:

CREATE SCHEMA OL37nnCOL (where *nn* is your assigned team number)

__ 4. In your ISQL session, enter **CALL QCMD**. You should see an IBM i command entry display. Press **F11** to display the full command entry screen.

__ 5. Change your current library to OL37nnCOL by entering the following command:

CHGCURLIB OL37nnCOL (where *nn* is your team number)

__ 6. Change your user profile so that your schema becomes your current library (*CURLIB) each time you sign on to the IBM i. Enter the following command:

CHGPRF CURLIB(OL37*nn*COL)

__ 7. From the student master schema OL37XXCOL, copy the source file QCLSRC and all its members to your source file QCLSRC in your student schema, OL37*nn*COL (where *nn* is your team number). Use the following command to copy the source file and all members to your schema (remember to change *nn* to your team number). Remember that when using prompting (F4) with commands, the object is first followed by the library unlike the command below which has library first then object name:

```
CPYF FROMFILE(OL37XXCOL/QCLSRC)
   TOFILE(OL37nnCOL/QCLSRC)
   FROMMBR(*ALL)
   TOMBR(*FROMMBR)
   MBROPT(*REPLACE)
   CRTFILE(*YES)
```


__ 8. From the student master schema (OL37XXCOL), copy the source file QSQLSRC and all its members to your source file QSQLSRC in your student schema, OL37*nn*COL (where *nn* is your team number). Use the following command to copy the source file and all members to your schema (remember to change *nn* to your team number):

```
CPYF FROMFILE(OL37XXCOL/QSQLSRC)
   TOFILE(OL37nnCOL/QSQLSRC)
   FROMMBR(*ALL)
   TOMBR(*FROMMBR)
   MBROPT(*REPLACE)
   CRTFILE(*YES)
```

__ 9. Use **CPYF** to copy the following file from the student master schema, OL37XXCOL, to your schema, OL37*nn*COL:

```
APINV_PF *FILE PF-DTA AP Invoice Physical File

CPYF FROMFILE(OL37XXCOL/APINV_PF)
   TOFILE(OL37nnCOL/APINV_PF)
   FROMMBR(*FIRST)
   TOMBR(*FIRST)
   MBROPT(*REPLACE)
CRTFILE(*YES)
```

__ 10. Enter the command **WRKMBRPDM** and press **F4**. Specify:

File QSQLSRC

Library OL37*nn*COL

(where *nn* is your team number). Then press **Enter**.

__ 11. Select option **2** =Edit for the following members per the instructions in the next step.

```
Member     Type   Text
REFTABLE   SQL    SQL Statements to RECREATE student tables for labs
```

__ a. On the SEU command line at the top of the edit member screen, type the following and press **Enter**. This command will find and change all occurrences of OL37XXCOL to your student team number. For example, team *nn* would become OL37*nn*COL.

```
C OL37XXCOL OL37nnCOL ALL
```

__ b. Press **F3** to exit and save these changes.

Note

You do not compile SQL source statements. They are prepared one by one prior to execution and then executed.

__ 12. Repeat the previous step to make the same changes to a second member:

```
Member     Type   Text
TABLEX     SQL    SQL Statements to insert student tables with data
```

Note

You should see an SEU message that says “String OL37XXCOL changed 141 times.”

__ 13. Execute the following command to create the exercise tables in your schema:

```
RUNSQLSTM SRCFILE(OL37nnCOL/QSQLSRC) SRCMBR(REFTABLE)
COMMIT(*NONE) NAMING(*SQL)
```

Remember to change the *nn* to your team number.

Review the contents of the spooled output once the job is completed.

Next, execute the following command to insert values (data) in the columns of each table in your student schema:

```
RUNSQLSTM SRCFILE(OL37nnCOL/QSQLSRC) SRCMBR(TABLEX)
COMMIT(*NONE) NAMING(*SQL)
```

Review the contents of your output queue once the job is completed.

__ 14. Optionally, should you need to refresh all of your tables at any time during the exercises, perform the following steps:

__ a. Enter the command **WRKMBRPDM** and press **F4**. Specify:

File QCLSRC

Library OL37*nn*COL

(where *nn* is your team number). Then press **Enter**.

__ b. Select option **14**=Compile for the following member.

```
Member     Type   Text
DELTABLES CLP Delete Student Corporate DB Tables
```

Review the contents of your output queue to verify the compile of the program.

__ c. Whenever you need to refresh your exercise tables, run the CL program with the command, **CALL DELTABLES**.

__ d. **RUNSQLSTM SRCFILE(OL37*nn*COL/QSQLSRC) SRCMBR(REFTABLE) COMMIT(*NONE) NAMING(*SQL)**

__ e. **RUNSQLSTM SRCFILE(OL37*nn*COL/QSQLSRC) SRCMBR(TABLEX) COMMIT(*NONE) NAMING(*SQL)**

Remember to change the *nn* above to your team number.

You have just learned that you can execute SQL statements outside of interactive SQL using the **RUNSQLSTM** command. You noted that all you need to do is to place your SQL statements in a source member (using SEU). The statements are parsed and run one statement at a time.

However, the SELECT statement cannot be run using **RUNSQLSTM**.

__ 15. Press **F3** to exit QCMD. You should be returned to interactive SQL.

Familiarize yourself with the data

__ 16. Look in Appendix A to review the definition of the columns in each table.

__ 17. Open System i Navigator.

__ 18. Expand the **Database** branch, then expand the next branch of the tree.

__ 19. Right-click **Schemas** and select **Select schemas to display**.

__ 20. On the **Select** schemas to display panel, enter your schema OL37*nn*COL then click the **Add** button. You should now see your schema listed in the right window for Selected schemas. Click **OK** to close this window.

__ 21. Next, expand the **Schemas** branch then expand the link for your schema. Then click the **Tables** link for your schema. You should find a copy of the EMPLOYEE, DEPARTMENT, PROJECT, and EMP_ACT tables in your student schema.

__ 22. If you would like to view table definition details, you can either right-click a table and select **Definition**, or double left-click the table. A right-click with selection of **Description** will display table properties in similar fashion to the IBM i command **DSPFD** (Display File Description). A right-click, **View Contents** or a right-click, **Edit Contents** will allow viewing or editing access to the data in the table, respectively.

Additional information that describes the tables can also be found in your lecture notebook.

Coding SELECT statements

Your manager has learned that you are taking an SQL course and that one of the topics is using the SELECT statement. You are asked to code the statement that will tell us which departments in our company pay an average salary higher than 20,000.00 for the jobs performed in each department. For these departments, you want to know the department

number, job title, and average salary sorted in descending sequence by the average salary. You are only interested in the salaries of non-managers.

You might determine that this is a challenging task. Let us try to solve the problem by analyzing the various components of the problem. In the steps that follow, execute the instructions and be prepared to discuss your answers in a lab review.

__ 23. In this exercise, use a 5250 emulation session.

__ 24. Next, start the interactive SQL interface.

__ a. On the IBM i command line, enter **STRSQL** and press **F4** to review your session defaults.

__ b. Change the **Library** option parameter to your schema OL37*nn*COL (where *nn* is your team number)**.**

__ c. Press **Enter** to start your SQL session.

__ 25. Familiarize yourself with the table:

On the line to the right of the arrow (===>), code the SELECT statement to display *all columns and all rows* in the EMPLOYEE table. If you need assistance or are not sure how to code this statement, look at the appendix section in this lab guide for the answer.

Review the columns in the table. Look at the JOB, WORKDEPT, and SALARY columns specifically. Use **F19** and **F20** to scroll left and right through the data values. Use the **Page Up** and **Page Down** keys to scroll up and down.

You should see a subset of the rows and columns similar to the following:

EMPNO	FIRSTNME	MIDINIT	LASTNAME	WORKDEPT	PHONENO
000010	Christine	I	Haas	A00	3978
000020	Michael	L	Thompson	B01	3476
000030	Sally	A	Kwan	C01	4738
000050	John	B	Geyer	E01	6789
000060	Irving	F	Stern	D11	6423
000070	Eva	D	Pulaski	D21	7831
000090	Eileen	W	Henderson	E11	5498
000100	Theodore	Q	Spenser	E21	0972
000110	Vicenzo	G	Lucchesi	A00	3490
000120	Sean		O Connell	A00	2167
000130	Delores	M	Quintana	C01	4578
000140	Heather	A	Nicholls	C01	1793
000150	Bruce		Adamson	D11	4510
000160	Elizabeth	R	Pianka	D11	3782
000170	Masatoshi	J	Yoshimura	D11	2890
000180	Marilyn	S	Scoutten	D11	1682

Press **Enter** to return to the Enter SQL Statements panel.

__ 26. Next, code the SQL statements to select only the columns that you want.

Now that you have seen all columns, code the SELECT statement to display only the columns listed below. Your result should look like the following:

```
WORKDEPT  JOB              SALARY
A00       PRES          52,750.00
B01       MANAGER       41,250.00
C01       MANAGER       38,250.00
E01       MANAGER       40,175.00
D11       MANAGER       32,250.00
D21       MANAGER       36,170.00
E11       MANAGER       29,750.00
E21       MANAGER       26,150.00
A00       SALESREP      46,500.00
A00       CLERK         29,250.00
C01       ANALYST       23,800.00
C01       ANALYST       28,420.00
D11       DESIGNER      25,280.00
D11       DESIGNER      22,250.00
D11       DESIGNER      24,680.00
D11       DESIGNER      21,340.00
              :
              :
```

If you need assistance or are not sure how to code this statement, look for the answer in the appendix section of this lab guide.

What changed from the previous step? What are the differences in selecting some specific columns compared to using the SELECT * in the previous step?

_____

_____

_____

_____

Rule for coding SELECT

Remember the order in which clauses must be coded in a SELECT:

```
SELECT ....
   FROM ....
      WHERE ....
      GROUP BY ....
      HAVING ....
      ORDER BY ....
```

The clauses that you code in a SELECT statement must be coded in this exact order. If you code the order of the clauses incorrectly, you will see an error message.

__ 27. Next, code the SQL statement to include only non-managers This means that you want to exclude the PRES and MANAGER job titles.

```
select workdept, job, salary from employee
   where job not in ('PRES', 'MANAGER')
```

You will have more rows than this subset, but your result set should look similar to the following:

```
WORKDEPT  JOB             SALARY
A00       SALESREP     46,500.00
A00       CLERK        29,250.00
C01       ANALYST      23,800.00
C01       ANALYST      28,420.00
D11       DESIGNER     25,280.00
D11       DESIGNER     22,250.00
              :
              :
```

What changed from the previous step? What did the WHERE clause do?

_____

_____

_____

_____

Solving the rest of the problem

So far, you have a result set that contains the salary paid to individual non-managers in each department for each job.

You need to determine the average salary for each job in each department and then determine whether that average is greater than 20,000.00. In order to do this, you need to GROUP the result set by JOB within WORKDEPT.

__ 28. To accomplish this requirement, code the following statement:

```
select workdept, job, salary from employee
   where job not in ('PRES', 'MANAGER')
   group by workdept, job
```

Enter this statement and run it.

What happened and why did it happen? (The answer is provided in the appendix section of this lab guide.)

_____

_____

The necessary SQL clause to accomplish the requirement for this lab step is covered in the next unit.

If you add the function to determine the average of the salaries in the departments, you will solve the requirement posed in this lab step. We discuss some of the many functions in SQL in the next unit. For now, know that there is a function that you can use to determine the average of a column. It is named AVG.

__ 29. Enter the following statement:

```
select workdept, job, avg(salary) from employee
   where job not in ('PRES', 'MANAGER')
   group by workdept, job
```

Your result set should look like the following:

```
WORKDEPT  JOB                          AVG ( SALARY )
A00       CLERK        29,250.00000000000000000000000000
A00       SALESREP     46,500.00000000000000000000000000
C01       ANALYST      26,110.00000000000000000000000000
D11       DESIGNER     23,731.25000000000000000000000000
D21       CLERK        22,950.00000000000000000000000000
E11       OPERATOR     18,810.00000000000000000000000000
E21       FILEREP      23,053.33333333333333333333333333
```

What do you have now?

_____

_____

What other steps must you perform in order to get the desired result?

_____

_____

__ 30. In this step, we look at the ORDER BY clause and how you can use it to further customize the look of your data. The next requirement is the need to ORDER the results in descending order by average salary. Type the following:

```
select workdept, job, avg(salary) from employee
   where job not in ('PRES', 'MANAGER')
   group by workdept, job
   order by avg(salary) desc
```

Your result set should look like this:

```
WORKDEPT  JOB                          AVG ( SALARY )
A00       SALESREP     46,500.00000000000000000000000000
A00       CLERK        29,250.00000000000000000000000000
C01       ANALYST      26,110.00000000000000000000000000
D11       DESIGNER     23,731.25000000000000000000000000
E21       FILEREP      23,053.33333333333333333333333333
D21       CLERK        22,950.00000000000000000000000000
E11       OPERATOR     18,810.00000000000000000000000000
```

Complete the problem

__ 31. Finally, the last requirement calls for only those departments with an average salary greater than 20,000.00. Code and run the following SQL statement:

```
select workdept, job, avg(salary) from employee
    where job not in ('PRES', 'MANAGER')
    group by workdept, job
    having avg(salary) > 20000.00
    order by avg(salary) desc
```

Your result set should look like this:

```
WORKDEPT  JOB                          AVG ( SALARY )
  A00     SALESREP     46,500.00000000000000000000000000
  A00     CLERK        29,250.00000000000000000000000000
  C01     ANALYST      26,110.00000000000000000000000000
  D11     DESIGNER     23,731.25000000000000000000000000
  E21     FILEREP      23,053.33333333333333333333333333
  D21     CLERK        22,950.00000000000000000000000000
```

You can now tell your manager that you have the answer to his question.

End of exercise

Instructor exercise review/wrap-up

Review this exercise with the class going through each of the answers that they should have written down in their lab notebooks.

Exercise 3. Using the SELECT statement

Estimated time

01:00

What this exercise is about

This exercise provides an opportunity to code and test your understanding of the SQL SELECT statement using an interactive SQL interface, either STRSQL or System i Navigator.

What you should be able to do

At the end of the lab, you should be able to:

- Code simple SELECT statements
- Use the ORDER BY and WHERE clauses in a SELECT statement

Introduction

The Corporate Data tables are used in this exercise as well as subsequent exercises. During this exercise, you will become more familiar with the tables.

Choose your SQL environment

In this exercise, as well as subsequent exercises, you can use **either** System i Navigator's Run SQL Scripts facility, or the 5250 interactive STRSQL interface, or a combination of both.

Decide which interface you want to use, and perform **one** of the following (do not perform both since the labs are a mirror image of each other):

Exercise instructions for 5250 emulator session

or

Exercise instructions for System i Navigator

For each query, a problem is stated followed by the desired output. The output format that you see will depend on which tool you use to perform the exercise.

Remember to save your SQL script or your STRSQL session. You can then initiate your sessions and run your SELECT statements with the advantage of duplicating and using prior statements without having to recode your SELECTs from scratch.

The answers to all of the lab questions (that is, how to code an SQL statement to produce the desired outcome) are listed in the answers section of the appendix.

Exercise instructions for 5250 emulator session

__ 1. Start a 5250 emulation session and sign on with your OL37xx user ID and password.

__ 2. First, set up your SQL environment. Issue the **STRSQL** command then press **F4**.

__ 3. Next, set the following parameters:

Naming convention: *SYS

Date format: *ISO

Press **Enter**.

In the 5250 interface, if you encounter any problems with the date column in a result set, you can always verify (and change) your date format by pressing **F13** while in the interactive interface.

Code SELECT statements

__ 4. Code the SELECT statement that will display the following columns from the Employee table:

- Last name
- First name
- Work department
- Date of birth
- Hire date
- Salary of each employee in the EMPLOYEE table whose salary is equal to 30,000 or more.

__ a. Press **Enter** to process your statement.

Your output should look like this:

LASTNAME	FIRSTNME	WORKDEPT	BIRTHDATE	HIREDATE	SALARY
Haas	Christine	A00	1933-08-24	1965-01-01	52,750.00
Thompson	Michael	B01	1948-02-02	1973-10-10	41,250.00
Kwan	Sally	C01	1941-05-11	1975-04-05	38,250.00
Geyer	John	E01	1925-09-15	1949-08-17	40,175.00
Stern	Irving	D11	1945-07-07	1973-09-14	32,250.00
Pulaski	Eva	D21	1953-05-26	1980-09-30	36,170.00
Lucchesi	Vicenzo	A00	1929-11-05	1958-05-16	46,500.00

Note that if you do not see the date columns in the format above, you will have to change your session defaults such that the date format is specified *ISO.

__ 5. Modify the SELECT statement that you just coded to:

- List those employees whose salary is less than 20,000
- Order your output by first name within last name

__ a. Remember that you can use the **F9** key to retrieve the previous SQL statement. Make any necessary modifications, then press **Enter** to rerun the statement.

Your output should look like this:

```
LASTNAME        FIRSTNME     WORKDEPT  BIRTHDATE   HIREDATE       SALARY
Johnson         Sybil          D21     1936-10-05  1975-09-11   17,250.00
Jones           William        D11     1953-02-23  1979-04-11   18,270.00
Mehta           Ramlal         E21     1932-08-11  1965-07-07   19,950.00
Parker          John           E11     1946-07-09  1980-05-30   15,340.00
Setrignt        Maude          E11     1931-04-21  1964-09-12   15,900.00
Smith           Daniel         D21     1939-11-12  1969-10-30   19,180.00
Smith           Philip         E11     1936-10-27  1972-06-19   17,750.00
```

__ 6. List all columns in the DEPARTMENT table where the manager number is blank or unknown.

Your output should look like this:

```
DEPTNO  DEPTNAME                       MGRNO   ADMRDEPT
  D01    Development Center                       A00
```

__ 7. Code the statement to produce the following from the DEPARTMENT table. SELECT only those departments whose reporting department (ADMRDEPT) is A00:

```
DEPTNO  DEPTNAME                       MGRNO   ADMRDEPT
 A00    SPIFFY Computer Service Div.   000010    A00
 B01    Planning                       000020    A00
 C01    Information Center             000030    A00
 D01    Development Center                       A00
 E01    Support Services               000050    A00
```

__ 8. Working with the DEPARTMENT table, list the department number and department name where the department column contains the word “Service”.

Your output should look like this:

```
DEPTNO  DEPTNAME
 A00    SPIFFY Computer Service Div.
 E01    Support Services
```

__ 9. Working with the EMPLOYEE file, list:

- Employee last name
- First name
- Work department

- Phone number
- Choose only those employees in department E01 through E11. (Make sure to exclude E21.)

  Your output should look like the following listing:

```
LASTNAME        FIRSTNME      WORKDEPT  PHONENO

Geyer           John            E01       6789
Henderson       Eileen          E11       5498
Schneider       Ethel           E11       8997
Parker          John            E11       4502
Smith           Philip          E11       2095
Setrignt        Maude           E11       3332
```

__ 10. Using the EMPLOYEE file, code the SELECT statement to produce the following:

```
LASTNAME        FIRSTNME      WORKDEPT  PHONENO

Geyer           John            E01       6789
Henderson       Eileen          E11       5498
Spenser         Theodore        E21       0972
Schneider       Ethel           E11       8997
Parker          John            E11       4502
Smith           Philip          E11       2095
Setrignt        Maude           E11       3332
Mehta           Ramlal          E21       9990
Lee             Wing            E21       2103
Gounot          Jason           E21       5698
```

__ 11. From the EMPLOYEE table, select or list:

__ a. Only those employees in work departments B01, C01, and D01

__ b. Calculate the sum of salary plus commission.

__ c. Sequence the results by the total of salary plus commission in descending sequence.

Your report should look like the following:

```
LASTNAME        FIRSTNME      WORKDEPT  SALARY + COMM

Thompson        Michael         B01         44,550.00
Kwan            Sally           C01         41,310.00
Nicholls        Heather         C01         30,694.00
Quintana        Delores         C01         25,704.00
```

Note

There are no employees in Department D01, as the sample results show. D01 exists in the DEPARTMENT table, but no rows exist in the EMPLOYEE table.

__ 12. Using the EMPLOYEE table, code the statement to:

- Display last name, salary. and work department only
- Whose monthly salary is greater than 3000.00
- List all employees by LASTNAME.

Your results should look like this:

```
LASTNAME                 SALARY   WORKDEPT

Geyer                 40,175.00      E01
Haas                  52,750.00      A00
Kwan                  38,250.00      C01
Lucchesi              46,500.00      A00
Pulaski               36,170.00      D21
Thompson              41,250.00      B01
```

__ 13. Using the EMPLOYEE table, code the SELECT statement that:

- Lists all male employees
- Whose monthly salary is less than 1600.00
- List the rows in descending order of monthly salary.

Your results should look like this:

```
EMPNO   LASTNAME                                  SALARY / 12

000250  Smith                 1,598.333333333333333333333333
000210  Jones                 1,522.500000000000000000000000
000300  Smith                 1,479.166666666666666666666666
000290  Parker                1,278.333333333333333333333333
```

__ 14. Code the SQL statement that answers the question: What percentage of total earnings is commission for our sales representatives? List the last name and the percentage.

Your results should look like this:

```
LASTNAME                                Numeric Expression

Lucchesi                                     7.27699530516
```


__ 15. Code the SQL statement that lists all information in the department table for department E01 or for all departments that report to E01 (reporting department is column ADMRDEPT).

Your results should look like this:

```
DEPTNO  DEPTNAME                  MGRNO   ADMRDEPT

 E01    Support Services          000050    A00
 E11    Operations                000090    E01
 E21    Software Support          000100    E01
```

__ 16. Code the SQL statement that answers the question: Which employees (including managers) have an annual salary greater than 40000.00 or are managers with an education level less than 16?

Your results should be similar to:

```
LASTNAME                SALARY   JOB       EDUCLVL

Haas                 52,750.00   PRES           18
Thompson             41,250.00   MANAGER        18
Geyer                40,175.00   MANAGER        16
Spenser              26,150.00   MANAGER        14
Lucchesi             46,500.00   SALESREP       19
```

Exercise instructions for System i Navigator

__ 1. Click the **System i Navigator** icon and expand (start a connection to) your IBM i. When prompted sign on with your OL37nn user ID and password.

__ 2. First, set up your SQL environment. Expand the **Databases** branch. Next, expand the branch below Databases (note the name of this branch will be the name assigned to the relational database on your IBM i using the **WRKRDBDIRE**), then expand schemas.

   __ a. Right-click **Schemas** then click **Select schemas to display.**

   __ b. On the Select schemas to display panel, specify `OL37nnCOL` (where *nn* is your team number), then click **Add** if it is not already listed.

   __ c. Also specify or add OL37V7COL and OL37XXCOL (separated with a comma: OL37V7COL, OL37XXCOL); remember in this case, XX is the actual team for the student master collection (do not change this to your team number).

   __ d. Click **OK** to add these collections. You should now see all three listed under schemas.

__ 3. Now return to the second branch (named Relational Database branch) off of Databases and right-click it. Click **Run SQL Scripts**.

   __ a. On the Run SQL Scripts panel, click the **Connection** menu option, then select **JDBC Settings** from the drop-down.

   __ b. On the JDBC Settings panel **System** tab,

      i. Set the SQL default schema parameter to your schema OL37*nn*COL (where *nn* is your team number) by selecting it from the drop-down.

      ii. For the schema list parameter enter the class schemas (note these two values are separated by a comma: OL37V7COL, OL37XXCOL).

   __ c. Next click the **Format** tab and specify:

      i. Naming convention: `*SQL`

      ii. Time: `*ISO`

      iii. Date: `*ISO`

__ 4. Click **Connect** to save these changes.

Code SELECT statements

__ 5. You should now be back at the Run SQL Scripts panel.

__ 6. Code the SELECT statement that will display the following columns from the Employee table.

   For this step and all of the remaining Navigator steps there are two options when typing in your SQL statements. First, you can directly type in the SELECT statement

on this panel which requires that you know the format of the statement. Remember to end your SELECT statement with a semicolon (;). Second, you can type in your SQL statement, in this case the SELECT statement, then press the **F4** key to call up the SQL Assist wizard.

- Last name
- First name
- Work department
- Date of birth
- Hire date
- Salary of each employee in the EMPLOYEE table whose salary is equal to 30,000 or more

__ a. Click **Run** then click **All** from the drop-down menu.

Your output should look like this:

LASTNAME	FIRSTNME	WORKDEPT	BIRTHDATE	HIREDATE	SALARY
Haas	Christine	A00	1933-08-24	1965-01-01	52,750.00
Thompson	Michael	B01	1948-02-02	1973-10-10	41,250.00
Kwan	Sally	C01	1941-05-11	1975-04-05	38,250.00
Geyer	John	E01	1925-09-15	1949-08-17	40,175.00
Stern	Irving	D11	1945-07-07	1973-09-14	32,250.00
Pulaski	Eva	D21	1953-05-26	1980-09-30	36,170.00
Lucchesi	Vicenzo	A00	1929-11-05	1958-05-16	46,500.00

Note that if you do not see the date columns in the format above, you must change your session defaults such that the date format is specified *ISO.

__ 7. Modify the SELECT statement that you just coded to:

- List those employees whose salary is less than 20,000
- Order your output by first name within last name

Your output should look like this:

LASTNAME	FIRSTNME	WORKDEPT	BIRTHDATE	HIREDATE	SALARY
Johnson	Sybil	D21	1936-10-05	1975-09-11	17,250.00
Jones	William	D11	1953-02-23	1979-04-11	18,270.00
Mehta	Ramlal	E21	1932-08-11	1965-07-07	19,950.00
Parker	John	E11	1946-07-09	1980-05-30	15,340.00
Setrignt	Maude	E11	1931-04-21	1964-09-12	15,900.00
Smith	Daniel	D21	1939-11-12	1969-10-30	19,180.00
Smith	Philip	E11	1936-10-27	1972-06-19	17,750.00

__ 8. List all columns in the DEPARTMENT table where the manager number is blank or unknown.

Your output should look like this:

```
DEPTNO  DEPTNAME                        MGRNO   ADMRDEPT

 D01    Development Center                       A00
```

__ 9. Code the statement to produce the following from the DEPARTMENT table. SELECT only those departments whose reporting department (ADMRDEPT) is A00:

```
DEPTNO  DEPTNAME                        MGRNO   ADMRDEPT

 A00    SPIFFY Computer Service Div.    000010   A00
 B01    Planning                        000020   A00
 C01    Information Center              000030   A00
 D01    Development Center                       A00
 E01    Support Services                000050   A00
```

__ 10. Working with the DEPARTMENT table, list the department number and department name where the department column contains the word “Service”.

Your output should look like this:

```
DEPTNO  DEPTNAME

 A00    SPIFFY Computer Service Div.
 E01    Support Services
```

__ 11. Working with the EMPLOYEE file, list the:

- Employee last name
- First name
- Work department
- Phone number
- Choose only those employees in department E01 through E11. (Make sure to exclude E21.)

Your output should look like the following listing:

```
LASTNAME        FIRSTNME      WORKDEPT  PHONENO

Geyer           John            E01      6789
Henderson       Eileen          E11      5498
Schneider       Ethel           E11      8997
Parker          John            E11      4502
Smith           Philip          E11      2095
Setrignt        Maude           E11      3332
```


__ 12. Using the EMPLOYEE file, code the SELECT statement to produce the following:

LASTNAME	FIRSTNME	WORKDEPT	PHONENO
Geyer	John	E01	6789
Henderson	Eileen	E11	5498
Spenser	Theodore	E21	0972
Schneider	Ethel	E11	8997
Parker	John	E11	4502
Smith	Philip	E11	2095
Setrignt	Maude	E11	3332
Mehta	Ramlal	E21	9990
Lee	Wing	E21	2103
Gounot	Jason	E21	5698

__ 13. From the EMPLOYEE table, select or list:

__ a. Only those employees in work departments B01, C01, and D01

__ b. Calculate the sum of salary plus commission.

__ c. Sequence the results by the total of salary plus commission in descending sequence.

Your report should look like the following:

LASTNAME	FIRSTNME	WORKDEPT	SALARY + COMM
Thompson	Michael	B01	44,550.00
Kwan	Sally	C01	41,310.00
Nicholls	Heather	C01	30,694.00
Quintana	Delores	C01	25,704.00

Note

There are no employees in Department D01, as the sample results show. D01 exists in the DEPARTMENT table, but no rows exist in the EMPLOYEE table.

__ 14. Using the EMPLOYEE table, code the statement to:

- Display last name, salary, and work department only
- Whose monthly salary is greater than 3000.00
- List all employees by LASTNAME.

Your results should look like this:

```
LASTNAME                SALARY    WORKDEPT

Geyer                40,175.00     E01
Haas                 52,750.00     A00
Kwan                 38,250.00     C01
Lucchesi             46,500.00     A00
Pulaski              36,170.00     D21
Thompson             41,250.00     B01
```

__ 15. Using the EMPOYEE table, code the SELECT statement that:

- Lists all male employees
- Whose monthly salary is less than 1600.00
- List the rows in descending order of monthly salary.

Your results should look like this:

```
EMPNO   LASTNAME                                SALARY / 12

000250  Smith              1,598.333333333333333333333333
000210  Jones              1,522.500000000000000000000000
000300  Smith              1,479.166666666666666666666666
000290  Parker             1,278.333333333333333333333333
```

__ 16. Code the SQL statement that answers the question: What percentage of total earnings is commission for our sales representatives? List last name and the percentage.

Your results should look like this:

```
LASTNAME                              Numeric Expression

Lucchesi                                   7.27699530516
```

__ 17. Code the SQL statement that lists all information in the department table for department E01 or for all departments that report to E01 (reporting department is column ADMRDEPT).

Your results should look like this:

```
DEPTNO  DEPTNAME                    MGRNO    ADMRDEPT

 E01    Support Services            000050     A00
 E11    Operations                  000090     E01
 E21    Software Support            000100     E01
```


__ 18. Code the SQL statement that answers the question: Which employees (including managers) have an annual salary greater than 40000.00 or are managers with an education level less than 16?

Your results should be similar to:

```
LASTNAME                SALARY    JOB        EDUCLVL

Haas                 52,750.00    PRES            18
Thompson             41,250.00    MANAGER         18
Geyer                40,175.00    MANAGER         16
Spenser              26,150.00    MANAGER         14
Lucchesi             46,500.00    SALESREP        19
```

End of exercise

Exercise 4. Using SQL functions

Estimated time

00:45

What this exercise is about

This exercise provides an opportunity to use SQL column and scalar functions within SELECT statements.

What you should be able to do

At the end of the lab, you should be able to:

- Code column functions
- Code scalar functions

Choose your SQL environment

In this exercise, as well as subsequent exercises, you can use **either** System i Navigator's Run SQL Scripts facility, or the 5250 interactive STRSQL interface, or a combination of both.

If you have not yet used the Navigator interface (Run SQL Scripts), revisit the **Lab Exercise, Using the SELECT Statement instructions for System i Navigator** and perform the steps numbered one through four (1 - 4) in order to establish your Navigator SQL environment. The instructions that follow can be used in either environment.

Exercise instructions for using SQL functions

For each query, a problem is stated followed by the desired output. The output format that you see will depend on which tool you use to perform the exercise.

Remember to save your SQL script or your STRSQL session. You can then initiate your sessions and run your SELECT statements with the advantage of duplicating and using prior statements without having to recode your SELECTs from scratch.

The answers to all of the lab questions (that is, how to code an SQL statement to produce the desired outcome) are listed in the Answers section of the appendix.

Using SQL Functions

Coding functions

__ 1. Initiate your 5250 or Navigator session. Verify that your current library is set to OL37*nn*COL (where *nn* is your team number).

__ 2. Make sure that your interactive SQL or Navigator JDBC settings are set the way you want.

__ 3. Display the total, average, minimum, and maximum salaries for our employees. Your output should be similar to this:

```
SUM(SALARY)  AVG(SALARY)                          MIN(SALARY)  MAX(SALARY)

873,715.00   27,303.59375000000000000000000000 15,340.00    52,750.00
```

__ 4. Code the SQL statement to determine the employee with the last name that is lowest in the collating sequence. Your output should be similar to this:

```
Adamson
```

__ 5. Code the SQL statement to determine how many departments in which our employees work are in our company? You should get an answer equal to 8.

__ 6. Code the SQL statement to determine the average salary for each job classification. Your output should be similar to this:

```
JOB          AVERAGE_SALARY

ANALYST      26,110.00000000000000000000000000
CLERK        24,000.00000000000000000000000000
DESIGNER     23,731.25000000000000000000000000
FILEREP      23,053.33333333333333333333333333
MANAGER      34,856.42857142857142857142857142
OPERATOR     18,810.00000000000000000000000000
PRES         52,750.00000000000000000000000000
SALESREP     46,500.00000000000000000000000000
```

__ 7. Modify the previous SELECT to list those jobs with an average salary greater than 35000.00. Your output should be similar to this:

```
JOB          AVERAGE_SALARY

PRES         52,750.00000000000000000000000000
SALESREP     46,500.00000000000000000000000000
```

__ 8. Code the SQL statement to list those departments with three or more employees other than the President of the company. Also calculate and display the average salary for the selected work departments. Then list the number of employees in descending order. Your output should be similar to this:

```
WORKDEPT              AVERAGE_SALARY      NUMBER_EMPLOYEES

D11      24,677.777777777777777777777                    9
D21      25,153.333333333333333333333                    6
E11      20,998.000000000000000000000                    5
E21      23,827.500000000000000000000                    4
C01      30,156.666666666666666666666                    3
```

__ 9. Modify the above SQL statement so that the average salary column is displayed with only the first two decimal places. Your output should be similar to this:

```
WORKDEPT  AVERAGE_SALARY  NUMBER_EMPLOYEES

  D11         24,677.77                 9
  D21         25,153.33                 6
  E11         20,998.00                 5
  E21         23,827.50                 4
  C01         30,156.66                 3
```

__ 10. Modify the above SQL statement so that the average salary is rounded. You may have to check the reference material for the function that performs rounding. Your results should be similar to this:

```
WORKDEPT  AVERAGE_SALARY  NUMBER_EMPLOYEES

  D11         24,677.78                 9
  D21         25,153.33                 6
  E11         20,998.00                 5
  E21         23,827.50                 4
  C01         30,156.67                 3
```

__ 11. Modify the SQL statement again so that if the average salary column is less than 25,000.00 the flag “LOW” is displayed. If the average salary is greater than 30,000.00, the flag “HIGH” is displayed. For all other average values, display “MEDIUM”. Your output should be similar to this:

```
WORKDEPT  AVERAGE_SALARY  FLAG    NUMBER_EMPLOYEES

  D11         24,677.78   LOW                    9
  D21         25,153.33   MEDIUM                 6
  E11         20,998.00   LOW                    5
  E21         23,827.50   LOW                    4
  C01         30,156.67   HIGH                   3
```


__ 12. Code the SQL statement to list rows from the EMPLOYEE table, concatenating first and last name; calculate the age (in years) of each employee; show the salary in character form with a currency prefix or suffix for your country. You can be as creative with column headings as you want to risk. There are many ways to solve this problem. Results should look similar to the following (note that the value of the age will vary depending on the date that you are attending this class):

```
String Expression          Numeric Expression  String Expression

Christine Haas                         66      52750.00   USD
Michael Thompson                       51      41250.00   USD
Sally Kwan                             58      38250.00   USD
John Geyer                             74      40175.00   USD
Irving Stern                           54      32250.00   USD
Eva Pulaski                            46      36170.00   USD
Eileen Henderson                       58      29750.00   USD
Theodore Spenser                       43      26150.00   USD
Vicenzo Lucchesi                       70      46500.00   USD
Sean O Connell                         57      29250.00   USD
Delores Quintana                       74      23800.00   USD
Heather Nicholls                       53      28420.00   USD
:                                      :       :
```

End of exercise

Exercise 5. JOIN and UNION

Estimated time

00:30

What this exercise is about

This exercise provides an opportunity to code SELECT statements using JOINs and UNIONs.

What you should be able to do

At the end of the lab, you should be able to:

- Code JOINs
- Code UNIONs

Choose your SQL environment

In this exercise, as well as subsequent exercises, you can use **either** System i Navigator's Run SQL Scripts facility, or the 5250 interactive STRSQL interface, or a combination of both.

If you have not yet used the Navigator interface (Run SQL Scripts), revisit the **Lab Exercise, Using the SELECT Statement instructions for System i Navigator** and perform the steps numbered one through four (1 - 4) in order to establish your Navigator SQL environment. The instructions that follow can be used in either environment.

Exercise instructions for JOIN and UNION

For each query, a problem is stated followed by the desired output. The output format that you see will depend on which tool you use to perform the exercise.

Remember to save your SQL script or your STRSQL session. You can then initiate your sessions and run your SELECT statements with the advantage of duplicating and using prior statements without having to recode your SELECTs from scratch.

The answers to all of the lab questions (that is, how to code an SQL statement to produce the desired outcome) are listed in the Answers section of the appendix.

Exercise instructions for JOIN and UNION

Coding SQL JOINs and UNIONs

__ 1. Initiate your 5250 or Navigator session. Verify that your current library is set to OL37*nn*COL (where *nn* is your team number)

__ 2. Code the SQL statement that lists the last name and job of all employees who work in any departments that contain the characters “Plan” in the first four positions. Your output should be similar to this:

```
LASTNAME        JOB       DEPTNAME

Thompson        MANAGER   Planning
```

__ 3. Code the SQL statement that lists the department number and last name of managers who report to department D01. Your output should be similar to this:

```
DEPTNO  LASTNAME

 D11    Stern
 D21    Pulaski
```

__ 4. Code the SQL statement that lists the names of all employees who work in the same department as the employee with the last name of Adamson. Your output should be similar to this:

```
LASTNAME        FIRSTNME      WORKDEPT

Stern           Irving          D11
Adamson         Bruce           D11
Pianka          Elizabeth       D11
Yoshimura       Masatoshi       D11
Scoutten        Marilyn         D11
Walker          James           D11
Brown           David           D11
Jones           William         D11
Lutz            Jennifer        D11
```

__ 5. Code the SQL statement that lists the average salary earned by men and by women for each department. Organize the output as shown below:

```
DEPTNO DEPTNAME                          AVG ( SALARY )   SEX

A00    SPIFFY Computer Service Div. 52,750.00000000000000000000000   F
A00    SPIFFY Computer Service Div. 37,875.00000000000000000000000   M
B01    Planning                     41,250.00000000000000000000000   M
C01    Information Center           30,156.66666666666666666666666   F
D11    Manufacturing Systems        24,778.33333333333333333333333   M
D11    Manufacturing Systems        24,476.66666666666666666666666   F
D21    Administration Systems       26,933.33333333333333333333333   F
D21    Administration Systems       23,373.33333333333333333333333   M
E01    Support Services             40,175.00000000000000000000000   M
E11    Operations                   23,966.66666666666666666666666   F
E11    Operations                   16,545.00000000000000000000000   M
E21    Software Support             23,827.50000000000000000000000   M
```

__ 6. Using a UNION clause, code the SQL statement to produce a list of all non-managers in work departments B01, C01, and E21. In the same report, follow the above and list all the managers. Use literals to produce the NON-MANAGER and MANAGER labels in the display. Your output should be similar to this:

```
EMPNO   LASTNAME        FIRSTNME      JOB_CLASS

000130  Quintana        Delores       NON-MANAGER
000140  Nicholls        Heather       NON-MANAGER
000320  Mehta           Ramlal        NON-MANAGER
000330  Lee             Wing          NON-MANAGER
000340  Gounot          Jason         NON-MANAGER
000020  Thompson        Michael       MANAGER
000030  Kwan            Sally         MANAGER
000100  Spenser         Theodore      MANAGER
```

__ 7. Next, code the SQL statement to join two tables, EMP_ACT and EMPLOYEE, to determine which employees are assigned to activities in EMP_ACT and have commission earnings >= 2000.00 for the year. Code your SELECT to display the data that follows. Notice the comma (,) between the last name and first name. Order

your output in ascending order by employee number within project number. Your output should be similar to this:

```
PROJNO  EMPNO   String Expression     COMM

AD3100  000010  Haas, Christine       4,220.00
AD3110  000070  Pulaski, Eva          2,893.00
AD3111  000240  Marino, Salvatore     2,301.00
AD3113  000270  Perez, Maria          2,190.00
IF1000  000030  Kwan, Sally           3,060.00
IF1000  000140  Nicholls, Heather     2,274.00
IF2000  000030  Kwan, Sally           3,060.00
IF2000  000140  Nicholls, Heather     2,274.00
MA2100  000010  Haas, Christine       4,220.00
MA2100  000110  Lucchesi, Vicenzo     3,720.00
MA2110  000010  Haas, Christine       4,220.00
MA2111  000200  Brown, David          2,217.00
MA2111  000220  Lutz, Jennifer        2,387.00
MA2112  000150  Adamson, Bruce        2,022.00
OP1000  000050  Geyer, John           3,214.00
OP1010  000090  Henderson, Eileen     2,390.00
OP1010  000280  Schneider, Ethel      2,100.00
OP2000  000050  Geyer, John              3,214.00
OP2010  000100  Spenser, Theodore        2,002.00
OP2012  000330  Lee, Wing               2,030.00
PL2100  000020  Thompson, Michael       3,300.00
```

__ 8. Code the SQL statement to list employees who are not responsible for any projects. Your results should look similar to this:

```
EMPNO   LASTNAME
000110  Lucchesi
000120  O Connell
000130  Quintana
000140  Nicholls
000170  Yoshimura
000180  Scoutten
000190  Walker
000200  Brown
000210  Jones
000240  Marino
000260  Johnson
000280  Schneider
000290  Parker
000300  Smith
000310  Setrignt
```

__ 9. Code the SQL statement to list the headcount and total salary for the departments that begin with A, D, or E. We only want a single row for each department category. Your output should be similar to this:

```
Dept. A Headcount =    COUNT ( * )   Total Salary = SUM ( SALARY )

Dept. A Headcount =              3   Total Salary = 128,500.00
Dept. D Headcount =             15   Total Salary = 373,020.00
Dept. E Headcount =             10   Total Salary = 240,475.00
```

Explain how the headings were created on the display.

_____

_____

_____

_____

End of exercise

Exercise 6. Coding subSELECTs

Estimated time

00:30

What this exercise is about

This exercise provides an opportunity to code SELECT statements.

What you should be able to do

At the end of the lab, you should be able to:

- Code subSELECTs

Choose your SQL environment

In this exercise, as well as subsequent exercises, you can use **either** System i Navigator's Run SQL Scripts facility, or the 5250 interactive STRSQL interface, or a combination of both.

If you have not yet used the Navigator interface (Run SQL scripts), revisit the **Lab Exercise, Using the SELECT Statement instructions for System i Navigator** and perform the steps numbered one through four (1 - 4) in order to establish your Navigator SQL environment. The instructions that follow can be used in either environment.

Exercise instructions for coding subSELECTs

For each query, a problem is stated followed by the desired output. The output format that you see will depend on which tool you use to perform the exercise.

Remember to save your SQL script or your STRSQL session. You can then initiate your sessions and run your SELECT statements with the advantage of duplicating and using prior statements without having to recode your SELECTs from scratch.

The answers to all of the lab questions (that is, how to code an SQL statement to produce the desired outcome) are listed in the answers section of the appendix.

Exercise instructions for coding subSELECTs

Coding SQL subSELECTs and subqueries

__ 1. Code the SQL statement to display a list of all employees who work in the same department as an employee with the last name of “Brown”. Use a subSELECT to solve this problem. Remember that there might be more than one person with the last name of “Brown”.

Your output should look similar to this:

```
WORKDEPT  LASTNAME

D11       Stern
D11       Adamson
D11       Pianka
D11       Yoshimura
D11       Scoutten
D11       Walker
D11       Brown
D11       Jones
D11       Lutz
```

__ 2. Code the SQL statement to determine the first employee hired in our company.

Your output should be similar to this:

```
FIRSTNME       LASTNAME          PHONENO  HIREDATE    JOB

Jason          Gounot             5698    1947-05-05  FILEREP
```

__ 3. Code the SQL statement to list the last name, job, and work department number for each employee who works in any department whose name includes the word “Center”.

Your output should look similar to this:

```
LASTNAME          JOB       WORKDEPT

Kwan              MANAGER     C01
Quintana          ANALYST     C01
Nicholls          ANALYST     C01
```

__ 4. Code the SQL statement to list those departments whose average salary for the employees in that work department is lower than the average for all employees in the company.

Your output should look similar to this:

```
WORKDEPT                        AVG ( SALARY )

  D11          24,677.777777777777777777777
  D21          25,153.333333333333333333333
  E11          20,998.000000000000000000000
  E21          23,827.500000000000000000000
```

__ 5. For this step of the exercise, code the SQL statement that lists only the group of departments that begin with an “E”. Another requirement is to list those female employees whose salaries are greater than the average of the male employees in the same group of departments.

Your output should look similar to this:

```
EMPNO    FIRSTNME       LASTNAME          WORKDEPT        SALARY

000090   Eileen         Henderson           E11         29,750.00
000280   Ethel          Schneider           E11         26,250.00
```

__ 6. Next, code the SQL statement that lists all work departments in the company. List the female employees whose salaries are less than the average of the male employees in the same department.

Your output should look similar to this:

```
EMPNO    FIRSTNME       LASTNAME          WORKDEPT        SALARY

000160   Elizabeth      Pianka              D11         22,250.00
000180   Marilyn        Scoutten            D11         21,340.00
000260   Sybil          Johnson             D21         17,250.00
000310   Maude          Setrignt            E11         15,900.00
```

More subSELECTs (optional)

__ 7. Code the SQL statement that lists those employees whose salary in work department E11 is lower than the average department salary for all departments in the company.

Your output should look similar to this:

```
FIRSTNME       LASTNAME                SALARY

John           Parker               15,340.00
Philip         Smith                17,750.00
Maude          Setrignt             15,900.00
```


__ 8. Code the SQL statement that lists those employees (other than managers or the president of the company) whose commission is the highest commission earned in each department. Your output should look similar to this:

WORKDEPT	LASTNAME	FIRSTNME	JOB	COMM	BONUS
A00	Lucchesi	Vicenzo	SALESREP	3,720.00	900.00
C01	Nicholls	Heather	ANALYST	2,274.00	600.00
D11	Lutz	Jennifer	DESIGNER	2,387.00	600.00
D21	Marino	Salvatore	CLERK	2,301.00	600.00
E11	Schneider	Ethel	OPERATOR	2,100.00	500.00
E21	Lee	Wing	FILEREP	2,030.00	500.00

End of exercise

Exercise 7. Using SQL Query Manager

Estimated time

00:45

What this exercise is about

This exercise provides an opportunity to use SQL Query Manager.

What you should be able to do

At the end of the lab, you should be able to:

- Create and modify Query Manager procedures
- Create and modify Query Manager report forms

Exercise instructions for 5250 emulator session

__ 1. Start a 5250 emulation session and sign on with your OL37nn user ID and password.

__ 2. If it is not already set, change your current library to OL37*nn*COL (where *nn* is your team number).

__ 3. Enter the command **STRQM** to start Query Manager.

__ 4. Choose option **10** = Work with Query Manager Profiles to modify your profile, OL37nn. Note that if you are the administrator of the system, you can modify any of the profiles for which you are authorized.

__ 5. Make the following changes (you might have to page down):

- Default library for QM objects: OL37*nn*COL
- Default library for QM tables: OL37*nn*COL
- Default query creation mode: SQL

  (The default is Prompted, but we will not be using this method.)

Then press **Enter**.

__ 6. From the Query Manager menu, select option **1** = Work with Query Manager Queries. On the next display, select option **1** = Create to create a new query named QMLAB.

__ 7. In the edit query display, enter the following SQL SELECT statement:

```
SELECT DISTINCT PROJNO, EMP_ACT.EMPNO,
    LASTNAME||', '||FIRSTNME, SALARY
 FROM EMP_ACT JOIN EMPLOYEE
  ON EMP_ACT.EMPNO = EMPLOYEE.EMPNO
   WHERE COMM >= 2000.00
 ORDER BY PROJNO, EMPNO
```

__ 8. When you are done, press **F15** = Check syntax. If any errors are detected, then check your typing and make sure the statement you entered matches the instructions in the previous step.

__ 9. Once you receive the message that the SQL statement is valid, then press the **F3** = Exit to exit and save your QMLAB query.

Run the QM query

__ 10. There are several ways to run a query. You can also do it by executing the QM command **RUN QUERY** from the command line and then pressing **F4**. Fill in the name of the query you want to run. Or, from this screen you can choose option **9** = Run to execute or run your query. Run your query using either of these options.

A copy of the results obtained using the system default form follows:

```
PROJNO  EMPNO   CONCAT                              SALARY
------ ------ ---------------------------- -------------
AD3100 000010 Haas, Christine 52,750.00
AD3110 000070 Pulaski, Eva 36,170.00
AD3111 000240 Marino, Salvatore 28,760.00
AD3113 000270 Perez, Maria 27,380.00
IF1000 000030 Kwan, Sally 38,250.00
IF1000 000140 Nicholls, Heather 28,420.00
IF2000 000030 Kwan, Sally 38,250.00
IF2000 000140 Nicholls, Heather 28,420.00
MA2100 000010 Haas, Christine 52,750.00
MA2100 000110 Lucchesi, Vicenzo 46,500.00
MA2110 000010 Haas, Christine 52,750.00
MA2111 000200 Brown, David 27,740.00
MA2111 000220 Lutz, Jexxifer 29,840.00
MA2112 000150 Adamson, Bruce 25,280.00
OP1000 000050 Geyer, John 40,175.00
OP1010 000090 Henderson, Eileen 29,750.00
OP1010 000280 Schneider, Ethel 26,250.00
OP2000 000050 Geyer, John 40,175.00
OP2010 000100 Spenser, Theodore 26,150.00
OP2012 000330 Lee, Wing 25,370.00
PL2100 000020 Thompson, Michael 41,250.00
```

__ 11. Exit and save the active data.

__ 12. From the Work with Query Manager Queries screen, enter option **9** next to your QMLAB query and press **Enter**. Explore the options that are available to you when you run your query. You can display your results or you can choose to write them to a spool file.

__ 13. Choose the option to send your output to your printer then press **Enter** to run your query.

__ 14. Look at your spool file to review your results.

Hint

You do not have to exit QM to view the spool file member. Note that the **F24** = More keys function is available, press it now. Next press **F22** = QM Statement which will open a window where you can enter SQL statements. Notice that F21= System command is one of the allowed function keys in this window. Press **F21** and then enter the WRKSPLF system command to view your spool file entries.

Modify the output

__ 15. Return to the main **QM** menu.

__ 16. Select the option to work with QM report forms.

__ 17. Choose option **1** = Create create a new form that will be named QMLABFORM.

__ 18. Your assignment in this lab step is to produce the report that looks similar to the one below. No detailed instructions will be provided. If you need help, review the lab answers at the end of this lab. Your report can look similar to the one below or even better. Try different options.

Note: QM assumes that your form is tied to the query QMLAB that you just created. You can check this while you are editing the report form by pressing **F13** to display the active query (Edit Query):

```
02/17/07 12:46:54      Employees Due Raises          Page 1

Project  Employee   Employee
Number    Number      Name                            Salary
-------- ---------- ------------------------ ------------
AD3100   000010     Haas, Christine               52,750.00

AD3110   000070     Pulaski, Eva                  36,170.00
AD3111   000240     Marino, Salvatore             28,760.00
AD3113   000270     Perez, Maria                  27,380.00
IF1000   000030     Kwan, Sally                   38,250.00
IF1000   000140     Nicholls, Heather             28,420.00
IF2000   000030     Kwan, Sally                   38,250.00
IF2000   000140     Nicholls, Heather             28,420.00
MA2100   000010     Haas, Christine               52,750.00
MA2100   000110     Lucchesi, Vicenzo             46,500.00
MA2110   000010     Haas, Christine               52,750.00
MA2111   000200     Brown, David                  27,740.00
MA2111   000220     Lutz, Jexxifer                29,840.00
MA2112   000150     Adamson, Bruce                25,280.00
OP1000   000050     Geyer, John                   40,175.00
OP1010   000090     Henderson, Eileen             29,750.00
OP1010   000280     Schneider, Ethel              26,250.00
OP2000   000050     Geyer, John                   40,175.00
OP2010   000100     Spenser, Theodore             26,150.00
OP2012   000330     Lee, Wing                     25,370.00
PL2100   000020     Thompson, Michael             41,250.00
         ==========                            ============
                 21                            742,380.00
```


Suggestions:

1. Make sure that your output fits within 80 print positions. If your page width is too wide, your output will be split into two spool file members.
2. You can modify the column sizes allocated on the report.
3. Add your country's currency symbol to the salary column.
4. Experiment with subtotals by project.

End of exercise

Exercise 8. Enhancing an SQL table

Estimated time

00:30

What this exercise is about

In this exercise, you continue to work with the STULST table that you created in the previous exercise.

What you should be able to do

At the end of the lab, you should be able to:

- Use SQL DDL statements to:
  - Add LABELS to columns in an existing table
  - INSERT rows of data to a table
  - Use ALTER to add a column to existing table

Introduction

In this exercise, you will enhance the STULST table again using familiar IBM i features.

Exercise instructions for 5250 emulator session

__ 1. Start a 5250 emulation session and sign on with your OL37nn user ID and password.

__ 2. Manage the STULST SQL table.

__ a. Using a procedure similar to the one you used in the first exercise, code a separate source member in your `QSQLSRC` file of your OL37*nn*LIB (where *nn* is your team number) for each of the following SQL scripts:

```
LABELS          Define a label for each of the columns in
                STULST
                 Hint - Use the LABEL ON COLUMN ... SQL
                statement.

INPSTUDTA       Add three rows to STULST
                 Hint - Use the INSERT INTO ... VALUES ... SQL
                statement.

CHGSTULST       Add a new column CLASS_CODE to STULST, with the
                following attributes:

                 Type   = CHAR
                 Length = 4
                 not null with default
                 Hint - Use the ALTER TABLE ... ADD ... SQL
                statement.
```

__ b. Execute each of these scripts using RUNSQLSTM.

__ c. Review the contents of your table using the **`RUNQRY`** command:

```
RUNQRY *N STULST
```

End of exercise

Exercise 9. Maintaining database objects

Estimated time

01:30

What this exercise is about

This exercise provides an opportunity to code SQL statements to manipulate data.

What you should be able to do

At the end of the lab, you should be able to:

- Code UPDATE and DELETE statements
- Code INSERT statements using a VALUE or a SELECT clause

Introduction

Reference your lecture notebook as necessary.

Exercise instructions for 5250 emulator session

Create TESTEMP table

__ 1. Start a 5250 emulation session and sign on with your OL37nn user ID and password.

__ 2. Create table `TESTEMP` in your schema, OL37*nn*COL**,** with the following columns:

EMPNO	Char(6)	Not null
LASTNAME	Varchar(15)	Not null
WORKDEPT	Char(3)	
HIREDATE	Date	
SALARY	Decimal(9,2)	
BONUS	Decimal(9,2)	

Add new employees to the table

__ 3. Three new employees have joined the company. Their data is:

a. Data for the first employee:

- Employee name Mr. Smith
- Employee numbers 000111
- Last name SMITH
- Department number C01
- Date hired June 6, 2000
- Salary $25,000
- Bonus: $0

b. Data for the second employee:

- Employee name Ms. Baker
- Employee numbers 000222
- Last name: BAKER
- Department number A00
- Date hired June 6, 2000
- Salary $28,000
- Bonus NULL

c. Data for the last employee:

- Employee name Ms. Thomas

- Employee numbers 000333
- Last name THOMAS
- Department number D11
- Date hired June 6, 2000
- Salary $33,000
- Bonus: $0

Code and execute the SQL statement to add these new employees to the TESTEMP table.

__ 4. Code and execute the SQL statement to insert data into the TESTEMP table by copying the appropriate columns in the EMPLOYEE table for those employees that have an employee number less than or equal to 50.

__ 5. Run the `SELECT * FROM TESTEMP` statement to verify that you have successfully performed the statements in the previous steps. If there are any errors, return to the step in question and debug the SQL statement.

Maintain data

__ 6. Mr. Smith received a bonus of $500. Code and execute the SQL statement to make this change in the employee data record.

__ 7. All of the employees in department C01 have done a good job this year. Therefore, they will receive a salary increase of $1,000. Code and execute the SQL statement to make this change to the appropriate employee records.

__ 8. After a short time, Mr. Smith decides to leave the company. Code and execute the SQL statement to make this change to the appropriate employee records.

__ 9. Code and execute the SQL statement to insert data into the TESTEMP table by copying the columns in the EMPLOYEE table where the employees have an employee number greater than 50.

__ 10. Code and execute the SQL statement to make the appropriate changes for the following situation:

```
Theodore Spenser is assigned to department E01.
Mrs. Brown joined the company. For Mrs. Brown enter the following
data:
```

- Employee number: 360
- Last name: BROWN
- Department number: D01
- Date hired: Date when data is entered
- Salary: $45,000

- Bonus: unknown

__ 11. Run `SELECT * FROM TESTEMP` to verify that you have successfully performed the statements in the previous steps. If there are any errors, return to the step in question and debug the SQL statement.

__ 12. Delete all rows from the TESTEMP table and examine the contents.

__ 13. Run `SELECT * FROM TESTEMP` to verify that you were successful.

__ 14. Code and execute the SQL statement to delete the TESTEMP table from your OL37xxCOL collection. Check the contents of your collection to verify that the table is no longer on the system.

End of exercise

Exercise 10. Change CHAIN to SELECT

Estimated time

00:45

What this exercise is about

This exercise provides an opportunity to modify an existing RPG IV program. You will change the native CHAIN opcode to a SELECT statement.

What you should be able to do

At the end of the lab, you should be able to:

- Create the SQL objects for your lab environment
- Modify a simple RPG IV program to use SQL rather than native I/O.

Introduction

In this exercise, you create all the SQL objects that you will use in the subsequent exercises. You create your schema, **OL38*nn*COL**, where *nn* is your assigned team number for this class. Once you have created your schema, you create the tables and populate them with data from the course master library. Then you are given an RPG IV program that performs an inquiry into an item master file. You will modify the program by changing an RPG CHAIN specification to an SQL SELECT statement.

We will use RPG IV fixed format code in this exercise. If you are unfamiliar with RPG IV, you should be able to perform most of this exercise without assistance. Please ask your instructor for any assistance that you require.

Exercise instructions

Create your schema

In this exercise, you may use the interactive SQL interface (STRSQL) or System i Navigator, as you desire. You may find you favor one tool over the other or you may choose to experiment with both tools in order to compare them. No detailed instructions are provided for the use of either tool.

Whenever you are prompted for your user ID (**OL38***nn*) and password (**OL38**), enter them. The **OL38** password is set to expire at initial signon; please change it to something meaningful, to you.Then:

__ 1. Create your student SQL schema, **OL38*nn*COL**.

__ 2. Add a description for the library, schema **for Team nn**.

Create tables and insert values

In this step, you will create four tables in your schema, **OL38*nn*COL**.

__ 3. Change your current library to **OL38*nn*COL**. using both the `CHGCURLIB` and the `CHGPRF` commands.

__ 4. Create (or use CPYF to copy the QSQLSRC file from OL38XXCOL) a source file named QSQLSRC in your student schema OL38*nn*COL.

__ 5. From the student master schema (OL38XXCOL), copy all source members in the source file QSQLSRC to your source file QSQLSRC in your student schema, **OL38*nn*COL**.

__ 6. Edit these members in your QSQLSRC file:

```
Member    Type   Text

CRTTABLE  SQL     SQL Statements to CREATE student tables for labs
REFTABLE  SQL    SQL Statements to RECREATE student tables for labs
TABLEX    SQL     SQL Statements to insert student tables with data
```

__ 7. For each member, use your editor of choice to find and change all occurrences of **OL38XXCOL** to **OL38*nn*COL**, your student schema For example, SEU allows you to use the SEU command:

```
C OL38XXCOL OL38nnCOL ALL
```

Note

Note: Remember that you do not compile SQL source statements. They are prepared one by one prior to execution and then executed.

__ 8. Once you have completed the find/change, execute the following commands as shown below:

__ a. `RUNSQLSTM SRCFILE(OL38nnCOL/QSQLSRC) SRCMBR(CRTTABLE) COMMIT(*NONE) NAMING(*SQL)`

This will create some of the exercise tables in your schema. Do not forget to change the *nn* above to your team number.

__ b. `RUNSQLSTM SRCFILE(OL38nnCOL/QSQLSRC) SRCMBR(TABLEX) COMMIT(*NONE) NAMING(*SQL)`

This will insert values in the columns of each table in your student schema.

__ c. You have now created the following tables and have populated them with values:

- EMPLOYEE
- DEPARTMENT
- PROJECT
- EMP_ACT

Reminder

REFRESH

Should you need to refresh the data in your tables, it will be necessary to execute the TABLEX SQL statements again:

```
RUNSQLSTM SRCFILE(OL38nnCOL/QSQLSRC) SRCMBR(TABLEX)
COMMIT(*NONE) NAMING(*SQL)
```

This will refresh the values in the columns of each table in your student schema. Do not forget to change the *nn* above to your team number.

__ d. Use CPYF to copy the following files from the student master schema, **OL38XXCOL**, to your schema, **OL38*nn*COL**:

```
APFLDREF *FILE PF-DTA Accounts Payable Field Reference File
APINV_PF *FILE PF-DTA AP Invoice Physical File
```

Copy the display file source

__ 9. Verify/change your current library to OL38*nn*COL.

__ 10. Copy the source file, QDDSSRC in OL38XXCOL and all its members to your schema, **OL38*nn*COL**.

Review the display file source

__ 11. Review the source of display file ITEMINQDSP in your schema below.

```
A                                      REF(*LIBL/ITEM_PF)
A                                      CA03(03 'Exit')
 **
A          R PROMPT
A            PGMNAM        10A  O  3  7
A                                   3 35'Item Inquiry'
A                                      COLOR(WHT)
A                                   3 64DATE
A                                      EDTCDE(Y)
A                                   8 20'Item Number . . . . . .:'
A            ITMNBR    R     D  I  8 45
A  40                                  ERRMSG('Item not found on file - pl -
A                                      ease correct' 40)
A                                  20  7'Press Enter to continue'
A                                  21  7'F3=Exit'
A                                      COLOR(BLU)
 **
A          R DETAIL
A                                      CA12(12 'Cancel')
A            PGMNAM        10A  O  3  7
A                                   3 35'Item Inquiry'
A                                      COLOR(WHT)
A                                   3 64DATE
A                                      EDTCDE(Y)
A                                   8 20'Item Number . . . . . .:'
A            ITMNBR    R        O  8 45
A                                   9 20'Description . . . . . .:'
A            ITMDESCR  R        O  9 45
A                                  11 20'Quantity on hand  . . .:'
A            ITMQTYOH  R        O 11 45
A  30                                  DSPATR(HI)
A                                  12 20'Quantity on order . . .:'
A            ITMQTYOO  R        O 12 45
A  30                                  DSPATR(HI)
A                                  14 20'Supplier Number . . . .:'
A            VNDNBR    R        O 14 45
A                                  15 20'Supplier Catalogue No .:'
A            ITMVNDCAT#R        O 15 45
A  30                              17  7'** NOTE ** The total quantity avai-
A                                      lable is low (<20) - reorder'
A                                      DSPATR(HI)
A                                  21  7'F3=Exit   F12=Cancel'
A                                      COLOR(BLU)
```


__ 12. Notice the following about the display file:

- Record format names are PROMPT and DETAIL.
- Input variable is ITMNBR.
- Function keys 3 and 12 are used.

Create display file ITEMINQDSP

__ 13. Create display file `ITEMINQDSP`, in your library OL38*nn*COL. Verify that it is created successfully.

Copy the RPGLE program source

__ 14. Copy the source file, `QRPGLESRC` in OL38XXCOL and all its members to your schema, **OL38**nn**COL**.

__ 15. Confirm that you have a source member, ITEMINQR, in your `QRPGLESRC` file.

Copy the ITEM_PF physical file

__ 16. Copy this file from OL38XXCOL to your schema, OL38*nn*COL.

Review and test the program ITEMINQR

__ 17. Study the existing source, shown below, in source member ITEMINQR, source file QRPGLESRC in your library OL38*nn*COL.

```
 ** Item Master File
FItem_PF   IF   E           K Disk
 ** Display File
FItemInqDspCF   E             Workstn

C                   Dow       NOT *In03
C     ItmNbr        Chain     Item_PF                          40----

C                   If        NOT *In40
 ** Display details
C                   Dow       NOT *In12
C                   Eval      *In30 = (ItmQtyOH + ItmQtyOO) < 20
C                   Exfmt     Detail

C                   IF        *In03
C                   Eval      *InLR = *ON
C                   Return
C                   Endif
C                   Enddo
C                   Endif

 ** No Item record found or F12 pressed - display prompt
C                   Eval      *In12 = *OFF
C                   Exfmt     Prompt
C                   Enddo
C                   Eval      *InLR = *ON
C                   Return

 ****************************************************************
C     *Inzsr        Begsr
C                   Eval      PgmNam = 'ITEMINQ'
C                   Exfmt     Prompt
C                   Endsr
```

__ 18. Your job will be to make all the necessary changes to this program so that the CHAIN to the Item_PF will be performed in a SELECT rather than a CHAIN.

Compile the program as-is and test it. When you run the program, use any item number in the range 20001 through 20050.

Note

You might want to use System i Navigator to view the table in order to understand the data. You can view the table, then run the program while you leave the view active.

Change from CHAIN to SELECT

__ 19. Now you are ready to modify the program. Make a copy of the program, ITEMINQR, and name it ITEMINQS in your source file.

__ 20. Change the member type from RPGLE to SQLRPGLE.

__ 21. Modify ITEMINQS to perform the I/O using SQL rather than RPG. You can leave the file definition as it is for this exercise.

```
C/Exec SQL
C+ select itmnbr, itmdescr, itmqtyoh, itmqtyoo, vndnbr,
itmvndcat#
C+        into :itmnbr, :itmdescr, :itmqtyoh,
C+             :itmqtyoo, :vndnbr, :itmvndcat#
C+        from item_pf
C+        where itmnbr = :itmnbr
C/End-exec
```

Can you explain how the fields that are prefixed by the colon (:) are used?

_____

_____

_____

__ 22. You may also ignore error handling based on indicator 40. Simply code an EVAL to set the value of *in40 to *off for now.

__ 23. Prepare and create the program. You may use the following command, or PDM option **14**, followed by **F4**=Prompt and entering the parameters appropriately:

```
CRTSQLRPGI OBJ(OL38nnCOL/ITEMINQS)
           SRCFILE(OL38nnCOL/QRPGLESRC)
           COMMIT(*NONE)
           OUTPUT(*PRINT)
           DBGVIEW(*SOURCE)
```

You could also try embedding the SQL statement SET OPTION in your program to handle the above SQL parameters of the compile. Here is an example:

```
SET OPTION COMMIT= *NONE, OUTPUT = *PRINT, DBGVIEW = *SOURCE
```

__ 24. Verify that your program compiled successfully and correct any errors. Review the spool file output from the SQL preprocessor. Notice that all the SQL is now commented out and replaced with calls to QSQROUTE plus variable assignments.

Test your program ITEMINQS

__ 25. Run your program ITEMINQS.

__ 26. Enter any valid item number (20001 through 20050).

__ 27. Enter an invalid item number (any number other than those above). What happens and why?

_____

_____

_____

_____

__ 28. What other code do you think needs to be included before we put this program in production?

_____

_____

_____

_____

__ 29. You probably included the file definition for ITEM_PF. Does SQL need this file definition? Why or why not?

_____

_____

_____

_____

End of exercise

Exercise Review/Wrapup

There are a number of points that you should review with the students:

1. The students should notice that using the SELECT was fairly easy but involved more coding than the CHAIN.
2. There is no error handling, so when an invalid item number is entered, either blank or zero fields will be displayed or you will see the values from the previous valid number displayed.
3. Proper error handling would involve using SQLCOD and the SQLCA data structure. We will cover error handling in a later topic of this unit. The students will implement error handling in the next exercise.
4. The file description spec is not required since SQL does not use it. Thus, our coding would have been simplified.
5. Remind them that the fields prefixed by colon (:) are called **Host Variables**. We will discuss them further in the next topic.

In closing, tell the class that using SQL to perform I/O is simple to code once we know the rules. There are some things that are done differently, but the same things that you must do in traditional programs are done in programs that use embedded SQL.

Exercise 11. Changing native to SQL I/O

Estimated time

00:45

What this exercise is about

This exercise provides an opportunity to modify an existing RPG program. You will change the native I/O to SQL statements.

What you should be able to do

At the end of the lab, you should be able to:

- Modify an existing RPG IV program to use SQL rather than native I/O
- Explain the areas of an RPG program that will need to be modified to support SQL I/O

Introduction

In this exercise, you are given an RPG IV program that gives an additional bonus to our best performing employees. You will modify the program by changing RPG specifications where required to support SQL I/O.

We will use RPG IV code in this exercise. If you are unfamiliar with RPG IV, you should be able to perform most of this exercise without assistance. Please ask your instructor for any assistance that you require.

Exercise instructions

Review the display file source

__ 1. Change your current library to OL38*nn*COL.

__ 2. Verify that the source member, BONUSDSPF, is in your schema, **OL38*nn*COL**.

__ 3. Review the source of display file BONUSDSPF below.

```
A                                      INDARA
A                                      CA03(03)
A          R BONUSFMT
A                                    2 22'Employee Bonus'
A                                    6  8'Enter employee number....:'
A                                    7  8'Enter amount of bonus.....-
A                                       ..:'
A            EMPNO     R        I  6 49REFFLD(EMPLOYEE/EMPNO EMPLOYEE)
A            ADDLBONUS      9Y 2I  7 49
A
A  90                                  ERRMSG('Invalid employee - reenter')
A                                   21  8'Press Enter to Continue'
A                                   22  8'F3=Exit'
A                                       COLOR(BLU)

A          R BONUSDET
A                                       CA12(12)
A                                    3 30'Addional Bonus Applied'
A                                       DSPATR(HI)
A                                    3 70DATE
A                                       EDTCDE(Y)
A                                    8 20'Employee Number:'
A            EMPNO     R        O  8 47
A                                   10 20'Additional Bonus:'
A            ADDLBONUS R        O 10 47EDTCDE(K)
A                                   11 20'Current Total Bonus:'
A            BONUS     R        O 11 47REFFLD(EMPLOYEE/BONUS EMPLOYEE)
A                                       EDTCDE(K)
A                                   22  8'F3=Exit  F12=Cancel'
A                                       COLOR(BLU)
```

__ 4. Notice the following about the display file:

- Record format names are BONUSFMT and BONUSDET.
- Input variables are EMPNO and ADDLBONUS.
- Function keys 3 and 12 are used.

Create display file BONUSDSPF

__ 5. Create display file BONUSDSPF, in your library OL38*nn*COL. Verify that it is created successfully.

Create an Index over the EMPLOYEE Table

__ 6. You recall that Tables created by SQL are Physical Files without a key. Verify this by running the DSPFD command against EMPLOYEE. Send the output to your OUTQ as you may want to reference it later.

__ 7. Create an index, EMPLOYEEI. The index should be based on the employee number column, EMPNO, and should be unique and in ascending order.

Review and Test the Program BONUSRPGR

__ 8. Find a member, BONUSRPGR, in your source file.

__ 9. Study the existing source, shown below, in source member BONUSRPGR, source file QRPGLESRC in your library OL38*nn*COL.

```
F*  Display file BONUSDSPF is declared.
FBonusDSPF CF   E             WorkStn InDDS(Indicators)
FEmployeeI UF   E           k Disk

D Indicators      DS
D   Exit                        1N   Overlay(Indicators:03)
D   Cancel                      1N   Overlay(Indicators:12)
D   Error                       1N   Overlay(Indicators:90)

C*  Format BONUSFMT from display file BONUSDSPF is written and read.
C*  The user will key values into ADDLBONUS and EMPNO.

C                   Exfmt     BonusFmt
C                   Dow       Not Exit

C     EmpNo         Chain     EmployeeI
C                   Eval      Error = Not %Found(EmployeeI)

C                   Dow       %Found(EmployeeI)
C* For valid employee, calculate new bonus, update record
C* and display result on workstation
C                   Eval      Bonus = Bonus + AddlBonus
C                   Update    Employee
C                   Exfmt     BonusDet

C                   Select
C                   When      Exit
C                   Eval      *InLR = *ON
C                   Return

C                   When      Cancel
C                   Leave
C                   EndSl

C                   EndDo
C                   Exfmt     BonusFmt
C                   EndDo

C                   Eval      *InLR = *ON
C                   Return
```

__ 10. Your job will be to make all the necessary changes to this program so that all existing I/O and file definitions will be handled using SQL. To assist you, do the following:

__ a. Compile the program as-is and test it. When you run the program, use any employee number in the range 000010 through 000400 (employee numbers are incremented by 10) and any bonus amount you like. You should check the value of the bonus of the impacted employee before and after the additional bonus is awarded.

Note

You might want to use System i Navigator to view the table before and after you award the bonus. You can view the table, then run the program while you leave the view active. To refresh your view, click **View** and select **Refresh**.

Change from native I/O to SQL I/O

__ 11. Now you are ready to modify the program. Make a copy of the program, BONUSRPGR, and name it BONUSRPGS in your source file.

__ 12. Modify BONUSRPGS to perform the I/O using SQL rather than RPG. There are several areas to consider: member type, file definition, fields that are defined by externally described files, SQL error handling versus native RPG error handling, and the SQL statements themselves.

__ 13. Prepare and create the program. You may use the following command, or PDM option 14, followed by F4=Prompt and completing the parameters appropriately:

```
CRTSQLRPGI OBJ(OL38nnCOL/BONUSRPGS)
           SRCFILE(OL38nnCOL/QRPGLESRC)
           COMMIT(*NONE)
           OUTPUT(*PRINT)
           DBGVIEW(*SOURCE)
```

__ 14. Verify that your program compiled successfully and correct any errors. Review the spool file output from the SQL preprocessor. Notice that all the SQL is now commented out and replaced with calls to QSQROUTE plus variable assignments.

Test your program BONUSRPGS

__ 15. Display the EMPLOYEE table. Decide which employee should get the additional bonus and how much it should be.

__ 16. Run your program BONUSRPGS.

__ 17. Enter the employee number and additional bonus. The additional bonus field is a numeric only, type **Y** field. You can only key the numbers 0-9, plus (+), minus (-), decimal (.), comma (,) and space into the field.

__ 18. Display the EMPLOYEE table as you did earlier. Verify that it has been updated correctly. If you would like to refresh your files, follow the procedure described earlier in the lab exercise guide. You can also modify the values of columns using iSeries Navigator.

If your values are not updated as expected, check that you have specified COMMIT(*NONE) in the CRTSQLRPGI command above.

Better Solution

Modify your program to include the SQL statements below:

```
/EXEC SQL
/+ SET OPTION COMMIT=*NONE
/End-Exec
```

Adding this code eliminates the need to change the **CRTSQLRPGI** command parameters. Note this is not promptable in SEU!

__ 19. If you have any problems, use the ILE debugger to debug your program. It might be interesting to debug your program even if it works! When you are in the debugger, notice what you can and cannot debug. Is it possible to put a breakpoint on an SQL statement? How do you get the value of a host variable?

End of exercise

Exercise review/wrapup

There are a number of points that you should review with the students:

1. Our exercise deals with a single row at a time. A program that could produce a multi row result set must be coded differently. As a matter of interest, if a SELECT INTO were to derive a multi row result set, the value of SQLCOD would be -811.
2. Make sure that any file-dependent error handling and error BIFs are replaced with the equivalent SQL error handling.
3. Because SQL can include expressions in a statement, SQL is far more powerful than RPG when performing I/O. You may be able to eliminate RPG code that can be performed in SQL such as we can see in the UPDATE statement.
4. Note the use of host variables in the program.
5. Point out that this program uses Static SQL.
6. When you use SQL I/O to replace a file that is externally described, it will be necessary to define any fields that are referenced in the file in the D-specs.
7. Finally, confess to the students that there is a bug in the application. The native RPG IV program gets an error message on the RPG update opcode if you press **ENTER** in the detail format.

   What is important is that the error is deliberate. Note that an SQL update does not cause the same problem. An SQL update statement does not require a READ/CHAIN to position the file cursor. While the coding may not be error-proof, it clearly demonstrates this difference.

Exercise 12. Using a cursor

Estimated time

01:00

What this exercise is about

In this exercise, you will modify a working program that applies a bonus to all employees in a department as specified by the user.

This program is similar to the bonus program you used previously. Each employees record is updated. As well, all impacted employees are displayed to the user in a subfile.

What you should be able to do

At the end of the lab, you should be able to:

- Modify an existing RPG IV program to use an SQL cursor
- Implement basic error handling and use SQLCOD
- Explain the areas of an RPG program that will need to be modified to support an SQL cursor

Introduction

In this exercise, you are given an RPG IV program that gives an additional bonus to our best performing department. You will modify the program by changing RPG specifications where required to support SQL I/O, an SQL cursor and any desired error handling.

Exercise instructions

Review the display file source

__ 1. Review the source of display file BONUSDSPFD, that should be in you schema:

```
A                                      REF(*LIBL/EMPLOYEE)
A                                      CA03(03)


 ** Function Keys
A          R FNCKEY
A                                      OVERLAY
A                                   22  5'F3=Exit'
A                                       COLOR(BLU)


 ** Message for empty subfile
A          R MSG
A                                      OVERLAY
A                                   12 32'No Records Found'
A                                       DSPATR(HI)

 ** Prompt for Department Record Format
A          R BONUSFMT
A                                     2 22'Department Bonus'
A                                     6  8'Enter department number... :'
A            WORKDEPT  R        I  6 49
A                                     7  8'Enter amount of bonus..... .......-
A                                        ..:'
A            ADDLBONUS      9Y 2I  7 49
A
A  90                                  ERRMSG('Invalid department   reent-
A                                      er' 90)
A                                   21  8'Press Enter to Continue'
A                                   22  8'F3=Exit'
A                                       COLOR(BLU)

 ** Subfile Record format
A          R DATA                      SFL
A            WORKDEPT  R        O  9  5
A            EMPNO     R        O  9 20
A            ADDLBONUS R        O  9 30REFFLD(ADDLBONUS *SRC)
A                                      EDTCDE(K)
A            BONUS     R        O  9 50REFFLD(EMPLOYEE/BONUS EMPLOYEE)
A                                      EDTCDE(K)


 ** Subfile Control format
A          R CONTROL                   SFLCTL(DATA)
A                                      SFLSIZ(0011)
A                                      SFLPAG(0010)
A  75                                  SFLCLR
A  85                                  SFLDSPCTL
A  95                                  SFLDSP
```



```
A  40                                  SFLEND(*MORE)
A                                      OVERLAY
A                                3  5DATE
A                                      EDTCDE(Y)
A                                3 35'Bonus List'
A                                      DSPATR(HI)
A                                3 66USER
A                                4 66SYSNAME
A                                6  5'Work Dept.'
A                                6 20'Employee'
A                                6 30'Additional Bonus'
A                                6 50'Bonus'
```

__ 2. Notice the following about the display file:

- Record format names are BONUSFMT, FNCKEY, MSG, DATA and CONTROL.
- Input variables are WORKDEPT and ADDLBONUS.
- Function key 3 is used.

Create display file BONUSDSPFD

__ 3. Create display file BONUSDSPFD, in your library OL38*nn*COL. Verify that it is created successfully.

Create an index over the EMPLOYEE table

__ 4. Create an index, EMPLOYEED, over the EMPLOYEE table The index should be based on the work department column, WORKDEPT. It is not unique but should be in ascending order.

Review and test the program BONUSDPTR

__ 5. Study the existing source, shown below, in source member BONUSDPTR, source file QRPGLESRC in your library OL38*nn*COL.

```
 ** Display file BONUSDSPFD is declared.
FBonusDSPFDCF   E             WorkStn Sfile(Data:Rrn)
F
FEmployeeD UF   E           k Disk

D Rrn             S              4  0 Inz(1)

 ** Format BONUSFMT from display file BONUSDSPFD is written and read.
 ** The user will key values into ADDLBONUS and WORKDEPT

C                   Exfmt     BonusFmt
C                   Dow       Not *in03

C     WorkDept      Chain     EmployeeD
C                   Eval      *in90 = Not %Found(EmployeeD)

 ** For valid Workdept, fill subfile.
C                   Dow       %Found(EmployeeD)

 ** For each employee, calculate new bonus, update record
C                   Eval      Bonus = Bonus + AddlBonus
C                   Update    Employee
C                   Write     Data

C     WorkDept      ReadE     EmployeeD
C                   Eval      Rrn = Rrn + 1
C                   Eval      *in40 = %Eof(EmployeeD)
C                   If        %Eof(EmployeeD)
C                   Leave
C                   Else
C                   EndIf
C                   Enddo

 ** Display subfile if file not empty
C                   Write     FncKey
C                   Select

C                   When      Rrn > 1
C                   Eval      *In85 = *ON
C                   Eval      *In95 = *ON

C                   Dow       NOT *In03
C                   Exfmt     Control
C                   Enddo

C                   When      Rrn <= 1
C                   Eval      *In85 = *ON
C                   Eval      *In95 = *OFF
C                   Write     Msg
C                   Exfmt     Control

C                   Endsl
```



```
C                   If        *in03
C                   Eval      *InLR = *ON
C                   Return

C                   EndIf

C                   EndDo

C                   Eval      *InLR = *ON
C                   Return
```

__ 6. Your job will be to make all the necessary changes to this program so that all existing I/O and file definitions will be handled using SQL. To assist you, do the following:

__ a. Compile the program as-is and test it. When you run the program, use any work department (such as A00, B01, C01) to test the program and to see how it works. Use any bonus amount you like. You should check the value of the bonus of the impacted employee before and after the additional bonus is awarded.

Note: You might want to use iSeries Navigator to view the Employee table before and after you award the bonus. You can view the table, then run the program while you leave the view active. To refresh your view, click *View* and select *Refresh*.

__ b. As necessary refresh the values of the EMPLOYEE table using RUNSQLSTM and the member to refresh values, TABLEX.

Change from native I/O to SQL I/O

__ 7. Now you are ready to modify the program. Make a copy of the program, BONUSDPTR, and name it BONUSDPTS in your source file.

__ 8. Modify BONUSDPTS to perform the I/O using SQL rather than RPG. There are several areas to consider in addition to those areas you discovered in the previous exercise:

- Is the file definition of EmployeeD still necessary?
- You will need to define a cursor and manage its use in your program to process the result set.
- An SQL update statement can process more than one row concurrently. Decide whether you want to use this facility or replace the existing RPG UPDATE opcode and update a single row.
- Use SQLCOD to test for success, record not found, end of file.
- Add error handling that is as comprehensive as you would like to test or use a WHENEVER as a generic error handler.

__ 9. Prepare and create the program. You may use the following command, or PDM option 14, followed by F4=Prompt and completing the parameters appropriately. You

may also specify some options using the H-spec and the SET statement in SQL if you like:

```
CRTSQLRPGI OBJ(OL38nnCOL/BONUSDPTS)
           SRCFILE(OL38nnCOL/QRPGLESRC)
           COMMIT(*NONE)
           OUTPUT(*PRINT)
           CLOSQLCSR(*ENDMOD)
           DATFMT(*ISO)
           DBGVIEW(*SOURCE)
```

__ 10. Verify that your program compiled successfully and correct any errors. Review the spool file output from the SQL preprocessor.

Test your program BONUSDPTS

__ 11. Display the EMPLOYEE table. Decide which department should get the additional bonus and how much it should be.

__ 12. Run your program BONUSDPTS.

__ 13. Enter the department number and the amount of the additional bonus.

__ 14. Display the EMPLOYEE table as you did earlier. Verify that it has been updated correctly. If you would like to refresh your files, follow the procedure described earlier in the lab exercise guide. Note that if your program displays the new bonus amount before you perform the UPDATE of the table, the table will show the old bonus amount until you press **F3** to end the program.

__ 15. If your values do not get updated as expected, check that you specified COMMIT(*NONE) in the `CRTSQLRPGI` command above.

__ 16. If you have any problems, use the ILE debugger to debug your program.

End of exercise

Exercise review/wrapup

There are a number of points that you should review with the students:

1. Did they close the cursor explicitly or implicitly? If anyone did an implicit close, ask the student(s) to explain any unexpected behavior.
2. What kind of error handling did the students include? Ask various students to explain what they did and why they did it.
3. Did anyone need to debug their program? Explain the behavior of the ILE debugger when working with member type SQLRPGLE.

Exercise 13. Dynamic embedded SQL

Estimated time

02:00

What this exercise is about

In this exercise, you add code to an existing program using the dynamic fixed list select.

This program lists employees for a department in a subfile. The user enters the department number and can also specify the order of the rows selected (by employee number or by surname).

What you should be able to do

At the end of the lab, you should be able to:

- Create an RPG IV program that includes dynamic fixed list select
- Include basic error handling and use SQLCOD
- Explain the considerations when using dynamic SQL and compare the use of dynamic versus static SQL

Introduction

You will use a program that has the subfile logic written and will add the SQL statements to return a result set that will load the subfile. You will use a parameter marker and you will dynamically build the ORDER BY clause.

Instructor exercise overview

Explain the existing program shell and display file provided to the students. You might demonstrate the solution program, DYNFSELS in the course library. Be sure to emphasize that they must build the SQL statement using a parameter marker for the WORKDEPT as well as use a character string to build the ORDER BY clause.

Show them that the fields for the sequence are set to protected once the user has chosen his order the first time but that the user can change the department number over and over. You cannot change the ORDER BY once the SQL statement is prepared unless you write your code so that it prepares the statement more than once. Our program is written so that the statement is prepared once but the WORKDEPT is a parameter than can be supplied over and over to the prepared statement.

The most difficult coding in the lab is building and preparing the SQL Statement string. The students need to decide what sequence to use and then build the SQLString variable. Talk about this and, if necessary, work on the board to show how the statement might be created. Once the statement string has been completely built, it must be prepared.

Exercise instructions

Review existing source members

__ 1. Change your current library to OL38*nn*COL.

__ 2. Locate copies of the display file, **DYNFSDSPF**, and the RPG IV procedure, **DYNFSEL**, in your library.

__ 3. Review the source of display file, **DYNFSDSPF**, below:

```
A                                      REF(EMPLOYEE)
A                                      CA03(03)
A                                      INDARA

 ** Subfile Record format

A          R RECORD                    SFL
A            WORKDEPT  R        O  9  5
A            EMPNO     R        O  9 20
A            LASTNAME  R        O  9 55

 ** Subfile Control format

A          R CONTROL                   SFLCTL(RECORD)
A                                      SFLSIZ(0011)
A                                      SFLPAG(0010)
A  85                                  SFLDSPCTL
A  95                                  SFLDSP
A  75                                  SFLCLR
A  90                                  SFLEND(*MORE)
A                                      OVERLAY PUTOVR

A                                   2  7DATE   EDTCDE(Y)
A                                   2 32'Employee List'
A                                       DSPATR(HI)
A                                   2 65USER
A                                   4  2'Enter Work Department:'
A            WORKDEPT  R        I  4 30
A                                   5  2'Sequence:'
A                                   5 15'Last Name?'
A            NAMEORDER      1A  B  5 30OVRDTA
A  30                                  DSPATR(PR)
A                                   5 35'Employee Number?'
A            EMPORDER       1A  B  5 55OVRDTA
A  30                                  DSPATR(PR)
A
A                                   7  5'Work Dept.'
A                                   7 20'Employee'
A*                                  7 28'First Name'
A                                   7 55'Surname'

 ** Function key narrative

A          R FNKEYS
A                                  23  7'F3=Exit'

 ** MESSAGE FOR EMPTY SUBFILE

A          R MSG
A                                      OVERLAY
A                                  12 32'No Records Found'
A                                      DSPATR(HI)
```


__ 4. Notice the following about the display file:

- Record format names RECORD and CONTROL.
- Input variables are WORKDEPT and NAMEORDER and EMPORDER for sequence (ORDER BY clause). Notice that the NAMEORDER and EMPORDER fields can be made protected.
- Function key 3 is used to terminate the application.

Create display file DYNFSDSPF

__ 5. Create display file DYNFSDSPF, in your library OL38*nn*COL. Verify that it is created successfully.

Review the RPGLE program source, DYNFSEL

__ 6. Review the existing source member, **DYNFSEL**, in your schema and make sure that the member type is SQLRPGLE:

```
H DftActGrp(*No) ActGrp(*Caller)

FDynFSDSPF CF   E             Workstn Sfile(Record:Rrn)
F                                     IndDS(WkStnIndics)

D WkStnIndics     DS
D  Exit                   3      3N
D  SflEnd                90     90N
D  SflDspCtl             85     85N
D  SflDsp                95     95N
D  SflClr                75     75N
D  Protect               30     30N

D Rrn             S              4  0 INZ
D EmptySfl        S               N
D Sequence        S             20A

D SqlString       S            100A

C/Exec Sql
C+ set option commit=*none
C/End-Exec

 /FREE
  DoW not Exit;
    ExSR Search;
  EndDo;

  ExSR CloseCursor;
  *InLr = *on;

   // subroutines
  BegSR *InzSR;     // Initialization subroutine
    Write FNKeys;
    SflDspCtl = *On;
    Exfmt Control;
    ExSR Prepare;
    ExSR Declare;
    Protect = *on;
  Endsr;

  BegSR Search;
    ExSr SFLClear;         // Clear subfile for new search

    ExSR OpenCursor;       // Position file cursor
                           // using search key
    ExSR FetchRow;         // Fetch first row


   Rrn = 1;               // Rrn set to 1 even if we are at EOF
   Exsr Fill;             // Fill Subfile
   Exsr Prompt;           // Prompt for new department
 EndSR;
```



```
Begsr Fill;

 // Load entire subfile
 DoW  SqlCod <> 100 AND (Rrn <= 9999);
    Write Record;
    ExSR FetchRow;       // Process remaining rows
    Rrn = Rrn + 1;
  Enddo;

  EmptySfl = (Rrn <= 1); // No records to display?
  SflEnd = (SqlCod = 100); // Also should consider whether or not Sfl has
                           // been filled. Future enhancement.
 Endsr;

 Begsr Prompt;

   If NOT EmptySfl;

     SflDspCtl = *ON; // Display Subfile
     SflDsp = *ON;

   Else;

     SflDspCtl = *ON; // No records to view - display message
     SflDsp = *OFF;
     Write Msg;
   Endif;


   Exfmt Control;
 Endsr;

 Begsr SFLCLear;   // Subfile clear subroutine
   SflClr = *on;
   SflDsp = *OFF;
   SflDspCtl = *OFF;
   Write Control;  // New search - clear subfile
   SflClr = *off;
   ExSR CloseCursor;
 EndSR;
/END-FREE
  //
  // Exercise is to code the SQL Statements for each of these subroutines
  //
  // Build Select String Variable and Prepare it once
C     Prepare       BegSR
  Add code
C                   EndSR
  // Declare Cursor
C     Declare       BegSR
  Add code
C                   EndSR
  // Open Cursor
C     OpenCursor    BegSR
  Add code
```

```
C                   EndSR
  // Process a row
C     FetchRow      BegSR
  Add code
C                   EndSR
  // Close cursor
C     CloseCursor   BegSR
  Add code
C                   EndSR
```

__ 7. Notice each of the subroutines in the RPG IV program. For each of the subroutines, add the appropriate SQL statements to perform the task.

__ 8. Review the logic of the existing program. Basically, the user enters a value for the **WorkDept**. The user also specifies the order in which the retrieved rows are to be displayed in the subfile. **EmpNo** order is the default. Once the user has selected an order, the two entry fields are changed to protected. The user can enter more departments to display. **F3** exits the program.

Compile and test the program DYNFSEL

__ 9. Notice that your program has the appropriate compiler options set in the H-spec and in the SQL SET statement.

__ 10. Verify that your program compiled successfully and correct any errors. Review the spool file output from the SQL preprocessor.

__ 11. Open iSeries Navigator and display the contents of the Employee table. Use this to choose various values for **WorkDept** to enter when you run your program.

__ 12. When you run the program, use any value of work department (such as A00, B01, C01) and specify your sequence to test the program and to see how it works.

Note

Watch for and check SQLCOD error messages; you may encounter unexpected values when you debug the program (if necessary). Check the manual for the description of the error. If you cannot find the message number, go to:

`http://publib.boulder.ibm.com/iseries/v5r2/ic2924/index.htm`

Click **Database** and **SQL Message Finder**.

__ 13. If you have any problems, use the ILE debugger to debug your program.

List SQL statement processing order

__ 14. As you recall from the lecture, the dynamic statements for a fixed list select must be executed in a very specific order. List the statements and the order below. Try to list them using your program as a guide:

_____

_____

_____

_____

_____

_____

_____

_____

End of exercise

Exercise Review/Wrapup

There are a number of points that you should review with the students:

1. Did anyone need to debug his or her program? Ask the students to discuss the behavior of the ILE debugger when working with member type SQLRPGLE.
2. Did anyone get any SQLCODes that needed to be looked up? What were they? How did they discover what caused the problem? How did the students fix the problems?
3. Discuss the order in which SQL statements must be executed for a Dynamic Fixed List Select:
   a. Prepare Select statement
   b. Declare cursor for result set
   c. Open Cursor
   d. Fetch rows one at a time
   e. Close cursor

Exercise 14. Create a stored procedure

Estimated time

00:30

What this exercise is about

This exercise provides an opportunity to create and run an external stored procedure. You will be provided with an RPG IV program that you will compile and use to create a *PGM. You will code and create the stored procedure that will call this program.

What you should be able to do

At the end of the lab, you should be able to:

- Create an RPG IV *PGM object that can be called by a SQL Client using a stored procedure
- Create an external stored procedure
- Use System i Navigator to test a stored procedure

Exercise instructions

Review the RPG IV program

__ 1. Confirm that you have a source member, APINVRSET in your QRPGLESRC file.

```
C/EXEC SQL
C+ Declare C1 Cursor For
C+ SELECT PoNbr, VndNbr, ApInvNbr, ApCheck#
C+    FROM ApInv_PF
C/END-EXEC


C/Exec Sql
C+  Open C1
C/End-Exec
C*
C                   If         SqlCod <> 100
C/Exec Sql
C+  Set Result Sets Cursor C1
C/End-Exec

C                   EndIf

C                   Eval      *InLR = *ON
C                   Return
```

__ 2. Notice that it is very similar to the program that we described in lecture *Stored Procedure with Parameter*. This version of the program does not accept a parameter and returns a result set of more than one row. Use the lecture notes as a guide in the rest of this exercise. Make sure that your current library is OL38nnCOL.

Create the *PGM object

__ 3. Compile this source using option **15** of PDM with the steps below or using H-spec keywords and the SQL SET OPTION statement. Please answer the questions in

relation to the way you create the module. You can also create the module using these steps, giving special attention to the underlined parameters:

```
                      Create SQL ILE RPG Object (CRTSQLRPGI)

Type choices, press Enter.

Object . . . . . . . . . . . . . . OBJ          > APINVRSET
  Library  . . . . . . . . . . . .               >   OL38nnCOL
Source file  . . . . . . . . . . . SRCFILE       > QRPGLESRC
  Library  . . . . . . . . . . . .               >   OL38nnCOL
Source member  . . . . . . . . . . SRCMBR        > APINVRSET
Commitment control . . . . . . . . COMMIT          *NONE
Relational database  . . . . . . . RDB             *LOCAL
Compile type . . . . . . . . . . . OBJTYPE       > *MODULE
Listing output . . . . . . . . . . OUTPUT          *PRINT
Text 'description' . . . . . . . . TEXT            *SRCMBRTXT


                            Additional Parameters

Debugging view . . . . . . . . . . DBGVIEW         *SOURCE
```

__ 4. Why are these parameters important?

_____

_____

__ 5. Create a *PGM object using either PDM option **26** next to the module or the command below, again giving special attention to the underlined parameters:

```
                          Create Program (CRTPGM)

Type choices, press Enter.

Program  . . . . . . . . . . . . . PGM           > APINVRSET
  Library  . . . . . . . . . . . .               >   OL38nnCOL
Module . . . . . . . . . . . . . . MODULE          *PGM
  Library  . . . . . . . . . . . .
                           + for more values

Text 'description' . . . . . . . . TEXT            *ENTMODTXT


                            Additional Parameters

Activation group . . . . . . . . . ACTGRP        > *CALLER
```

__ 6. Why is this parameter important?

_____

Code stored procedure

Use the stored procedures in the Student Notes as a guide. What you need to create is an SP that will produce a result set with no parameters. Use the **Run SQL Scripts** facility.

__ 7. Double-click the **System i Navigator** icon on your PC.

__ 8. Double-click the class server name.

__ 9. Sign on if prompted using your assigned userid and password.

__ 10. Once signed on, you should see the **Database** icon. Expand it.

__ 11. Right-click the system name under the **Database** icon and select **Run SQL Scripts**.

__ 12. When the window appears, System i select **Connection**, then **JDBC Setup**.

__ 13. Click the **format** tab and make sure that the naming convention is set to `*SYS` (system naming convention).

__ 14. Click **OK**.

__ 15. In the white space of the **Run SQL Scripts** window, enter the statements to create your stored procedure. It should be named **APRESULT** and should be specified in your **OL38*nn*COL** (use fully qualified names). Note that the table that is used, **APINV_PF**, should be specified in the same schema.

__ 16. Do not forget to include a semicolon as needed in your stored procedure as well as at the end of the last statement.

Note

- You may choose to use Interactive SQL or RUNSQLSTM rather than iSeries Navigator to create the procedure.
- If you are recreating the procedure because of a problem or an enhancement, you must DROP the existing procedure first.

Create stored procedure

__ 17. Now that you have entered the statements to create your Stored Procedure, you must run them.

__ 18. From the **Run** option on the menu bar, select **All**.

__ 19. If your Stored Procedure is created (you will see the message **Statement ran successfully**), proceed to the next step.

__ 20. If you have problems, notice the messages and attempt to correct them. If you need any help, please ask the instructor.

Test stored procedure

__ 21. In the **Run SQL Scripts** window, enter the **CALL** statement to run your Stored Procedure.

__ 22. Highlight your **Call** statement.

__ 23. Run this statement using the **Run** option on the menu bar. Select **Run Selected**.

__ 24. Once you see the result set displayed in the **Results Viewer**, you have completed the exercise.

End of exercise

Appendix A. Tables used for SQL exercises

APINV_PF table

Column Name	Description
PONBR	Purch Order Number
VNDNBR	Vendor Number
APINVNBR	Vendor Invoice Number
APDATE	Date Completed
POTOTAMT	Amount
APDISCOUNT	Invoice Discount Available
APNETPAID	Net Amount Paid
APSTATUS	Accounts Payable Status
APDATEPAID	Date Paid
APCHECK#	Check (Cheque) Number
APDUEDATE	Due Date

Data in APINV_PF table

Purch Order Number	Vend Num	Vendor Invoice Number	Date Compl YYYYMMDD	Purch Order Amount	Inv Disc Avail	Net Amount Paid	AP Sts	Date Paid YYYYMMDD	Check Number	Due Date YYYYMMDD
300,071	10,005	5000	1996-04-01	1000.00	50.00	950.00	P	1996-04-11	70,007	1996-04-11
300,072	10,005	1005	1996-06-01	2000.00	300.00	.00		9999-12-31	0	1996-06-30
300,073	10,015	1580	1998-04-01	3000.00	150.00	2850.00	P	1998-04-11	7,008	1998-04-11
300,074	10,020	2007	1998-05-15	4000.00	200.00	4000.00	P	1999-06-01	7,009	1998-05-25
300,075	10,025	25697	1998-03-15	500.00	50.00	.00		9999-12-31	0	1998-03-31
300,076	10,030	301	1998-08-03	600.00	60.00	.00		9999-12-31	0	1998-08-15
300,077	10,035	350001	1998-02-01	700.00	70.00	630.00	P	1998-03-15	7,010	1998-03-01
300,078	10,040	40079	1998-09-03	800.00	80.00	.00		9999-12-31	0	1998-09-24
300,079	10,045	45090	1998-06-03	900.00	90.00	.00		9999-12-31	0	1998-06-25
300,080	10,050	500777	1998-07-22	100.00	5.00	.00		9999-12-31	0	1998-08-15
300,081	10,005	5001	1998-06-26	2000.00	200.00	.00	D	9999-12-31	0	1998-07-15
300,082	10,010	1006	1998-05-23	3000.00	450.00	.00		9999-12-31	0	1998-06-16
300,083	10,015	1583	1998-09-02	4000.00	200.00	3800.00	P	1998-09-30	7,011	1998-09-30
300,084	10,020	2013	1998-01-31	5000.00	250.00	4750.00	P	1998-03-15	7,012	1998-02-14
300,085	10,025	25797	1998-04-22	6000.00	600.00	.00		9999-12-31	0	1998-05-03
300,086	10,030	388	1998-08-31	7000.00	700.00	.00		9999-12-31	0	1998-09-23
300,087	10,035	350007	1998-08-21	1000.00	100.00	.00		9999-12-31	0	1998-09-03
300,088	10,040	40100	1998-05-03	1500.00	150.00	.00		9999-12-31	0	1998-06-23
300,089	10,045	45601	1998-09-03	1400.00	140.00	.00		9999-12-31	0	1998-09-30
300,090	10,050	50789	1998-09-03	1200.00	60.00	.00		9999-12-31	0	1998-10-02
300,091	10,005	5009	1998-09-03	1400.00	90.00	.00		9999-12-31	0	1998-10-14
300,092	10,010	1099	1998-09-03	100.00	15.00	.00		9999-12-31	0	1998-10-31
300,093	10,015	1599	1998-09-03	800.00	40.00	.00		9999-12-31	0	1998-10-03
300,094	10,020	2077	1998-09-03	700.00	35.00	.00		9999-12-31	0	1998-09-30
300,095	10,025	25900	1998-09-03	600.00	60.00	.00		9999-12-31	0	1998-09-25
300,096	10,030	399	1998-09-03	500.00	50.00	.00		9999-12-31	0	1998-11-03
300,097	10,035	35019	1998-07-03	400.00	40.00	400.00	P	1998-07-31	7,013	1998-07-26

DEPARTMENT table

Column Name	Description
DEPTNO	Department number or ID.
DEPTNAME	A name describing the general activities of the department.
MGRNO	Employee number (EMPNO) of the department manager.
ADMRDEPT	The department (DEPTNO) to which this department reports; the department at the highest level reports to itself.

Data in DEPARTMENT table

DEPTNO	DEPTNAME	MGRNO	ADMRDEPT
******	********	*****	********
A00	SPIFFY COMPUTER SERVICE DIV.	000010	A00
B01	PLANNING	000020	A00
C01	INFORMATION CENTER	000030	A00
D01	DEVELOPMENT CENTER	?	A00
D11	MANUFACTURING SYSTEMS	000060	D01
D21	ADMINISTRATION SYSTEMS	000070	D01
E01	SUPPORT SERVICES	000050	A00
E11	OPERATIONS	000090	E01
E21	SOFTWARE SUPPORT	000100	E01

EMPLOYEE table

Column Name	Description
EMPNO	Employee number
FIRSTNME	First name of employee
MIDINIT	Middle initial of employee
LASTNAME	Last name of employee
WORKDEPT	ID of department in which the employee works
PHONENO	Employee telephone number
HIREDATE	Date of hire
JOB	Job held by the employee
EDLEVEL	Number of years of formal education
SEX	Sex of the employee (M or F)
BIRTHDATE	Date of birth
SALARY	Yearly salary in dollars
BONUS	Yearly bonus in dollars
COMM	Yearly commission in dollars

Data in EMPLOYEE table

EMP NO	FIRST NAME	MID INIT	LAST NAME	WORK DEPT	PHONE NO	HIRE DATE	ED JOB	LEVEL	SEX	BIRTH DATE	SALARY	BONUS	COMM
******	*****	****	****	****	*****	**********	***	*****	***	**********	******	*****	****
000010	CHRISTINE	I	HAAS	A00	3978	1965-01-01	PRES	18	F	1933-08-24	52750	1000	4220
000020	MICHAEL	L	THOMPSON	B01	3476	1973-10-10	MANAGER	18	M	1948-02-02	41250	800	3300
000030	SALLY	A	KWAN	C01	4738	1975-04-05	MANAGER	20	F	1941-05-11	38250	800	3060
000050	JOHN	B	GEYER	E01	6789	1949-08-17	MANAGER	16	M	1925-09-15	40175	800	3214
000060	IRVING	F	STERN	D11	6423	1973-09-14	MANAGER	16	M	1945-07-07	32250	500	2580
000070	EVA	D	PULASKI	D21	7831	1980-09-30	MANAGER	16	F	1953-05-26	36170	700	2893
000090	EILEEN	W	HENDERSON	E11	5498	1970-08-15	MANAGER	16	F	1941-05-15	29750	600	2380
000100	THEODORE	Q	SPENSER	E21	0972	1980-06-19	MANAGER	14	M	1956-12-18	26150	500	2092
000110	VINCENZO	G	LUCCHESSI	A00	3490	1958-05-16	SALESREP	19	M	1929-11-05	46500	900	3720
000120	SEAN		O'CONNELL	A00	2167	1963-12-05	CLERK	14	M	1942-10-18	29250	600	2340
000130	DOLORES	M	QUINTANA	C01	4578	1971-07-28	ANALYST	16	F	1925-09-15	23800	500	1904
000140	HEATHER	A	NICHOLLS	C01	1793	1976-12-15	ANALYST	18	F	1946-01-19	28420	600	2274
000150	BRUCE		ADAMSON	D11	4510	1972-02-12	DESIGNER	16	M	1947-05-17	25280	500	2022
000160	ELIZABETH	R	PIANKA	D11	3782	1977-10-11	DESIGNER	17	F	1955-04-12	22250	400	1780
000170	MASATOSHI	J	YOSHIMURA	D11	2890	1978-09-15	DESIGNER	16	M	1951-01-05	24680	500	1974
000180	MARILYN	S	SCOUTTEN	D11	1682	1973-07-07	DESIGNER	17	F	1949-02-21	21340	500	1707
000190	JAMES	H	WALKER	D11	2986	1974-07-26	DESIGNER	16	M	1952-06-25	20450	400	1636
000200	DAVID		BROWN	D11	4501	1966-03-03	DESIGNER	16	M	1941-05-29	27740	600	2217
000210	WILLIAM	T	JONES	D11	0942	1979-04-11	DESIGNER	17	M	1953-02-23	18270	400	1462
000220	JENNIFER	K	LUTZ	D11	0672	1968-08-29	DESIGNER	18	F	1948-03-19	29840	600	2387
000230	JAMES	J	JEFFERSON	D21	2094	1966-11-21	CLERK	14	M	1935-05-30	22180	400	1774
000240	SALVATORE	M	MARINO	D21	3780	1979-12-05	CLERK	17	M	1954-03-31	28760	600	2301
000250	DANIEL	S	SMITH	D21	0961	1969-10-30	CLERK	15	M	1939-11-12	19180	400	1534
000260	SYBIL	P	JOHNSON	D21	8953	1975-09-11	CLERK	16	F	1936-10-05	17250	300	1380
000270	MARIA	L	PEREZ	D21	9001	1980-09-30	CLERK	15	F	1953-05-26	27380	500	2190
000280	ETHEL	R	SCHNEIDER	E11	8997	1967-03-24	OPERATOR	17	F	1936-03-28	26250	500	2100
000290	JOHN	R	PARKER	E11	4502	1980-05-30	OPERATOR	12	M	1946-07-09	15340	300	1227
000300	PHILIP	X	SMITH	E11	2095	1972-06-19	OPERATOR	14	M	1936-10-27	17750	400	1420
000310	MAUDE	F	SETRIGHT	E11	3332	1964-09-12	OPERATOR	12	F	1931-04-21	15900	300	1272
000320	RAMLAL	V	MEHTA	E21	9990	1965-07-07	FILEREP	16	M	1932-08-11	19950	400	1596
000330	WING		LEE	E21	2103	1976-02-23	FILEREP	14	M	1941-07-18	25370	500	2030
000340	JASON	R	GOUNOT	E21	5698	1947-05-05	FILEREP	16	M	1926-05-17	23840	500	1907

EMP_ACT table

Column Name	Description
EMPNO	Employee ID number
PROJNO	PROJNO of the project to which the employee is assigned
ACTNO	ID of an activity within a project to which an employee is assigned
EMPTIME	A proportion of the employee's full time (between 0.00 and 1.00) to be spent on the project from EMSTDATE to EMENDATE
EMSTDATE	Start date of the activity
EMENDATE	Completion date of the activity

Data in EMP_ACT table

EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
000010	AD3100	10	.50	1982-01-01	1982-07-01
000070	AD3110	10	1.00	1982-01-01	1983-02-01
000230	AD3111	60	1.00	1982-01-01	1982-03-15
000230	AD3111	60	.50	1982-03-15	1982-04-15
000230	AD3111	70	.50	1982-03-15	1982-10-15
000230	AD3111	80	.50	1982-04-15	1982-10-15
000230	AD3111	180	1.00	1982-10-15	1983-01-01
000240	AD3111	70	1.00	1982-02-15	1982-09-15
000240	AD3111	80	1.00	1982-09-15	1983-01-01
000250	AD3112	60	1.00	1982-01-01	1982-02-01
000250	AD3112	60	.50	1982-02-01	1982-03-15
000250	AD3112	60	.50	1982-12-01	1983-01-01
000250	AD3112	60	1.00	1983-01-01	1983-02-01
000250	AD3112	70	.50	1982-02-01	1982-03-15
000250	AD3112	70	1.00	1982-03-15	1982-08-15
000250	AD3112	70	.25	1982-08-15	1982-10-15
000250	AD3112	80	.25	1982-08-15	1982-10-15
000250	AD3112	80	.50	1982-10-15	1982-12-01
000250	AD3112	180	.50	1982-08-15	1983-01-01
000260	AD3113	70	.50	1982-06-15	1982-07-01
000260	AD3113	70	1.00	1982-07-01	1983-02-01
000260	AD3113	80	1.00	1982-01-01	1982-03-01
000260	AD3113	80	.50	1982-03-01	1982-04-15
000260	AD3113	180	.50	1982-03-01	1982-04-15
000260	AD3113	180	1.00	1982-04-15	1982-06-01
000260	AD3113	180	.50	1982-06-01	1982-07-01
000270	AD3113	60	.50	1982-03-01	1982-04-01
000270	AD3113	60	1.00	1982-04-01	1982-09-01
000270	AD3113	60	.25	1982-09-01	1982-10-15
000270	AD3113	70	.75	1982-09-01	1982-10-15
000270	AD3113	70	1.00	1982-10-15	1983-02-01
000270	AD3113	80	1.00	1982-01-01	1982-03-01
000270	AD3113	80	.50	1982-03-01	1982-04-01
000030	IF1000	10	.50	1982-06-01	1983-01-01
000130	IF1000	90	1.00	1982-01-01	1982-10-01
000130	IF1000	100	.50	1982-10-01	1983-01-01
000140	IF1000	90	.50	1982-10-01	1983-01-01
000030	IF2000	10	.50	1982-01-01	1983-01-01
000140	IF2000	100	1.00	1982-01-01	1982-03-01
000140	IF2000	100	.50	1982-03-01	1982-07-01
000140	IF2000	110	.50	1982-03-01	1982-07-01
000140	IF2000	110	.50	1982-10-01	1983-01-01
000010	MA2100	10	.50	1982-01-01	1982-11-01
000110	MA2100	20	1.00	1982-01-01	1982-03-01
000010	MA2110	10	1.00	1982-01-01	1983-02-01
000200	MA2111	50	1.00	1982-01-01	1982-06-15
000200	MA2111	60	1.00	1982-06-15	1983-02-01
000220	MA2111	40	1.00	1982-01-01	1983-02-01
000150	MA2112	60	1.00	1982-01-01	1982-07-15
000150	MA2112	180	1.00	1982-07-15	1983-02-01
000170	MA2112	60	1.00	1982-01-01	1983-06-01
000170	MA2112	70	1.00	1982-06-01	1983-02-01
000190	MA2112	70	1.00	1982-02-01	1982-10-01
000190	MA2112	80	1.00	1982-10-01	1983-10-01
000160	MA2113	60	1.00	1982-07-15	1983-02-01
000170	MA2113	80	1.00	1982-01-01	1983-02-01
000180	MA2113	70	1.00	1982-04-01	1982-06-15
000210	MA2113	80	.50	1982-10-01	1983-02-01
000210	MA2113	180	.50	1982-10-01	1983-02-01
000050	OP1000	10	.25	1982-01-01	1983-02-01
000090	OP1010	10	1.00	1982-01-01	1983-02-01
000280	OP1010	130	1.00	1982-01-01	1983-02-01
000290	OP1010	130	1.00	1982-01-01	1983-02-01
000300	OP1010	130	1.00	1982-01-01	1983-02-01
000310	OP1010	130	1.00	1982-01-01	1983-02-01
000050	OP2010	10	.75	1982-01-01	1983-02-01
000100	OP2010	10	1.00	1982-01-01	1983-02-01
000320	OP2011	140	.75	1982-01-01	1983-02-01
000320	OP2011	150	.25	1982-01-01	1983-02-01
000330	OP2012	140	.25	1982-01-01	1983-02-01
000330	OP2012	160	.75	1982-01-01	1983-02-01
000340	OP2013	140	.50	1982-01-01	1983-02-01
000340	OP2013	170	.50	1982-01-01	1983-02-01
000020	PL2100	30	1.00	1982-01-01	1982-09-15

PROJECT table

Column Name	Description
PROJNO	Project number
PROJNAME	Project name
DEPTNO	Department number of the department responsible for the project
RESPEMP	Employee number of the person responsible for the project
PRSTAFF	Estimated mean staffing
PRSTDATE	Estimated start date of the project
PRENDATE	Estimated end date of the project
MAJPROJ	Controlling project number for sub projects

Data in PROJECT table

PROJNO	PROJNAME	DEPTNO	RESPEMP	PRSTAFF	PRSTDATE	PRENDATE	MAJPROJ
******	********	******	*******	*******	********	********	*******
AD3100	ADMIN SERVICES	D01	000010	6.5	1982-01-01	1983-02-01	?
AD3110	GENERAL ADMIN SYSTEMS	D21	000070	6	1982-01-01	1983-02-01	AD3100
AD3111	PAYROLL PROGRAMMING	D21	000230	2	1982-01-01	1983-02-01	AD3110
AD3112	PERSONNEL PROGRAMMING	D21	000250	1	1982-01-01	1983-02-01	AD3110
AD3113	ACCOUNT PROGRAMMING	D21	000270	2	1982-01-01	1983-02-01	AD3110
IF1000	QUERY SERVICES	C01	000030	2	1982-01-01	1983-02-01	?
IF2000	USER EDUCATION	C01	000030	1	1982-01-01	1983-02-01	?
MA2100	WELD LINE AUTOMATION	D01	000010	12	1982-01-01	1983-02-01	?
MA2110	W L PROGRAMMING	D11	000060	9	1982-01-01	1983-02-01	MA2100
MA2111	W L PROGRAM DESIGN	D11	000220	2	1982-01-01	1982-12-01	MA2110
MA2112	W L ROBOT DESIGN	D11	000150	3	1982-01-01	1982-12-01	MA2110
MA2113	W L PROD CONT PROGS	D11	000160	3	1982-02-15	1982-12-01	MA2110
OP1000	OPERATION SUPPORT	E01	000050	6	1982-01-01	1983-02-01	?
OP1010	OPERATION	E11	000090	5	1982-01-01	1983-02-01	OP1000
OP2000	GEN SYSTEMS SERVICES	E01	000050	5	1982-01-01	1983-02-01	?
OP2010	SYSTEMS SUPPORT	E21	000100	4	1982-01-01	1983-02-01	OP2000
OP2011	SCP SYSTEMS SUPPORT	E21	000320	1	1982-01-01	1983-02-01	OP2010
OP2012	APPLICATIONS SUPPORT	E21	000330	1	1982-01-01	1983-02-01	OP2010
OP2013	DB/DC SUPPORT	E21	000340	1	1982-01-01	1983-02-01	OP2010
PL2100	WELD LINE PLANNING	B01	000020	1	1982-01-01	1982-09-15	MA2100

Appendix B. Machine exercises: Sample solutions

Answers to exercise 1: Getting to know SQL

```
****************************************************************
Step 6 - CREATE STULST TABLE (MEMBER CRTSTULST)
****************************************************************
create table ol37nnlib/stulst
      (first_name char(15) not null,
      mid_init char(1) ,
      surname char(25) not null,
      cust_name varchar(50) not null,
      cust_no char(10) ,
      team_no numeric(2,0) not null,
      crse_code char(5) not null,
      crse_date date not null);
****************************************************************
```

Answers to exercise 2: How SELECT works

```
****************************************************************
* step 25 - Select all of the columns of data.
*
     select * from employee
*
****************************************************************
* step 26 - Select only a specific subset of columns of data.
*
     select workdept, job, salary from employee
*
****************************************************************
* step 27 - See those job classifications that are non-managers.
*
     select workdept, job, salary
        from employee
        where job not in ('PRES', 'MANAGER')

****************************************************************
* step 28
*
```

This statement does not work because GROUP BY needs unique values by job within department.

Since SALARY has an occurrence for each employee, you need to determine the average of SALARY.

The specific error is: Column SALARY or expression in SELECT is not valid.

```
****************************************************************
* step 29
*
```

We have a single row for each JOB classification within a department. For each row, an average salary has been Calculated.

We need to sequence the average salary column in descending order.

Answers to exercise 3: Coding SELECT statements

```
*************************************************************
* 5250 section       - step 4
* Navigator section  - step 6
***
SELECT LASTNAME, FIRSTNME, WORKDEPT, BIRTHDATE, HIREDATE, SALARY
        FROM employee
        WHERE salary >= 30000.00

*************************************************************
* 5250 section       - step 5
* Navigator section  - step 7
***
SELECT LASTNAME, FIRSTNME, WORKDEPT, BIRTHDATE, HIREDATE, SALARY
        FROM employee WHERE salary < 20000.00
        order by lastname, firstnme

*************************************************************
* 5250 section        - step 6
* Navigator section   - step 8
***
SELECT *
        FROM department
        where mgrno = ' ' or mgrno is null

*************************************************************
* 5250 section        - step 7
* Navigator section   - step 9
***
select *
        from department
        where admrdept = 'A00'

*************************************************************
* 5250 section         - step 8
* Navigator section    - step 10
***
select deptno, deptname
        from department
        where deptname like '%Service%'
```

```
************************************************************
* 5250 section        - step 9
* Navigator section   - step 11
***
SELECT LASTNAME, FIRSTNME, WORKDEPT, phoneno
        FROM employee
        WHERE workdept between 'E01' and 'E11'

************************************************************
* 5250 section        - step 10
* Navigator section   - step 12
***
SELECT LASTNAME, FIRSTNME, WORKDEPT, phoneno
        FROM employee

        where workdept like 'E%'

************************************************************
* 5250 section         - step 11
* Navigator section    - step 13
***
SELECT LASTNAME, FIRSTNME, WORKDEPT, salary + comm
        FROM employee
        where workdept in ('B01', 'C01', 'D01')
        order by 4 desc

************************************************************
* 5250 section         - step 12
* Navigator section    - step 14
***
SELECT LASTNAME, salary, workdept
        FROM employee
        where salary/12 >3000
        order by lastname

************************************************************
* 5250 section         - step 13
* Navigator section    - step 15
***
select empno, lastname, salary/12
        from employee
        where salary/12 < 1600 and sex = 'M'
        order by 3 desc
```

```
*************************************************************
* 5250 section         - step 14
* Navigator section    - step 16
***
select lastname, (100 * comm)/(salary + comm + bonus)
        from employee
        where job = 'SALESREP'

*************************************************************
* 5250 section         - step 15
* Navigator section    - step 17
***
select *
        from department
        where deptno='E01' or ADMRDEPT='E01'

*************************************************************
* 5250 section        - step 16
* Navigator section   - step 18
***
select lastname, salary, job, educlvl
        from employee
        where salary > 40000
           or (job = 'MANAGER' and educlvl < 16)
```

Answers to exercise 4: Using SQL functions

```
*********************************************************
*                    - step 3
***

select sum(salary), avg(salary), min(salary), max(salary)
         from employee

*********************************************************
*                    - step 4
***

select min(lastname) from employee

*********************************************************
*                    - step 5
***

select count(distinct workdept) from employee

*********************************************************
*                     - step 6
***

select job, avg(salary) as Average_Salary
         from employee
         group by job

*********************************************************
*                     - step 7
***

select job,avg(salary) as Average_Salary
         from employee
         group by job
         having avg(salary) > 35000

*********************************************************
*                       - step 8
***

select workdept, avg(salary) as Average_salary,
         count(*) as number_employees
```

```
        from employee
        where job <> 'PRES'
        group by workdept
        having count(*) >= 3
        order by 3 desc

*******************************************************
*                      - step 9
***

select workdept, cast(avg(salary) as decimal(7,2))
        as Average_salary,
           count(*) as number_employees
        from employee
        where job <> 'PRES'
        group by workdept
        having count(*) >= 3
        order by 3 desc

*******************************************************
*                      - step 10
***

select workdept, cast(round(avg(salary),2) as decimal (7,2))
        as Average_salary,
           count(*) as number_employees
           from employee
           where job <> 'PRES'
           group by workdept
           having count(*) >= 3
           order by 3 desc



*******************************************************

*                       - step 11
***

select workdept, cast(round(avg(salary),2) as decimal (7,2))
        as Average_salary, case
           when avg(salary)< 25000.00 then 'LOW'
        when avg(salary)> 30000.00 then 'HIGH'
        else 'MEDIUM'
```

```
        end as flag,
        count(*) as number_employees
        from employee
        where job <> 'PRES'
        group by workdept
        having count(*) >= 3
        order by 4 desc

********************************************************
*                       - step 12
***

select rtrim(firstnme)||' '||rtrim(lastname),
        year(curdate())-year(birthdate),
        char(salary)||'USD'
        from employee
```

Answers to exercise 5: JOINs and UNIONs

```
*********************************************************
*                      - step 2
***

select lastname, job, deptname
        from employee, department
        where workdept=deptno and deptname like '%Plan%'
     --OR--
   select lastname, job, deptname
        from employee inner join department
        on workdept=deptno where deptname like '%Plan%'

*********************************************************
*                       - step 3
***

select deptno, lastname
        from employee, department
        where job='MANAGER'
           and admrdept='D01'
           and deptno=workdept

     --OR--

   select deptno, lastname
        from employee inner join department
        on deptno=workdept
           where job='MANAGER'
           and admrdept='D01'

*********************************************************
*                       - step 4
***
select e.lastname, e.firstnme, e.workdept
        from employee e, employee a
        where e.workdept=a.workdept and a.lastname='Adamson'

     --OR--

   select e.lastname, e.firstnme, e.workdept
        from employee e inner join employee a
        on e.workdept=a.workdept where a.lastname='Adamson'
```

```
*********************************************************
*                     - step 5
***

select deptno, deptname, avg(salary), sex

         from department, employee
         where workdept=deptno
         group by deptno, deptname, sex
         order by deptno, 3 desc

      --OR--

select deptno, deptname, avg(salary), sex
         from department inner join employee
         on workdept=deptno
         group by deptno, deptname, sex
         order by deptno, 3 desc

*********************************************************
*                      - step 6
***

select empno, lastname,firstnme, 'NON-MANAGER' as job_title
         from employee
         where job<>'MANAGER' and workdept in ('B01','C01','E21')
         UNION ALL
         select empno, lastname,firstnme, 'MANAGER'
         from employee
         where job = 'MANAGER' and workdept in ('B01','C01','E21')

*********************************************************
*                       - step 7
***

SELECT DISTINCT PROJNO, EMP_ACT.EMPNO,
         LASTNAME||', '||FIRSTNME, COMM
         FROM EMP_ACT, EMPLOYEE
         WHERE EMP_ACT.EMPNO = EMPLOYEE.EMPNO
         AND COMM >= 2000.00
         ORDER BY PROJNO, EMPNO
```

```
*********************************************************
*                      - step 8
***


SELECT EMPNO, LASTNAME
         FROM EMPLOYEE EXCEPTION JOIN PROJECT
         ON EMPNO = RESPEMP

*****************************************************************
*                       - step 9
***

select 'Dept. A Headcount =', count(*), 'Total Salary =',
         sum(salary)
         from employee

               where workdept like 'A__'
            UNION ALL
            select 'Dept. D Headcount =', count(*), 'Total Salary =',
               sum(salary)
               from employee
               where workdept like 'D__'
            UNION ALL
            select 'Dept. E Headcount =', count(*), 'Total Salary =',
               sum(salary)
               from employee
               where workdept like 'E__'
```

The headings are based on the column headings of the initial (top level) SELECT statement.

Answers for exercise 6: Coding subSELECTs

```
*********************************************************
*                        - step 1
***

select workdept, lastname
         from employee
         where workdept in (select workdept
            from employee
            where lastname='Brown')

*********************************************************
*                        - step 2
***

select firstnme, lastname, phoneno, hiredate, job
         from employee
         where hiredate = (select min(hiredate)
            from employee)

*********************************************************
*                        - step 3
***

*********************************************************
select lastname, job, workdept
         from employee
         where workdept in (select deptno
            from department
            where deptname like '%Center%')

*********************************************************
*                        - step 4
***

select workdept, avg(salary)
         from employee
         group by workdept
         having avg(salary) < (select avg(salary)
            from employee)
```

```
*******************************************************
*                      - step 5
***

select empno, firstnme, lastname, workdept, salary
        from employee
        where workdept like 'E%'
        and sex='F'
        and salary > (select avg(salary)
        from employee
        where workdept like 'E%'
        and sex='M')


*******************************************************
*                     - step 6
***

select empno, firstnme, lastname, workdept, salary
        from employee x
        where sex='F'
        and salary < (select avg(salary)
           from employee
           where workdept = x.workdept
           and sex='M')


*******************************************************
*                     - step 7
***

select firstnme, lastname, salary
        from employee
        where workdept = 'E11'
        and salary < all (select avg(salary)
           from employee
           group by workdept)


*******************************************************
*                      - step 8
***
```

```
select workdept, lastname, firstnme, job, comm, bonus
         from employee c
         where job not in ('MANAGER', 'PRES')
         and comm = (select max(comm)
            from employee
            where workdept = c.workdept
            and job not in ('MANAGER', 'PRES'))
```

Answers to exercise 7: Using SQL Query Manager

```
**************************************************
*                      - step 18
*
***
```

A solution for the QMLABFORM follows:

```
**************************************************

Column Usage and Editing

                          Edit Column Formatting

Type information, press Enter.
For Usage and Edit, press F4 for list.
For Heading, press F4 for prompt.

Column          Heading              Usage Edit Seq Indent Width
PROJNO          Project_Number       COUNT        1    2       7
EMPNO           Employee_Number                   2    2       8
SEL1            Employee_Name                     3    2      29
SALARY          Salary               SUM   K2     4    2      13

**************************************************

F11 = Edit Column Headings:

                          Edit Column Formatting

Type information, press Enter.
For Usage and Edit, press F4 for list.
For Heading, press F4 for prompt.

Column          Heading
PROJNO          Project_Number
EMPNO           Employee_Number
SEL1            Employee_Name
SALARY          Salary

**************************************************
```

```
Page Heading:
                          Edit Page Heading

Type choices, press Enter.
  Blank lines before . . . . . . . . . . . . . 0     0-999
  Blank lines after . . . . . . . . . . . . . . 2       0-999

Page text: Use &col, &DATE, &TIME, and &PAGE to cause variable insertion.

Line Align          Page Heading Text
  1  left           &DATE &TIME
  1  CENTER         Employees Due Raises
  1  right          Page &PAGE


**************************************************
```

Answers to exercise 8: Enhancing an SQL table

```
******************************************************
Define Column LABELS for the STULST Table
******************************************************

LABEL ON COLUMN OL37nnLIB/STULST
(first_name is 'First Name',
mid_init is 'Middle_Initial',
surname is 'Surname',
cust_name is 'Customer Name',
cust_no is 'Customer_Number',
team_no is 'Team Number',
crse_code is 'Course Code',
crse_date is 'Course Date')

******************************************************
Insert rows of data to a table
******************************************************

INSERT INTO OL37nnLIB/STULST VALUES( 'John','J', 'Smith',
'ABC Widget Company', '991', 99, 'S6137', '2007-02-20')

******************************************************
Add a new column
******************************************************

ALTER TABLE OL37nnLIB/STULST
     ADD CLASS_CODE CHAR(4) NOT NULL WITH DEFAULT
```

Answers to exercise 9: Maintaining database objects

```
*****************************************************
*                      - step 2
***

      CREATE TABLE OL37nnCOL/TESTEMP
      (EMPNO CHAR(6) NOT NULL,
      LASTNAME VARCHAR(15) NOT NULL,
      WORKDEPT CHAR(3) ,
      HIREDATE DATE ,
      SALARY DECIMAL(9,2) ,
      BONUS DECIMAL(9,2) )

*****************************************************
*                      - step 3
***

      INSERT INTO OL37nnCOL/TESTEMP
      (EMPNO, LASTNAME, WORKDEPT, HIREDATE, SALARY, BONUS)
      VALUES ('000111', 'SMITH', 'C01', '06/06/2000', 25000, 0)

      INSERT INTO OL37nnCOL/TESTEMP
      (EMPNO, LASTNAME, WORKDEPT, HIREDATE, SALARY, BONUS)
      VALUES('000333', 'THOMAS', 'D11', '06/06/2000', 33000, 0)

      INSERT INTO OL37nn1COL/TESTEMP
      (EMPNO, LASTNAME, WORKDEPT, HIREDATE, SALARY)
      VALUES ('000222', 'BAKER', 'A00', '06/06/2000', 28000)
```

Note that the insert for Baker's row does not list the BONUS column in the column list, nor is its value specified in the VALUES clause. As a result, the BONUS for Baker's row will be marked as unknown (NULL).

```
*****************************************************
*                      - step 4
***
      INSERT INTO OL37nnCOL/TESTEMP
      (EMPNO,LASTNAME,WORKDEPT,HIREDATE,SALARY,BONUS )
      SELECT EMPNO, LASTNAME, WORKDEPT, HIREDATE, SALARY, BONUS
      FROM OL37nnCOL/EMPLOYEE
      WHERE EMPNO <= '000050'
```

Or:

```
INSERT INTO OL37nnCOL/TESTEMP
SELECT EMPNO, LASTNAME, WORKDEPT, HIREDATE, SALARY, BONUS
FROM OL37nnCOL/EMPLOYEE
WHERE EMPNO <= '000050'


*******************************************************
*                     - step 6
***

UPDATE OL37xxCOL/TESTEMP
SET BONUS = 500
WHERE EMPNO = '000111'

*******************************************************
*                    - step 7
***
UPDATE OL37nnCOL/TESTEMP
SET SALARY = SALARY + 1000
WHERE WORKDEPT = 'C01'

*******************************************************
*                    - step 8
***

DELETE FROM OL37nnCOL/TESTEMP
WHERE EMPNO = '000111'

*******************************************************
*                     - step 9
***
INSERT INTO OL37nnCOL/TESTEMP
(EMPNO,LASTNAME,WORKDEPT,HIREDATE,SALARY,BONUS)
SELECT EMPNO, LASTNAME, WORKDEPT, HIREDATE, SALARY, BONUS
FROM OL37nnCOL/EMPLOYEE
WHERE EMPNO > '000050'
```

Or:

```
INSERT INTO OL37xxCOL/TESTEMP
SELECT EMPNO, LASTNAME, WORKDEPT, HIREDATE, SALARY, BONUS
FROM OL37nnCOL/EMPLOYEE
WHERE EMPNO > '000050'
```

Rows inserted: 28

```
******************************************************
*                  - step 10a
***
     UPDATE OL37nnCOL/TESTEMP
     SET WORKDEPT = 'E01'
     WHERE EMPNO = '000100'

******************************************************
*                  - step 10b
***

     INSERT INTO OL37nnCOL/TESTEMP
     VALUES ('000360', 'BROWN ', 'D01', CURRENT DATE, 45000, NULL )
```

Note: For the above syntax to work, you must provide a value for every column, in the sequence in which the values must be inserted.

or

```
INSERT INTO OL37xxCOL/TESTEMP
(EMPNO, LASTNAME, WORKDEPT, HIREDATE, SALARY )
VALUES ('000360', 'BROWN ', 'D01', CURRENT DATE, 45000 )
```

Note: Column names and data values have a one-to-one correspondence. Therefore, the first value will go into the first column named, and the second value will go into the second column named, and so on. Columns receiving their default values might be omitted from this syntax.

```
******************************************************
*                  - step 12
***

     DELETE FROM OL37nnCOL/TESTEMP

******************************************************
*                   - step 14
***
     DROP TABLE OL37nnCOL/TESTEMP
```

Exercise 10. Change CHAIN to SELECT

```
 ** Item Master File
F*Item_PF   IF   E           K Disk
 ** Display File
FItemInqDspCF   E             Workstn

C                   Dow       NOT *In03
C*    ItmNbr        Chain     Item_PF                          40
C/Exec SQL
C+ select itmnbr, itmdescr, itmqtyoh, itmqtyoo, vndnbr, itmvndcat#
C+  into :itmnbr,:itmdescr,:itmqtyoh,
C+       :itmqtyoo,:vndnbr,:itmvndcat#
C+        from item_pf
C+        where itmnbr = :itmnbr
C/End-exec

c                   Eval      *in40 = *off

C                   If        NOT *In40
 ** Display details
C                   Dow       NOT *In12
C                   Eval      *In30 = (ItmQtyOH + ItmQtyOO) < 20
C                   Exfmt     Detail

C                   IF        *In03
C                   Eval      *InLR = *ON
C                   Return
C                   Endif
C                   Enddo
C                   Endif

 ** No Item record found or F12 pressed - display prompt
C                   Eval      *In12 = *OFF
C                   Exfmt     Prompt
C                   Enddo

C                   Eval      *InLR = *ON
C                   Return

 *************************************************************
C     *Inzsr        Begsr
C                   Eval      PgmNam = 'ITEMINQ'
C                   Exfmt     Prompt
C                   Endsr
```

Exercise 11. Changing native to SQL I/O

7. CREATE UNIQUE INDEX employeei ON EMPLOYEE (EMPNO ASC)

```
 *  Display file BONUSDSPF is declared.
FBonusDSPF CF   E             WorkStn InDDS(Indicators)
F* Comment out the F spec; not using file I/O
F*mployeeI UF   E           k Disk

D Indicators      DS
D   Exit                         1N   Overlay(Indicators:03)
D   Cancel                       1N   Overlay(Indicators:12)
D   Error                        1N   Overlay(Indicators:90)

C*  Format BONUSFMT from display file BONUSDSPF is written and read.
C*  The user will key values into ADDLBONUS and EMPNO.

C                   Exfmt     BonusFmt
C                   Dow       Not Exit

C*    EmpNo         Chain     EmployeeI
C* Replace the CHAIN with a SELECT.
C/EXEC SQL
C+ select Bonus
C+        INTO :bonus
C+        FROM Employee
C+        WHERE empno=:empno
C/END-EXEC
C* Change error handling to SQL
C*                  Eval      Error = Not %Found(EmployeeI)
C*                  Dow       %Found(EmployeeI)

C                   Eval      Error = (SQLCod <> 0)

C                   Dow       SqlCod = 0
C* For valid employee, calculate new bonus, update record
C* and display result on workstation

C                   Eval      Bonus = Bonus + AddlBonus
C*                  Update    Employee
C/EXEC SQL
C+ UPDATE EMPLOYEE
C+        SET BONUS = :BONUS
C+        WHERE empno = :empno
C/END-EXEC
C                   Exfmt     BonusDet

C                   Select
C                   When      Exit
C                   Eval      *InLR = *ON
C                   Return

C                   When      Cancel
C                   Leave
```

```
C                   EndSl

C                   EndDo

C                   Exfmt     BonusFmt
C                   EndDo

C                   Eval      *InLR = *ON
C                   Return
```

Exercise 12. Using a cursor

```
4. CREATE INDEX employeed ON EMPLOYEE (WORKDEPT ASC)
```

```
** Compile options:
 **   DATFMT(*ISO) DATSEP(-) COMMIT(*NONE) CLOSQLCSR(*ENDMOD)

 ** Display file BONUSDSPFD is declared.
FBonusDSPFDCF   E             WorkStn Sfile(Data:Rrn)
F
 ** No longer require the LF. Can process using SELECT
F*mployeeD UF   E           k Disk

D Rrn             S              4  0 Inz(1)
D ErrorMsg        S             50A   Inz('An error has occurred +
D                                         - SQLCODE = ') Varying

 ** Format BONUSFMT from display file BONUSDSPFD is written and read.
 ** The user will key values into ADDLBONUS and WORKDEPT

C                   Exfmt     BonusFmt
C                   Dow       Not *in03

 ** Replace the CHAIN with SELECT; need to declare cursor
 ** and open it also.
C*    WorkDept      Chain     EmployeeD
C*                  Eval      *in90 = Not %Found(EmployeeD)

 ** Monitor for Errors
C/Exec Sql
C+ WHENEVER SQLERROR GOTO ERRORLBL
C/End-Exec

 ** Declare Cursor
C/Exec Sql
C+ DECLARE DEPTCSR CURSOR FOR
C+   SELECT workdept,empno,bonus
C+     FROM Employee
C+     Where workdept=:workdept
C/End-Exec

 ** Open Cursor
C/Exec Sql
C+ OPEN DeptCSR
C/End-Exec

 ** Fetch first Row
C/Exec Sql
C+ FETCH NEXT FROM DeptCSR
C+   INTO :workdept, :empno, :bonus
C/End-Exec

 ** For valid Workdept, fill subfile.
```

```
C*                  Dow       %Found(EmployeeD)

 ** Load entire Subfile
C                   Dow       SqlCod <> 100

 ** For each employee, calculate new bonus, update record
C                   Eval      Bonus = Bonus + AddlBonus
 ** Rather than update individual rows, we can use a single SQL
 ** UPDATE statement to update all employees in a department
C*                  Update    Employee
C                   Write     Data

 ** Use the FETCH statement to process rows and load the subfile
C*    WorkDept      ReadE     EmployeeD
C*                  If        %Eof(EmployeeD)

 ** Fetch next Row
C/Exec Sql
C+ FETCH NEXT FROM DeptCSR
C+   INTO :workdept, :empno, :bonus
C/End-Exec
** replace with a test for SQLCOD
C                   Eval      Rrn = Rrn + 1
C                   Eval      *in40 = (SQLCOD = 100)
C                   If        (SQLCOD = 100)
C                   Leave
C                   Else
C                   EndIf
C                   Enddo

 ** Display subfile if file not empty
C                   Write     FncKey
C                   Select

C                   When      Rrn > 1
C                   Eval      *In85 = *ON
C                   Eval      *In95 = *ON

C                   Dow       NOT *In03
C                   Exfmt     Control
 ** Update table with new bonus
C/Exec SQL
C+  Update Employee set bonus = (bonus + :addlbonus)
C+         where workdept = :workdept
C/End-exec
C                   Enddo

C                   When      Rrn <= 1
C                   Eval      *In85 = *ON
C                   Eval      *In95 = *OFF
C                   Write     Msg
C                   Exfmt     Control

C                   Endsl

C                   If        *in03
```

```
 ** Close the cursor and return

 ** Close Cursor
C/Exec Sql
C+ CLOSE DeptCSR
C/End-Exec
C                   Eval      *InLR = *ON
C                   Return

C                   EndIf

C                   EndDo


C     ErrorLbl      Tag
C                   Eval      *in90 = *on
C                   Eval      ErrorMsg = ErrorMsg +
C                                 %TrimL(%EditC(SqlCod:'1'))
C     ErrorMsg      Dsply     '*REQUESTER'

 ** Close the cursor and return
 ** Close Cursor
C/Exec Sql
C+ CLOSE DeptCSR
C/End-Exec
C                   Eval      *InLR = *ON
C                   Return
```

Exercise 13. Dynamic embedded SQL

```
H DftActGrp(*No) ActGrp(*Caller)

FDynFSDSPF CF   E             Workstn Sfile(Record:Rrn)
F                                     IndDS(WkStnIndics)

D WkStnIndics     DS
D  Exit                   3      3N
D  SflEnd                90     90N
D  SflDspCtl             85     85N
D  SflDsp                95     95N
D  SflClr                75     75N
D  Protect               30     30N

D Rrn             S              4  0 INZ
D EmptySfl        S               N
D Sequence        S             20A

D SqlString       S            100A

C/Exec Sql
C+ set option commit=*none
C/End-Exec

 /FREE
  DoW not Exit;
    ExSR Search;
  EndDo;

  ExSR CloseCursor;
  *InLr = *on;

   // subroutines
  BegSR *InzSR;     // Initialization subroutine
    Write FNKeys;
    SflDspCtl = *On;
    Exfmt Control;
    ExSR Prepare;
    ExSR Declare;
    Protect = *on;
  Endsr;

  BegSR Search;
    ExSr SFLClear;         // Clear subfile for new search

    ExSR OpenCursor;       // Position file cursor
                           // using search key
    ExSR FetchRow;         // Fetch first row


   Rrn = 1;               // Rrn set to 1 even if we are at EOF
   Exsr Fill;             // Fill Subfile
   Exsr Prompt;           // Prompt for new department
 EndSR;
```

```
 Begsr Fill;

  // Load entire subfile
  DoW  SqlCod <> 100 AND (Rrn <= 9999);
     Write Record;
     ExSR FetchRow;       // Process remaining rows
     Rrn = Rrn + 1;
   Enddo;

   EmptySfl = (Rrn <= 1); // No records to display?
   SflEnd = (SqlCod = 100); // Also should consider whether or not Sfl has
                            // been filled. Future enhancement.
  Endsr;

  Begsr Prompt;

    If NOT EmptySfl;

      SflDspCtl = *ON; // Display Subfile
      SflDsp = *ON;

    Else;

      SflDspCtl = *ON; // No records to view - display message
      SflDsp = *OFF;
      Write Msg;
    Endif;

    Exfmt Control;
  Endsr;

  Begsr SFLCLear;   // Subfile clear subroutine
    SflClr = *on;
    SflDsp = *OFF;
    SflDspCtl = *OFF;
    Write Control;  // New search - clear subfile
    SflClr = *off;
    ExSR CloseCursor;
  EndSR;
 /END-FREE
  // Build Select String Variable and Prepare it once
C     Prepare       BegSR
C                   Select
  // Process sequence desired. Default is Empno
C                   When      NameOrder <> ' '
C                   Eval      Sequence = 'Lastname, Empno'
C                   Other
C                   Eval      Sequence = 'Empno'
C                   EndSl
  // Build SQL Select statement
C                   Eval      SqlString =
C                             'SELECT EMPNO, LASTNAME, WORKDEPT ' +
C                             'FROM EMPLOYEE ' +
C                             'WHERE WORKDEPT = ? ' +
C                             'ORDER BY ' + %TrimR(Sequence)
```

```
C/EXEC SQL
C+ PREPARE SQLSTMT FROM :SQLSTRING
C/END-EXEC
C                   EndSR
  // Declare Cursor
C     Declare       BegSR
C/EXEC SQL
C+ DECLARE SQLCSR CURSOR FOR SQLSTMT
C/END-EXEC
C                   EndSR
  // Open Cursor
C     OpenCursor    BegSR
C/EXEC SQL
C+ OPEN SQLCSR USING :WORKDEPT
C/END-EXEC
C                   EndSR
  // Process a row
C     FetchRow      BegSR
C/EXEC SQL
C+ FETCH SQLCSR INTO :EMPNO, :LASTNAME, :WORKDEPT
C/END-EXEC
C                   EndSR
  // Close cursor
C     CloseCursor   BegSR
C/EXEC SQL
C+ CLOSE SQLCSR
C/END-EXEC
C                   EndSR
```

Exercise 14. Create a stored procedure

4. Commitment Control is not implemented in this application. If you want to be able to debug your program, you must specify DBGVIEW *SOURCE.

6. In order to be able to see the result set, both the SP and the RPG IV program must run in the same activation group. The reason is that the cursor would be closed by the system when the activation group holding the RPG IV program ends. If the SP is in a different activation group, it would not see the cursor.

```
create procedure ol38nncol/apresult
           result set 1
           language rpgle
           external name ol38nncol/apinvrset;

   call ol38nncol/apresult;
```