

---

# **Appendix A**

## **Practices and Solutions**

---

# Table of Contents

Practices for Lesson I.....	3
Practice I-1: Introduction .....	4
Practice Solutions I-1: Introduction .....	5
Practices for Lesson 1 .....	11
Practice 1-1: Retrieving Data Using the SQL SELECT Statement .....	12
Practice Solutions 1-1: Retrieving Data Using the SQL SELECT Statement .....	16
Practices for Lesson 2 .....	19
Practice 2-1: Restricting and Sorting Data.....	20
Practice Solutions 2-1: Restricting and Sorting Data .....	24
Practices for Lesson 3 .....	27
Practice 3-1: Using Single-Row Functions to Customize Output .....	28
Practice Solutions 3-1: Using Single-Row Functions to Customize Output .....	32
Practices for Lesson 4 .....	35
Practice 4-1: Using Conversion Functions and Conditional Expressions .....	36
Practice Solutions 4-1: Using Conversion Functions and Conditional Expressions ....	39
Practices for Lesson 5 .....	41
Practice 5-1: Reporting Aggregated Data Using the Group Functions.....	42
Practice Solutions 5-1: Reporting Aggregated Data Using the Group Functions .....	45
Practices for Lesson 6 .....	48
Practice 6-1: Displaying Data from Multiple Tables Using Joins .....	49
Practice Solutions 6-1: Displaying Data from Multiple Tables Using Joins .....	52
Practices for Lesson 7 .....	54
Practice 7-1: Using Subqueries to Solve Queries .....	55
Practice Solutions 7-1: Using Subqueries to Solve Queries .....	57
Practices for Lesson 8 .....	59
Practice 8-1: Using the Set Operators.....	60
Practice Solutions 8-1: Using the Set Operators.....	62
Practices for Lesson 9 .....	64
Practice 9-1: Manipulating Data .....	65
Practice Solutions 9-1: Manipulating Data .....	69
Practices for Lesson 10 .....	73
Practice 10-1: Using DDL Statements to Create and Manage Tables .....	74
Practice Solutions 10-1: Using DDL Statements to Create and Manage Tables .....	76
Practices for Lesson 11 .....	79
Practice 11-1: Creating Other Schema Objects .....	80
Practice Solutions 11-1: Creating Other Schema Objects .....	82
Practices for Appendix F .....	84
Practice F-1: Oracle Join Syntax.....	85
Practice Solutions F-1: Oracle Join Syntax .....	88

## Practices for Lesson I

In this practice, you perform the following:

- Start Oracle SQL Developer and create a new connection to the `ora1` account.
- Use Oracle SQL Developer to examine data objects in the `ora1` account. The `ora1` account contains the HR schema tables.

Note the following location for the lab files:

`\home\oracle\labs\sql1\labs`

If you are asked to save any lab files, save them in this location.

In any practice, there may be exercises that are prefaced with the phrases “If you have time” or “If you want an extra challenge.” Work on these exercises only if you have completed all other exercises within the allocated time and would like a further challenge to your skills.

Perform the practices slowly and precisely. You can experiment with saving and running command files. If you have any questions at any time, ask your instructor.

### Note

- 1) All written practices use Oracle SQL Developer as the development environment. Although it is recommended that you use Oracle SQL Developer, you can also use SQL\*Plus that is available in this course.
- 2) For any query, the sequence of rows retrieved from the database may differ from the screenshots shown.

## ***Practice I-1: Introduction***

This is the first of many practices in this course. The solutions (if you require them) can be found at the end of this practice. Practices are intended to cover most of the topics that are presented in the corresponding lesson.

### **Starting Oracle SQL Developer**

- 1) Start Oracle SQL Developer using the SQL Developer desktop icon.

### **Creating a New Oracle SQL Developer Database Connection**

- 2) To create a new database connection, in the Connections Navigator, right-click Connections. Select New Connection from the menu. The New/Select Database Connection dialog box appears.
- 3) Create a database connection using the following information:
  - a) Connection Name: myconnection
  - b) Username: ora1
  - c) Password: ora1
  - d) Hostname: localhost
  - e) Port: 1521
  - f) SID: ORCL

Ensure that you select the Save Password check box.

### **Testing and Connecting Using the Oracle SQL Developer Database Connection**

- 4) Test the new connection.
- 5) If the status is Success, connect to the database using this new connection.

### **Browsing the Tables in the Connections Navigator**

- 6) In the Connections Navigator, view the objects available to you in the Tables node. Verify that the following tables are present:

COUNTRIES  
DEPARTMENTS  
EMPLOYEES  
JOB\_GRADES  
JOB\_HISTORY  
JOBS  
LOCATIONS  
REGIONS

- 7) Browse the structure of the EMPLOYEES table.
- 8) View the data of the DEPARTMENTS table.

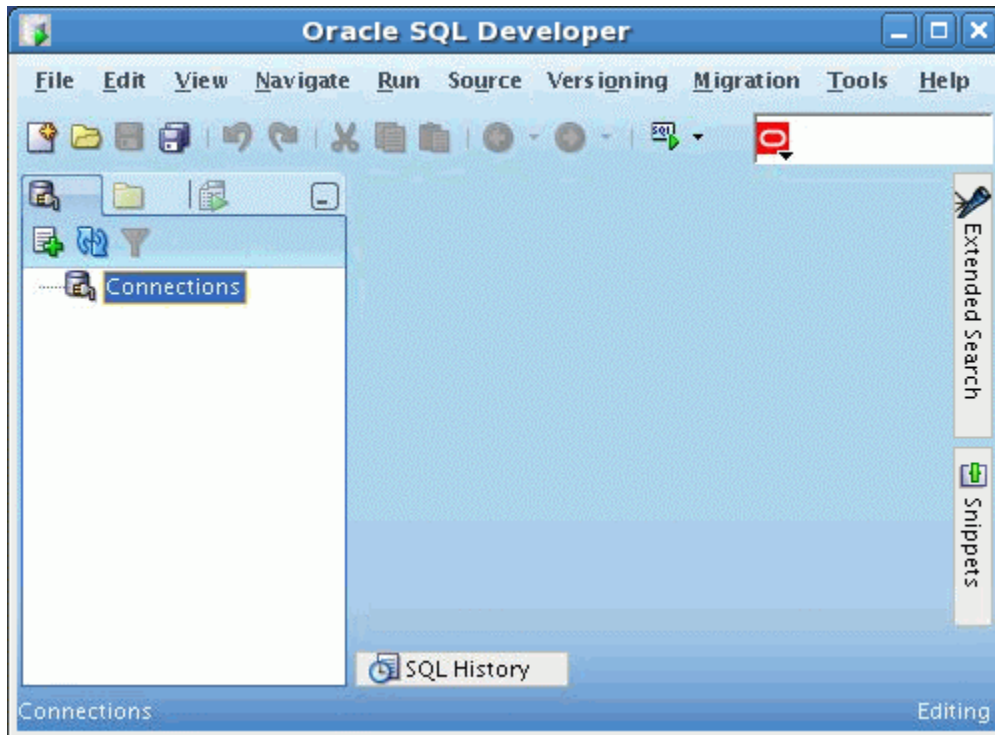
## ***Practice Solutions I-1: Introduction***

### **Starting Oracle SQL Developer**

- 1) Start Oracle SQL Developer using the SQL Developer desktop icon.
  - a) Double-click the SQL Developer desktop icon.

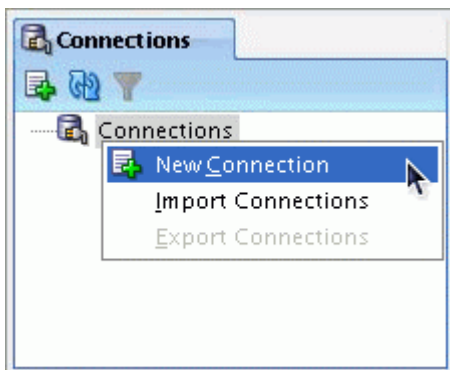


The SQL Developer Interface appears.



### **Creating a New Oracle SQL Developer Database Connection**

- 2) To create a new database connection, in the Connections Navigator, right-click Connections and select New Connection from the menu.



## Practice Solutions I-1: Introduction (continued)

The New / Select Database Connection dialog box appears.

The screenshot shows the 'New / Select Database Connection' dialog box. The 'Connection Name' field is empty. The 'Username' and 'Password' fields are empty. The 'Save Password' checkbox is unchecked. The 'Oracle' tab is selected. The 'Role' dropdown is set to 'default'. The 'Connection Type' dropdown is set to 'Basic'. The 'OS Authentication', 'Kerberos Authentication', and 'Proxy Connection' checkboxes are unchecked. The 'Hostname' field is 'localhost'. The 'Port' field is '1521'. The 'SID' radio button is selected, and the 'Service name' radio button is unselected. The 'Status:' label is at the bottom left. The buttons at the bottom are 'Help', 'Save', 'Clear', 'Test', 'Connect', and 'Cancel'.

3) Create a database connection using the following information:

- a) Connection Name: myconnection
- b) Username: ora1
- c) Password: ora1
- d) Hostname: localhost
- e) Port: 1521
- f) SID: ORCL

Ensure that you select the Save Password check box.

## Practice Solutions I-1: Introduction (continued)

**New / Select Database Connection**

Connection Name: myconnection

Username: ora1

Password: \*\*\*\*

☒ Save Password

Oracle

Role: default

Connection Type: Basic

☐ OS Authentication

☐ Kerberos Authentication

☐ Proxy Connection

Hostname: localhost

Port: 1521

☒ SID: orcl

☐ Service name:

Status:

Help Save Clear Test Connect Cancel

### Testing and Connecting Using the Oracle SQL Developer Database Connection

4) Test the new connection.

**New / Select Database Connection**

Connection Name: myconnection

Username: ora1

Password: \*\*\*\*

☒ Save Password

Oracle

Role: default

Connection Type: Basic

☐ OS Authentication

☐ Kerberos Authentication

☐ Proxy Connection

Hostname: localhost

Port: 1521

☒ SID: orcl

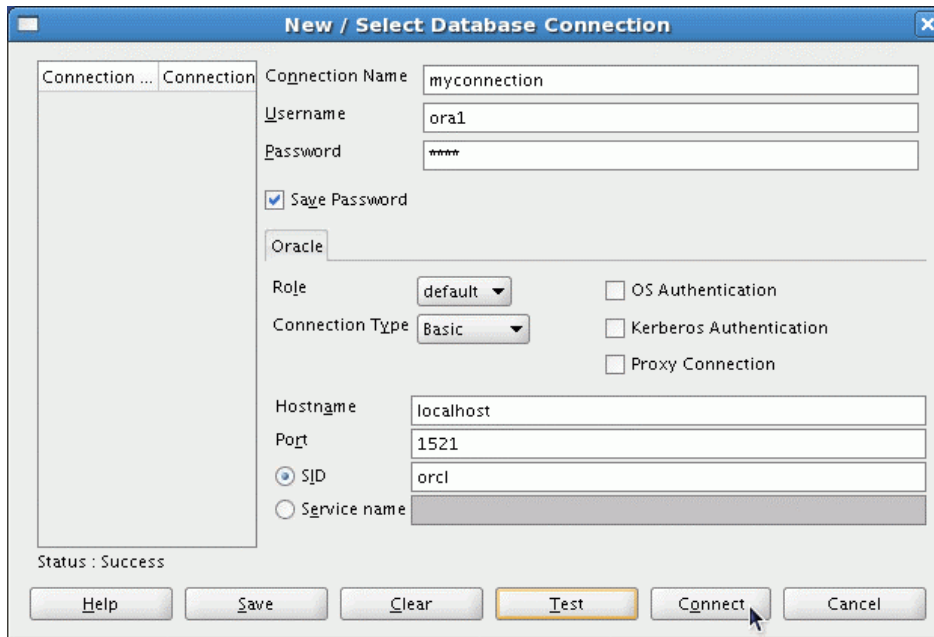
☐ Service name:

Status: Success

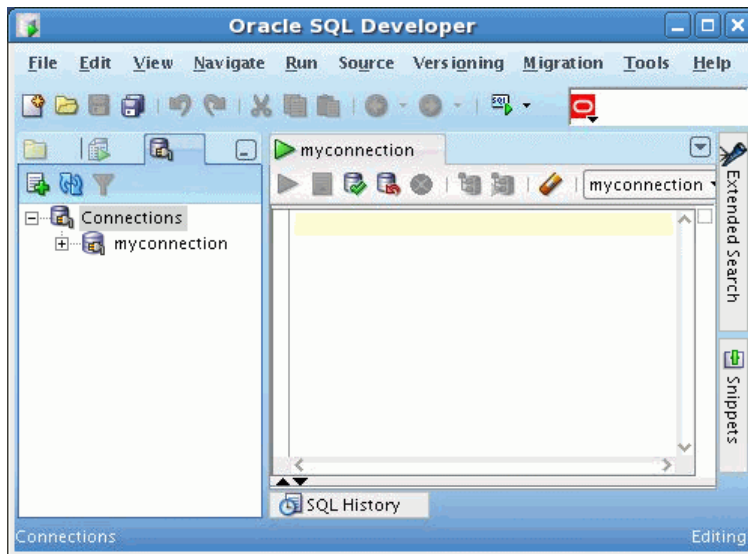
Help Save Clear Test Connect Cancel

5) If the status is Success, connect to the database using this new connection.

## Practice Solutions I-1: Introduction (continued)



When you create a connection, a SQL Worksheet for that connection opens automatically.



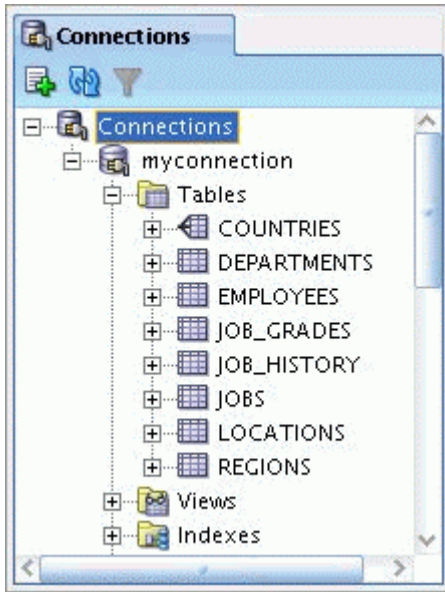
### Browsing the Tables in the Connections Navigator

- 6) In the Connections Navigator, view the objects available to you in the Tables node. Verify that the following tables are present:

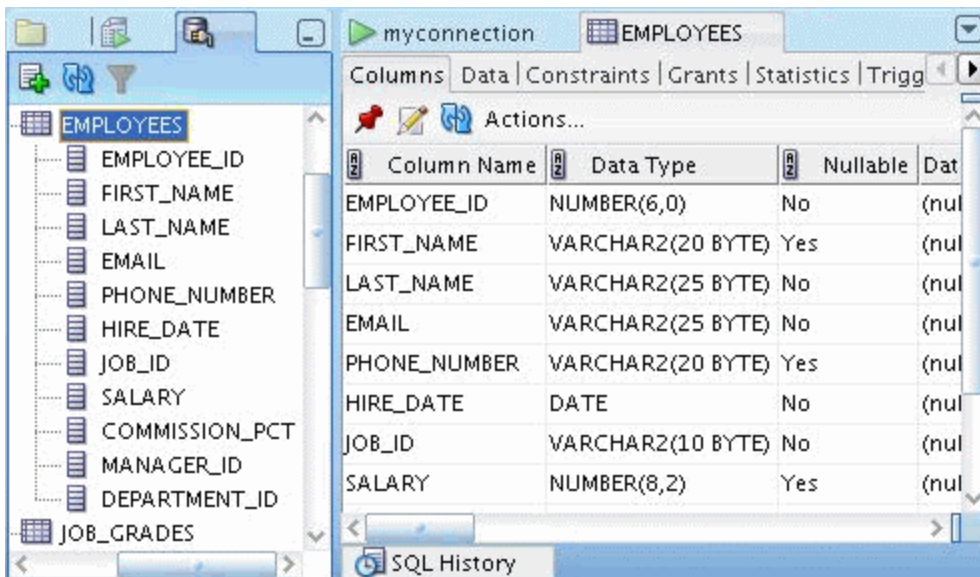
COUNTRIES  
DEPARTMENTS  
EMPLOYEES  
JOB\_GRADES  
JOB\_HISTORY  
JOBS  
LOCATIONS  
REGIONS



## Practice Solutions I-1: Introduction (continued)

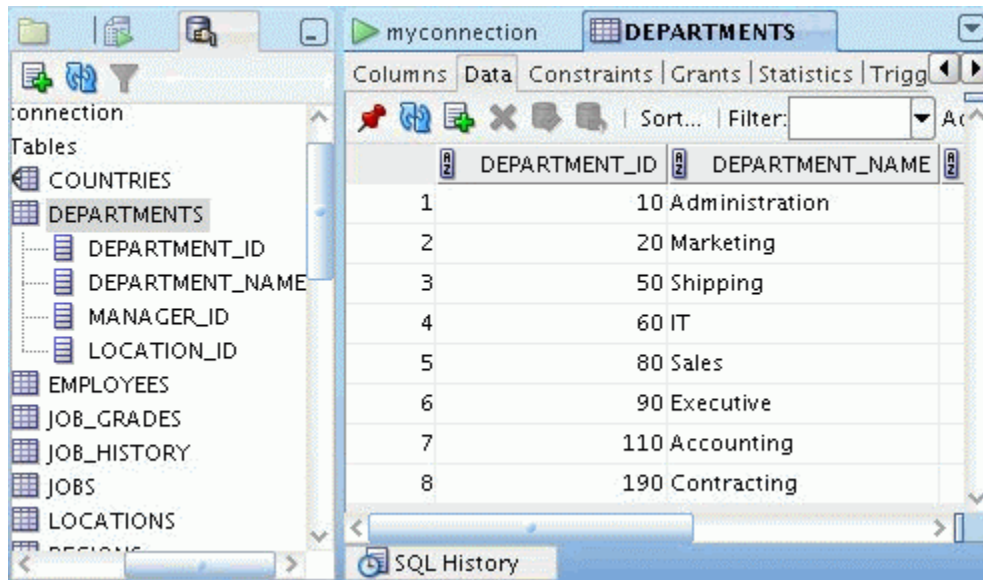


7) Browse the structure of the EMPLOYEES table.



8) View the data of the DEPARTMENTS table.

## Practice Solutions I-1: Introduction (continued)



The screenshot shows the Oracle SQL Developer interface. On the left, the 'Tables' pane lists database objects: COUNTRIES, DEPARTMENTS, EMPLOYEES, JOB\_GRADES, JOB\_HISTORY, JOBS, LOCATIONS, and REGIONS. The 'DEPARTMENTS' table is selected. The main pane displays the 'Data' tab for the 'DEPARTMENTS' table, showing a list of 8 rows. The columns are DEPARTMENT\_ID and DEPARTMENT\_NAME. The data is as follows:

DEPARTMENT_ID	DEPARTMENT_NAME
1	10 Administration
2	20 Marketing
3	50 Shipping
4	60 IT
5	80 Sales
6	90 Executive
7	110 Accounting
8	190 Contracting

At the bottom of the interface, there is an 'SQL History' pane.

## Practices for Lesson 1

In this practice, you write simple `SELECT` queries. The queries cover most of the `SELECT` clauses and operations that you learned in this lesson.

## ***Practice 1-1: Retrieving Data Using the SQL `SELECT` Statement***

### **Part 1**

Test your knowledge:

- 1) The following `SELECT` statement executes successfully:

```
SELECT last_name, job_id, salary AS Sal
FROM   employees;
```

True/False

- 2) The following `SELECT` statement executes successfully:

```
SELECT *
FROM   job_grades;
```

True/False

- 3) There are four coding errors in the following statement. Can you identify them?

```
SELECT      employee_id, last_name
sal x 12    ANNUAL SALARY
FROM        employees;
```

### **Part 2**

Note the following points before you begin with the practices:

- Save all your lab files at the following location:  
/home/oracle/labs/sql1/labs
- Enter your SQL statements in a SQL Worksheet. To save a script in SQL Developer, make sure that the required SQL worksheet is active and then from the File menu, select Save As to save your SQL statement as a lab\_<lessonno>\_<stepno>.sql script. When you are modifying an existing script, make sure that you use Save As to save it with a different file name.
- To run the query, click the Execute Statement icon in the SQL Worksheet. Alternatively, you can press [F9]. For DML and DDL statements, use the Run Script icon or press [F5].
- After you have executed the query, make sure that you do not enter your next query in the same worksheet. Open a new worksheet.

You have been hired as a SQL programmer for Acme Corporation. Your first task is to create some reports based on data from the Human Resources tables.

## Practice 1-1: Retrieving Data Using the SQL SELECT Statement (continued)

- 4) Your first task is to determine the structure of the DEPARTMENTS table and its contents.

Name	Null	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

4 rows selected

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

- 5) Determine the structure of the EMPLOYEES table.

Name	Null	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

11 rows selected

The HR department wants a query to display the last name, job ID, hire date, and employee ID for each employee, with the employee ID appearing first. Provide an alias STARTDATE for the HIRE\_DATE column. Save your SQL statement to a file named lab\_01\_05.sql so that you can dispatch this file to the HR department.

- 6) Test your query in the lab\_01\_05.sql file to ensure that it runs correctly.

**Note:** After you have executed the query, make sure that you do not enter your next query in the same worksheet. Open a new worksheet.

## Practice 1-1: Retrieving Data Using the SQL SELECT Statement (continued)

	EMPLOYEE_ID	LAST_NAME	JOB_ID	STARTDATE
1	200	Whalen	AD_ASST	17-SEP-87
2	201	Hartstein	MK_MAN	17-FEB-96
3	202	Fay	MK_REP	17-AUG-97
4	205	Higgins	AC_MGR	07-JUN-94
5	206	Gietz	AC_ACCOUNT	07-JUN-94

...

19	176 Taylor	SA_REP	24-MAR-98
20	178 Grant	SA_REP	24-MAY-99

- 7) The HR department wants a query to display all unique job IDs from the EMPLOYEES table.

	JOB_ID
1	AC_ACCOUNT
2	AC_MGR
3	AD_ASST
4	AD_PREP
5	AD_VP
6	IT_PROG
7	MK_MAN
8	MK_REP
9	SA_MAN
10	SA_REP
11	ST_CLERK
12	ST_MAN

### Part 3

If you have time, complete the following exercises:

- 8) The HR department wants more descriptive column headings for its report on employees. Copy the statement from lab\_01\_05.sql to a new SQL Worksheet. Name the column headings Emp #, Employee, Job, and Hire Date, respectively. Then run the query again.

	Emp #	Employee	Job	Hire Date
1	200	Whalen	AD_ASST	17-SEP-87
2	201	Hartstein	MK_MAN	17-FEB-96
3	202	Fay	MK_REP	17-AUG-97
4	205	Higgins	AC_MGR	07-JUN-94
5	206	Gietz	AC_ACCOUNT	07-JUN-94

...

## Practice 1-1: Retrieving Data Using the SQL SELECT Statement (continued)

19	176 Taylor	SA_REP	24-MAR-98
20	178 Grant	SA_REP	24-MAY-99

- 9) The HR department has requested a report of all employees and their job IDs. Display the last name concatenated with the job ID (separated by a comma and space) and name the column Employee and Title.

	Employee and Title
1	Abel, SA_REP
2	Davies, ST_CLERK
3	De Haan, AD_VP
4	Ernst, IT_PROG
5	Fay, MK_REP

...

19	Whalen, AD_ASST
20	Zlotkey, SA_MAN

If you want an extra challenge, complete the following exercise:

- 10) To familiarize yourself with the data in the EMPLOYEES table, create a query to display all the data from that table. Separate each column output by a comma. Name the column title THE\_OUTPUT.

	THE_OUTPUT
1	200,Jennifer,Whalen,JWHALEN,515.123.4444,AD_ASST,101,17-SEP-87,4400,,10
2	201,Michael,Hartstein,MHARTSTE,515.123.5555,MK_MAN,100,17-FEB-96,13000,,20
3	202,Pat,Fay,PFAY,603.123.6666,MK_REP,201,17-AUG-97,6000,,20
4	205,Shelley,Higgins,SHIGGINS,515.123.8080,AC_MGR,101,07-JUN-94,12000,,110
5	206,William,Gietz,WGIETZ,515.123.8181,AC_ACCOUNT,205,07-JUN-94,8300,,110

...

19	176,Jonathon,Taylor,JTAYLOR,011.44.1644.429265,SA_REP,149,24-MAR-98,8600,,2,80
20	178,Kimberely,Grant,KGRANT,011.44.1644.429263,SA_REP,149,24-MAY-99,7000,,15,

## ***Practice Solutions 1-1: Retrieving Data Using the SQL SELECT Statement***

### **Part 1**

Test your knowledge:

- 1) The following SELECT statement executes successfully:

```
SELECT last_name, job_id, salary AS Sal
FROM   employees;
```

**True/False**

- 2) The following SELECT statement executes successfully:

```
SELECT *
FROM   job_grades;
```

**True/False**

- 3) There are four coding errors in the following statement. Can you identify them?

```
SELECT      employee_id, last_name
sal x 12    ANNUAL SALARY
FROM        employees;
```

- **The EMPLOYEES table does not contain a column called sal. The column is called SALARY.**
- **The multiplication operator is \*, not x, as shown in line 2.**
- **The ANNUAL SALARY alias cannot include spaces. The alias should read ANNUAL\_SALARY or should be enclosed within double quotation marks.**
- **A comma is missing after the LAST\_NAME column.**

### **Part 2**

You have been hired as a SQL programmer for Acme Corporation. Your first task is to create some reports based on data from the Human Resources tables.

- 4) Your first task is to determine the structure of the DEPARTMENTS table and its contents.

- a. To determine the DEPARTMENTS table structure:

```
DESCRIBE departments
```



## Practice Solutions 1-1: Retrieving Data Using the SQL SELECT Statement (continued)

- b. To view the data contained in the DEPARTMENTS table:

```
SELECT *  
FROM departments;
```

- 5) Determine the structure of the EMPLOYEES table.

```
DESCRIBE employees
```

The HR department wants a query to display the last name, job ID, hire date, and employee ID for each employee, with the employee ID appearing first. Provide an alias STARTDATE for the HIRE\_DATE column. Save your SQL statement to a file named lab\_01\_05.sql so that you can dispatch this file to the HR department.

```
SELECT employee_id, last_name, job_id, hire_date StartDate  
FROM employees;
```

- 6) Test your query in the lab\_01\_05.sql file to ensure that it runs correctly.

```
SELECT employee_id, last_name, job_id, hire_date StartDate  
FROM employees;
```

- 7) The HR department wants a query to display all unique job IDs from the EMPLOYEES table.

```
SELECT DISTINCT job_id  
FROM employees;
```

### Part 3

If you have time, complete the following exercises:

- 8) The HR department wants more descriptive column headings for its report on employees. Copy the statement from lab\_01\_05.sql to a new SQL Worksheet. Name the column headings Emp #, Employee, Job, and Hire Date, respectively. Then run the query again.

```
SELECT employee_id "Emp #", last_name "Employee",  
       job_id "Job", hire_date "Hire Date"  
FROM employees;
```

- 9) The HR department has requested a report of all employees and their job IDs. Display the last name concatenated with the job ID (separated by a comma and space) and name the column Employee and Title.

```
SELECT last_name||', '||job_id "Employee and Title"  
FROM employees;
```

## ***Practice Solutions 1-1: Retrieving Data Using the SQL SELECT Statement (continued)***

If you want an extra challenge, complete the following exercise:

- 10) To familiarize yourself with the data in the EMPLOYEES table, create a query to display all the data from that table. Separate each column output by a comma. Name the column title THE\_OUTPUT.

```
SELECT employee_id || ',' || first_name || ',' || last_name
       || ',' || email || ',' || phone_number || ',' || job_id
       || ',' || manager_id || ',' || hire_date || ',' ||
       || salary || ',' || commission_pct || ',' ||
department_id
       THE_OUTPUT
FROM   employees;
```

---

## Practices for Lesson 2

---

In this practice, you build more reports, including statements that use the `WHERE` clause and the `ORDER BY` clause. You make the SQL statements more reusable and generic by including the ampersand substitution.

## Practice 2-1: Restricting and Sorting Data

The HR department needs your assistance in creating some queries.

- 1) Because of budget issues, the HR department needs a report that displays the last name and salary of employees who earn more than \$12,000. Save your SQL statement as a file named `lab_02_01.sql`. Run your query.

	A Z	LAST_NAME	A Z	SALARY
1		Hartstein		13000
2		King		24000
3		Kochhar		17000
4		De Haan		17000

- 2) Open a new SQL Worksheet. Create a report that displays the last name and department number for employee number 176. Run the query.

	A Z	LAST_NAME	A Z	DEPARTMENT_ID
1		Taylor		80

- 3) The HR department needs to find high-salary and low-salary employees. Modify `lab_02_01.sql` to display the last name and salary for any employee whose salary is not in the range of \$5,000 to \$12,000. Save your SQL statement as `lab_02_03.sql`.

	A Z	LAST_NAME	A Z	SALARY
1		Whalen		4400
2		Hartstein		13000
3		King		24000
4		Kochhar		17000
5		De Haan		17000
6		Lorentz		4200
7		Rajs		3500
8		Davies		3100
9		Matos		2600
10		Vargas		2500

- 4) Create a report to display the last name, job ID, and hire date for employees with the last names of Matos and Taylor. Order the query in ascending order by the hire date.

	A Z	LAST_NAME	A Z	JOB_ID	A Z	HIRE_DATE
1		Matos		ST_CLERK		15-MAR-98
2		Taylor		SA_REP		24-MAR-98

## Practice 2-1: Restricting and Sorting Data (continued)

- 5) Display the last name and department ID of all employees in departments 20 or 50 in ascending alphabetical order by name.

	A Z LAST_NAME	A Z DEPARTMENT_ID
1	Davies	50
2	Fay	20
3	Hartstein	20
4	Matos	50
5	Mourgos	50
6	Rajs	50
7	Vargas	50

- 6) Modify lab\_02\_03.sql to display the last name and salary of employees who earn between \$5,000 and \$12,000, and are in department 20 or 50. Label the columns Employee and Monthly Salary, respectively. Save lab\_02\_03.sql as lab\_02\_06.sql again. Run the statement in lab\_02\_06.sql.

	A Z Employee	A Z Monthly Salary
1	Fay	6000
2	Mourgos	5800

- 7) The HR department needs a report that displays the last name and hire date for all employees who were hired in 1994.

	A Z LAST_NAME	A Z HIRE_DATE
1	Higgins	07-JUN-94
2	Gietz	07-JUN-94

- 8) Create a report to display the last name and job title of all employees who do not have a manager.

	A Z LAST_NAME	A Z JOB_ID
1	King	AD_PRES

- 9) Create a report to display the last name, salary, and commission of all employees who earn commissions. Sort data in descending order of salary and commissions. Use the column's numeric position in the ORDER BY clause.

	A Z LAST_NAME	A Z SALARY	A Z COMMISSION_PCT
1	Abel	11000	0.3
2	Zlotkey	10500	0.2
3	Taylor	8600	0.2
4	Grant	7000	0.15

## Practice 2-1: Restricting and Sorting Data (continued)

- 10) Members of the HR department want to have more flexibility with the queries that you are writing. They would like a report that displays the last name and salary of employees who earn more than an amount that the user specifies after a prompt. Save this query to a file named `lab_02_10.sql`. If you enter 12000 when prompted, the report displays the following results:

	A Z	LAST_NAME	A Z	SALARY
1		Hartstein		13000
2		King		24000
3		Kochhar		17000
4		De Haan		17000

- 11) The HR department wants to run reports based on a manager. Create a query that prompts the user for a manager ID and generates the employee ID, last name, salary, and department for that manager's employees. The HR department wants the ability to sort the report on a selected column. You can test the data with the following values:

manager\_id = 103, sorted by last\_name:

	A Z	EMPLOYEE_ID	A Z	LAST_NAME	A Z	SALARY	A Z	DEPARTMENT_ID
1		104		Ernst		6000		60
2		107		Lorentz		4200		60

manager\_id = 201, sorted by salary:

	A Z	EMPLOYEE_ID	A Z	LAST_NAME	A Z	SALARY	A Z	DEPARTMENT_ID
1		202		Fay		6000		20

manager\_id = 124, sorted by employee\_id:

	A Z	EMPLOYEE_ID	A Z	LAST_NAME	A Z	SALARY	A Z	DEPARTMENT_ID
1		141		Rajs		3500		50
2		142		Davies		3100		50
3		143		Matos		2600		50
4		144		Vargas		2500		50

If you have time, complete the following exercises:

- 12) Display all employee last names in which the third letter of the name is "a."

	A Z	LAST_NAME
1		Grant
2		Whalen

## Practice 2-1: Restricting and Sorting Data (continued)

- 13) Display the last names of all employees who have both an “a” and an “e” in their last name.

	LAST_NAME
1	Davies
2	De Haan
3	Hartstein
4	Whalen

If you want an extra challenge, complete the following exercises:

- 14) Display the last name, job, and salary for all employees whose jobs are either those of a sales representative or of a stock clerk, and whose salaries are not equal to \$2,500, \$3,500, or \$7,000.

	LAST_NAME	JOB_ID	SALARY
1	Abel	SA_REP	11000
2	Taylor	SA_REP	8600
3	Davies	ST_CLERK	3100
4	Matos	ST_CLERK	2600

- 15) Modify lab\_02\_06.sql to display the last name, salary, and commission for all employees whose commission is 20%. Save lab\_02\_06.sql as lab\_02\_15.sql again. Rerun the statement in lab\_02\_15.sql.

	Employee	Monthly Salary	COMMISSION_PCT
1	Zlotkey	10500	0.2
2	Taylor	8600	0.2

## Practice Solutions 2-1: Restricting and Sorting Data

The HR department needs your assistance in creating some queries.

- 1) Because of budget issues, the HR department needs a report that displays the last name and salary of employees earning more than \$12,000. Save your SQL statement as a file named `lab_02_01.sql`. Run your query.

```
SELECT last_name, salary
FROM employees
WHERE salary > 12000;
```

- 2) Open a new SQL Worksheet. Create a report that displays the last name and department number for employee number 176.

```
SELECT last_name, department_id
FROM employees
WHERE employee_id = 176;
```

- 3) The HR department needs to find high-salary and low-salary employees. Modify `lab_02_01.sql` to display the last name and salary for all employees whose salary is not in the range \$5,000 through \$12,000. Save your SQL statement as `lab_02_03.sql`.

```
SELECT last_name, salary
FROM employees
WHERE salary NOT BETWEEN 5000 AND 12000;
```

- 4) Create a report to display the last name, job ID, and hire date for employees with the last names of Matos and Taylor. Order the query in ascending order by hire date.

```
SELECT last_name, job_id, hire_date
FROM employees
WHERE last_name IN ('Matos', 'Taylor')
ORDER BY hire_date;
```

- 5) Display the last name and department ID of all employees in departments 20 or 50 in ascending alphabetical order by name.

```
SELECT last_name, department_id
FROM employees
WHERE department_id IN (20, 50)
ORDER BY last_name ASC;
```

- 6) Modify `lab_02_03.sql` to list the last name and salary of employees who earn between \$5,000 and \$12,000, and are in department 20 or 50. Label the columns Employee and Monthly Salary, respectively. Save `lab_02_03.sql` as `lab_02_06.sql` again. Run the statement in `lab_02_06.sql`.

```
SELECT last_name "Employee", salary "Monthly Salary"
FROM employees
WHERE salary BETWEEN 5000 AND 12000
AND department_id IN (20, 50);
```



## Practice Solutions 2-1: Restricting and Sorting Data (continued)

- 7) The HR department needs a report that displays the last name and hire date for all employees who were hired in 1994.

```
SELECT    last_name, hire_date
FROM      employees
WHERE     hire_date LIKE '%94';
```

- 8) Create a report to display the last name and job title of all employees who do not have a manager.

```
SELECT    last_name, job_id
FROM      employees
WHERE     manager_id IS NULL;
```

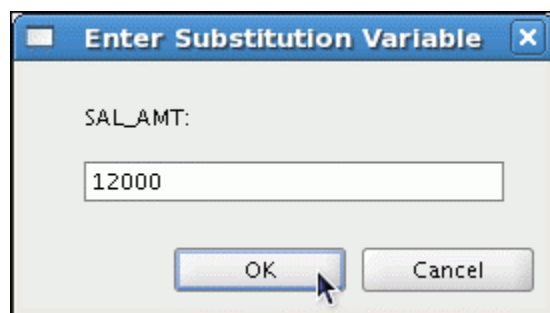
- 9) Create a report to display the last name, salary, and commission for all employees who earn commissions. Sort data in descending order of salary and commissions. Use the column's numeric position in the ORDER BY clause.

```
SELECT    last_name, salary, commission_pct
FROM      employees
WHERE     commission_pct IS NOT NULL
ORDER BY 2 DESC, 3 DESC;
```

- 10) Members of the HR department want to have more flexibility with the queries that you are writing. They would like a report that displays the last name and salary of employees who earn more than an amount that the user specifies after a prompt. (You can use the query created in practice exercise 1 and modify it.) Save this query to a file named lab\_02\_10.sql.

```
SELECT    last_name, salary
FROM      employees
WHERE     salary > &sal_amt;
```

Enter 12000 when prompted for a value in a dialog box. Click OK.



- 11) The HR department wants to run reports based on a manager. Create a query that prompts the user for a manager ID and generates the employee ID, last name, salary, and department for that manager's employees. The HR department wants the ability to sort the report on a selected column. You can test the data with the following values:

manager\_id = 103, sorted by last\_name  
manager\_id = 201, sorted by salary  
manager\_id = 124, sorted by employee\_id

## Practice Solutions 2-1: Restricting and Sorting Data (continued)

```
SELECT employee_id, last_name, salary, department_id
FROM employees
WHERE manager_id = &mgr_num
ORDER BY &order_col;
```

If you have the time, complete the following exercises:

- 12) Display all employee last names in which the third letter of the name is “a.”

```
SELECT    last_name
FROM      employees
WHERE     last_name LIKE '__a%';
```

- 13) Display the last names of all employees who have both an “a” and an “e” in their last name.

```
SELECT    last_name
FROM      employees
WHERE     last_name LIKE '%a%'
AND       last_name LIKE '%e%';
```

If you want an extra challenge, complete the following exercises:

- 14) Display the last name, job, and salary for all employees whose job is that of a sales representative or a stock clerk, and whose salary is not equal to \$2,500, \$3,500, or \$7,000.

```
SELECT    last_name, job_id, salary
FROM      employees
WHERE     job_id IN ('SA_REP', 'ST_CLERK')
AND       salary NOT IN (2500, 3500, 7000);
```

- 15) Modify lab\_02\_06.sql to display the last name, salary, and commission for all employees whose commission amount is 20%. Save lab\_02\_06.sql as lab\_02\_15.sql again. Rerun the statement in lab\_02\_15.sql.

```
SELECT    last_name "Employee", salary "Monthly Salary",
          commission_pct
FROM      employees
WHERE     commission_pct = .20;
```

---

## Practices for Lesson 3


---

This practice provides a variety of exercises using different functions that are available for character, number, and date data types.

### Practice 3-1: Using Single-Row Functions to Customize Output





- 1) Write a query to display the system date. Label the column `Date`.

**Note:** If your database is remotely located in a different time zone, the output will be the date for the operating system on which the database resides.

	 Date
1	10-JUN-09

- 2) The HR department needs a report to display the employee number, last name, salary, and salary increased by 15.5% (expressed as a whole number) for each employee. Label the column `New Salary`. Save your SQL statement in a file named `lab_03_02.sql`.






- 3) Run your query in the `lab_03_02.sql` file.

	 EMPLOYEE_ID	 LAST_NAME	 SALARY	 New Salary
1	200	Whalen	4400	5082
2	201	Hartstein	13000	15015
3	202	Fay	6000	6930
4	205	Higgins	12000	13860
5	206	Gietz	8300	9587

...

19	176	Taylor	8600	9933
20	178	Grant	7000	8085

- 4) Modify your query `lab_03_02.sql` to add a column that subtracts the old salary from the new salary. Label the column `Increase`. Save the contents of the file as `lab_03_04.sql`. Run the revised query.



	 EMPLOYEE_ID	 LAST_NAME	 SALARY	 New Salary	 Increase
1	200	Whalen	4400	5082	682
2	201	Hartstein	13000	15015	2015
3	202	Fay	6000	6930	930
4	205	Higgins	12000	13860	1860
5	206	Gietz	8300	9587	1287

...



19	176	Taylor	8600	9933	1333
20	178	Grant	7000	8085	1085

### Practice 3-1: Using Single-Row Functions to Customize Output (continued)

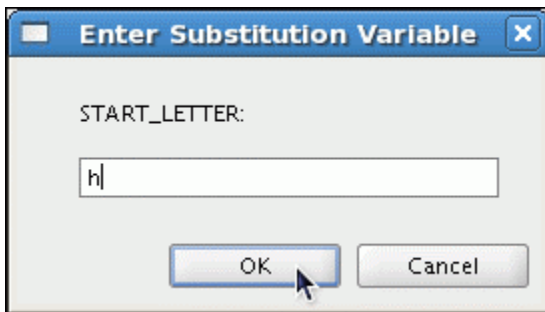
- 5) Write a query that displays the last name (with the first letter in uppercase and all the other letters in lowercase) and the length of the last name for all employees whose name starts with the letters “J,” “A,” or “M.” Give each column an appropriate label. Sort the results by the employees’ last names.

	 Name	 Length
1	Abel	4
2	Matos	5
3	Mourgos	7



Rewrite the query so that the user is prompted to enter a letter that the last name starts with. For example, if the user enters “H” (capitalized) when prompted for a letter, then the output should show all employees whose last name starts with the letter “H.”

	 Name	 Length
1	Hartstein	9
2	Higgins	7
3	Hunold	6

Modify the query such that the case of the entered letter does not affect the output. The entered letter must be capitalized before being processed by the `SELECT` query.



The dialog box titled "Enter Substitution Variable" has a close button (X) in the top right corner. It contains a label "START\_LETTER:" followed by a text input field. The input field contains the lowercase letter "h". At the bottom, there are two buttons: "OK" and "Cancel". A mouse cursor is pointing at the "OK" button.

	 Name	 Length
1	Hartstein	9
2	Higgins	7
3	Hunold	6

- 6) The HR department wants to find the duration of employment for each employee. For each employee, display the last name and calculate the number of months between today and the date on which the employee was hired. Label the column as `MONTHS_WORKED`. Order your results by the number of months employed. Round the number of months up to the closest whole number.

**Note:** Because this query depends on the date when it was executed, the values in the `MONTHS_WORKED` column will differ for you.

### Practice 3-1: Using Single-Row Functions to Customize Output (continued)

	LAST_NAME	MONTHS_WORKED
1	Zlotkey	112
2	Mourgos	115
3	Grant	121
4	Lorentz	124
5	Vargas	131

...

19	Whalen	261
20	King	264

If you have time, complete the following exercises:

- 7) Create a query to display the last name and salary for all employees. Format the salary to be 15 characters long, left-padded with the \$ symbol. Label the column SALARY.

	LAST_NAME	SALARY
1	Whalen	\$\$\$\$\$\$\$\$\$\$\$4400
2	Hartstein	\$\$\$\$\$\$\$\$\$\$\$13000
3	Fay	\$\$\$\$\$\$\$\$\$\$\$6000
4	Higgins	\$\$\$\$\$\$\$\$\$\$\$12000
5	Gietz	\$\$\$\$\$\$\$\$\$\$\$8300

...

19	Taylor	\$\$\$\$\$\$\$\$\$\$\$8600
20	Grant	\$\$\$\$\$\$\$\$\$\$\$7000

- 8) Create a query that displays the first eight characters of the employees' last names and indicates the amounts of their salaries with asterisks. Each asterisk signifies a thousand dollars. Sort the data in descending order of salary. Label the column EMPLOYEES\_AND\_THEIR\_SALARIES.

	EMPLOYEES_AND_THEIR_SALARIES
1	King *****
2	Kochhar *****
3	De Haan *****
4	Hartstei *****
5	Higgins *****

...

19	Matos ***
20	Vargas **

### ***Practice 3-1: Using Single-Row Functions to Customize Output (continued)***

- 9) Create a query to display the last name and the number of weeks employed for all employees in department 90. Label the number of weeks column `TENURE`. Truncate the number of weeks value to 0 decimal places. Show the records in descending order of the employee's tenure.

**Note:** The `TENURE` value will differ as it depends on the date on which you run the query.

	LAST_NAME	TENURE
1	King	1147
2	Kochhar	1028
3	De Haan	856

## Practice Solutions 3-1: Using Single-Row Functions to Customize Output

- 1) Write a query to display the system date. Label the column Date.

**Note:** If your database is remotely located in a different time zone, the output will be the date for the operating system on which the database resides.

```
SELECT sysdate "Date"
FROM dual;
```

- 2) The HR department needs a report to display the employee number, last name, salary, and salary increased by 15.5% (expressed as a whole number) for each employee. Label the column New Salary. Save your SQL statement in a file named lab\_03\_02.sql.

```
SELECT employee_id, last_name, salary,
       ROUND(salary * 1.155, 0) "New Salary"
FROM employees;
```

- 3) Run your query in the file lab\_03\_02.sql.

```
SELECT employee_id, last_name, salary,
       ROUND(salary * 1.155, 0) "New Salary"
FROM employees;
```

- 4) Modify your query lab\_03\_02.sql to add a column that subtracts the old salary from the new salary. Label the column Increase. Save the contents of the file as lab\_03\_04.sql. Run the revised query.

```
SELECT employee_id, last_name, salary,
       ROUND(salary * 1.155, 0) "New Salary",
       ROUND(salary * 1.155, 0) - salary "Increase"
FROM employees;
```

- 5) Write a query that displays the last name (with the first letter in uppercase and all the other letters in lowercase) and the length of the last name for all employees whose name starts with the letters "J," "A," or "M." Give each column an appropriate label. Sort the results by the employees' last names.

```
SELECT INITCAP(last_name) "Name",
       LENGTH(last_name) "Length"
FROM employees
WHERE last_name LIKE 'J%'
OR last_name LIKE 'M%'
OR last_name LIKE 'A%'
ORDER BY last_name ;
```

Rewrite the query so that the user is prompted to enter a letter that starts the last name. For example, if the user enters H (capitalized) when prompted for a letter, then the output should show all employees whose last name starts with the letter "H."



## Practice Solutions 3-1: Using Single-Row Functions to Customize Output (continued)

```
SELECT  INITCAP(last_name) "Name",
        LENGTH(last_name) "Length"
FROM    employees
WHERE   last_name LIKE '&start_letter%'
ORDER BY last_name;
```

Modify the query such that the case of the entered letter does not affect the output. The entered letter must be capitalized before being processed by the SELECT query.

```
SELECT  INITCAP(last_name) "Name",
        LENGTH(last_name) "Length"
FROM    employees
WHERE   last_name LIKE UPPER('&start_letter%' )
ORDER BY last_name;
```

- 6) The HR department wants to find the duration of employment for each employee. For each employee, display the last name and calculate the number of months between today and the date on which the employee was hired. Label the column MONTHS\_WORKED. Order your results by the number of months employed. Round the number of months up to the closest whole number.

**Note:** Because this query depends on the date when it was executed, the values in the MONTHS\_WORKED column will differ for you.

```
SELECT last_name, ROUND(MONTHS_BETWEEN(
        SYSDATE, hire_date)) MONTHS_WORKED
FROM    employees
ORDER BY months_worked;
```

If you have the time, complete the following exercises:

- 7) Create a query to display the last name and salary for all employees. Format the salary to be 15 characters long, left-padded with the \$ symbol. Label the column SALARY.

```
SELECT last_name,
        LPAD(salary, 15, '$') SALARY
FROM    employees;
```

- 8) Create a query that displays the first eight characters of the employees' last names and indicates the amounts of their salaries with asterisks. Each asterisk signifies a thousand dollars. Sort the data in descending order of salary. Label the column EMPLOYEES\_AND\_THEIR\_SALARIES.

```
SELECT rpad(last_name, 8) || ' ' ||
        rpad(' ', salary/1000+1, '*')
        EMPLOYEES_AND_THEIR_SALARIES
FROM    employees
ORDER BY salary DESC;
```

### ***Practice Solutions 3-1: Using Single-Row Functions to Customize Output (continued)***

- 9) Create a query to display the last name and the number of weeks employed for all employees in department 90. Label the number of weeks column `TENURE`. Truncate the number of weeks value to 0 decimal places. Show the records in descending order of the employee's tenure.

**Note:** The `TENURE` value will differ as it depends on the date when you run the query.

```
SELECT last_name, trunc((SYSDATE-hire_date)/7) AS TENURE
FROM   employees
WHERE  department_id = 90
ORDER BY TENURE DESC
```

---

## Practices for Lesson 4

---

This practice provides a variety of exercises using `TO_CHAR` and `TO_DATE` functions, and conditional expressions such as `DECODE` and `CASE`. Remember that for nested functions, the results are evaluated from the innermost function to the outermost function.

## Practice 4-1: Using Conversion Functions and Conditional Expressions

- 1) Create a report that produces the following for each employee:  
 <employee last name> earns <salary> monthly but wants <3 times salary.>. Label the column Dream Salaries.

	Dream Salaries
1	Whalen earns \$4,400.00 monthly but wants \$13,200.00.
2	Hartstein earns \$13,000.00 monthly but wants \$39,000.00.
3	Fay earns \$6,000.00 monthly but wants \$18,000.00.
4	Higgins earns \$12,000.00 monthly but wants \$36,000.00.
5	Gietz earns \$8,300.00 monthly but wants \$24,900.00.

...

19	Taylor earns \$8,600.00 monthly but wants \$25,800.00.
20	Grant earns \$7,000.00 monthly but wants \$21,000.00.

- 2) Display each employee's last name, hire date, and salary review date, which is the first Monday after six months of service. Label the column REVIEW. Format the dates to appear in the format similar to "Monday, the Thirty-First of July, 2000."

	LAST_NAME	HIRE_DATE	REVIEW
1	Whalen	17-SEP-87	Monday, the Twenty-First of March, 1988
2	Hartstein	17-FEB-96	Monday, the Nineteenth of August, 1996
3	Fay	17-AUG-97	Monday, the Twenty-Third of February, 1998
4	Higgins	07-JUN-94	Monday, the Twelfth of December, 1994
5	Gietz	07-JUN-94	Monday, the Twelfth of December, 1994

...

19	Taylor	24-MAR-98	Monday, the Twenty-Eighth of September, 1998
20	Grant	24-MAY-99	Monday, the Twenty-Ninth of November, 1999

- 3) Display the last name, hire date, and day of the week on which the employee started. Label the column DAY. Order the results by the day of the week, starting with Monday.

	LAST_NAME	HIRE_DATE	DAY
1	Grant	24-MAY-99	MONDAY
2	Ernst	21-MAY-91	TUESDAY
3	Taylor	24-MAR-98	TUESDAY
4	Rajs	17-OCT-95	TUESDAY
5	Mourgos	16-NOV-99	TUESDAY

...

19	Matos	15-MAR-98	SUNDAY
20	Fay	17-AUG-97	SUNDAY

## Practice 4-1: Using Conversion Functions and Conditional Expressions (continued)

- 4) Create a query that displays the employees' last names and commission amounts. If an employee does not earn commission, show "No Commission." Label the column COMM.

	LAST_NAME	COMM
1	Whalen	No Commission
2	Hartstein	No Commission
3	Fay	No Commission
4	Higgins	No Commission
5	Gietz	No Commission

...

16	Vargas	No Commission
17	Zlotkey	.2
18	Abel	.3
19	Taylor	.2
20	Grant	.15

If you have time, complete the following exercises:

- 5) Using the DECODE function, write a query that displays the grade of all employees based on the value of the column JOB\_ID, using the following data:

Job	Grade
AD_PRES	A
ST_MAN	B
IT_PROG	C
SA_REP	D
ST_CLERK	E
None of the above	0

	JOB_ID	GRADE
1	AC_ACCOUNT	0
2	AC_MGR	0
3	AD_ASST	0
4	AD_PRES	A
5	AD_VP	0
6	AD_VP	0
7	IT_PROG	C

...



14	SA_REP	D
15	SA_REP	D

...

19	ST_CLERK	E
20	ST_MAN	B

### ***Practice 4-1: Using Conversion Functions and Conditional Expressions (continued)***

6) Rewrite the statement in the preceding exercise by using the CASE syntax.

	 JOB_ID	 GRADE
1	AC_ACCOUNT	0
2	AC_MGR	0
3	AD_ASST	0
4	AD_PREP	A
5	AD_VP	0
6	AD_VP	0
7	IT_PROG	C

...

14	SA_REP	D
15	SA_REP	D

...

19	ST_CLERK	E
20	ST_MAN	B

## Practice Solutions 4-1: Using Conversion Functions and Conditional Expressions

- 1) Create a report that produces the following for each employee:  
<employee last name> earns <salary> monthly but wants <3 times salary>. Label the column Dream Salaries.

```
SELECT last_name || ' earns '  
       || TO_CHAR(salary, 'fm$99,999.00')  
       || ' monthly but wants '  
       || TO_CHAR(salary * 3, 'fm$99,999.00')  
       || '. ' "Dream Salaries"  
FROM   employees;
```

- 2) Display each employee's last name, hire date, and salary review date, which is the first Monday after six months of service. Label the column REVIEW. Format the dates to appear in the format similar to "Monday, the Thirty-First of July, 2000."

```
SELECT last_name, hire_date,  
       TO_CHAR(NEXT_DAY(ADD_MONTHS(hire_date, 6), 'MONDAY'),  
               'fmDay, "the" Ddspth "of" Month, YYYY') REVIEW  
FROM   employees;
```

- 3) Display the last name, hire date, and day of the week on which the employee started. Label the column DAY. Order the results by the day of the week, starting with Monday.

```
SELECT last_name, hire_date,  
       TO_CHAR(hire_date, 'DAY') DAY  
FROM   employees  
ORDER BY TO_CHAR(hire_date - 1, 'd');
```

- 4) Create a query that displays the employees' last names and commission amounts. If an employee does not earn commission, show "No Commission." Label the column COMM.

```
SELECT last_name,  
       NVL(TO_CHAR(commission_pct), 'No Commission') COMM  
FROM   employees;
```

- 5) Using the DECODE function, write a query that displays the grade of all employees based on the value of the JOB\_ID column, using the following data:

<b>Job</b>	<b>Grade</b>
AD_PRES	A
ST_MAN	B
IT_PROG	C
SA_REP	D
ST_CLERK	E
None of the above	0

## ***Practice Solutions 4-1: Using Conversion Functions and Conditional Expressions (continued)***

```
SELECT job_id, decode (job_id,  
                        'ST_CLERK', 'E',  
                        'SA_REP',   'D',  
                        'IT_PROG',  'C',  
                        'ST_MAN',   'B',  
                        'AD_PRES',  'A',  
                        '0') GRADE  
FROM employees;
```

- 6) Rewrite the statement in the preceding exercise by using the CASE syntax.

```
SELECT job_id, CASE job_id  
                WHEN 'ST_CLERK' THEN 'E'  
                WHEN 'SA_REP'   THEN 'D'  
                WHEN 'IT_PROG'  THEN 'C'  
                WHEN 'ST_MAN'   THEN 'B'  
                WHEN 'AD_PRES'  THEN 'A'  
                ELSE '0' END GRADE  
FROM employees;
```



---

## Practices for Lesson 5

---

At the end of this practice, you should be familiar with using group functions and selecting groups of data.

## Practice 5-1: Reporting Aggregated Data Using the Group Functions

Determine the validity of the following three statements. Circle either True or False.

- 1) Group functions work across many rows to produce one result per group.  
True/False
- 2) Group functions include nulls in calculations.  
True/False
- 3) The WHERE clause restricts rows before inclusion in a group calculation.  
True/False

The HR department needs the following reports:

- 4) Find the highest, lowest, sum, and average salary of all employees. Label the columns Maximum, Minimum, Sum, and Average, respectively. Round your results to the nearest whole number. Save your SQL statement as lab\_05\_04.sql. Run the query.

	Maximum	Minimum	Sum	Average
1	24000	2500	175500	8775

- 5) Modify the query in lab\_05\_04.sql to display the minimum, maximum, sum, and average salary for each job type. Save lab\_05\_04.sql as lab\_05\_05.sql again. Run the statement in lab\_05\_05.sql.

	JOB_ID	Maximum	Minimum	Sum	Average
1	AC_MGR	12000	12000	12000	12000
2	AC_ACCOUNT	8300	8300	8300	8300
3	IT_PROG	9000	4200	19200	6400
4	ST_MAN	5800	5800	5800	5800
5	AD_ASST	4400	4400	4400	4400
6	AD_VP	17000	17000	34000	17000
7	MK_MAN	13000	13000	13000	13000
8	SA_MAN	10500	10500	10500	10500
9	MK_REP	6000	6000	6000	6000
10	AD_PRES	24000	24000	24000	24000
11	SA_REP	11000	7000	26600	8867
12	ST_CLERK	3500	2500	11700	2925

## Practice 5-1: Reporting Aggregated Data Using the Group Functions (continued)

- 6) Write a query to display the number of people with the same job.

	JOB_ID	COUNT(*)
1	AC_ACCOUNT	1
2	AC_MGR	1
3	AD_ASST	1
4	AD_PRES	1
5	AD_VP	2
6	IT_PROG	3
7	MK_MAN	1
8	MK_REP	1
9	SA_MAN	1
10	SA_REP	3
11	ST_CLERK	4
12	ST_MAN	1

Generalize the query so that the user in the HR department is prompted for a job title. Save the script to a file named lab\_05\_06.sql. Run the query. Enter IT\_PROG when prompted.

	JOB_ID	COUNT(*)
1	IT_PROG	3

- 7) Determine the number of managers without listing them. Label the column Number of Managers.

**Hint:** Use the MANAGER\_ID column to determine the number of managers.

	Number of Managers
1	8

- 8) Find the difference between the highest and lowest salaries. Label the column DIFFERENCE.

	DIFFERENCE
1	21500

If you have time, complete the following exercises:

- 9) Create a report to display the manager number and the salary of the lowest-paid employee for that manager. Exclude anyone whose manager is not known. Exclude any groups where the minimum salary is \$6,000 or less. Sort the output in descending order of salary.

	MANAGER_ID	MIN(SALARY)
1	102	9000
2	205	8300
3	149	7000

## Practice 5-1: Reporting Aggregated Data Using the Group Functions (continued)

If you want an extra challenge, complete the following exercises:

- 10) Create a query to display the total number of employees and, of that total, the number of employees hired in 1995, 1996, 1997, and 1998. Create appropriate column headings.

	2	TOTAL	2	1995	2	1996	2	1997	2	1998
1		20	1		2		2		3	

- 11) Create a matrix query to display the job, the salary for that job based on department number, and the total salary for that job, for departments 20, 50, 80, and 90, giving each column an appropriate heading.

	2	Job	2	Dept 20	2	Dept 50	2	Dept 80	2	Dept 90	2	Total
1		AC_MGR		(null)		(null)		(null)		(null)		12000
2		AC_ACCOUNT		(null)		(null)		(null)		(null)		8300
3		IT_PROG		(null)		(null)		(null)		(null)		19200
4		ST_MAN		(null)		5800		(null)		(null)		5800
5		AD_ASST		(null)		(null)		(null)		(null)		4400
6		AD_VP		(null)		(null)		(null)		34000		34000
7		MK_MAN		13000		(null)		(null)		(null)		13000
8		SA_MAN		(null)		(null)		10500		(null)		10500
9		MK_REP		6000		(null)		(null)		(null)		6000
10		AD_PRES		(null)		(null)		(null)		24000		24000
11		SA_REP		(null)		(null)		19600		(null)		26600
12		ST_CLERK		(null)		11700		(null)		(null)		11700

## ***Practice Solutions 5-1: Reporting Aggregated Data Using the Group Functions***

Determine the validity of the following three statements. Circle either True or False.

- 1) Group functions work across many rows to produce one result per group.  
**True/False**
- 2) Group functions include nulls in calculations.  
**True/False**
- 3) The WHERE clause restricts rows before inclusion in a group calculation.  
**True/False**

The HR department needs the following reports:

- 4) Find the highest, lowest, sum, and average salary of all employees. Label the columns Maximum, Minimum, Sum, and Average, respectively. Round your results to the nearest whole number. Save your SQL statement as lab\_05\_04.sql. Run the query.

```
SELECT ROUND(MAX(salary),0) "Maximum",  
       ROUND(MIN(salary),0) "Minimum",  
       ROUND(SUM(salary),0) "Sum",  
       ROUND(AVG(salary),0) "Average"  
FROM   employees;
```

- 5) Modify the query in lab\_05\_04.sql to display the minimum, maximum, sum, and average salary for each job type. Save lab\_05\_04.sql as lab\_05\_05.sql again. Run the statement in lab\_05\_05.sql.

```
SELECT job_id, ROUND(MAX(salary),0) "Maximum",  
       ROUND(MIN(salary),0) "Minimum",  
       ROUND(SUM(salary),0) "Sum",  
       ROUND(AVG(salary),0) "Average"  
FROM   employees  
GROUP BY job_id;
```

- 6) Write a query to display the number of people with the same job.

```
SELECT job_id, COUNT(*)  
FROM   employees  
GROUP BY job_id;
```

Generalize the query so that the user in the HR department is prompted for a job title. Save the script to a file named lab\_05\_06.sql. Run the query. Enter IT\_PROG when prompted and click OK.

```
SELECT job_id, COUNT(*)  
FROM   employees  
WHERE  job_id = '&job_title'  
GROUP BY job_id;
```

## ***Practice Solutions 5-1: Reporting Aggregated Data Using the Group Functions (continued)***

- 7) Determine the number of managers without listing them. Label the column Number of Managers.

**Hint:** Use the MANAGER\_ID column to determine the number of managers.

```
SELECT COUNT(DISTINCT manager_id) "Number of Managers"
FROM   employees;
```

- 8) Find the difference between the highest and lowest salaries. Label the column DIFFERENCE.

```
SELECT   MAX(salary) - MIN(salary) DIFFERENCE
FROM     employees;
```

If you have the time, complete the following exercises:

- 9) Create a report to display the manager number and the salary of the lowest-paid employee for that manager. Exclude anyone whose manager is not known. Exclude any groups where the minimum salary is \$6,000 or less. Sort the output in descending order of salary.

```
SELECT   manager_id, MIN(salary)
FROM     employees
WHERE    manager_id IS NOT NULL
GROUP BY manager_id
HAVING   MIN(salary) > 6000
ORDER BY MIN(salary) DESC;
```

If you want an extra challenge, complete the following exercises:

- 10) Create a query that will display the total number of employees and, of that total, the number of employees hired in 1995, 1996, 1997, and 1998. Create appropriate column headings.

```
SELECT   COUNT(*) total,
          SUM(DECODE(TO_CHAR(hire_date,
'YYYY'),1995,1,0)) "1995",
          SUM(DECODE(TO_CHAR(hire_date,
'YYYY'),1996,1,0)) "1996",
          SUM(DECODE(TO_CHAR(hire_date,
'YYYY'),1997,1,0)) "1997",
          SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),1998,1,0)) "1998"
FROM     employees;
```

### ***Practice Solutions 5-1: Reporting Aggregated Data Using the Group Functions (continued)***

- 11) Create a matrix query to display the job, the salary for that job based on the department number, and the total salary for that job, for departments 20, 50, 80, and 90, giving each column an appropriate heading.

```
SELECT    job_id "Job",  
          SUM(DECODE(department_id , 20, salary)) "Dept 20",  
          SUM(DECODE(department_id , 50, salary)) "Dept 50",  
          SUM(DECODE(department_id , 80, salary)) "Dept 80",  
          SUM(DECODE(department_id , 90, salary)) "Dept 90",  
          SUM(salary) "Total"  
FROM      employees  
GROUP BY  job_id;
```

## Practices for Lesson 6

This practice is intended to give you experience in extracting data from more than one table using the SQL:1999-compliant joins.



## Practice 6-1: Displaying Data from Multiple Tables Using Joins

- 1) Write a query for the HR department to produce the addresses of all the departments. Use the `LOCATIONS` and `COUNTRIES` tables. Show the location ID, street address, city, state or province, and country in the output. Use a `NATURAL JOIN` to produce the results.

	LOCATION_ID	STREET_ADDRESS	CITY	STATE_PROVINCE	COUNTRY_NAME
1		1400 2014 Jabberwocky Rd	Southlake	Texas	United States of America
2		1500 2011 Interiors Blvd	South San Francisco	California	United States of America
3		1700 2004 Charade Rd	Seattle	Washington	United States of America
4		1800 460 Bloor St. W.	Toronto	Ontario	Canada
5		2500 Magdalen Centre, The Oxford Science Park	Oxford	Oxford	United Kingdom

- 2) The HR department needs a report of only those employees with corresponding departments. Write a query to display the last name, department number, and department name for these employees.

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Davies	50	Shipping
5	Vargas	50	Shipping

...

18	Higgins	110	Accounting
19	Gietz	110	Accounting

- 3) The HR department needs a report of employees in Toronto. Display the last name, job, department number, and the department name for all employees who work in Toronto.

	LAST_NAME	JOB_ID	DEPARTMENT_ID	DEPARTMENT_NAME
1	Hartstein	MK_MAN	20	Marketing
2	Fay	MK_REP	20	Marketing

- 4) Create a report to display employees' last name and employee number along with their manager's last name and manager number. Label the columns `Employee`, `Emp#`, `Manager`, and `Mgr#`, respectively. Save your SQL statement as `lab_06_04.sql`. Run the query.

	Employee	EMP#	Manager	Mgr#
1	Hunold	103	De Haan	102
2	Fay	202	Hartstein	201
3	Gietz	206	Higgins	205
4	Lorentz	107	Hunold	103
5	Ernst	104	Hunold	103

...

## Practice 6-1: Displaying Data from Multiple Tables Using Joins (continued)

18 Taylor	176 Zlotkey	149
19 Abel	174 Zlotkey	149

- 5) Modify `lab_06_04.sql` to display all employees including King, who has no manager. Order the results by the employee number. Save your SQL statement as `lab_06_05.sql`. Run the query in `lab_06_05.sql`.

Employee	EMP#	Manager	Mgr#
1 King	100 (null)	(null)	
2 Kochhar	101 King		100
3 De Haan	102 King		100
4 Hunold	103 De Haan		102
5 Ernst	104 Hunold		103

...

19 Higgins	205 Kochhar		101
20 Gietz	206 Higgins		205

- 6) Create a report for the HR department that displays employee last names, department numbers, and all the employees who work in the same department as a given employee. Give each column an appropriate label. Save the script to a file named `lab_06_06.sql`.

DEPARTMENT	EMPLOYEE	COLLEAGUE
1	20 Fay	Hartstein
2	20 Hartstein	Fay
3	50 Davies	Matos
4	50 Davies	Mourgos
5	50 Davies	Rajs

...

41	110 Gietz	Higgins
42	110 Higgins	Gietz

- 7) The HR department needs a report on job grades and salaries. To familiarize yourself with the `JOB_GRADES` table, first show the structure of the `JOB_GRADES` table. Then create a query that displays the name, job, department name, salary, and grade for all employees.

DESC JOB_GRADES		
Name	Null	Type
-----		
GRADE_LEVEL		VARCHAR2(3)
LOWEST_SAL		NUMBER
HIGHEST_SAL		NUMBER
3 rows selected		

## Practice 6-1: Displaying Data from Multiple Tables Using Joins (continued)

	LAST_NAME	JOB_ID	DEPARTMENT_NAME	SALARY	GRADE_LEVEL
1	King	AD_PRES	Executive	24000	E
2	Kochhar	AD_VP	Executive	17000	E
3	De Haan	AD_VP	Executive	17000	E
4	Hartstein	MK_MAN	Marketing	13000	D
5	Higgins	AC_MGR	Accounting	12000	D

...

18	Matos	ST_CLERK	Shipping	2600	A
19	Vargas	ST_CLERK	Shipping	2500	A

If you want an extra challenge, complete the following exercises:

- 8) The HR department wants to determine the names of all the employees who were hired after Davies. Create a query to display the name and hire date of any employee hired after employee Davies.

	LAST_NAME	HIRE_DATE
1	Fay	17-AUG-97
2	Lorentz	07-FEB-99
3	Mourgos	16-NOV-99
4	Matos	15-MAR-98
5	Vargas	09-JUL-98
6	Zlotkey	29-JAN-00
7	Taylor	24-MAR-98
8	Grant	24-MAY-99

- 9) The HR department needs to find the names and hire dates of all the employees who were hired before their managers, along with their managers' names and hire dates. Save the script to a file named `lab_06_09.sql`.

	LAST_NAME	HIRE_DATE	LAST_NAME_1	HIRE_DATE_1
1	Whalen	17-SEP-87	Kochhar	21-SEP-89
2	Hunold	03-JAN-90	De Haan	13-JAN-93
3	Vargas	09-JUL-98	Mourgos	16-NOV-99
4	Matos	15-MAR-98	Mourgos	16-NOV-99
5	Davies	29-JAN-97	Mourgos	16-NOV-99
6	Rajs	17-OCT-95	Mourgos	16-NOV-99
7	Grant	24-MAY-99	Zlotkey	29-JAN-00
8	Taylor	24-MAR-98	Zlotkey	29-JAN-00
9	Abel	11-MAY-96	Zlotkey	29-JAN-00

## ***Practice Solutions 6-1: Displaying Data from Multiple Tables Using Joins***

- 1) Write a query for the HR department to produce the addresses of all the departments. Use the `LOCATIONS` and `COUNTRIES` tables. Show the location ID, street address, city, state or province, and country in the output. Use a `NATURAL JOIN` to produce the results.

```
SELECT location_id, street_address, city, state_province,
country_name
FROM   locations
NATURAL JOIN countries;
```

- 2) The HR department needs a report of all employees. Write a query to display the last name, department number, and department name for all the employees.

```
SELECT last_name, department_id, department_name
FROM   employees
JOIN   departments
USING (department_id);
```

- 3) The HR department needs a report of employees in Toronto. Display the last name, job, department number, and department name for all employees who work in Toronto.

```
SELECT e.last_name, e.job_id, e.department_id,
d.department_name
FROM   employees e JOIN departments d
ON     (e.department_id = d.department_id)
JOIN   locations l
ON     (d.location_id = l.location_id)
WHERE  LOWER(l.city) = 'toronto';
```

- 4) Create a report to display employees' last names and employee number along with their managers' last names and manager number. Label the columns `Employee`, `Emp#`, `Manager`, and `Mgr#`, respectively. Save your SQL statement as `lab_06_04.sql`. Run the query.

```
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id  "Mgr#"
FROM   employees w join employees m
ON     (w.manager_id = m.employee_id);
```

- 5) Modify `lab_06_04.sql` to display all employees including King, who has no manager. Order the results by the employee number. Save your SQL statement as `lab_06_05.sql`. Run the query in `lab_06_05.sql`.

```
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id  "Mgr#"
FROM   employees w
LEFT   OUTER JOIN employees m
ON     (w.manager_id = m.employee_id)
ORDER BY 2;
```

## **Practice Solutions 6-1: Displaying Data from Multiple Tables Using Joins (continued)**

- 6) Create a report for the HR department that displays employee last names, department numbers, and all the employees who work in the same department as a given employee. Give each column an appropriate label. Save the script to a file named lab\_06\_06.sql. Run the query.

```
SELECT e.department_id department, e.last_name employee,  
       c.last_name colleague  
FROM   employees e JOIN employees c  
ON     (e.department_id = c.department_id)  
WHERE  e.employee_id <> c.employee_id  
ORDER BY e.department_id, e.last_name, c.last_name;
```

- 7) The HR department needs a report on job grades and salaries. To familiarize yourself with the JOB\_GRADES table, first show the structure of the JOB\_GRADES table. Then create a query that displays the name, job, department name, salary, and grade for all employees.

```
DESC JOB_GRADES  
  
SELECT e.last_name, e.job_id, d.department_name,  
       e.salary, j.grade_level  
FROM   employees e JOIN departments d  
ON     (e.department_id = d.department_id)  
JOIN   job_grades j  
ON     (e.salary BETWEEN j.lowest_sal AND j.highest_sal);
```

If you want an extra challenge, complete the following exercises:

- 8) The HR department wants to determine the names of all employees who were hired after Davies. Create a query to display the name and hire date of any employee hired after employee Davies.

```
SELECT e.last_name, e.hire_date  
FROM   employees e JOIN employees davies  
ON     (davies.last_name = 'Davies')  
WHERE  davies.hire_date < e.hire_date;
```

- 9) The HR department needs to find the names and hire dates for all employees who were hired before their managers, along with their managers' names and hire dates. Save the script to a file named lab\_06\_09.sql.

```
SELECT w.last_name, w.hire_date, m.last_name, m.hire_date  
FROM   employees w JOIN employees m  
ON     (w.manager_id = m.employee_id)  
WHERE  w.hire_date < m.hire_date;
```

---

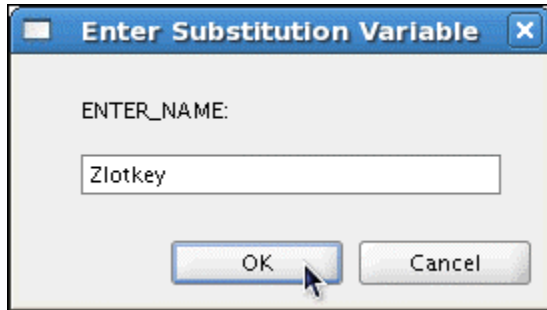
## Practices for Lesson 7

---

In this practice, you write complex queries using nested `SELECT` statements.  
For practice questions, you may want to create the inner query first. Make sure that it runs and produces the data that you anticipate before you code the outer query.

## Practice 7-1: Using Subqueries to Solve Queries

- 1) The HR department needs a query that prompts the user for an employee last name. The query then displays the last name and hire date of any employee in the same department as the employee whose name they supply (excluding that employee). For example, if the user enters Zlotkey, find all employees who work with Zlotkey (excluding Zlotkey).



	LAST_NAME	HIRE_DATE
1	Abel	11-MAY-96
2	Taylor	24-MAR-98

- 2) Create a report that displays the employee number, last name, and salary of all employees who earn more than the average salary. Sort the results in order of ascending salary.

	EMPLOYEE_ID	LAST_NAME	SALARY
1	103	Hunold	9000
2	149	Zlotkey	10500
3	174	Abel	11000
4	205	Higgins	12000
5	201	Hartstein	13000
6	102	De Haan	17000
7	101	Kochhar	17000
8	100	King	24000

- 3) Write a query that displays the employee number and last name of all employees who work in a department with any employee whose last name contains the letter "u." Save your SQL statement as lab\_07\_03.sql. Run your query.

	EMPLOYEE_ID	LAST_NAME
1	124	Mourgos
2	141	Rajs
3	142	Davies
4	143	Matos
5	144	Vargas
6	103	Hunold
7	104	Ernst
8	107	Lorentz

### Practice 7-1: Using Subqueries to Solve Queries (continued)

- 4) The HR department needs a report that displays the last name, department number, and job ID of all employees whose department location ID is 1700.

	LAST_NAME	DEPARTMENT_ID	JOB_ID
1	Whalen	10	AD_ASST
2	King	90	AD_PRES
3	Kochhar	90	AD_VP
4	De Haan	90	AD_VP
5	Higgins	110	AC_MGR
6	Gietz	110	AC_ACCOUNT

Modify the query so that the user is prompted for a location ID. Save this to a file named `lab_07_04.sql`.

- 5) Create a report for HR that displays the last name and salary of every employee who reports to King.

	LAST_NAME	SALARY
1	Hartstein	13000
2	Kochhar	17000
3	De Haan	17000
4	Mourgos	5800
5	Zlotkey	10500

- 6) Create a report for HR that displays the department number, last name, and job ID for every employee in the Executive department.

	DEPARTMENT_ID	LAST_NAME	JOB_ID
1	90	King	AD_PRES
2	90	Kochhar	AD_VP
3	90	De Haan	AD_VP

- 7) Create a report that displays a list of all employees whose salary is more than the salary of any employee from department 60.

If you have the time, complete the following exercise:

- 8) Modify the query in `lab_07_03.sql` to display the employee number, last name, and salary of all employees who earn more than the average salary, and who work in a department with any employee whose last name contains a "u." Save `lab_07_03.sql` as `lab_07_08.sql` again. Run the statement in `lab_07_08.sql`.

	EMPLOYEE_ID	LAST_NAME	SALARY
1	103	Hunold	9000



## Practice Solutions 7-1: Using Subqueries to Solve Queries

- 1) The HR department needs a query that prompts the user for an employee last name. The query then displays the last name and hire date of any employee in the same department as the employee whose name they supply (excluding that employee). For example, if the user enters Zlotkey, find all employees who work with Zlotkey (excluding Zlotkey).

```
UNDEFINE Enter_name

SELECT last_name, hire_date
FROM   employees
WHERE  department_id = (SELECT department_id
                        FROM   employees
                        WHERE  last_name = '&&Enter_name')
AND    last_name <> '&Enter_name';
```

- 2) Create a report that displays the employee number, last name, and salary of all employees who earn more than the average salary. Sort the results in order of ascending salary.

```
SELECT employee_id, last_name, salary
FROM   employees
WHERE  salary > (SELECT AVG(salary)
                 FROM   employees)
ORDER BY salary;
```

- 3) Write a query that displays the employee number and last name of all employees who work in a department with any employee whose last name contains a “u.” Save your SQL statement as lab\_07\_03.sql. Run your query.

```
SELECT employee_id, last_name
FROM   employees
WHERE  department_id IN (SELECT department_id
                        FROM   employees
                        WHERE  last_name like '%u%');
```

- 4) The HR department needs a report that displays the last name, department number, and job ID of all employees whose department location ID is 1700.

```
SELECT last_name, department_id, job_id
FROM   employees
WHERE  department_id IN (SELECT department_id
                        FROM   departments
                        WHERE  location_id = 1700);
```

## **Practice Solutions 7-1: Using Subqueries to Solve Queries (continued)**

Modify the query so that the user is prompted for a location ID. Save this to a file named lab\_07\_04.sql.

```
SELECT last_name, department_id, job_id
FROM   employees
WHERE  department_id IN (SELECT department_id
                        FROM   departments
                        WHERE  location_id =
                        &Enter_location);
```

- 5) Create a report for HR that displays the last name and salary of every employee who reports to King.

```
SELECT last_name, salary
FROM   employees
WHERE  manager_id = (SELECT employee_id
                    FROM   employees
                    WHERE  last_name = 'King');
```

- 6) Create a report for HR that displays the department number, last name, and job ID for every employee in the Executive department.

```
SELECT department_id, last_name, job_id
FROM   employees
WHERE  department_id IN (SELECT department_id
                        FROM   departments
                        WHERE  department_name =
                        'Executive');
```

- 7) Create a report that displays a list of all employees whose salary is more than the salary of any employee from department 60.

```
SELECT last_name FROM employees
WHERE salary > ANY (SELECT salary
                   FROM employees
                   WHERE department_id=60);
```

If you have the time, complete the following exercise:

- 8) Modify the query in lab\_07\_03.sql to display the employee number, last name, and salary of all employees who earn more than the average salary and who work in a department with any employee whose last name contains a “u.” Save lab\_07\_03.sql to lab\_07\_08.sql again. Run the statement in lab\_07\_08.sql.

```
SELECT employee_id, last_name, salary
FROM   employees
WHERE  department_id IN (SELECT department_id
                        FROM   employees
                        WHERE  last_name like '%u%')
AND    salary > (SELECT AVG(salary)
                FROM   employees);
```

---

## Practices for Lesson 8

---

In this practice, you write queries using the set operators.

### Practice 8-1: Using the Set Operators

- 1) The HR department needs a list of department IDs for departments that do not contain the job ID ST\_CLERK. Use the set operators to create this report.

	DEPARTMENT_ID
1	10
2	20
3	60
4	80
5	90
6	110
7	190

- 2) The HR department needs a list of countries that have no departments located in them. Display the country ID and the name of the countries. Use the set operators to create this report.

	COUNTRY_ID	COUNTRY_NAME
1	DE	Germany

- 3) Produce a list of jobs for departments 10, 50, and 20, in that order. Display the job ID and department ID by using the set operators.

	JOB_ID	DEPARTMENT_ID
1	AD_ASST	10
2	ST_MAN	50
3	ST_CLERK	50
4	MK_MAN	20
5	MK_REP	20

- 4) Create a report that lists the employee IDs and job IDs of those employees who currently have a job title that is the same as their job title when they were initially hired by the company (that is, they changed jobs, but have now gone back to doing their original job).

	EMPLOYEE_ID	JOB_ID
1	176	SA_REP
2	200	AD_ASST

- 5) The HR department needs a report with the following specifications:
- Last name and department ID of all employees from the EMPLOYEES table, regardless of whether or not they belong to a department
  - Department ID and department name of all departments from the DEPARTMENTS table, regardless of whether or not they have employees working in them

Write a compound query to accomplish this.

### Practice 8-1: Using the Set Operators (continued)

	LAST_NAME	DEPARTMENT_ID	TO_CHAR(NULL)
1	Abel	80	(null)
2	Davies	50	(null)
3	De Haan	90	(null)
4	Ernst	60	(null)
5	Fay	20	(null)
6	Gietz	110	(null)
7	Grant	(null)	(null)
8	Hartstein	20	(null)
9	Higgins	110	(null)
10	Hunold	60	(null)
11	King	90	(null)
12	Kochhar	90	(null)
13	Lorentz	60	(null)
14	Matos	50	(null)
15	Mourgos	50	(null)
16	Rajs	50	(null)
17	Taylor	80	(null)
18	Vargas	50	(null)
19	Whalen	10	(null)
20	Zlotkey	80	(null)
21	(null)	10	Administration
22	(null)	20	Marketing
23	(null)	50	Shipping
24	(null)	60	IT
25	(null)	80	Sales
26	(null)	90	Executive
27	(null)	110	Accounting
28	(null)	190	Contracting

## ***Practice Solutions 8-1: Using the Set Operators***

- 1) The HR department needs a list of department IDs for departments that do not contain the job ID ST\_CLERK. Use the set operators to create this report.

```
SELECT department_id
FROM departments
MINUS
SELECT department_id
FROM employees
WHERE job_id = 'ST_CLERK';
```

- 2) The HR department needs a list of countries that have no departments located in them. Display the country ID and the name of the countries. Use the set operators to create this report.

```
SELECT country_id, country_name
FROM countries
MINUS
SELECT l.country_id, c.country_name
FROM locations l JOIN countries c
ON (l.country_id = c.country_id)
JOIN departments d
ON d.location_id=l.location_id;
```

- 3) Produce a list of jobs for departments 10, 50, and 20, in that order. Display job ID and department ID using the set operators.

```
SELECT distinct job_id, department_id
FROM employees
WHERE department_id = 10
UNION ALL
SELECT DISTINCT job_id, department_id
FROM employees
WHERE department_id = 50
UNION ALL
SELECT DISTINCT job_id, department_id
FROM employees
WHERE department_id = 20
```

- 4) Create a report that lists the employee IDs and job IDs of those employees who currently have a job title that is the same as their job title when they were initially hired by the company (that is, they changed jobs, but have now gone back to doing their original job).

```
SELECT      employee_id, job_id
FROM        employees
INTERSECT
SELECT      employee_id, job_id
FROM        job_history;
```

### ***Practice Solutions 8-1: Using the Set Operators (continued)***

5) The HR department needs a report with the following specifications:

- Last name and department ID of all the employees from the EMPLOYEES table, regardless of whether or not they belong to a department
- Department ID and department name of all the departments from the DEPARTMENTS table, regardless of whether or not they have employees working in them

Write a compound query to accomplish this.

```
SELECT last_name,department_id,TO_CHAR(null)
FROM   employees
UNION
SELECT TO_CHAR(null),department_id,department_name
FROM   departments;
```

---

## Practices for Lesson 9

---

In this practice, you add rows to the MY\_EMPLOYEE table, update and delete data from the table, and control your transactions. You run a script to create the MY\_EMPLOYEE table.



## Practice 9-1: Manipulating Data

The HR department wants you to create SQL statements to insert, update, and delete employee data. As a prototype, you use the MY\_EMPLOYEE table before giving the statements to the HR department.

**Note:** For all the DML statements, use the Run Script icon (or press [F5]) to execute the query. This way you get to see the feedback messages on the Script Output tabbed page. For SELECT queries, continue to use the Execute Statement icon or press [F9] to get the formatted output on the Results tabbed page.

### Insert data into the MY\_EMPLOYEE table.

- 1) Run the statement in the lab\_09\_01.sql script to build the MY\_EMPLOYEE table used in this practice.
- 2) Describe the structure of the MY\_EMPLOYEE table to identify the column names.

DESCRIBE my_employee		
Name	Null	Type
-----	-----	-----
ID	NOT NULL	NUMBER(4)
LAST_NAME		VARCHAR2(25)
FIRST_NAME		VARCHAR2(25)
USERID		VARCHAR2(8)
SALARY		NUMBER(9,2)
5 rows selected		

- 3) Create an INSERT statement to add the *first row* of data to the MY\_EMPLOYEE table from the following sample data. Do not list the columns in the INSERT clause. *Do not enter all rows yet.*

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audrey	aropebur	1550

- 4) Populate the MY\_EMPLOYEE table with the second row of the sample data from the preceding list. This time, list the columns explicitly in the INSERT clause.
- 5) Confirm your addition to the table.

### Practice 9-1: Manipulating Data (continued)

	ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	1	Patel	Ralph	rpatel	895
2	2	Dancs	Betty	bdancs	860

- 6) Write an INSERT statement in a dynamic reusable script file to load the remaining rows into the MY\_EMPLOYEE table. The script should prompt for all the columns (ID, LAST\_NAME, FIRST\_NAME, USERID, and SALARY). Save this script to a lab\_09\_06.sql file.
- 7) Populate the table with the next two rows of the sample data listed in step 3 by running the INSERT statement in the script that you created.
- 8) Confirm your additions to the table.

	ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	1	Patel	Ralph	rpatel	895
2	2	Dancs	Betty	bdancs	860
3	3	Biri	Ben	bbiri	1100
4	4	Newman	Chad	cnewman	750

- 9) Make the data additions permanent.

#### Update and delete data in the MY\_EMPLOYEE table.

- 10) Change the last name of employee 3 to Drexler.
- 11) Change the salary to \$1,000 for all employees who have a salary less than \$900.
- 12) Verify your changes to the table.

	ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	1	Patel	Ralph	rpatel	1000
2	2	Dancs	Betty	bdancs	1000
3	3	Drexler	Ben	bbiri	1100
4	4	Newman	Chad	cnewman	1000

- 13) Delete Betty Dancs from the MY\_EMPLOYEE table.
- 14) Confirm your changes to the table.

	ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	1	Patel	Ralph	rpatel	1000
2	3	Drexler	Ben	bbiri	1100
3	4	Newman	Chad	cnewman	1000






## Practice 9-1: Manipulating Data (continued)

15) Commit all pending changes.

### Control data transaction to the MY\_EMPLOYEE table.

16) Populate the table with the last row of the sample data listed in step 3 by using the statements in the script that you created in step 6. Run the statements in the script.

17) Confirm your addition to the table.

		ID		LAST_NAME		FIRST_NAME		USERID		SALARY
1		1		Patel		Ralph		rpatel		1000
2		3		Drexler		Ben		bbiri		1100
3		4		Newman		Chad		cnewman		1000
4		5		Ropeburn		Audrey		aropebur		1550






18) Mark an intermediate point in the processing of the transaction.

19) Delete all the rows from the MY\_EMPLOYEE table.

20) Confirm that the table is empty.

21) Discard the most recent DELETE operation without discarding the earlier INSERT operation.

22) Confirm that the new row is still intact.

		ID		LAST_NAME		FIRST_NAME		USERID		SALARY
1		1		Patel		Ralph		rpatel		1000
2		3		Drexler		Ben		bbiri		1100
3		4		Newman		Chad		cnewman		1000
4		5		Ropeburn		Audrey		aropebur		1550

23) Make the data addition permanent.

If you have the time, complete the following exercise:

24) Modify the lab\_09\_06.sql script such that the USERID is generated automatically by concatenating the first letter of the first name and the first seven characters of the last name. The generated USERID must be in lowercase. Therefore, the script should not prompt for the USERID. Save this script to a file named lab\_09\_24.sql.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
6	Anthony	Mark	manthony	1230

25) Run the lab\_09\_24.sql script to insert the following record:

26) Confirm that the new row was added with correct USERID.

### ***Practice 9-1: Manipulating Data (continued)***

	 ID	 LAST_NAME	 FIRST_NAME	 USERID	 SALARY
1	6	Anthony	Mark	manthony	1230

## Practice Solutions 9-1: Manipulating Data

Insert data into the MY\_EMPLOYEE table.

- 1) Run the statement in the lab\_09\_01.sql script to build the MY\_EMPLOYEE table used in this practice.
  - a) From File menu, select Open. In the Open dialog box, navigate to the /home/oracle/labs/sql1/labs folder, and then double-click lab\_09\_01.sql.
  - b) After the statement is opened in a SQL Worksheet, click the Run Script icon to run the script. You get a Create Table succeeded message on the Script Output tabbed page.
- 2) Describe the structure of the MY\_EMPLOYEE table to identify the column names.

```
DESCRIBE my_employee
```

- 3) Create an INSERT statement to add the first row of data to the MY\_EMPLOYEE table from the following sample data. Do not list the columns in the INSERT clause.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audrey	aropebur	1550

```
INSERT INTO my_employee  
VALUES (1, 'Patel', 'Ralph', 'rpatel', 895);
```

- 4) Populate the MY\_EMPLOYEE table with the second row of the sample data from the preceding list. This time, list the columns explicitly in the INSERT clause.

```
INSERT INTO my_employee (id, last_name, first_name,  
                          userid, salary)  
VALUES (2, 'Dancs', 'Betty', 'bdancs', 860);
```

- 5) Confirm your additions to the table.

```
SELECT *  
FROM my_employee;
```

## **Practice Solutions 9-1: Manipulating Data (continued)**

- 6) Write an INSERT statement in a dynamic reusable script file to load the remaining rows into the MY\_EMPLOYEE table. The script should prompt for all the columns (ID, LAST\_NAME, FIRST\_NAME, USERID, and SALARY). Save this script to a file named lab\_09\_06.sql.

```
INSERT INTO my_employee
VALUES (&p_id, '&p_last_name', '&p_first_name',
       '&p_userid', &p_salary);
```

- 7) Populate the table with the next two rows of sample data listed in step 3 by running the INSERT statement in the script that you created.

```
INSERT INTO my_employee
VALUES (&p_id, '&p_last_name', '&p_first_name',
       '&p_userid', &p_salary);
```

- 8) Confirm your additions to the table.

```
SELECT  *
FROM    my_employee;
```

- 9) Make the data additions permanent.

```
COMMIT;
```

### **Update and delete data in the MY\_EMPLOYEE table.**

- 10) Change the last name of employee 3 to Drexler.

```
UPDATE  my_employee
SET      last_name = 'Drexler'
WHERE    id = 3;
```

- 11) Change the salary to \$1,000 for all employees with a salary less than \$900.

```
UPDATE  my_employee
SET      salary = 1000
WHERE    salary < 900;
```

- 12) Verify your changes to the table.

```
SELECT  *
FROM    my_employee;
```

- 13) Delete Betty Dancs from the MY\_EMPLOYEE table.

```
DELETE
FROM    my_employee
WHERE   last_name = 'Dancs';
```

- 14) Confirm your changes to the table.

```
SELECT  *
FROM    my_employee;
```

## ***Practice Solutions 9-1: Manipulating Data (continued)***

15) Commit all pending changes.

```
COMMIT;
```

**Control data transaction to the MY\_EMPLOYEE table.**

16) Populate the table with the last row of the sample data listed in step 3 by using the statements in the script that you created in step 6. Run the statements in the script.

```
INSERT INTO my_employee  
VALUES (&p_id, '&p_last_name', '&p_first_name',  
        '&p_userid', &p_salary);
```

17) Confirm your addition to the table.

```
SELECT *  
FROM   my_employee;
```

18) Mark an intermediate point in the processing of the transaction.

```
SAVEPOINT step_17;
```

19) Delete all the rows from the MY\_EMPLOYEE table.

```
DELETE  
FROM   my_employee;
```

20) Confirm that the table is empty.

```
SELECT *  
FROM   my_employee;
```

21) Discard the most recent DELETE operation without discarding the earlier INSERT operation.

```
ROLLBACK TO step_17;
```

22) Confirm that the new row is still intact.

```
SELECT *  
FROM   my_employee;
```

23) Make the data addition permanent.

```
COMMIT;
```

## Practice Solutions 9-1: Manipulating Data (continued)

If you have time, complete the following exercise:

- 24) Modify the lab\_09\_06.sql script such that the USERID is generated automatically by concatenating the first letter of the first name and the first seven characters of the last name. The generated USERID must be in lowercase. Therefore, the script should not prompt for the USERID. Save this script to a file named lab\_09\_24.sql.

```
SET ECHO OFF
SET VERIFY OFF
INSERT INTO my_employee
VALUES (&p_id, '&p_last_name', '&p_first_name',
       lower(substr('&p_first_name', 1, 1) ||
       substr('&p_last_name', 1, 7)), &p_salary);
SET VERIFY ON
SET ECHO ON
UNDEFINE p_first_name
UNDEFINE p_last_name
```

- 25) Run the lab\_09\_24.sql script to insert the following record:

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
6	Anthony	Mark	manthony	1230

- 26) Confirm that the new row was added with the correct USERID.

```
SELECT *
FROM my_employee
WHERE ID='6';
```



## Practices for Lesson 10

Create new tables by using the `CREATE TABLE` statement. Confirm that the new table was added to the database. You also learn to set the status of a table as `READ ONLY` and then revert to `READ/WRITE`.

**Note:** For all the DDL and DML statements, click the Run Script icon (or press [F5]) to execute the query in SQL Developer. This way you get to see the feedback messages on the Script Output tabbed page. For `SELECT` queries, continue to click the Execute Statement icon or press [F9] to get the formatted output on the Results tabbed page.

## Practice 10-1: Using DDL Statements to Create and Manage Tables

- 1) Create the DEPT table based on the following table instance chart. Save the statement in a script called lab\_10\_01.sql, and then execute the statement in the script to create the table. Confirm that the table is created.

<b>Column Name</b>	ID	NAME
<b>Key Type</b>	Primary key	
<b>Nulls/Unique</b>		
<b>FK Table</b>		
<b>FK Column</b>		
<b>Data type</b>	NUMBER	VARCHAR2
<b>Length</b>	7	25

Name	Null	Type
-----	-----	-----
ID	NOT NULL	NUMBER(7)
NAME		VARCHAR2(25)

- 2) Populate the DEPT table with data from the DEPARTMENTS table. Include only columns that you need.
- 3) Create the EMP table based on the following table instance chart. Save the statement in a script called lab\_10\_03.sql, and then execute the statement in the script to create the table. Confirm that the table is created.

<b>Column Name</b>	ID	LAST_NAME	FIRST_NAME	DEPT_ID
<b>Key Type</b>				
<b>Nulls/Unique</b>				
<b>FK Table</b>				DEPT
<b>FK Column</b>				ID
<b>Data type</b>	NUMBER	VARCHAR2	VARCHAR2	NUMBER
<b>Length</b>	7	25	25	7

Name	Null	Type
-----	-----	-----
ID		NUMBER(7)
LAST_NAME		VARCHAR2(25)
FIRST_NAME		VARCHAR2(25)
DEPT_ID		NUMBER(7)

## ***Practice 10-1: Using DDL Statements to Create and Manage Tables (continued)***

- 4) Create the EMPLOYEES2 table based on the structure of the EMPLOYEES table. Include only the EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, SALARY, and DEPARTMENT\_ID columns. Name the columns in your new table ID, FIRST\_NAME, LAST\_NAME, SALARY, and DEPT\_ID, respectively.
- 5) Alter the EMPLOYEES2 table status to read-only.
- 6) Try to insert the following row in the EMPLOYEES2 table:

ID	FIRST_NAME	LAST_NAME	SALARY	DEPT_ID
34	Grant	Marcie	5678	10

You get the following error message:

```
Error starting at line 1 in command:
INSERT INTO employees2
VALUES (34, 'Grant','Marcie',5678,10)
Error at Command Line:1 Column:12
Error report:
SQL Error: ORA-12081: update operation not allowed on table "ORA1"."EMPLOYEES2"
12081. 00000 - "update operation not allowed on table \"%s\".\"%s\""
*Cause:      An attempt was made to update a read-only materialized view.
*Action:     No action required. Only Oracle is allowed to update a
              read-only materialized view.
```

- 7) Revert the EMPLOYEES2 table to the read/write status. Now, try to insert the same row again. You should get the following messages:

```
ALTER TABLE employees2 succeeded.
1 rows inserted
```

- 8) Drop the EMPLOYEES2 table.

## Practice Solutions 10-1: Using DDL Statements to Create and Manage Tables

- 1) Create the DEPT table based on the following table instance chart. Save the statement in a script called lab\_10\_01.sql, and then execute the statement in the script to create the table. Confirm that the table is created.

Column Name	ID	NAME
Key Type	Primary key	
Nulls/Unique		
FK Table		
FK Column		
Data type	NUMBER	VARCHAR2
Length	7	25

```
CREATE TABLE dept
(id    NUMBER(7) CONSTRAINT department_id_pk PRIMARY KEY,
name  VARCHAR2(25));
```

To confirm that the table was created and to view its structure, issue the following command:

```
DESCRIBE dept
```

- 2) Populate the DEPT table with data from the DEPARTMENTS table. Include only those columns that you need.

```
INSERT INTO dept
SELECT department_id, department_name
FROM departments;
```

- 3) Create the EMP table based on the following table instance chart. Save the statement in a script called lab\_10\_03.sql, and then execute the statement in the script to create the table. Confirm that the table is created.

Column Name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Key Type				
Nulls/Unique				
FK Table				DEPT
FK Column				ID
Data type	NUMBER	VARCHAR2	VARCHAR2	NUMBER
Length	7	25	25	7

## Practice Solutions 10-1: Using DDL Statements to Create and Manage Tables (continued)

```
CREATE TABLE emp
(id          NUMBER(7),
last_name    VARCHAR2(25),
first_name   VARCHAR2(25),
dept_id      NUMBER(7)
CONSTRAINT emp_dept_id_FK REFERENCES dept (id)
);
```

To confirm that the table was created and to view its structure:

```
DESCRIBE emp
```

- 4) Create the EMPLOYEES2 table based on the structure of the EMPLOYEES table. Include only the EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, SALARY, and DEPARTMENT\_ID columns. Name the columns in your new table ID, FIRST\_NAME, LAST\_NAME, SALARY, and DEPT\_ID, respectively.

```
CREATE TABLE employees2 AS
SELECT employee_id id, first_name, last_name, salary,
       department_id dept_id
FROM   employees;
```

- 5) Alter the EMPLOYEES2 table status to read-only.

```
ALTER TABLE employees2 READ ONLY
```

- 6) Try to insert the following row in the EMPLOYEES2 table.

ID	FIRST_NAME	LAST_NAME	SALARY	DEPT_ID
34	Grant	Marcie	5678	10

Note, you will get the “Update operation not allowed on table” error message. Therefore, you will not be allowed to insert any row into the table because it is assigned a read-only status.

```
INSERT INTO employees2
VALUES (34, 'Grant', 'Marcie', 5678, 10)
```

- 7) Revert the EMPLOYEES2 table to the read/write status. Now try to insert the same row again.

Now, because the table is assigned a READ WRITE status, you will be allowed to insert a row into the table.

```
ALTER TABLE employees2 READ WRITE

INSERT INTO employees2
VALUES (34, 'Grant', 'Marcie', 5678, 10)
```

## ***Practice Solutions 10-1: Using DDL Statements to Create and Manage Tables (continued)***

8) Drop the EMPLOYEES2 table.

**Note:** You can even drop a table that is in the READ ONLY mode. To test this, alter the table again to READ ONLY status, and then issue the DROP TABLE command. The table EMPLOYEES2 will be dropped.

```
DROP TABLE employees2;
```

---

## Practices for Lesson 11

---

Part 1 of this lesson's practice provides you with a variety of exercises in creating, using, and removing views. Complete questions 1–6 of this lesson.

Part 2 of this lesson's practice provides you with a variety of exercises in creating and using a sequence, an index, and a synonym. Complete questions 7–10 of this lesson.

## Practice 11-1: Creating Other Schema Objects

### Part 1

- 1) The staff in the HR department wants to hide some of the data in the EMPLOYEES table. Create a view called EMPLOYEES\_VU based on the employee numbers, employee last names, and department numbers from the EMPLOYEES table. The heading for the employee name should be EMPLOYEE.
- 2) Confirm that the view works. Display the contents of the EMPLOYEES\_VU view.

	EMPLOYEE_ID	EMPLOYEE	DEPARTMENT_ID
1	200	Whalen	10
2	201	Hartstein	20
3	202	Fay	20
4	205	Higgins	110
5	206	Gietz	110

...

19	205	Higgins	110
20	206	Gietz	110

- 3) Using your EMPLOYEES\_VU view, write a query for the HR department to display all employee names and department numbers.

	EMPLOYEE	DEPARTMENT_ID
1	King	90
2	Kochhar	90
3	De Haan	90
4	Hunold	60
5	Ernst	60

...

19	Higgins	110
20	Gietz	110

- 4) Department 50 needs access to its employee data. Create a view named DEPT50 that contains the employee numbers, employee last names, and department numbers for all employees in department 50. You have been asked to label the view columns EMPNO, EMPLOYEE, and DEPTNO. For security purposes, do not allow an employee to be reassigned to another department through the view.
- 5) Display the structure and contents of the DEPT50 view.

DESCRIBE dept50		
Name	Null	Type
-----		
EMPNO	NOT NULL	NUMBER(6)
EMPLOYEE	NOT NULL	VARCHAR2(25)
DEPTNO		NUMBER(4)



### ***Practice 11-1: Creating Other Schema Objects (continued)***

EMPNO	EMPLOYEE	DEPTNO
124	Mourgos	50
141	Rajs	50
142	Davies	50
143	Matos	50
144	Vargas	50

- 6) Test your view. Attempt to reassign Matos to department 80.

#### **Part 2**

- 7) You need a sequence that can be used with the `PRIMARY KEY` column of the `DEPT` table. The sequence should start at 200 and have a maximum value of 1,000. Have your sequence increment by 10. Name the sequence `DEPT_ID_SEQ`.
- 8) To test your sequence, write a script to insert two rows in the `DEPT` table. Name your script `lab_11_08.sql`. Be sure to use the sequence that you created for the `ID` column. Add two departments: Education and Administration. Confirm your additions. Run the commands in your script.
- 9) Create a nonunique index on the `NAME` column in the `DEPT` table.
- 10) Create a synonym for your `EMPLOYEES` table. Call it `EMP`.

## Practice Solutions 11-1: Creating Other Schema Objects

### Part 1

- 1) The staff in the HR department wants to hide some of the data in the EMPLOYEES table. Create a view called EMPLOYEES\_VU based on the employee numbers, employee last names, and department numbers from the EMPLOYEES table. The heading for the employee name should be EMPLOYEE.

```
CREATE OR REPLACE VIEW employees_vu AS
  SELECT employee_id, last_name employee, department_id
  FROM employees;
```

- 2) Confirm that the view works. Display the contents of the EMPLOYEES\_VU view.

```
SELECT *
FROM   employees_vu;
```

- 3) Using your EMPLOYEES\_VU view, write a query for the HR department to display all employee names and department numbers.

```
SELECT employee, department_id
FROM   employees_vu;
```

- 4) Department 50 needs access to its employee data. Create a view named DEPT50 that contains the employee numbers, employee last names, and department numbers for all employees in department 50. They have requested that you label the view columns EMPNO, EMPLOYEE, and DEPTNO. For security purposes, do not allow an employee to be reassigned to another department through the view.

```
CREATE VIEW dept50 AS
  SELECT employee_id empno, last_name employee,
         department_id deptno
  FROM   employees
 WHERE  department_id = 50
 WITH CHECK OPTION CONSTRAINT emp_dept_50;
```

- 5) Display the structure and contents of the DEPT50 view.

```
DESCRIBE dept50

SELECT *
FROM   dept50;
```

- 6) Test your view. Attempt to reassign Matos to department 80.

```
UPDATE dept50
SET    deptno = 80
WHERE  employee = 'Matos';
```

The error is because the DEPT50 view has been created with the WITH CHECK OPTION constraint. This ensures that the DEPTNO column in the view is protected from being changed.

## ***Practice Solutions 11-1: Creating Other Schema Objects (continued)***

### **Part 2**

- 7) You need a sequence that can be used with the primary key column of the DEPT table. The sequence should start at 200 and have a maximum value of 1,000. Have your sequence increment by 10. Name the sequence DEPT\_ID\_SEQ.

```
CREATE SEQUENCE dept_id_seq
  START WITH 200
  INCREMENT BY 10
  MAXVALUE 1000;
```

- 8) To test your sequence, write a script to insert two rows in the DEPT table. Name your script lab\_11\_08.sql. Be sure to use the sequence that you created for the ID column. Add two departments: Education and Administration. Confirm your additions. Run the commands in your script.

```
INSERT INTO dept
VALUES (dept_id_seq.nextval, 'Education');

INSERT INTO dept
VALUES (dept_id_seq.nextval, 'Administration');
```

- 9) Create a nonunique index on the NAME column in the DEPT table.

```
CREATE INDEX dept_name_idx ON dept (name);
```

- 10) Create a synonym for your EMPLOYEES table. Call it EMP.

```
CREATE SYNONYM emp FOR EMPLOYEES;
```

---

## Practices for Appendix F

---

This practice is intended to give you practical experience in extracting data from more than one table using the Oracle join syntax.

## Practice F-1: Oracle Join Syntax

- 1) Write a query for the HR department to produce the addresses of all the departments. Use the LOCATIONS and COUNTRIES tables. Show the location ID, street address, city, state or province, and country in the output. Run the query.

	LOCATION_ID	STREET_ADDRESS	CITY	STATE_PROVINCE	COUNTRY_NAME
1	1400	2014 Jabberwocky Rd	Southlake	Texas	United States of America
2	1500	2011 Interiors Blvd	South San Francisco	California	United States of America
3	1700	2004 Charade Rd	Seattle	Washington	United States of America
4	1800	460 Bloor St. W.	Toronto	Ontario	Canada
5	2500	Magdalen Centre, The Oxford Science Park	Oxford	Oxford	United Kingdom

- 2) The HR department needs a report of all employees. Write a query to display the last name, department number, and department name for all employees. Run the query.

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Davies	50	Shipping
5	Vargas	50	Shipping

...

18	Higgins	110	Accounting
19	Gietz	110	Accounting

- 3) The HR department needs a report of employees in Toronto. Display the last name, job, department number, and department name for all employees who work in Toronto.

	LAST_NAME	JOB_ID	DEPARTMENT_ID	DEPARTMENT_NAME
1	Hartstein	MK_MAN	20	Marketing
2	Fay	MK_REP	20	Marketing

- 4) Create a report to display the employees' last names and employee number along with their managers' last names and manager number. Label the columns Employee, Emp#, Manager, and Mgr#, respectively. Save your SQL statement as lab\_f\_04.sql.

	Employee	EMP#	Manager	Mgr#
1	Hunold	103	De Haan	102
2	Fay	202	Hartstein	201
3	Gietz	206	Higgins	205
4	Lorentz	107	Hunold	103
5	Ernst	104	Hunold	103

...

18	Taylor	176	Zlotkey	149
19	Abel	174	Zlotkey	149

## Practice F-1: Oracle Join Syntax (continued)

- 5) Modify `lab_f_04.sql` to display all employees including King, who has no manager. Order the results by the employee number. Save your SQL statement as `lab_f_05.sql`. Run the query in `lab_f_05.sql`.

	Employee	EMP#	Manager	Mgr#
1	Hunold	103	De Haan	102
2	Fay	202	Hartstein	201
3	Gietz	206	Higgins	205
4	Lorentz	107	Hunold	103
5	Ernst	104	Hunold	103

...

19	Abel	174	Zlotkey	149
20	King	100	(null)	(null)

- 6) Create a report for the HR department that displays employee last names, department numbers, and all employees who work in the same department as a given employee. Give each column an appropriate label. Save the script to a file named `lab_f_06.sql`.

	DEPARTMENT	EMPLOYEE	COLLEAGUE
1	20	Fay	Hartstein
2	20	Hartstein	Fay
3	50	Davies	Matos
4	50	Davies	Mourgos
5	50	Davies	Rajs

...

39	90	Kochhar	De Haan
40	90	Kochhar	King
41	110	Gietz	Higgins
42	110	Higgins	Gietz

- 7) The HR department needs a report on job grades and salaries. To familiarize yourself with the `JOB_GRADES` table, first show the structure of the `JOB_GRADES` table. Then create a query that displays the name, job, department name, salary, and grade for all employees.

Name	Null	Type
-----	-----	-----
GRADE_LEVEL		VARCHAR2(3)
LOWEST_SAL		NUMBER
HIGHEST_SAL		NUMBER

## Practice F-1: Oracle Join Syntax (continued)

	LAST_NAME	JOB_ID	DEPARTMENT_NAME	SALARY	GRADE_LEVEL
1	King	AD_PRES	Executive	24000	E
2	De Haan	AD_VP	Executive	17000	E
3	Kochhar	AD_VP	Executive	17000	E
4	Hartstein	MK_MAN	Marketing	13000	D
5	Higgins	AC_MGR	Accounting	12000	D

...

18	Matos	ST_CLERK	Shipping	2600	A
19	Vargas	ST_CLERK	Shipping	2500	A

If you want an extra challenge, complete the following exercises:

- 8) The HR department wants to determine the names of all employees who were hired after Davies. Create a query to display the name and hire date of any employee hired after employee Davies.

	LAST_NAME	HIRE_DATE
1	Lorentz	07-FEB-99
2	Mourgos	16-NOV-99
3	Matos	15-MAR-98
4	Vargas	09-JUL-98
5	Zlotkey	29-JAN-00
6	Taylor	24-MAR-98
7	Grant	24-MAY-99
8	Fay	17-AUG-97

- 9) The HR department needs to find the names and hire dates for all employees who were hired before their managers, along with their managers' names and hire dates. Save the script to a file named lab\_f\_09.sql.

	LAST_NAME	HIRE_DATE	LAST_NAME_1	HIRE_DATE_1
1	Whalen	17-SEP-87	Kochhar	21-SEP-89
2	Hunold	03-JAN-90	De Haan	13-JAN-93
3	Vargas	09-JUL-98	Mourgos	16-NOV-99
4	Matos	15-MAR-98	Mourgos	16-NOV-99
5	Davies	29-JAN-97	Mourgos	16-NOV-99
6	Rajs	17-OCT-95	Mourgos	16-NOV-99
7	Grant	24-MAY-99	Zlotkey	29-JAN-00
8	Taylor	24-MAR-98	Zlotkey	29-JAN-00
9	Abel	11-MAY-96	Zlotkey	29-JAN-00

## Practice Solutions F-1: Oracle Join Syntax

- 1) Write a query for the HR department to produce the addresses of all the departments. Use the LOCATIONS and COUNTRIES tables. Show the location ID, street address, city, state or province, and country in the output. Run the query.

```
SELECT location_id, street_address, city, state_province,
country_name
FROM   locations, countries
WHERE  locations.country_id = countries.country_id;
```

- 2) The HR department needs a report of all employees. Write a query to display the last name, department number, and department name for all employees. Run the query.

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e, departments d
WHERE  e.department_id = d.department_id;
```

- 3) The HR department needs a report of employees in Toronto. Display the last name, job, department number, and department name for all employees who work in Toronto.

```
SELECT e.last_name, e.job_id, e.department_id,
d.department_name
FROM   employees e, departments d , locations l
WHERE  e.department_id = d.department_id
AND    d.location_id = l.location_id
AND    LOWER(l.city) = 'toronto';
```

- 4) Create a report to display the employee last name and the employee number along with the last name of the employee's manager and manager number. Label the columns Employee, Emp#, Manager, and Mgr#, respectively. Save your SQL statement as lab\_f\_04.sql.

```
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id  "Mgr#"
FROM   employees w, employees m
WHERE  w.manager_id = m.employee_id;
```

- 5) Modify lab\_f\_04.sql to display all employees including King, who has no manager. Order the results by the employee number. Save the SQL statement as lab\_f\_05.sql. Run the query in lab\_f\_05.sql.

```
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id  "Mgr#"
FROM   employees w, employees m
WHERE  w.manager_id = m.employee_id (+);
```



## **Practice Solutions F-1: Oracle Join Syntax (continued)**

- 6) Create a report for the HR department that displays employee last names, department numbers, and all the employees who work in the same department as a given employee. Give each column an appropriate label. Save the script to a file named lab\_f\_06.sql.

```
SELECT e1.department_id department, e1.last_name employee,  
       e2.last_name colleague  
FROM   employees e1, employees e2  
WHERE  e1.department_id = e2.department_id  
AND    e1.employee_id <> e2.employee_id  
ORDER BY e1.department_id, e1.last_name, e2.last_name;
```

- 7) The HR department needs a report on job grades and salaries. To familiarize yourself with the JOB\_GRADES table, first show the structure of the JOB\_GRADES table. Then create a query that displays the name, job, department name, salary, and grade for all employees.

```
DESC JOB_GRADES  
  
SELECT e.last_name, e.job_id, d.department_name,  
       e.salary, j.grade_level  
FROM   employees e, departments d, job_grades j  
WHERE  e.department_id = d.department_id  
AND    e.salary BETWEEN j.lowest_sal AND j.highest_sal;
```

If you want an extra challenge, complete the following exercises:

- 8) The HR department wants to determine the names of all employees hired after Davies. Create a query to display the name and hire date of any employee hired after Davies.

```
SELECT e.last_name, e.hire_date  
FROM   employees e , employees davies  
WHERE  davies.last_name = 'Davies'  
AND    davies.hire_date < e.hire_date;
```

- 9) The HR department needs to find the names and hire dates for all employees who were hired before their managers, along with their managers' names and hire dates. Label the columns Employee, Emp Hired, Manager, and Mgr Hired, respectively. Save the script to a file named lab\_f\_09.sql.

```
SELECT w.last_name, w.hire_date, m.last_name, m.hire_date  
FROM   employees w , employees m  
WHERE  w.manager_id = m.employee_id  
AND    w.hire_date < m.hire_date;
```

---

# **Appendix A**

## **Practices and Solutions**

---

## Table of Contents

Practices and Solutions for Lesson I .....	3
Practice I-1: Accessing SQL Developer Resources .....	4
Practice I-2: Using SQL Developer .....	5
Practice Solutions I-1: Accessing SQL Developer Resources .....	7
Practice Solutions I-2: Using SQL Developer .....	8
Practices and Solutions for Lesson 1 .....	17
Practice 1-1: Controlling User Access .....	17
Practice Solutions 1-1: Controlling User Access .....	20
Practices and Solutions for Lesson 2 .....	25
Practice 2-1: Managing Schema Objects .....	25
Practice Solutions 2-1: Managing Schema Objects .....	31
Practices and Solutions for Lesson 3 .....	39
Practice 3-1: Managing Objects with Data Dictionary Views .....	39
Practice Solutions 3-1: Managing Objects with Data Dictionary Views .....	43
Practices and Solutions for Lesson 4 .....	47
Practice 4-1: Manipulating Large Data Sets .....	47
Practice Solutions 4-1: Manipulating Large Data Sets .....	51
Practices and Solutions for Lesson 5 .....	56
Practice 5-1: Managing Data in Different Time Zones .....	56
Practice Solutions 5-1: Managing Data in Different Time Zones .....	59
Practices and Solutions for Lesson 6 .....	62
Practice 6-1: Retrieving Data by Using Subqueries .....	62
Practice Solutions 6-1: Retrieving Data by Using Subqueries .....	66
Practices and Solutions for Lesson 7 .....	70
Practice 7-1: Regular Expression Support .....	70
Practice Solutions 7-1: Regular Expression Support .....	72

## **Practices and Solutions for Lesson I**

In this practice, you review the available SQL Developer resources. You also learn about your user account that you use in this course. You then start SQL Developer, create a new database connection, and browse your HR tables. You also set some SQL Developer preferences, execute SQL statements, and execute an anonymous PL/SQL block by using SQL Worksheet. Finally, you access and bookmark the Oracle Database 11g documentation and other useful Web sites that you can use in this course.

## ***Practice I-1: Accessing SQL Developer Resources***

In this practice, you do the following:

- 1) Access the SQL Developer home page.
  - a. Access the online SQL Developer home page available at:  
[http://www.oracle.com/technology/products/database/sql\\_developer/index.html](http://www.oracle.com/technology/products/database/sql_developer/index.html)
  - b. Bookmark the page for easier future access.
- 2) Access the SQL Developer tutorial available online at:  
<http://st-curriculum.oracle.com/tutorial/SQLDeveloper/index.htm>. Then review the following sections and associated demos:
  - a) What to Do First
  - b) Working with Database Objects
  - c) Accessing Data

## ***Practice I-2: Using SQL Developer***

- 1) Start SQL Developer by using the desktop icon.
- 2) Create a database connection using the following information:
  - a) Connection Name: myconnection
  - b) Username: oraxx, where xx is the number of your PC (Ask your instructor to assign you an ora account out of the ora21-ora40 range of accounts.)
  - c) Password: oraxx
  - d) Hostname: localhost
  - e) Port: 1521
  - f) SID: orcl (or the value provided to you by the instructor)
- 3) Test the new connection. If the status is Success, connect to the database by using this new connection.
  - a) Click the Test button in the New/Select Database Connection window.
  - b) If the status is Success, click the Connect button.
- 4) Browse the structure of the EMPLOYEES table and display its data.
  - a) Expand the myconnection connection by clicking the plus sign next to it.
  - b) Expand the Tables icon by clicking the plus sign next to it.
  - c) Display the structure of the EMPLOYEES table.
  - d) View the data of the DEPARTMENTS table.
- 5) Execute some basic SELECT statements to query the data in the EMPLOYEES table in the SQL Worksheet area. Use both the Execute Statement (or press F9) and the Run Script (or press F5) icons to execute the SELECT statements. Review the results of both methods of executing the SELECT statements on the appropriate tabbed pages.
  - a) Write a query to select the last name and salary for any employee whose salary is less than or equal to \$3,000.
  - b) Write a query to display last name, job ID, and commission for all employees who are not entitled to receive a commission.
- 6) Set your script pathing preference to /home/oracle/labs/sql2.
  - a) Select Tools > Preferences > Database > Worksheet Parameters.
  - b) Enter the value in the Select default path to look for scripts field.
- 7) Enter the following in the Enter SQL Statement box.

```
SELECT employee_id, first_name, last_name,  
       FROM employees;
```
- 8) Save the SQL statement to a script file by using the File > Save As menu item.
  - a) Select File > Save As.
  - b) Name the file intro\_test.sql.

### ***Practice I-2: Using SQL Developer (continued)***

- c) Place the file under your /home/oracle/labs/sql2/labs folder.
- 9) Open and run `confidence.sql` from your /home/oracle/labs/sql2/labs folder, and observe the output.

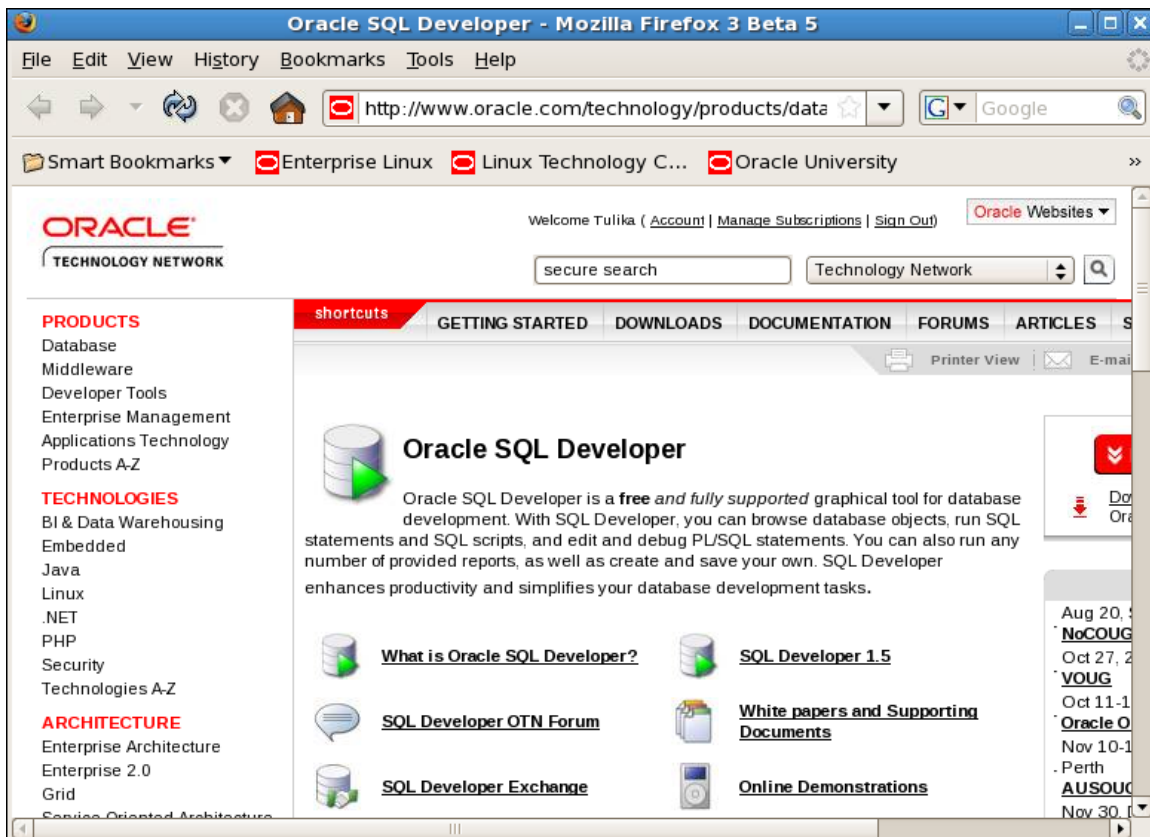
## Practice Solutions I-1: Accessing SQL Developer Resources

1) Access the SQL Developer home page.

a) Access the online SQL Developer home page available online at:

[http://www.oracle.com/technology/products/database/sql\\_developer/index.html](http://www.oracle.com/technology/products/database/sql_developer/index.html)

The SQL Developer home page is displayed as follows:



b) Bookmark the page for easier future access.

2) Access the SQL Developer tutorial available online at:

<http://st-curriculum.oracle.com/tutorial/SQLDeveloper/index.htm>

Then, review the following sections and associated demos:

a) What to Do First

b) Working with Database Objects

c) Accessing Data



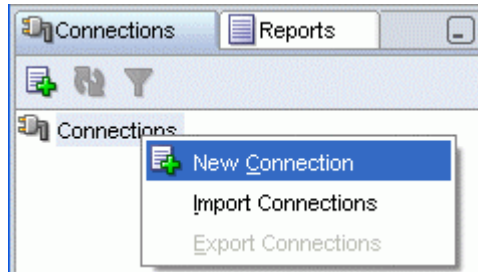
## ***Practice Solutions I-2: Using SQL Developer***

1) Start SQL Developer by using the desktop icon.



2) Create a database connection using the following information:

- a. Connection Name: myconnection
- b. Username: oraxx (Ask your instructor to assign you one ora account out of the ora21–ora40 range of accounts.)
- c. Password: oraxx
- d. Hostname: localhost
- e. Port: 1521
- f. SID: orcl (or the value provided to you by the instructor)



## Practice Solutions I-2: Using SQL Developer (continued)

Connection Name: myconnection  
Username: ora21  
Password: \*\*\*\*\*  
☒ Save Password  
Oracle  
Role: default  
Connection Type: Basic  
☐ OS Authentication  
☐ Kerberos Authentication  
☐ Proxy Connection  
Hostname: localhost  
Port: 1521  
☒ SID: orcl  
☐ Service name:   
Status :  
Buttons: Help, Save, Clear, Test, Connect, Cancel

3) Test the new connection. If the status is Success, connect to the database by using this new connection.

a) Click the Test button in the New/Select Database Connection window.

Status :  
Buttons: Help, Save, Clear, Test, Connect, Cancel

b) If the status is Success, click the Connect button.

Status : Success  
Buttons: Help, Save, Clear, Test, Connect, Cancel

### Browsing the Tables

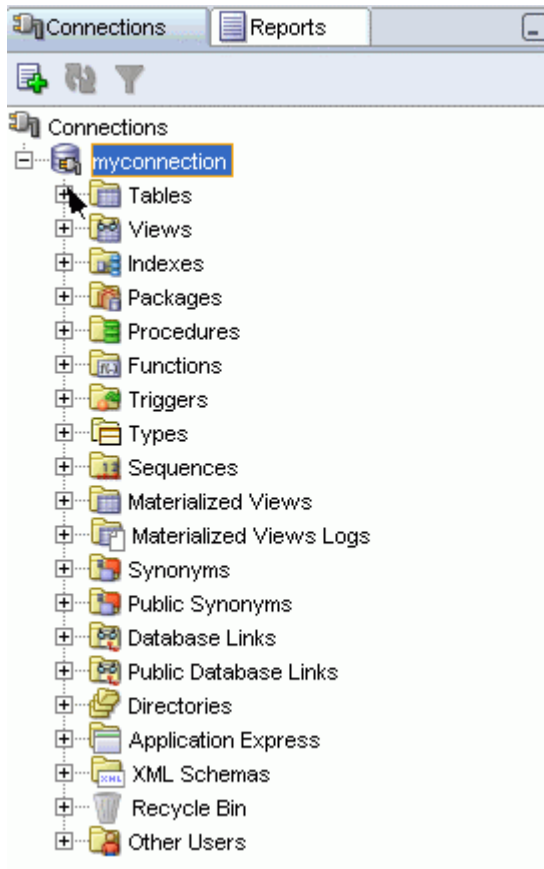
4) Browse the structure of the EMPLOYEES table and display its data.

a) Expand the myconnection connection by clicking the plus sign next to it.

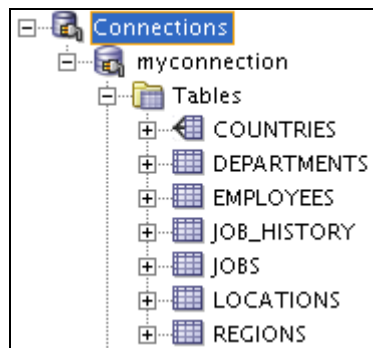
## Practice Solutions I-2: Using SQL Developer (continued)



b) Expand the Tables icon by clicking the plus sign next to it.



## Practice Solutions I-2: Using SQL Developer (continued)






c) Display the structure of the EMPLOYEES table.

Click the **EMPLOYEES** table. The Columns tab displays the columns in the EMPLOYEES table as follows:

myconnection

EMPLOYEES

Columns | Data | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Indexes | SQL

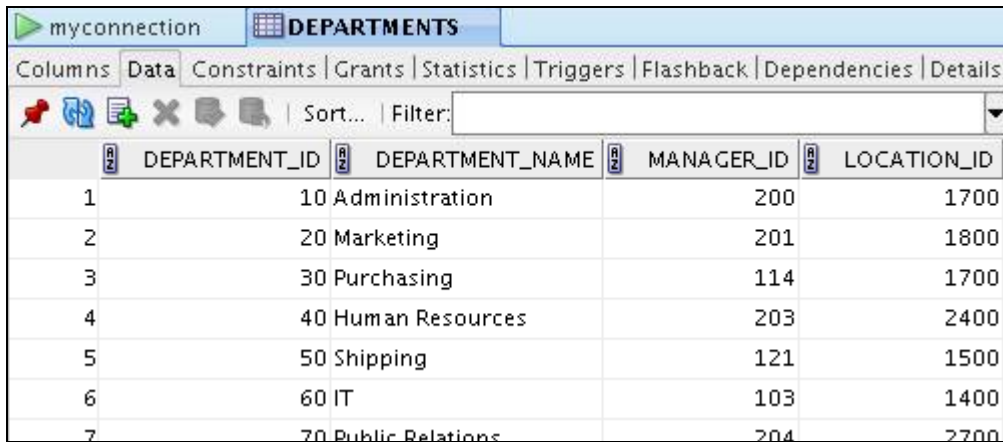
   Actions...

Column Name	Data Type	Nullable	Data Default	COLUMN ID	Primary Key	COMMENTS
EMPLOYEE_ID	NUMBER(6,0)	No	(null)	1	1	Primary key of employee
FIRST_NAME	VARCHAR2(20 BYTE)	Yes	(null)	2	(null)	First name of the employee
LAST_NAME	VARCHAR2(25 BYTE)	No	(null)	3	(null)	Last name of the employee
EMAIL	VARCHAR2(25 BYTE)	No	(null)	4	(null)	Email id of the employee
PHONE_NUMBER	VARCHAR2(20 BYTE)	Yes	(null)	5	(null)	Phone number of the employee
HIRE_DATE	DATE	No	(null)	6	(null)	Date when the employee was hired
JOB_ID	VARCHAR2(10 BYTE)	No	(null)	7	(null)	Current job of the employee
SALARY	NUMBER(8,2)	Yes	(null)	8	(null)	Monthly salary of the employee
COMMISSION_PCT	NUMBER(2,2)	Yes	(null)	9	(null)	Commission percentage of the employee
MANAGER_ID	NUMBER(6,0)	Yes	(null)	10	(null)	Manager id of the employee
DEPARTMENT_ID	NUMBER(4,0)	Yes	(null)	11	(null)	Department id where employee works

d) View the data of the DEPARTMENTS table.

In the Connections navigator, click the **DEPARTMENTS** table. Then click the Data tab.

## Practice Solutions I-2: Using SQL Developer (continued)



The screenshot shows the SQL Developer interface with the 'DEPARTMENTS' table selected. The 'Data' tab is active, displaying a grid of data. The columns are DEPARTMENT\_ID, DEPARTMENT\_NAME, MANAGER\_ID, and LOCATION\_ID. The data is as follows:

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	30	Purchasing	114	1700
4	40	Human Resources	203	2400
5	50	Shipping	121	1500
6	60	IT	103	1400
7	70	Public Relations	204	2700

- 5) Execute some basic `SELECT` statements to query the data in the `EMPLOYEES` table in the SQL Worksheet area. Use both the Execute Statement (or press F9) and the Run Script icons (or press F5) to execute the `SELECT` statements. Review the results of both methods of executing the `SELECT` statements on the appropriate tabbed pages.

- a) Write a query to select the last name and salary for any employee whose salary is less than or equal to \$3,000.

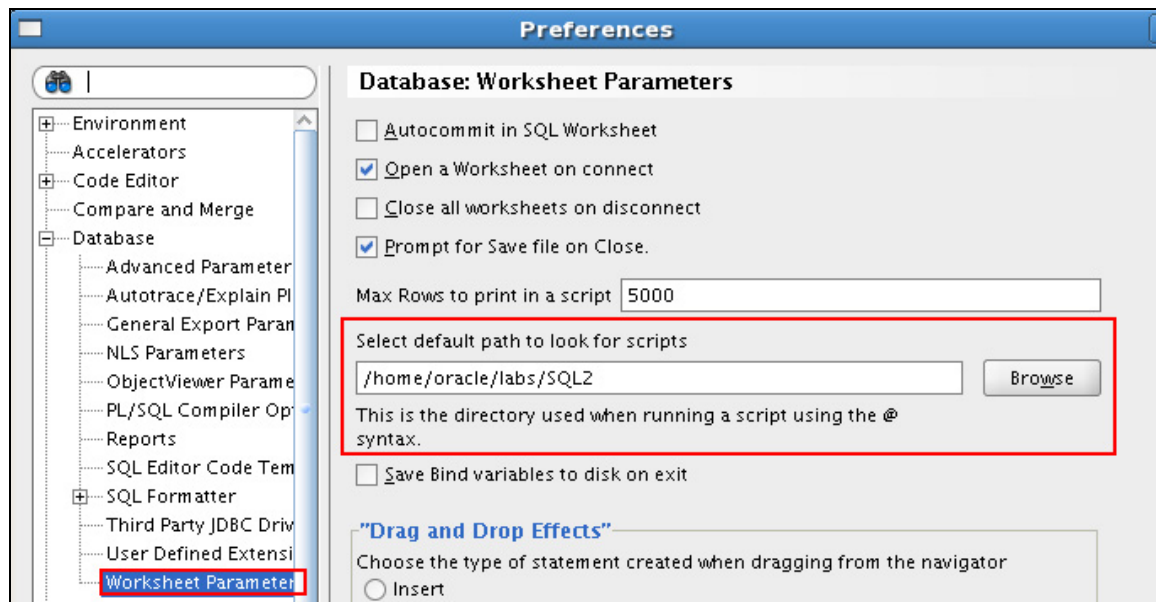
```
SELECT last_name, salary
FROM employees
WHERE salary <= 3000;
```

- b) Write a query to display last name, job ID, and commission for all employees who are not entitled to receive a commission.

```
SELECT last_name, job_id, commission_pct
FROM employees
WHERE commission_pct IS NULL;
```

- 6) Set your script pathing preference to `/home/oracle/labs/sql2`.
- a) Select `Tools > Preferences > Database > Worksheet Parameters`.
- b) Enter the value in the **Select default path to look for scripts** field. Then, click OK.

## Practice Solutions I-2: Using SQL Developer (continued)

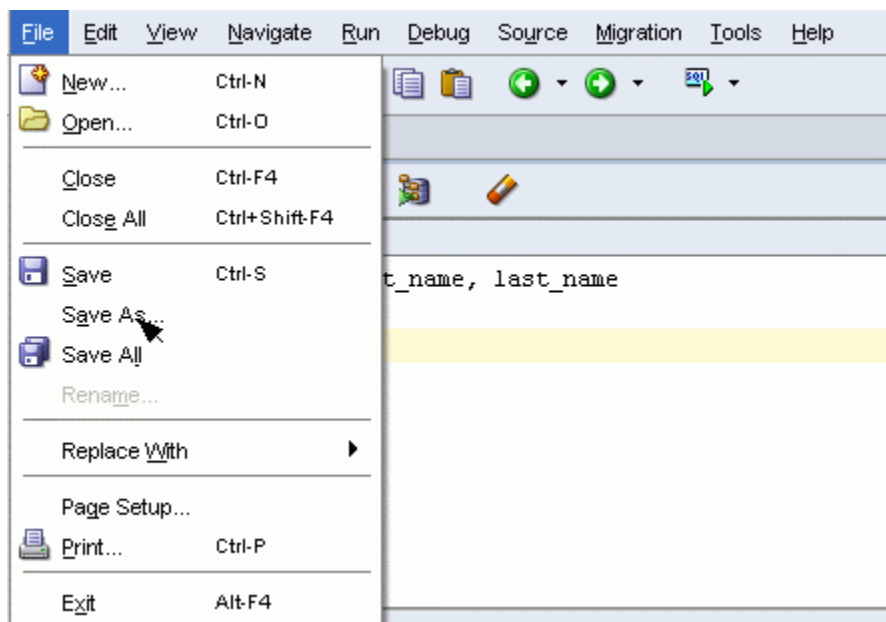


7) Enter the following SQL statement:

```
SELECT employee_id, first_name, last_name
FROM employees;
```

8) Save the SQL statement to a script file by using the File > Save As menu item.

a) Select File > Save As.

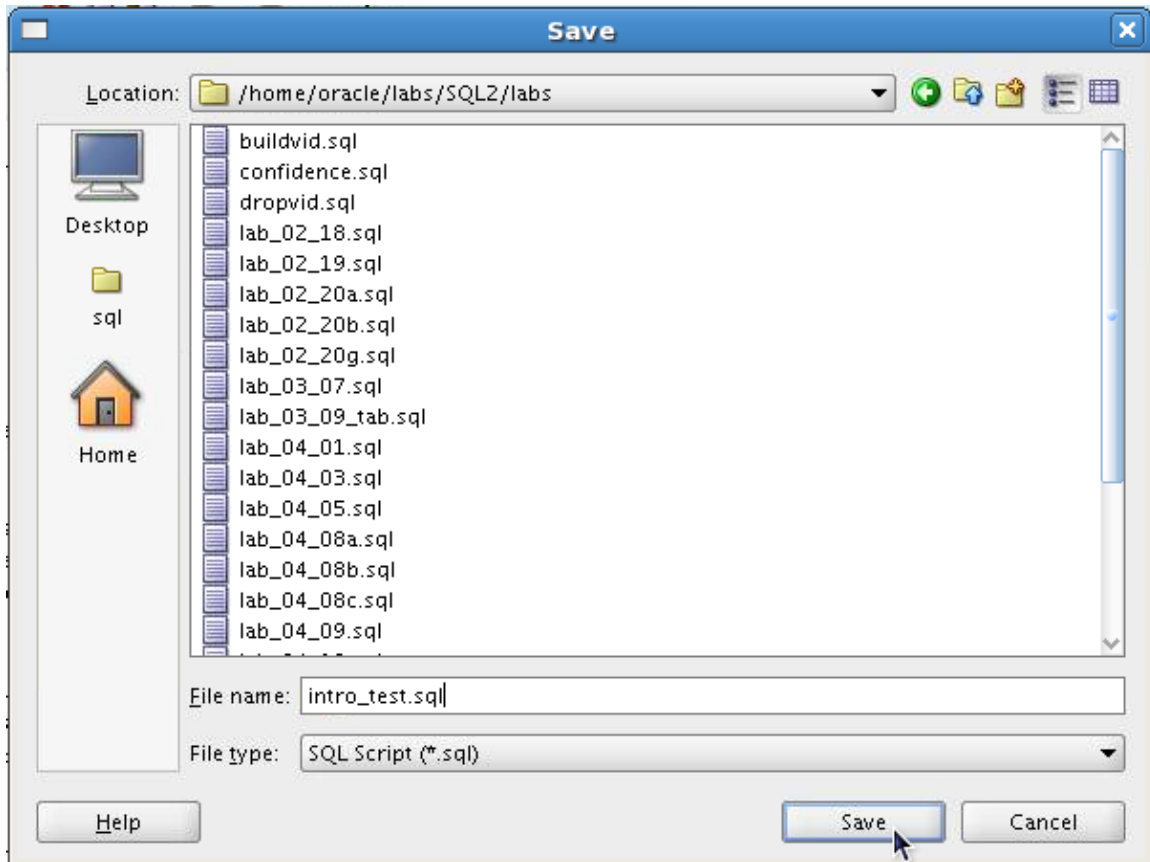


b) Name the file **intro\_test.sql**.

## Practice Solutions I-2: Using SQL Developer (continued)

Enter `intro_test.sql` in the File\_name text box.

c) Place the file under the `/home/oracle/labs/SQL2/labs` folder.

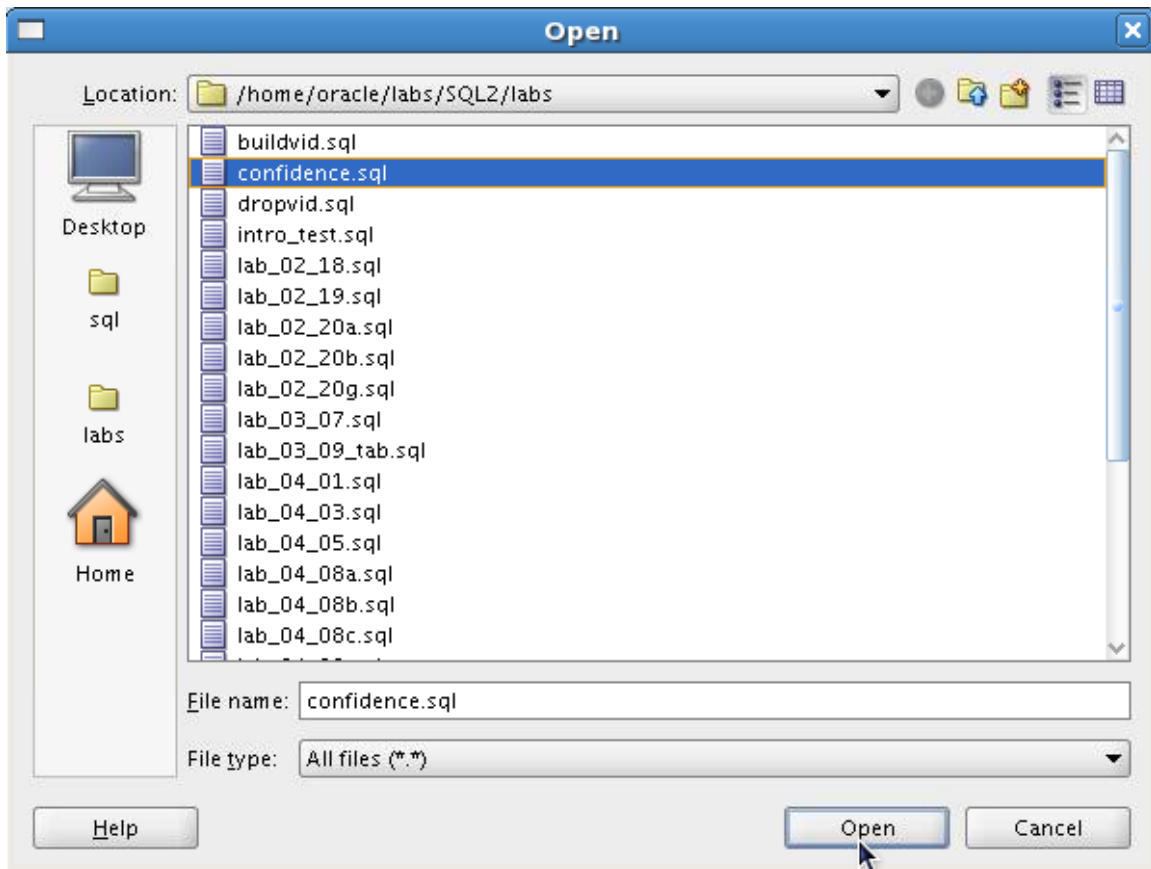


Then, click Save.

9) Open and run `confidence.sql` from your `/home/oracle/labs/SQL2/labs` folder and observe the output.

## Practice Solutions I-2: Using SQL Developer (continued)

Open the confidence.sql script file by using the File > Open menu item.



Then, press F5 to execute the script.

The following is the expected result:

```
COUNT (*)
-----
8

1 rows selected

COUNT (*)
-----
107

1 rows selected

COUNT (*)
-----
25

1 rows selected
```



## ***Practice Solutions I-2: Using SQL Developer (continued)***

```
COUNT (*)
-----
4
```

1 rows selected

```
COUNT (*)
-----
23
```

1 rows selected

```
COUNT (*)
-----
27
```

1 rows selected

```
COUNT (*)
-----
19
```

1 rows selected

```
COUNT (*)
-----
10
```

1 rows selected

## Practices and Solutions for Lesson 1

### ***Practice 1-1: Controlling User Access***

1. What privilege should a user be given to log on to the Oracle server? Is this a system privilege or an object privilege?  
\_\_\_\_\_
2. What privilege should a user be given to create tables?  
\_\_\_\_\_
3. If you create a table, who can pass along privileges to other users in your table?  
\_\_\_\_\_
4. You are the DBA. You create many users who require the same system privileges. What should you use to make your job easier?  
\_\_\_\_\_
5. What command do you use to change your password?  
\_\_\_\_\_
6. User21 is the owner of the EMP table and grants the DELETE privilege to User22 by using the WITH GRANT OPTION clause. User22 then grants the DELETE privilege on EMP to User23. User21 now finds that User23 has the privilege and revokes it from User22. Which user can now delete from the EMP table?  
\_\_\_\_\_
7. You want to grant SCOTT the privilege to update data in the DEPARTMENTS table. You also want to enable SCOTT to grant this privilege to other users. What command do you use?  
\_\_\_\_\_

**To complete question 8 and the subsequent ones, you need to connect to the database by using SQL Developer. If you are already not connected, do the following to connect:**

1. Click the SQL Developer desktop icon.
  2. In the Connections Navigator, use the **oraxx** account and the corresponding password provided by your instructor to log on to the database.
- 
8. Grant another user query privilege on your table. Then, verify whether that user can use the privilege.  
**Note:** For this exercise, team up with another group. For example, if you are user ora21, team up with another user ora22.
    - a. Grant another user privilege to view records in your REGIONS table. Include an option for this user to further grant this privilege to other users.
    - b. Have the user query your REGIONS table.
    - c. Have the user pass on the query privilege to a third user (for example, ora23).

### Practice 1-1: Controlling User Access (continued)

- d. Take back the privilege from the user who performs step b.
- Note:** Each team can run exercises 9 and 10 independently.
9. Grant another user query and data manipulation privileges on your COUNTRIES table. Make sure that the user cannot pass on these privileges to other users.
10. Take back the privileges on the COUNTRIES table granted to another user.
- Note:** For exercises 11 through 17, team up with another group.
11. Grant another user access to your DEPARTMENTS table. Have the user grant you query access to his or her DEPARTMENTS table.
12. Query all the rows in your DEPARTMENTS table.

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	30	Purchasing	114	1700
4	40	Human Resources	203	2400
5	50	Shipping	121	1500
6	60	IT	103	1400
7	70	Public Relations	204	2700
8	80	Sales	145	2500

...

13. Add a new row to your DEPARTMENTS table. Team 1 should add Education as department number 500. Team 2 should add Human Resources as department number 510. Query the other team's table.
14. Create a synonym for the other team's DEPARTMENTS table.
15. Query all the rows in the other team's DEPARTMENTS table by using your synonym.

Team 1 SELECT statement results:

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
15	150	Shareholder Services	(null)	1700
16	160	Benefits	(null)	1700
17	170	Manufacturing	(null)	1700
18	180	Construction	(null)	1700
19	190	Contracting	(null)	1700
20	200	Operations	(null)	1700
21	210	IT Support	(null)	1700
22	220	NOC	(null)	1700
23	230	IT Helpdesk	(null)	1700
24	240	Government Sales	(null)	1700
25	250	Retail Sales	(null)	1700
26	260	Recruiting	(null)	1700
27	270	Payroll	(null)	1700
28	510	Human Resources	(null)	(null)

## Practice 1-1: Controlling User Access (continued)

Team 2 SELECT statement results:

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
15	150	Shareholder Services	(null)	1700
16	160	Benefits	(null)	1700
17	170	Manufacturing	(null)	1700
18	180	Construction	(null)	1700
19	190	Contracting	(null)	1700
20	200	Operations	(null)	1700
21	210	IT Support	(null)	1700
22	220	NOC	(null)	1700
23	230	IT Helpdesk	(null)	1700
24	240	Government Sales	(null)	1700
25	250	Retail Sales	(null)	1700
26	260	Recruiting	(null)	1700
27	270	Payroll	(null)	1700
28	500	Education	(null)	(null)

16. Revoke the SELECT privilege from the other team.

17. Remove the row that you inserted into the DEPARTMENTS table in step 13 and save the changes.

## ***Practice Solutions 1-1: Controlling User Access***

To complete question 8 and the subsequent ones, you need to connect to the database by using SQL Developer.

1. What privilege should a user be given to log on to the Oracle server? Is this a system or an object privilege?  
**The CREATE SESSION system privilege**
2. What privilege should a user be given to create tables?  
**The CREATE TABLE privilege**
3. If you create a table, who can pass along privileges to other users in your table?  
**You can, or anyone you have given those privileges to, by using WITH GRANT OPTION**
4. You are the DBA. You create many users who require the same system privileges.  
What should you use to make your job easier?  
**Create a role containing the system privileges and grant the role to the users.**
5. What command do you use to change your password?  
**The ALTER USER statement**
6. User21 is the owner of the EMP table and grants DELETE privileges to User22 by using the WITH GRANT OPTION clause. User22 then grants DELETE privileges on EMP to User23. User21 now finds that User23 has the privilege and revokes it from User22. Which user can now delete data from the EMP table?  
**Only User21**
7. You want to grant SCOTT the privilege to update data in the DEPARTMENTS table. You also want to enable SCOTT to grant this privilege to other users. What command do you use?  
**GRANT UPDATE ON departments TO scott WITH GRANT OPTION;**

## Practice Solutions 1-1: Controlling User Access (continued)

8. Grant another user query privilege on your table. Then, verify whether that user can use the privilege.

**Note:** For this exercise, team up with another group. For example, if you are user ora21, team up with another user ora22.

- a) Grant another user privilege to view records in your REGIONS table. Include an option for this user to further grant this privilege to other users.

*Team 1 executes this statement:*

```
GRANT select
ON regions
TO <team2_oraxx> WITH GRANT OPTION;
```

- b) Have the user query your REGIONS table.

*Team 2 executes this statement:*

```
SELECT * FROM <team1_oraxx>.regions;
```

- c) Have the user pass on the query privilege to a third user (for example, ora23).

*Team 2 executes this statement.*

```
GRANT select
ON <team1_oraxx>.regions
TO <team3_oraxx>;
```

- d) Take back the privilege from the user who performs step b.

*Team 1 executes this statement.*

```
REVOKE select
ON regions
FROM <team2_oraxx>;
```

9. Grant another user query and data manipulation privileges on your COUNTRIES table. Make sure the user cannot pass on these privileges to other users.

*Team 1 executes this statement.*

```
GRANT select, update, insert
ON COUNTRIES
TO <team2_oraxx>;
```

## Practice Solutions 1-1: Controlling User Access (continued)

10. Take back the privileges on the COUNTRIES table granted to another user.

*Team 1 executes this statement.*

```
REVOKE select, update, insert ON COUNTRIES FROM <team2_oraxx>;
```

**Note:** For the exercises 11 through 17, team up with another group.

11. Grant another user access to your DEPARTMENTS table. Have the user grant you query access to his or her DEPARTMENTS table.

*Team 2 executes the GRANT statement.*

```
GRANT select  
ON departments  
TO <team1_oraxx>;
```

*Team 1 executes the GRANT statement.*

```
GRANT select  
ON departments  
TO <team2_oraxx>;
```

Here, <team1\_oraxx> is the username of Team 1 and <team2\_oraxx> is the username of Team 2.

12. Query all the rows in your DEPARTMENTS table.

```
SELECT  *  
FROM    departments;
```

13. Add a new row to your DEPARTMENTS table. Team 1 should add Education as department number 500. Team 2 should add Human Resources as department number 510. Query the other team's table.

*Team 1 executes this INSERT statement.*

```
INSERT INTO departments(department_id, department_name)  
VALUES (500, 'Education');  
COMMIT;
```

*Team 2 executes this INSERT statement.*

```
INSERT INTO departments(department_id, department_name)  
VALUES (510, 'Human Resources');  
COMMIT;
```

## Practice Solutions 1-1: Controlling User Access (continued)

14. Create a synonym for the other team's DEPARTMENTS table.

*Team 1 creates a synonym named team2.*

```
CREATE SYNONYM team2
      FOR <team2_oraxx>.DEPARTMENTS;
```

*Team 2 creates a synonym named team1.*

```
CREATE SYNONYM team1
      FOR <team1_oraxx>. DEPARTMENTS;
```

15. Query all the rows in the other team's DEPARTMENTS table by using your synonym.

*Team 1 executes this SELECT statement.*

```
SELECT *
      FROM   team2;
```

*Team 2 executes this SELECT statement.*

```
SELECT *
      FROM   team1;
```

16. Revoke the SELECT privilege from the other team.

*Team 1 revokes the privilege.*

```
REVOKE select
      ON departments
      FROM   <team2_oraxx>;
```

*Team 2 revokes the privilege.*

```
REVOKE select
      ON departments
      FROM   <team1_oraxx>;
```



## ***Practice Solutions 1-1: Controlling User Access (continued)***

17. Remove the row that you inserted into the DEPARTMENTS table in step 8 and save the changes.

*Team 1 executes this DELETE statement.*

```
DELETE FROM departments  
WHERE department_id = 500;  
COMMIT;
```

*Team 2 executes this DELETE statement.*

```
DELETE FROM departments  
WHERE department_id = 510;  
COMMIT;
```

## Practices and Solutions for Lesson 2

### Practice 2-1: Managing Schema Objects

In this practice, you use the `ALTER TABLE` command to modify columns and add constraints. You use the `CREATE INDEX` command to create indexes when creating a table, along with the `CREATE TABLE` command. You create external tables.

1. Create the `DEPT2` table based on the following table instance chart. Enter the syntax in the SQL Worksheet. Then, execute the statement to create the table. Confirm that the table is created.

Column Name	ID	NAME
Key Type		
Nulls/Unique		
FK Table		
FK Column		
Data type	NUMBER	VARCHAR2
Length	7	25

Name	Null	Type
-----	-----	-----
ID		NUMBER (7)
NAME		VARCHAR2 (25)
2 rows selected		

2. Populate the `DEPT2` table with data from the `DEPARTMENTS` table. Include only the columns that you need.
3. Create the `EMP2` table based on the following table instance chart. Enter the syntax in the SQL Worksheet. Then execute the statement to create the table. Confirm that the table is created.

Column Name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Key Type				
Nulls/Unique				
FK Table				
FK Column				
Data type	NUMBER	VARCHAR2	VARCHAR2	NUMBER
Length	7	25	25	7

## Practice 2-1: Managing Schema Objects (continued)

Name	Null	Type
-----		
ID		NUMBER(7)
LAST_NAME		VARCHAR2(25)
FIRST_NAME		VARCHAR2(25)
DEPT_ID		NUMBER(7)
4 rows selected		

4. Modify the EMP2 table to allow for longer employee last names. Confirm your modification.

Name	Null	Type
-----		
ID		NUMBER(7)
LAST_NAME		VARCHAR2(50)
FIRST_NAME		VARCHAR2(25)
DEPT_ID		NUMBER(7)
4 rows selected		

5. Create the EMPLOYEES2 table based on the structure of the EMPLOYEES table. Include only the EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, SALARY, and DEPARTMENT\_ID columns. Name the columns in your new table ID, FIRST\_NAME, LAST\_NAME, SALARY, and DEPT\_ID, respectively.
6. Drop the EMP2 table.
7. Query the recycle bin to see whether the table is present.

	ORIGINAL_NAME	OPERATION	DROPTIME
-----			
17	EMP_NEW_SAL	DROP	2009-05-22:14:44:15
18	EMP2	DROP	2009-05-22:14:57:57

8. Restore the EMP2 table to a state before the DROP statement.

Name	Null	Type
-----		
ID		NUMBER(7)
LAST_NAME		VARCHAR2(50)
FIRST_NAME		VARCHAR2(25)
DEPT_ID		NUMBER(7)
4 rows selected		

9. Drop the FIRST\_NAME column from the EMPLOYEES2 table. Confirm your modification by checking the description of the table.

## Practice 2-1: Managing Schema Objects (continued)

Name	Null	Type
ID		NUMBER(6)
LAST_NAME	NOT NULL	VARCHAR2(25)
SALARY		NUMBER(8,2)
DEPT_ID		NUMBER(4)
4 rows selected		

10. In the EMPLOYEES2 table, mark the DEPT\_ID column as UNUSED. Confirm your modification by checking the description of the table.

Name	Null	Type
ID		NUMBER(6)
LAST_NAME	NOT NULL	VARCHAR2(25)
SALARY		NUMBER(8,2)
3 rows selected		

11. Drop all the UNUSED columns from the EMPLOYEES2 table. Confirm your modification by checking the description of the table.
12. Add a table-level PRIMARY KEY constraint to the EMP2 table on the ID column. The constraint should be named at creation. Name the constraint my\_emp\_id\_pk.
13. Create a PRIMARY KEY constraint to the DEPT2 table using the ID column. The constraint should be named at creation. Name the constraint my\_dept\_id\_pk.
14. Add a foreign key reference on the EMP2 table that ensures that the employee is not assigned to a nonexistent department. Name the constraint my\_emp\_dept\_id\_fk.
15. Modify the EMP2 table. Add a COMMISSION column of the NUMBER data type, precision 2, scale 2. Add a constraint to the COMMISSION column that ensures that a commission value is greater than zero.
16. Drop the EMP2 and DEPT2 tables so that they cannot be restored. Verify the recycle bin.
17. Create the DEPT\_NAMED\_INDEX table based on the following table instance chart. Name the index for the PRIMARY KEY column as DEPT\_PK\_IDX.

Column Name	Deptno	Dname
Primary Key	Yes	
Data Type	Number	VARCHAR2
Length	4	30

18. Create an external table library\_items\_ext. Use the ORACLE\_LOADER access driver.

## Practice 2-1: Managing Schema Objects (continued)

**Note:** The `emp_dir` directory and `library_items.dat` file are already created for this exercise. `library_items.dat` has records in the following format:

```
2354, 2264, 13.21, 150,  
2355, 2289, 46.23, 200,  
2355, 2264, 50.00, 100,
```

- Open the `lab_02_18.sql` file. Observe the code snippet to create the `library_items_ext` external table. Then replace `<TODO1>`, `<TODO2>`, `<TODO3>`, and `<TODO4>` as appropriate and save the file as `lab_02_18_soln.sql`. Run the script to create the external table.
- Query the `library_items_ext` table.

	CATEGOR...	BOO...	BOOK_P...	QUAN...
1	2354	2264	13.21	150
2	2355	2289	46.23	200
3	2355	2264	50	100

- The HR department needs a report of the addresses of all departments. Create an external table as `dept_add_ext` using the `ORACLE_DATAPUMP` access driver. The report should show the location ID, street address, city, state or province, and country in the output. Use a `NATURAL JOIN` to produce the results.

**Note:** The `emp_dir` directory is already created for this exercise.

- Open the `lab_02_19.sql` file. Observe the code snippet to create the `dept_add_ext` external table. Then, replace `<TODO1>`, `<TODO2>`, and `<TODO3>` with the appropriate code. Replace `<oraxx_emp4.exp>` and `<oraxx_emp5.exp>` with the appropriate file names. For example, if you are the `ora21` user, your file names are `ora21_emp4.exp` and `ora21_emp5.exp`. Save the script as `lab_02_19_soln.sql`.
- Run the `lab_02_19_soln.sql` script to create the external table.
- Query the `dept_add_ext` table.

## Practice 2-1: Managing Schema Objects (continued)

	LOCAT...	STREET_ADDRESS	CITY	STATE_PROVINCE	COUNTRY_NAME
1	1000	1297 Via Cola di Rie	Roma	(null)	Italy
2	1100	93091 Calle della Testa	Venice	(null)	Italy
3	1200	2017 Shinjuku-ku	Tokyo	Tokyo Prefecture	Japan
4	1300	9450 Kamiya-cho	Hiroshima	(null)	Japan
5	1400	2014 Jabberwocky Rd	Southlake	Texas	United States of Amer
6	1500	2011 Interiors Blvd	South San Francisco	California	United States of Amer
7	1600	2007 Zagora St	South Brunswick	New Jersey	United States of Amer
8	1700	2004 Charade Rd	Seattle	Washington	United States of Amer

**Note:** When you perform the preceding step, two files `oraxx_emp4.exp` and `oraxx_emp5.exp` are created under the default directory `emp_dir`.

20. Create the `emp_books` table and populate it with data. Set the primary key as deferred and observe what happens at the end of the transaction.
- Run the `lab_02_20_a.sql` file to create the `emp_books` table. Observe that the `emp_books_pk` primary key is not created as deferrable.

```
create table succeeded.
```

- Run the `lab_02_20_b.sql` file to populate data into the `emp_books` table. What do you observe?

```
1 rows inserted

Error starting at line 2 in command:
insert into emp_books values(300,'Change Management')
Error report:
SQL Error: ORA-00001: unique constraint (ORA21.EMP_BOOKS_PK) violated
00001. 00000 - "unique constraint (%s.%s) violated"
*Cause:      An UPDATE or INSERT statement attempted to insert a duplicate key.
              For Trusted Oracle configured in DBMS MAC mode, you may see
              this message if a duplicate entry exists at a different level.
*Action:     Either remove the unique restriction or do not insert the key.
```

- Set the `emp_books_pk` constraint as deferred. What do you observe?

```
Error starting at line 1 in command:
set constraint emp_books_pk deferred
Error report:
SQL Error: ORA-02447: cannot defer a constraint that is not deferrable
02447. 00000 - "cannot defer a constraint that is not deferrable"
*Cause:      An attempt was made to defer a nondeferrable constraint
*Action:     Drop the constraint and create a new one that is deferrable
```

- Drop the `emp_books_pk` constraint.

```
alter table emp_books succeeded.
```

- Modify the `emp_books` table definition to add the `emp_books_pk` constraint as deferrable this time.

## Practice 2-1: Managing Schema Objects (continued)

```
alter table emp_books succeeded.
```

f. Set the emp\_books\_pk constraint as deferred.

```
set constraint succeeded.
```

g. Run the lab\_02\_20\_g.sql file to populate data into the emp\_books table.

What do you observe?

```
1 rows inserted  
1 rows inserted  
1 rows inserted
```

h. Commit the transaction. What do you observe?

```
Error report:  
SQL Error: ORA-02091: transaction rolled back  
ORA-00001: unique constraint (ORA21.EMP_BOOKS_PK) violated  
02091. 00000 - "transaction rolled back"  
*Cause:      Also see error 2092. If the transaction is aborted at a remote  
              site then you will only see 2091; if aborted at host then you will  
              see 2092 and 2091.  
*Action:     Add rollback segment and retry the transaction.
```

## Practice Solutions 2-1: Managing Schema Objects

1. Create the DEPT2 table based on the following table instance chart. Enter the syntax in the SQL Worksheet. Then, execute the statement to create the table. Confirm that the table is created.

Column Name	ID	NAME
Key Type		
Nulls/Unique		
FK Table		
FK Column		
Data type	NUMBER	VARCHAR2
Length	7	25

```
CREATE TABLE dept2
  (id NUMBER(7),
   name VARCHAR2(25));

DESCRIBE dept2
```

2. Populate the DEPT2 table with data from the DEPARTMENTS table. Include only the columns that you need.

```
INSERT INTO dept2
SELECT department_id, department_name
FROM departments;
```

3. Create the EMP2 table based on the following table instance chart. Enter the syntax in the SQL Worksheet. Then execute the statement to create the table. Confirm that the table is created.

Column Name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Key Type				
Nulls/Unique				
FK Table				
FK Column				
Data type	NUMBER	VARCHAR2	VARCHAR2	NUMBER
Length	7	25	25	7



## ***Practice Solutions 2-1: Managing Schema Objects (continued)***

```
CREATE TABLE emp2
(id          NUMBER(7),
 last_name   VARCHAR2(25),
 first_name  VARCHAR2(25),
 dept_id     NUMBER(7));

DESCRIBE emp2
```

4. Modify the EMP2 table to allow for longer employee last names. Confirm your modification.

```
ALTER TABLE emp2
MODIFY (last_name   VARCHAR2(50));

DESCRIBE emp2
```

5. Create the EMPLOYEES2 table based on the structure of the EMPLOYEES table. Include only the EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, SALARY, and DEPARTMENT\_ID columns. Name the columns in your new table ID, FIRST\_NAME, LAST\_NAME, SALARY, and DEPT\_ID, respectively.

```
CREATE TABLE employees2 AS
SELECT  employee_id id, first_name, last_name, salary,
        department_id dept_id
FROM    employees;
```

6. Drop the EMP2 table.

```
DROP TABLE emp2;
```

## ***Practice Solutions 2-1: Managing Schema Objects (continued)***

7. Query the recycle bin to see whether the table is present.

```
SELECT original_name, operation, droptime
FROM recyclebin;
```

8. Restore the EMP2 table to a state before the DROP statement.

```
FLASHBACK TABLE emp2 TO BEFORE DROP;
DESC emp2;
```

9. Drop the FIRST\_NAME column from the EMPLOYEES2 table. Confirm your modification by checking the description of the table.

```
ALTER TABLE employees2
DROP COLUMN first_name;

DESCRIBE employees2
```

10. In the EMPLOYEES2 table, mark the DEPT\_ID column as UNUSED. Confirm your modification by checking the description of the table.

```
ALTER TABLE    employees2
SET    UNUSED (dept_id);

DESCRIBE employees2
```

11. Drop all the UNUSED columns from the EMPLOYEES2 table. Confirm your modification by checking the description of the table.

```
ALTER TABLE employees2
DROP UNUSED COLUMNS;

DESCRIBE employees2
```

## Practice Solutions 2-1: Managing Schema Objects (continued)

12. Add a table-level PRIMARY KEY constraint to the EMP2 table on the ID column. The constraint should be named at creation. Name the constraint my\_emp\_id\_pk.

```
ALTER TABLE emp2
ADD CONSTRAINT my_emp_id_pk PRIMARY KEY (id);
```

13. Create a PRIMARY KEY constraint to the DEPT2 table using the ID column. The constraint should be named at creation. Name the constraint my\_dept\_id\_pk.

```
ALTER TABLE dept2
ADD CONSTRAINT my_dept_id_pk PRIMARY KEY(id);
```

14. Add a foreign key reference on the EMP2 table that ensures that the employee is not assigned to a nonexistent department. Name the constraint my\_emp\_dept\_id\_fk.

```
ALTER TABLE emp2
ADD CONSTRAINT my_emp_dept_id_fk
FOREIGN KEY (dept_id) REFERENCES dept2(id);
```

15. Modify the EMP2 table. Add a COMMISSION column of the NUMBER data type, precision 2, scale 2. Add a constraint to the COMMISSION column that ensures that a commission value is greater than zero.

```
ALTER TABLE emp2
ADD commission NUMBER(2,2)
CONSTRAINT my_emp_comm_ck CHECK (commission > 0);
```

16. Drop the EMP2 and DEPT2 tables so that they cannot be restored. Check in the recycle bin.

```
DROP TABLE emp2 PURGE;
DROP TABLE dept2 PURGE;

SELECT original_name, operation, droptime
FROM recyclebin;
```

17. Create the DEPT\_NAMED\_INDEX table based on the following table instance chart. Name the index for the PRIMARY KEY column as DEPT\_PK\_IDX.

Column Name	Deptno	Dname
Primary Key	Yes	
Data Type	Number	VARCHAR2
Length	4	30

## Practice Solutions 2-1: Managing Schema Objects (continued)

```
CREATE TABLE DEPT_NAMED_INDEX
(deptno NUMBER(4)
PRIMARY KEY USING INDEX
(CREATE INDEX dept_pk_idx ON
DEPT_NAMED_INDEX(deptno)),
dname VARCHAR2(30));
```

18. Create an external table `library_items_ext`. Use the `ORACLE_LOADER` access driver.

**Note:** The `emp_dir` directory and `library_items.dat` are already created for this exercise.

`library_items.dat` has records in the following format:

2354, 2264, 13.21, 150,

2355, 2289, 46.23, 200,

2355, 2264, 50.00, 100,

- a) Open the `lab_02_18.sql` file. Observe the code snippet to create the `library_items_ext` external table. Then, replace `<TODO1>`, `<TODO2>`, `<TODO3>`, and `<TODO4>` as appropriate and save the file as `lab_02_18_soln.sql`.  
Run the script to create the external table.

```
CREATE TABLE library_items_ext ( category_id  number(12)
                                , book_id  number(6)
                                , book_price number(8,2)
                                , quantity  number(8)
                                )

ORGANIZATION EXTERNAL
(TYPE ORACLE_LOADER
DEFAULT DIRECTORY emp_dir
ACCESS PARAMETERS (RECORDS DELIMITED BY NEWLINE
                   FIELDS TERMINATED BY ','))
LOCATION ('library_items.dat')
)
REJECT LIMIT UNLIMITED;
```

## Practice Solutions 2-1: Managing Schema Objects (continued)

b) Query the `library_items_ext` table.

```
SELECT * FROM library_items_ext;
```

19. The HR department needs a report of addresses of all the departments. Create an external table as `dept_add_ext` using the `ORACLE_DATAPUMP` access driver. The report should show the location ID, street address, city, state or province, and country in the output. Use a `NATURAL JOIN` to produce the results.

**Note:** The `emp_dir` directory is already created for this exercise.

- a) Open the `lab_02_19.sql` file. Observe the code snippet to create the `dept_add_ext` external table. Then, replace `<TODO1>`, `<TODO2>`, and `<TODO3>` with appropriate code. Replace `<oraxx_emp4.exp>` and `<oraxx_emp5.exp>` with appropriate file names. For example, if you are user `ora21`, your file names are `ora21_emp4.exp` and `ora21_emp5.exp`. Save the script as `lab_02_19_soln.sql`.

```
CREATE TABLE dept_add_ext (location_id,
                           street_address, city,
                           state_province,
                           country_name)

ORGANIZATION EXTERNAL(
TYPE ORACLE_DATAPUMP
DEFAULT DIRECTORY emp_dir
LOCATION ('oraxx_emp4.exp','oraxx_emp5.exp'))
PARALLEL
AS
SELECT location_id, street_address, city, state_province,
country_name
FROM locations
NATURAL JOIN countries;
```

**Note:** When you perform the preceding step, two files `oraxx_emp4.exp` and `oraxx_emp5.exp` are created under the default directory `emp_dir`.

Run the `lab_02_19_soln.sql` script to create the external table.

## Practice Solutions 2-1: Managing Schema Objects (continued)

b) Query the dept\_add\_ext table.

```
SELECT * FROM dept_add_ext;
```

20. Create the emp\_books table and populate it with data. Set the primary key as deferred and observe what happens at the end of the transaction.

a) Run the lab\_02\_20a.sql script to create the emp\_books table. Observe that the emp\_books\_pk primary key is not created as deferrable.

```
CREATE TABLE emp_books (book_id number,  
                          title varchar2(20), CONSTRAINT  
emp_books_pk PRIMARY KEY (book_id));
```

b) Run the lab\_02\_20b.sql script to populate data into the emp\_books table.

What do you observe?

```
INSERT INTO emp_books VALUES (300, 'Organizations');  
INSERT INTO emp_books VALUES (300, 'Change Management');
```

The first row is inserted. However, you see the ora-00001 error with the second row insertion.

c) Set the emp\_books\_pk constraint as deferred. What do you observe?

```
SET CONSTRAINT emp_books_pk DEFERRED;
```

You see the following error: “ORA-02447: Cannot defer a constraint that is not deferrable.”

d) Drop the emp\_books\_pk constraint.

```
ALTER TABLE emp_books DROP CONSTRAINT emp_books_pk;
```

e) Modify the emp\_books table definition to add the emp\_books\_pk constraint as deferrable this time.

```
ALTER TABLE emp_books ADD (CONSTRAINT emp_books_pk PRIMARY KEY  
(book_id) DEFERRABLE);
```

f) Set the emp\_books\_pk constraint as deferred.

```
SET CONSTRAINT emp_books_pk DEFERRED;
```

## ***Practice Solutions 2-1: Managing Schema Objects (continued)***

g) Run the lab\_02\_20g.sql script to populate data into the emp\_books table.

What do you observe?

```
INSERT INTO emp_books VALUES (300, 'Change Management');  
INSERT INTO emp_books VALUES (300, 'Personality');  
INSERT INTO emp_books VALUES (350, 'Creativity');
```

You see that all the rows are inserted.

h) Commit the transaction. What do you observe?

```
COMMIT;
```

You see that the transaction is rolled back.

## Practices and Solutions for Lesson 3

### Practice 3-1: Managing Objects with Data Dictionary Views

In this practice, you query the dictionary views to find information about objects in your schema.

1. Query the USER\_TABLES data dictionary view to see information about the tables that you own.

	TABLE_NAME
1	REGIONS
2	LOCATIONS
3	DEPARTMENTS
4	JOBS
5	EMPLOYEES
6	JOB_HISTORY
7	EMP_NEW_SAL
8	EMPLOYEES2
9	DEPT_NAMED_INDEX

...

2. Query the ALL\_TABLES data dictionary view to see information about all the tables that you can access. Exclude the tables that you own.

**Note:** Your list may not exactly match the following list:

	TABLE_NAME	OWNER
1	DUAL	SYS
2	SYSTEM_PRIVILEGE_MAP	SYS
3	TABLE_PRIVILEGE_MAP	SYS

...

98	PLAN_TABLE\$	SYS
99	WRI\$_ADV_ASA_RECO_DATA	SYS
100	PSTUBTBL	SYS

3. For a specified table, create a script that reports the column names, data types, and data types' lengths, as well as whether nulls are allowed. Prompt the user to enter the table name. Give appropriate aliases to the DATA\_PRECISION and DATA\_SCALE columns. Save this script in a file named lab\_03\_01.sql.

For example, if the user enters DEPARTMENTS, the following output results:



### Practice 3-1: Managing Objects with Data Dictionary Views (continued)

	COLUMN_NAME	DATA_TYPE	DATA_LENGTH	PRECISION	SCALE	NULLABLE
1	DEPARTMENT_ID	NUMBER	22	4	0	N
2	DEPARTMENT_NAME	VARCHAR2	30	(null)	(null)	N
3	MANAGER_ID	NUMBER	22	6	0	Y
4	LOCATION_ID	NUMBER	22	4	0	Y

4. Create a script that reports the column name, constraint name, constraint type, search condition, and status for a specified table. You must join the USER\_CONSTRAINTS and USER\_CONS\_COLUMNS tables to obtain all this information. Prompt the user to enter the table name. Save the script in a file named lab\_03\_04.sql. For example, if the user enters DEPARTMENTS, the following output results:

	COLUMN_NAME	CONSTRAINT_NAME	CONSTR...	SEARCH_CONDITION	STATUS
1	DEPARTMENT_NAME	DEPT_NAME_NN	C	"DEPARTMENT_NAME" IS NOT ...	ENABLED
2	DEPARTMENT_ID	DEPT_ID_PK	P	(null)	ENABLED
3	LOCATION_ID	DEPT_LOC_FK	R	(null)	ENABLED
4	MANAGER_ID	DEPT_MGR_FK	R	(null)	ENABLED

5. Add a comment to the DEPARTMENTS table. Then query the USER\_TAB\_COMMENTS view to verify that the comment is present.

	COMMENTS
1	Company department information including name, code, and location.

6. Create a synonym for your EMPLOYEES table. Call it EMP. Then find the names of all synonyms that are in your schema.

	SYNONYM_NAME	TABLE_OWNER	TABLE_NAME	DB_LINK
1	TEAM2	ORA22	DEPARTMENTS	(null)
2	EMP	ORA21	EMPLOYEES	(null)

7. Run lab\_03\_07.sql to create the dept50 view for this exercise. You need to determine the names and definitions of all the views in your schema. Create a report that retrieves view information: the view name and text from the USER\_VIEWS data dictionary view.

**Note:** The EMP\_DETAILS\_VIEW was created as part of your schema.

**Note:** You can see the complete definition of the view if you use Run Script (or press F5) in SQL Developer. If you use Execute Statement (or press F9) in SQL Developer, scroll horizontally in the result pane. If you use SQL\*Plus, to see more contents of a LONG column, use the SET LONG n command, where n is the value of the number of characters of the LONG column that you want to see.

### ***Practice 3-1: Managing Objects with Data Dictionary Views (continued)***

R2	VIEW_NAME	TEXT
1	DEPT50	SELECT employee_id empno, last_name employee, department_id deptno
2	EMP_DETAILS_VIEW	SELECT e.employee_id, e.job_id, e.manager_id, e.department_id, d.location_id,

8. Find the names of your sequences. Write a query in a script to display the following information about your sequences: sequence name, maximum value, increment size, and last number. Name the script `lab_03_08.sql`. Run the statement in your script.

[illegible]

Run the `lab_03_09_tab.sql` script as a prerequisite for exercises 9 through 11. Alternatively, open the script file to copy the code and paste it into your SQL Worksheet. Then execute the script. This script:

- Drops if there are existing tables DEPT2 and EMP2
- Creates the DEPT2 and EMP2 tables

**Note:** In Practice 2, you should have already dropped the DEPT2 and EMP2 tables so that they cannot be restored.

9. Confirm that both the DEPT2 and EMP2 tables are stored in the data dictionary.

	TABLE_NAME
1	DEPT2
2	EMP2




10. Confirm that the constraints were added by querying the `USER_CONSTRAINTS` view. Note the types and names of the constraints.

R2	CONSTRAINT_NAME	R2	CONSTRAINT_TYPE
1	MY_DEPT_ID_PK		P
2	MY_EMP_ID_PK		P
3	MY_EMP_DEPT_ID_FK		R

11. Display the object names and types from the USER\_OBJECTS data dictionary view for the EMP2 and DEPT2 tables.
12. Create the SALES\_DEPT table based on the following table instance chart. Name the index for the PRIMARY KEY column SALES\_PK\_IDX. Then query the data dictionary view to find the index name, table name, and whether the index is unique.

### ***Practice 3-1: Managing Objects with Data Dictionary Views (continued)***

<b>Column Name</b>	Team_Id	Location
<b>Primary Key</b>	Yes	
<b>Data Type</b>	Number	VARCHAR2
<b>Length</b>	3	30

 INDEX_NAME	 TABLE_NAME	 UNIQUENESS
1 SALES_PK_IDX	SALES_DEPT	NONUNIQUE

## ***Practice Solutions 3-1: Managing Objects with Data Dictionary Views***

1. Query the data dictionary to see information about the tables you own.

```
SELECT table_name
FROM   user_tables;
```

2. Query the dictionary view to see information about all the tables that you can access. Exclude tables that you own.

```
SELECT table_name, owner
FROM   all_tables
WHERE  owner <> 'ORAxX';
```

3. For a specified table, create a script that reports the column names, data types, and data types' lengths, as well as whether nulls are allowed. Prompt the user to enter the table name. Give appropriate aliases to the DATA\_PRECISION and DATA\_SCALE columns. Save this script in a file named lab\_03\_01.sql.

```
SELECT column_name, data_type, data_length,
       data_precision PRECISION, data_scale SCALE, nullable
FROM   user_tab_columns
WHERE  table_name = UPPER('&tab_name');
```

To test, run the script and enter DEPARTMENTS as the table name.

4. Create a script that reports the column name, constraint name, constraint type, search condition, and status for a specified table. You must join the USER\_CONSTRAINTS and USER\_CONS\_COLUMNS tables to obtain all this information. Prompt the user to enter the table name. Save the script in a file named lab\_03\_04.sql.

```
SELECT ucc.column_name, uc.constraint_name,
       uc.constraint_type,
       uc.search_condition, uc.status
FROM   user_constraints uc JOIN user_cons_columns ucc
ON     uc.table_name = ucc.table_name
AND    uc.constraint_name = ucc.constraint_name
AND    uc.table_name = UPPER('&tab_name');
```

To test, run the script and enter DEPARTMENTS as the table name.

## **Practice Solutions 3-1: Managing Objects with Data Dictionary Views (continued)**

5. Add a comment to the DEPARTMENTS table. Then query the USER\_TAB\_COMMENTS view to verify that the comment is present.

```
COMMENT ON TABLE departments IS
    'Company department information including name, code, and
    location.';

SELECT COMMENTS
FROM   user_tab_comments
WHERE  table_name = 'DEPARTMENTS';
```

6. Create a synonym for your EMPLOYEES table. Call it EMP. Then, find the names of all the synonyms that are in your schema.

```
CREATE SYNONYM emp FOR EMPLOYEES;
SELECT *
FROM   user_synonyms;
```

7. Run lab\_03\_07.sql to create the dept50 view for this exercise. You need to determine the names and definitions of all the views in your schema. Create a report that retrieves view information: the view name and text from the USER\_VIEWS data dictionary view.

**Note:** The EMP\_DETAILS\_VIEW was created as part of your schema.

**Note:** You can see the complete definition of the view if you use Run Script (or press F5) in SQL Developer. If you use Execute Statement (or press F9) in SQL Developer, scroll horizontally in the result pane. If you use SQL\*Plus to see more contents of a LONG column, use the SET LONG n command, where n is the value of the number of characters of the LONG column that you want to see.

```
SELECT   view_name, text
FROM     user_views;
```

## **Practice Solutions 3-1: Managing Objects with Data Dictionary Views (continued)**

8. Find the names of your sequences. Write a query in a script to display the following information about your sequences: sequence name, maximum value, increment size, and last number. Name the script `lab_03_08.sql`. Run the statement in your script.

```
SELECT    sequence_name, max_value, increment_by, last_number
FROM      user_sequences;
```

Run the `lab_03_09_tab.sql` script as a prerequisite for exercises 9 through 11. Alternatively, open the script file to copy the code and paste it into your SQL Worksheet. Then execute the script. This script:

- Drops the DEPT2 and EMP2 tables
- Creates the DEPT2 and EMP2 tables

**Note:** In Practice 2, you should have already dropped the DEPT2 and EMP2 tables so that they cannot be restored.

9. Confirm that both the DEPT2 and EMP2 tables are stored in the data dictionary.

```
SELECT    table_name
FROM      user_tables
WHERE     table_name IN ('DEPT2', 'EMP2');
```

10. Query the data dictionary to find out the constraint names and types for both the tables.

```
SELECT    constraint_name, constraint_type
FROM      user_constraints
WHERE     table_name IN ('EMP2', 'DEPT2');
```

11. Query the data dictionary to display the object names and types for both the tables.

```
SELECT    object_name, object_type
FROM      user_objects
WHERE     object_name LIKE 'EMP%'
OR        object_name LIKE 'DEPT%';
```

## ***Practice Solutions 3-1: Managing Objects with Data Dictionary Views (continued)***

12. Create the SALES\_DEPT table based on the following table instance chart. Name the index for the PRIMARY KEY column as SALES\_PK\_IDX. Then query the data dictionary view to find the index name, table name, and whether the index is unique.

Column Name	Team_Id	Location
Primary Key	Yes	
Data Type	Number	VARCHAR2
Length	3	30

```
CREATE TABLE SALES_DEPT
  (team_id NUMBER(3)
   PRIMARY KEY USING INDEX
   (CREATE INDEX sales_pk_idx ON
    SALES_DEPT(team_id),
    location VARCHAR2(30));

SELECT INDEX_NAME, TABLE_NAME, UNIQUENESS
FROM USER_INDEXES
WHERE TABLE_NAME = 'SALES_DEPT';
```

## Practices and Solutions for Lesson 4

### Practice 4-1: Manipulating Large Data Sets

In this practice, you perform multitable INSERT and MERGE operations, and track row versions.

1. Run the lab\_04\_01.sql script in the lab folder to create the SAL\_HISTORY table.
2. Display the structure of the SAL\_HISTORY table.

Name	Null	Type
EMPLOYEE_ID		NUMBER(6)
HIRE_DATE		DATE
SALARY		NUMBER(8,2)
3 rows selected		

3. Run the lab\_04\_03.sql script in the lab folder to create the MGR\_HISTORY table.
4. Display the structure of the MGR\_HISTORY table.

Name	Null	Type
EMPLOYEE_ID		NUMBER(6)
MANAGER_ID		NUMBER(6)
SALARY		NUMBER(8,2)
3 rows selected		

5. Run the lab\_04\_05.sql script in the lab folder to create the SPECIAL\_SAL table.
6. Display the structure of the SPECIAL\_SAL table.

Name	Null	Type
EMPLOYEE_ID		NUMBER(6)
SALARY		NUMBER(8,2)
2 rows selected		

7. a. Write a query to do the following:
  - Retrieve details such as the employee ID, hire date, salary, and manager ID of those employees whose employee ID is less than 125 from the EMPLOYEES table.
  - If the salary is more than \$20,000, insert details such as the employee ID and salary into the SPECIAL\_SAL table.



### Practice 4-1: Manipulating Large Data Sets (continued)

- Insert details such as the employee ID, hire date, and salary into the SAL\_HISTORY table.
- Insert details such as the employee ID, manager ID, and salary into the MGR\_HISTORY table.

- b. Display the records from the SPECIAL\_SAL table.

	EMPLOYEE_ID	SALARY
1	100	24000

- c. Display the records from the SAL\_HISTORY table.

	EMPLOYEE_ID	HIRE_DATE	SALARY
1	101	21-SEP-89	17000
2	102	13-JAN-93	17000
3	103	03-JAN-90	9000
4	104	21-MAY-91	6000
5	105	25-JUN-97	4800
6	106	05-FEB-98	4800
7	107	07-FEB-99	4200

- d. Display the records from the MGR\_HISTORY table.

	EMPLOYEE_ID	MANAGER_ID	SALARY
1	101	100	17000
2	102	100	17000
3	103	102	9000
4	104	103	6000
5	105	103	4800
6	106	103	4800
7	107	103	4200

8.

- Run the lab\_04\_08a.sql script in the lab folder to create the SALES\_WEEK\_DATA table.
- Run the lab\_04\_08b.sql script in the lab folder to insert records into the SALES\_WEEK\_DATA table.

## Practice 4-1: Manipulating Large Data Sets (continued)

- c. Display the structure of the SALES\_WEEK\_DATA table.

Name	Null	Type
ID		NUMBER(6)
WEEK_ID		NUMBER(2)
QTY_MON		NUMBER(8,2)
QTY_TUE		NUMBER(8,2)
QTY_WED		NUMBER(8,2)
QTY_THUR		NUMBER(8,2)
QTY_FRI		NUMBER(8,2)
7 rows selected		

- d. Display the records from the SALES\_WEEK\_DATA table.

ID	WEEK_ID	QTY_MON	QTY_TUE	QTY_WED	QTY_THUR	QTY_FRI	
1	200	6	2050	2200	1700	1200	3000

- e. Run the lab\_04\_08\_e.sql script in the lab folder to create the EMP\_SALES\_INFO table.
- f. Display the structure of the EMP\_SALES\_INFO table.

Name	Null	Type
ID		NUMBER(6)
WEEK		NUMBER(2)
QTY_SALES		NUMBER(8,2)
3 rows selected		

- g. Write a query to do the following:
- Retrieve details such as ID, week ID, sales quantity on Monday, sales quantity on Tuesday, sales quantity on Wednesday, sales quantity on Thursday, and sales quantity on Friday from the SALES\_WEEK\_DATA table.
  - Build a transformation such that each record retrieved from the SALES\_WEEK\_DATA table is converted into multiple records for the EMP\_SALES\_INFO table.
- Hint:** Use a pivoting INSERT statement.

- h. Display the records from the EMP\_SALES\_INFO table.

	AZ	ID	AZ	WEEK	AZ	QTY_SALES
1		200		6		2050
2		200		6		2200
3		200		6		1700
4		200		6		1200
5		200		6		3000

9. You have the data of past employees stored in a flat file called emp.data. You want to store the names and email IDs of all employees, past and present, in a table. To do

### Practice 4-1: Manipulating Large Data Sets (continued)

this, first create an external table called EMP\_DATA using the emp.dat source file in the emp\_dir directory. Use the lab\_04\_09.sql script to do this.

10. Next, run the lab\_04\_10.sql script to create the EMP\_HIST table.

- Increase the size of the email column to 45.
- Merge the data in the EMP\_DATA table created in the last lab into the data in the EMP\_HIST table. Assume that the data in the external EMP\_DATA table is the most up-to-date. If a row in the EMP\_DATA table matches the EMP\_HIST table, update the email column of the EMP\_HIST table to match the EMP\_DATA table row. If a row in the EMP\_DATA table does not match, insert it into the EMP\_HIST table. Rows are considered matching when the employee's first and last names are identical.
- Retrieve the rows from EMP\_HIST after the merge.

	FIRST_NAME	LAST_NAME	EMAIL
1	Ellen	Abel	EABEL
2	Sundar	Ande	SANDE
3	Mozhe	Atkinson	MATKINSO
4	David	Austin	DAUSTIN
5	Hermann	Baer	HBAER
6	Shelli	Baida	SBAIDA
7	Amit	Banda	ABANDA
8	Elizabeth	Bates	EBATES
9	Sarah	Bell	SBELL
10	David	Bernstein	DBERNSTE
11	Laura	Bissot	LBISSOT

11. Create the EMP3 table by using the lab\_04\_11.sql script. In the EMP3 table, change the department for Kochhar to 60 and commit your change. Next, change the department for Kochhar to 50 and commit your change. Track the changes to Kochhar by using the Row Versions feature.

	START_DATE	END_DATE	DEPARTMENT_ID
1	18-JUN-09 06.04.26.000000000 PM	(null)	50
2	18-JUN-09 06.04.26.000000000 PM	18-JUN-09 06.04.26.000000000 PM	60
3	(null)	18-JUN-09 06.04.26.000000000 PM	90

## ***Practice Solutions 4-1: Manipulating Large Data Sets***

1. Run the `lab_04_01.sql` script in the lab folder to create the `SAL_HISTORY` table.
2. Display the structure of the `SAL_HISTORY` table.

```
DESC sal_history
```

3. Run the `lab_04_03.sql` script in the lab folder to create the `MGR_HISTORY` table.
4. Display the structure of the `MGR_HISTORY` table.

```
DESC mgr_history
```

5. Run the `lab_04_05.sql` script in the lab folder to create the `SPECIAL_SAL` table.
6. Display the structure of the `SPECIAL_SAL` table.

```
DESC special_sal
```

7. a) Write a query to do the following:
  - Retrieve details such as the employee ID, hire date, salary, and manager ID of those employees whose employee ID is less than 125 from the `EMPLOYEES` table.
  - If the salary is more than \$20,000, insert details such as the employee ID and salary into the `SPECIAL_SAL` table.
  - Insert details such as the employee ID, hire date, and salary into the `SAL_HISTORY` table.
  - Insert details such as the employee ID, manager ID, and salary into the `MGR_HISTORY` table.

## Practice Solutions 4-1: Manipulating Large Data Sets (continued)

```
INSERT ALL
WHEN SAL > 20000 THEN
INTO special_sal VALUES (EMPID, SAL)
ELSE
INTO sal_history VALUES (EMPID, HIREDATE, SAL)
INTO mgr_history VALUES (EMPID, MGR, SAL)
SELECT employee_id EMPID, hire_date HIREDATE,
salary SAL, manager_id MGR
FROM employees
WHERE employee_id < 125;
```

- b) Display the records from the SPECIAL\_SAL table.

```
SELECT * FROM special_sal;
```

- c) Display the records from the SAL\_HISTORY table.

```
SELECT * FROM sal_history;
```

- d) Display the records from the MGR\_HISTORY table.

```
SELECT * FROM mgr_history;
```

8. a) Run the lab\_04\_08a.sql script in the lab folder to create the SALES\_WEEK\_DATA table.
- b) Run the lab\_04\_08b.sql script in the lab folder to insert records into the SALES\_WEEK\_DATA table.
- c) Display the structure of the SALES\_WEEK\_DATA table.

```
DESC sales_week_data
```

- d) Display the records from the SALES\_WEEK\_DATA table.

```
SELECT * FROM SALES_WEEK_DATA;
```

## Practice Solutions 4-1: Manipulating Large Data Sets (continued)

e) Run the lab\_04\_08\_e.sql script in the lab folder to create the EMP\_SALES\_INFO table.

f) Display the structure of the EMP\_SALES\_INFO table.

```
DESC emp_sales_info
```

g) Write a query to do the following:

- Retrieve details such as the employee ID, week ID, sales quantity on Monday, sales quantity on Tuesday, sales quantity on Wednesday, sales quantity on Thursday, and sales quantity on Friday from the SALES\_WEEK\_DATA table.
- Build a transformation such that each record retrieved from the SALES\_WEEK\_DATA table is converted into multiple records for the EMP\_SALES\_INFO table.

**Hint:** Use a pivoting INSERT statement.

```
INSERT ALL
  INTO emp_sales_info VALUES (id, week_id, QTY_MON)
  INTO emp_sales_info VALUES (id, week_id, QTY_TUE)
  INTO emp_sales_info VALUES (id, week_id, QTY_WED)
  INTO emp_sales_info VALUES (id, week_id, QTY_THUR)
  INTO emp_sales_info VALUES (id, week_id, QTY_FRI)
SELECT ID, week_id, QTY_MON, QTY_TUE, QTY_WED,
       QTY_THUR, QTY_FRI FROM sales_week_data;
```

h) Display the records from the EMP\_SALES\_INFO table.

```
SELECT * FROM emp_sales_info;
```

## **Practice Solutions 4-1: Manipulating Large Data Sets (continued)**

9. You have the data of past employees stored in a flat file called `emp.dat`. You want to store the names and email IDs of all employees past and present in a table. To do this, first create an external table called `EMP_DATA` using the `emp.dat` source file in the `emp_dir` directory. You can use the script in `lab_04_09.sql` to do this.

```
CREATE TABLE emp_data
  (first_name  VARCHAR2(20)
  ,last_name   VARCHAR2(20)
  , email      VARCHAR2(30)
  )
ORGANIZATION EXTERNAL
(
  TYPE oracle_loader
  DEFAULT DIRECTORY emp_dir
  ACCESS PARAMETERS
  (
    RECORDS DELIMITED BY NEWLINE CHARACTERSET US7ASCII
    NOBADFILE
    NOLOGFILE
    FIELDS
    ( first_name POSITION ( 1:20) CHAR
    , last_name POSITION (22:41) CHAR
    , email      POSITION (43:72) CHAR )
  )
  LOCATION ('emp.dat') ) ;
```

10. Next, run the `lab_04_10.sql` script to create the `EMP_HIST` table.
- a) Increase the size of the email column to 45.

```
ALTER TABLE emp_hist MODIFY email varchar(45);
```

- b) Merge the data in the `EMP_DATA` table created in the last lab into the data in the `EMP_HIST` table. Assume that the data in the external `EMP_DATA` table is the most up-to-date. If a row in the `EMP_DATA` table matches the `EMP_HIST` table, update the email column of the `EMP_HIST` table to match the `EMP_DATA` table row. If a row in the `EMP_DATA` table does not match, insert it into the `EMP_HIST` table. Rows are considered matching when the employee's first and last names are identical.

```
MERGE INTO EMP_HIST f USING EMP_DATA h
ON (f.first_name = h.first_name
AND f.last_name = h.last_name)
```

## Practice Solutions 4-1: Manipulating Large Data Sets (continued)

```
WHEN MATCHED THEN
  UPDATE SET f.email = h.email
WHEN NOT MATCHED THEN
  INSERT (f.first_name
        , f.last_name
        , f.email)
VALUES (h.first_name
      , h.last_name
      , h.email);
```

- c) Retrieve the rows from EMP\_HIST after the merge.

```
SELECT * FROM emp_hist;
```

11. Create the EMP3 table using the lab\_04\_11.sql script. In the EMP3 table, change the department for Kochhar to 60 and commit your change. Next, change the department for Kochhar to 50 and commit your change. Track the changes to Kochhar using the Row Versions feature.

```
UPDATE emp3 SET department_id = 60
WHERE last_name = 'Kochhar';
COMMIT;
UPDATE emp3 SET department_id = 50
WHERE last_name = 'Kochhar';
COMMIT;
```

```
SELECT VERSIONS_STARTTIME "START_DATE",
       VERSIONS_ENDTIME "END_DATE", DEPARTMENT_ID
FROM EMP3
      VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
WHERE LAST_NAME = 'Kochhar';
```




## Practices and Solutions for Lesson 5

### Practice 5-1: Managing Data in Different Time Zones


In this practice, you display time zone offsets, `CURRENT_DATE`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP`. You also set time zones and use the `EXTRACT` function.

1. Alter the session to set `NLS_DATE_FORMAT` to `DD-MON-YYYY HH24:MI:SS`.
2. a. Write queries to display the time zone offsets (`TZ_OFFSET`) for the following time zones.


- *US/Pacific-New*

 <code>TZ_OFFSET('US/PACIFIC-NEW')</code>
1 -07:00

- *Singapore*




 <code>TZ_OFFSET('SINGAPORE')</code>
1 +08:00

- *Egypt*

 <code>TZ_OFFSET('EGYPT')</code>
1 +03:00



- b. Alter the session to set the `TIME_ZONE` parameter value to the time zone offset of US/Pacific-New.
- c. Display `CURRENT_DATE`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP` for this session.
- d. Alter the session to set the `TIME_ZONE` parameter value to the time zone offset of Singapore.
- e. Display `CURRENT_DATE`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP` for this session.

**Note:** The output might be different based on the date when the command is executed.

 <code>CURRENT_DATE</code>	 <code>CURRENT_TIMESTAMP</code>	 <code>LOCALTIMESTAMP</code>
1 23-JUN-2009 15:12:08	23-JUN-09 03.12.08.000000000 PM +08:00	23-JUN-09 03.12.08.000000000 PM

**Note:** Observe in the preceding practice that `CURRENT_DATE`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP` are sensitive to the session time zone.

3. Write a query to display `DBTIMEZONE` and `SESSIONTIMEZONE`.

 <code>DBTIMEZONE</code>	 <code>SESSIONTIMEZONE</code>
1 +00:00	+08:00

4. Write a query to extract the `YEAR` from the `HIRE_DATE` column of the `EMPLOYEES` table for those employees who work in department 80.

## Practice 5-1: Managing Data in Different Time Zones (continued)

	LAST_NAME	EXTRACT(YEARFROMHIRE_DATE)
1	Russell	1996
2	Partners	1997
3	Errazuriz	1997
4	Cambrault	1999
5	Zlotkey	2000
6	Tucker	1997
7	Bernstein	1997

5. Alter the session to set NLS\_DATE\_FORMAT to DD-MON-YYYY.
6. Examine and run the lab\_05\_06.sql script to create the SAMPLE\_DATES table and populate it.
  - a. Select from the table and view the data.

	DATE_COL
1	23-JUN-2009

- b. Modify the data type of the DATE\_COL column and change it to TIMESTAMP. Select from the table to view the data.

	DATE_COL
1	23-JUN-09 02.14.52.000000000 PM

- c. Try to modify the data type of the DATE\_COL column and change it to TIMESTAMP WITH TIME ZONE. What happens?
7. Create a query to retrieve last names from the EMPLOYEES table and calculate the review status. If the year hired was 1998, display Needs Review for the review status; otherwise, display not this year! Name the review status column Review. Sort the results by the HIRE\_DATE column.  
**Hint:** Use a CASE expression with the EXTRACT function to calculate the review status.

	LAST_NAME	Review
1	King	not this year!
2	Whalen	not this year!
3	Kochhar	not this year!
4	Hunold	not this year!
5	Ernst	not this year!
6	De Haan	not this year!
7	Mavris	not this year!

...

### ***Practice 5-1: Managing Data in Different Time Zones (continued)***

8. Create a query to print the last names and the number of years of service for each employee. If the employee has been employed for five or more years, print 5 years of service. If the employee has been employed for 10 or more years, print 10 years of service. If the employee has been employed for 15 or more years, print 15 years of service. If none of these conditions match, print maybe next year! Sort the results by the HIRE\_DATE column. Use the EMPLOYEES table.

**Hint:** Use CASE expressions and TO\_YMINTERVAL.

	LAST_NAME		HIRE_DATE		SYSDATE		Awards
1	OConnell		21-JUN-1999		23-JUN-2009		10 years of service
2	Grant		13-JAN-2000		23-JUN-2009		5 years of service
3	Whalen		17-SEP-1987		23-JUN-2009		15 years of service
4	Hartstein		17-FEB-1996		23-JUN-2009		10 years of service
5	Fay		17-AUG-1997		23-JUN-2009		10 years of service
6	Mavris		07-JUN-1994		23-JUN-2009		15 years of service

...

## Practice Solutions 5-1: Managing Data in Different Time Zones

1. Alter the session to set NLS\_DATE\_FORMAT to DD-MON-YYYY HH24:MI:SS.

```
ALTER SESSION SET NLS_DATE_FORMAT =  
'DD-MON-YYYY HH24:MI:SS';
```

2. a. Write queries to display the time zone offsets (TZ\_OFFSET) for the following time zones: *US/Pacific-New*, *Singapore*, and *Egypt*.

*US/Pacific-New*

```
SELECT TZ_OFFSET ('US/Pacific-New') from dual;
```

*Singapore*

```
SELECT TZ_OFFSET ('Singapore') from dual;
```

*Egypt*

```
SELECT TZ_OFFSET ('Egypt') from dual;
```

- b. Alter the session to set the TIME\_ZONE parameter value to the time zone offset of US/Pacific-New.

```
ALTER SESSION SET TIME_ZONE = '-7:00';
```

- c. Display CURRENT\_DATE, CURRENT\_TIMESTAMP, and LOCALTIMESTAMP for this session.

**Note:** The output may be different based on the date when the command is executed.

```
SELECT CURRENT_DATE, CURRENT_TIMESTAMP,  
LOCALTIMESTAMP FROM DUAL;
```

- d. Alter the session to set the TIME\_ZONE parameter value to the time zone offset of Singapore.

```
ALTER SESSION SET TIME_ZONE = '+8:00';
```

- e. Display CURRENT\_DATE, CURRENT\_TIMESTAMP, and LOCALTIMESTAMP for this session.

**Note:** The output might be different, based on the date when the command is executed.

```
SELECT CURRENT_DATE, CURRENT_TIMESTAMP,  
LOCALTIMESTAMP FROM DUAL;
```

## ***Practice Solutions 5-1: Managing Data in Different Time Zones (continued)***

**Note:** Observe in the preceding practice that `CURRENT_DATE`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP` are all sensitive to the session time zone.

3. Write a query to display `DBTIMEZONE` and `SESSIONTIMEZONE`.

```
SELECT DBTIMEZONE, SESSIONTIMEZONE
FROM DUAL;
```

4. Write a query to extract `YEAR` from the `HIRE_DATE` column of the `EMPLOYEES` table for those employees who work in department 80.

```
SELECT last_name, EXTRACT (YEAR FROM HIRE_DATE)
FROM employees
WHERE department_id = 80;
```

5. Alter the session to set `NLS_DATE_FORMAT` to `DD-MON-YYYY`.

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY';
```

6. Examine and run the `lab_05_06.sql` script to create the `SAMPLE_DATES` table and populate it.

- a. Select from the table and view the data.

```
SELECT * FROM sample_dates;
```

- b. Modify the data type of the `DATE_COL` column and change it to `TIMESTAMP`.  
Select from the table to view the data.

```
ALTER TABLE sample_dates MODIFY date_col TIMESTAMP;
SELECT * FROM sample_dates;
```

- c. Try to modify the data type of the `DATE_COL` column and change it to `TIMESTAMP WITH TIME ZONE`. What happens?

```
ALTER TABLE sample_dates MODIFY date_col
TIMESTAMP WITH TIME ZONE;
```

## **Practice Solutions 5-1: Managing Data in Different Time Zones (continued)**

You are unable to change the data type of the DATE\_COL column because the Oracle server does not permit you to convert from TIMESTAMP to TIMESTAMP WITH TIMEZONE by using the ALTER statement.

7. Create a query to retrieve last names from the EMPLOYEES table and calculate the review status. If the year hired was 1998, display Needs Review for the review status; otherwise, display not this year! Name the review status column Review. Sort the results by the HIRE\_DATE column.

**Hint:** Use a CASE expression with the EXTRACT function to calculate the review status.

```
SELECT e.last_name
       , (CASE extract(year from e.hire_date)
           WHEN 1998 THEN 'Needs Review'
           ELSE 'not this year!'
         END ) AS "Review "
FROM   employees e
ORDER BY e.hire_date;
```

8. Create a query to print the last names and the number of years of service for each employee. If the employee has been employed five or more years, print 5 years of service. If the employee has been employed 10 or more years, print 10 years of service. If the employee has been employed 15 or more years, print 15 years of service. If none of these conditions match, print maybe next year! Sort the results by the HIRE\_DATE column. Use the EMPLOYEES table.

**Hint:** Use CASE expressions and TO\_YMINTERVAL.

```
SELECT e.last_name, hire_date, sysdate,
       (CASE
         WHEN (sysdate -TO_YMINTERVAL('15-0'))>=
              hire_date THEN      '15 years of service'
         WHEN (sysdate -TO_YMINTERVAL('10-0'))>= hire_date
              THEN      '10 years of service'
         WHEN (sysdate - TO_YMINTERVAL('5-0'))>= hire_date
              THEN '5 years of service'
         ELSE 'maybe next year!'
       END) AS "Awards"
FROM   employees e;
```

## Practices and Solutions for Lesson 6

### Practice 6-1: Retrieving Data by Using Subqueries

In this practice, you write multiple-column subqueries, and correlated and scalar subqueries. You also solve problems by writing the WITH clause.

1. Write a query to display the last name, department number, and salary of any employee whose department number and salary both match the department number and salary of any employee who earns a commission.

	LAST_NAME	DEPARTMENT_ID	SALARY
1	Russell	80	14000
2	Partners	80	13500
3	Errazuriz	80	12000

2. Display the last name, department name, and salary of any employee whose salary and commission match the salary and commission of any employee located in location ID 1700.

	LAST_NAME	DEPARTMENT_NAME	SALARY
1	Whalen	Administration	4400
2	Higgins	Accounting	12000
3	Greenberg	Finance	12000
4	Gietz	Accounting	8300

3. Create a query to display the last name, hire date, and salary for all employees who have the same salary and commission as Kochhar.

**Note:** Do not display Kochhar in the result set.

	LAST_NAME	HIRE_DATE	SALARY
1	De Haan	13-JAN-1993	17000

4. Create a query to display the employees who earn a salary that is higher than the salary of all the sales managers (JOB\_ID = 'SA\_MAN'). Sort the results from the highest to the lowest.

### Practice 6-1: Retrieving Data by Using Subqueries (continued)

	LAST_NAME	JOB_ID	SALARY
1	King	AD_PRES	24000
2	De Haan	AD_VP	17000
3	Kochhar	AD_VP	17000

5. Display details such as the employee ID, last name, and department ID of those employees who live in cities the names of which begin with *T*.

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	202	Fay	20
2	201	Hartstein	20

6. Write a query to find all employees who earn more than the average salary in their departments. Display last name, salary, department ID, and the average salary for the department. Sort by average salary and round to two decimals. Use aliases for the columns retrieved by the query as shown in the sample output.

	ENAME	SALARY	DEPTNO	DEPT_AVG
1	Fripp	8200	50	3475.5555555555555555555555555556
2	Kaufling	7900	50	3475.5555555555555555555555555556
3	Chung	3800	50	3475.5555555555555555555555555556
4	Mourgos	5800	50	3475.5555555555555555555555555556
5	Bell	4000	50	3475.5555555555555555555555555556
6	Rajs	3500	50	3475.5555555555555555555555555556
7	Bull	4100	50	3475.5555555555555555555555555556
8	Everett	3900	50	3475.5555555555555555555555555556

7. Find all employees who are not supervisors.
  - a. First, do this using the NOT EXISTS operator.



## Practice 6-1: Retrieving Data by Using Subqueries (continued)

R 2	LAST_NAME
1	Abel
2	Ande
3	Atkinson
4	Austin
5	Baer
6	Baida

- b. Can this be done by using the NOT IN operator? How, or why not?
8. Write a query to display the last names of the employees who earn less than the average salary in their departments.

R 2	LAST_NAME
1	Chen
2	Sciarra
3	Urman
4	Popp
5	Khoo
6	Baida

9. Write a query to display the last names of the employees who have one or more coworkers in their departments with later hire dates but higher salaries.

R 2	LAST_NAME
1	Vargas
2	Patel
3	Olson
4	Marlow
5	Landry
6	Perkins

10. Write a query to display the employee ID, last names, and department names of all the employees.

**Note:** Use a scalar subquery to retrieve the department name in the SELECT statement.



	<b>EMPLOYEE_ID</b>	<b>LAST_NAME</b>	<b>DEPARTMENT</b>
1	205	Higgins	Accounting
2	206	Gietz	Accounting
3	200	Whalen	Administration
4	100	King	Executive
5	101	Kochhar	Executive

...

### ***Practice 6-1: Retrieving Data by Using Subqueries (continued)***

105	196 Walsh	Shipping
106	197 Feeney	Shipping
107	178 Grant	(null)

11. Write a query to display the department names of those departments whose total salary cost is above one-eighth (1/8) of the total salary cost of the whole company. Use the WITH clause to write this query. Name the query SUMMARY.

	DEPARTMENT_NAME		DEPT_TOTAL
1	Sales		304500
2	Shipping		156400

## Practice Solutions 6-1: Retrieving Data by Using Subqueries

1. Write a query to display the last name, department number, and salary of any employee whose department number and salary match the department number and salary of any employee who earns a commission.

```
SELECT last_name, department_id, salary
FROM   employees
WHERE  (salary, department_id) IN
      (SELECT salary, department_id
       FROM   employees
       WHERE  commission_pct IS NOT NULL);
```

2. Display the last name, department name, and salary of any employee whose salary and commission match the salary and commission of any employee located in location ID1700.

```
SELECT e.last_name, d.department_name, e.salary
FROM   employees e, departments d
WHERE  e.department_id = d.department_id
AND    (salary, NVL(commission_pct,0)) IN
      (SELECT salary, NVL(commission_pct,0)
       FROM   employees e, departments d
       WHERE  e.department_id = d.department_id
       AND d.location_id = 1700);
```

3. Create a query to display the last name, hire date, and salary for all employees who have the same salary and commission as Kochhar.

**Note:** Do not display Kochhar in the result set.

```
SELECT last_name, hire_date, salary
FROM   employees
WHERE  (salary, NVL(commission_pct,0)) IN
      (SELECT salary, NVL(commission_pct,0)
       FROM   employees
       WHERE  last_name = 'Kochhar')
AND last_name != 'Kochhar';
```

4. Create a query to display the employees who earn a salary that is higher than the salary of all the sales managers (JOB\_ID = 'SA\_MAN'). Sort the results on salary from the highest to the lowest.

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  salary > ALL
      (SELECT salary
       FROM   employees
       WHERE  job_id = 'SA_MAN')
ORDER BY salary DESC;
```

## ***Practice Solutions 6-1: Retrieving Data by Using Subqueries (continued)***

5. Display details such as the employee ID, last name, and department ID of those employees who live in cities the names of which begin with *T*.

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  department_id IN (SELECT department_id
                        FROM departments
                        WHERE location_id IN
                              (SELECT location_id
                               FROM locations
                               WHERE city LIKE 'T%'));
```

6. Write a query to find all employees who earn more than the average salary in their departments. Display last name, salary, department ID, and the average salary for the department. Sort by average salary. Use aliases for the columns retrieved by the query as shown in the sample output.

```
SELECT e.last_name ename, e.salary salary,
       e.department_id deptno, AVG(a.salary) dept_avg
FROM   employees e, employees a
WHERE  e.department_id = a.department_id
AND    e.salary > (SELECT AVG(salary)
                  FROM   employees
                  WHERE  department_id = e.department_id )
GROUP BY e.last_name, e.salary, e.department_id
ORDER BY AVG(a.salary);
```

## Practice Solutions 6-1: Retrieving Data by Using Subqueries (continued)

7. Find all employees who are not supervisors.
  - a. First, do this by using the NOT EXISTS operator.

```
SELECT outer.last_name
FROM   employees outer
WHERE  NOT EXISTS (SELECT 'X'
                   FROM employees inner
                   WHERE inner.manager_id =
                       outer.employee_id);
```

- b. Can this be done by using the NOT IN operator? How, or why not?

```
SELECT outer.last_name
FROM   employees outer
WHERE  outer.employee_id
NOT IN (SELECT inner.manager_id
       FROM   employees inner);
```

This alternative solution is not a good one. The subquery picks up a NULL value, so the entire query returns no rows. The reason is that all conditions that compare a NULL value result in NULL. Whenever NULL values are likely to be part of the value set, *do not* use NOT IN as a substitute for NOT EXISTS.

8. Write a query to display the last names of the employees who earn less than the average salary in their departments.

```
SELECT last_name
FROM   employees outer
WHERE  outer.salary < (SELECT AVG(inner.salary)
                     FROM employees inner
                     WHERE inner.department_id
                         = outer.department_id);
```

## ***Practice Solutions 6-1: Retrieving Data by Using Subqueries (continued)***

9. Write a query to display the last names of employees who have one or more coworkers in their departments with later hire dates but higher salaries.

```
SELECT last_name
FROM employees outer
WHERE EXISTS (SELECT 'X'
              FROM employees inner
              WHERE inner.department_id =
                    outer.department_id
              AND inner.hire_date > outer.hire_date
              AND inner.salary > outer.salary);
```

10. Write a query to display the employee ID, last names, and department names of all employees.

**Note:** Use a scalar subquery to retrieve the department name in the SELECT statement.

```
SELECT employee_id, last_name,
       (SELECT department_name
        FROM departments d
        WHERE e.department_id =
              d.department_id ) department
FROM employees e
ORDER BY department;
```

11. Write a query to display the department names of those departments whose total salary cost is above one-eighth (1/8) of the total salary cost of the whole company. Use the WITH clause to write this query. Name the query SUMMARY.

```
WITH
summary AS (
  SELECT d.department_name, SUM(e.salary) AS dept_total
  FROM employees e, departments d
  WHERE e.department_id = d.department_id
  GROUP BY d.department_name)
SELECT department_name, dept_total
FROM summary
WHERE dept_total > ( SELECT SUM(dept_total) * 1/8
                   FROM summary )
ORDER BY dept_total DESC;
```

## Practices and Solutions for Lesson 7

### Practice 7-1: Regular Expression Support

In this practice, you use regular expressions functions to search for, replace, and manipulate data. You also create a new CONTACTS table and add a CHECK constraint to the p\_number column to ensure that phone numbers are entered into the database in a specific standard format.

1. Write a query to search the EMPLOYEES table for all the employees whose first names start with “Ki” or “Ko.”

	FIRST_NAME	LAST_NAME
1	Janette	King
2	Steven	King
3	Neena	Kochhar

2. Create a query that removes the spaces in the STREET\_ADDRESS column of the LOCATIONS table in the display. Use “Street Address” as the column heading.

	Street Address
1	1297ViaColadiRie
2	93091CalledeIlaTesta
3	2017Shinjuku-ku
4	9450Kamiya-cho
5	2014JabberwockyRd
6	2011InteriorsBlvd
7	2007ZagoraSt

3. Create a query that displays “St” replaced by “Street” in the STREET\_ADDRESS column of the LOCATIONS table. Be careful that you do not affect any rows that already have “Street” in them. Display only those rows that are affected.

	REGEXP_REPLACE(STREET_ADDRESS,'ST\$','STREET')
1	2007 Zagora Street
2	6092 Boxwood Street
3	12-98 Victoria Street
4	8204 Arthur Street

4. Create a contacts table and add a check constraint to the p\_number column to enforce the following format mask to ensure that phone numbers are entered into the database in the following standard format: (XXX) XXX-XXXX. The table should have the following columns:

- l\_name varchar2(30)
- p\_number varchar2(30)

### Practice 7-1: Regular Expression Support (continued)

5. Run the SQL script `lab_07_05.sql` to insert the following seven phone numbers into the `contacts` table. Which numbers are added?

<b>l_name</b> Column Value	<b>p_number</b> Column Value
NULL	'(650) 555-5555'
NULL	'(215) 555-3427'
NULL	'650 555-5555'
NULL	'650 555 5555'
NULL	'650-555-5555'
NULL	'(650)555-5555'
NULL	' (650) 555-5555'

6. Write a query to find the number of occurrences of the DNA pattern `ctc` in the string `gtctcgtctcgttctgtctgtcgttctg`. Ignore case-sensitivity.

COUNT_DNA	
1	2



## Practice Solutions 7-1: Regular Expression Support

1. Write a query to search the EMPLOYEES table for all employees whose first names start with “Ki” or “Ko.”

```
SELECT first_name, last_name
FROM employees
WHERE REGEXP_LIKE (last_name, '^K(i|o).');
```

2. Create a query that removes the spaces in the STREET\_ADDRESS column of the LOCATIONS table in the display. Use “Street Address” as the column heading.

```
SELECT regexp_replace (street_address, ' ', '') AS "Street
Address"
FROM locations;
```

3. Create a query that displays “St” replaced by “Street” in the STREET\_ADDRESS column of the LOCATIONS table. Be careful that you do not affect any rows that already have “Street” in them. Display only those rows, which are affected.

```
SELECT regexp_replace (street_address, 'St$',
'Street')
FROM locations
WHERE regexp_like (street_address, 'St');
```

4. Create a contacts table and add a check constraint to the p\_number column to enforce the following format mask to ensure that phone numbers are entered into the database in the following standard format: (XXX) XXX-XXXX. The table should have the following columns:

- l\_name varchar2(30)
- p\_number varchar2 (30)

```
CREATE TABLE contacts
(
  l_name      VARCHAR2(30),
  p_number    VARCHAR2(30)
  CONSTRAINT p_number_format
  CHECK ( REGEXP_LIKE ( p_number, '^\\(\\d{3}\\) \\d{3}-
\\d{4}$' ) )
);
```

## ***Practice Solutions 7-1: Regular Expression Support (continued)***

5. Run the `lab_07_05.sql` SQL script to insert the following seven phone numbers into the `contacts` table. Which numbers are added?  
**Only the first two `INSERT` statements use a format that conforms to the `c_contacts_pnf` constraint; the remaining statements generate `CHECK` constraint errors.**
6. Write a query to find the number of occurrences of the DNA pattern `ctc` in the string  
`gtctcgtctcgttctgtctgtcgttctg`. Use the alias `Count_DNA`. Ignore case-sensitivity.

```
SELECT REGEXP_COUNT('gtctcgtctcgttctgtctgtcgttctg','ctc')
AS Count_DNA
FROM dual;
```

## ***Practice Solutions 7-1: Regular Expression Support (continued)***

---

# **Appendix AP**

## **Additional Practices and Solutions**

---

## Table of Contents

Additional Practices .....	3
Practice 1-1 .....	4
Practice Solutions 1-1 .....	12
Case Study .....	17
Practice 2-1 .....	19
Practice Solutions 2-1 .....	27



## Practice 1-1

These exercises can be used for extra practice after you have discussed the following topics: basic SQL SELECT statement, basic SQL Developer commands, and SQL functions.

- 1) The HR department needs to find data for all the clerks who were hired after the year 1997.

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
1	143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98	ST_CLERK	2600
2	144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-98	ST_CLERK	2500

- 2) The HR department needs a report of employees who earn commission. Show the last name, job, salary, and commission of those employees. Sort the data by salary in descending order.

	LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
1	Abel	SA_REP	11000	0.3
2	Zlotkey	SA_MAN	10500	0.2
3	Taylor	SA_REP	8600	0.2
4	Grant	SA_REP	7000	0.15

- 3) For budgeting purposes, the HR department needs a report on projected raises. The report should display those employees who have no commission, but who have a 10% raise in salary (round off the salaries).

	New salary
1	The salary of King after a 10% raise is 26400
2	The salary of Kochhar after a 10% raise is 18700
3	The salary of De Haan after a 10% raise is 18700
4	The salary of Hunold after a 10% raise is 9900
5	The salary of Ernst after a 10% raise is 6600
6	The salary of Lorentz after a 10% raise is 4620
7	The salary of Mourgos after a 10% raise is 6380
8	The salary of Rajs after a 10% raise is 3850
9	The salary of Davies after a 10% raise is 3410
10	The salary of Matos after a 10% raise is 2860
11	The salary of Vargas after a 10% raise is 2750
12	The salary of Whalen after a 10% raise is 4840
13	The salary of Hartstein after a 10% raise is 14300
14	The salary of Fay after a 10% raise is 6600
15	The salary of Higgins after a 10% raise is 13200
16	The salary of Gietz after a 10% raise is 9130

### Practice 1-1 (continued)

- 4) Create a report of employees and their length of employment. Show the last names of all the employees together with the number of years and the number of completed months that they have been employed. Order the report by the length of their employment. The employee who has been employed the longest should appear at the top of the list.

	LAST_NAME	YEARS	MONTHS
1	King	22	0
2	Whalen	21	9
3	Kochhar	19	9
4	Hunold	19	6
5	Ernst	18	1
6	De Haan	16	6
7	Higgins	15	1
8	Gietz	15	1
9	Rajs	13	8
10	Hartstein	13	4
11	Abel	13	2
12	Davies	12	5
13	Fay	11	10
14	Matos	11	4
15	Taylor	11	3
16	Vargas	11	0
17	Lorentz	10	5
18	Grant	10	1
19	Mourgos	9	7
20	Zlotkey	9	5

- 5) Show those employees who have a last name starting with the letters “J,” “K,” “L,” or “M.”

	LAST_NAME
1	King
2	Kochhar
3	Lorentz
4	Matos
5	Mourgos



### Practice 1-1 (continued)

- 6) Create a report that displays all employees, and indicate with the words *Yes* or *No* whether they receive a commission. Use the DECODE expression in your query.

	A Z	LAST_NAME	A Z	SALARY	A Z	COMMISSION
1		King		24000	No	
2		Kochhar		17000	No	
3		De Haan		17000	No	
4		Hunold		9000	No	
5		Ernst		6000	No	
6		Lorentz		4200	No	
7		Mourgos		5800	No	
8		Rajs		3500	No	
9		Davies		3100	No	
10		Matos		2600	No	
11		Vargas		2500	No	
12		Zlotkey		10500	Yes	
13		Abel		11000	Yes	
14		Taylor		8600	Yes	
15		Grant		7000	Yes	
16		Whalen		4400	No	
17		Hartstein		13000	No	
18		Fay		6000	No	
19		Higgins		12000	No	
20		Gietz		8300	No	

These exercises can be used for extra practice after you have discussed the following topics: basic SQL SELECT statement, basic SQL Developer commands, SQL functions, joins, and group functions.

- 7) Create a report that displays the department name, location ID, last name, job title, and salary of those employees who work in a specific location. Prompt the user for the location. For example, if the user enters 1800, these are the results:

	A Z	DEPARTMENT_NAME	A Z	LOCATION_ID	A Z	LAST_NAME	A Z	JOB_ID	A Z	SALARY
1		Marketing		1800		Hartstein		MK_MAN		13000
2		Marketing		1800		Fay		MK_REP		6000

- 8) Find the number of employees who have a last name that ends with the letter “n.” Create two possible solutions.

	A Z	COUNT(*)
1		3

### Practice 1-1 (continued)

- 9) Create a report that shows the name, location, and number of employees for each department. Make sure that the report also includes departments without employees.

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	COUNT(E.EMPLOYEE_ID)
1	80	Sales	2500	3
2	110	Accounting	1700	2
3	10	Administration	1700	1
4	60	IT	1400	3
5	20	Marketing	1800	2
6	90	Executive	1700	3
7	50	Shipping	1500	5
8	190	Contracting	1700	0

- 10) The HR department needs to find the job titles in departments 10 and 20. Create a report to display the job IDs for those departments.

JOB_ID
AD_ASST
MK_MAN
MK_REP

- 11) Create a report that displays the jobs that are found in the Administration and Executive departments. Also display the number of employees for these jobs. Show the job with the highest number of employees first.

JOB_ID	FREQUENCY
AD_VP	2
AD PRES	1
AD_ASST	1

These exercises can be used for extra practice after you have discussed the following topics: basic SQL SELECT statements, basic SQL Developer commands, SQL functions, joins, group functions, and subqueries.

- 12) Show all the employees who were hired in the first half of the month (before the 16th of the month).

LAST_NAME	HIRE_DATE
De Haan	13-JAN-93
Hunold	03-JAN-90
Lorentz	07-FEB-99
Matos	15-MAR-98
Vargas	09-JUL-98
Abel	11-MAY-96
Higgins	07-JUN-94
Gietz	07-JUN-94

### Practice 1-1 (continued)

- 13) Create a report that displays the following for all employees: last name, salary, and salary expressed in terms of thousands of dollars.

	A Z LAST_NAME	A Z SALARY	A Z THOUSANDS
1	King	24000	24
2	Kochhar	17000	17
3	De Haan	17000	17
4	Hunold	9000	9
5	Ernst	6000	6
6	Lorentz	4200	4
7	Mourgos	5800	5
8	Rajs	3500	3
9	Davies	3100	3
10	Matos	2600	2
11	Vargas	2500	2
12	Zlotkey	10500	10
13	Abel	11000	11
14	Taylor	8600	8
15	Grant	7000	7
16	Whalen	4400	4
17	Hartstein	13000	13
18	Fay	6000	6
19	Higgins	12000	12
20	Gietz	8300	8

- 14) Show all the employees who have managers with a salary higher than \$15,000. Show the following data: employee name, manager name, manager salary, and salary grade of the manager.

	A Z LAST_NAME	A Z MANAGER	A Z SALARY	A Z GRADE_LEVEL
1	De Haan	King	24000	E
2	Hartstein	King	24000	E
3	Higgins	Kochhar	17000	E
4	Hunold	De Haan	17000	E
5	Kochhar	King	24000	E
6	Mourgos	King	24000	E
7	Whalen	Kochhar	17000	E
8	Zlotkey	King	24000	E

## Practice 1-1 (continued)

- 15) Show the department number, name, number of employees, and average salary of all the departments, together with the names, salaries, and jobs of the employees working in each department.

	DEPARTMENT_ID	DEPARTMENT_NAME	EMPLOYEES	AVG_SAL	LAST_NAME	SALARY	JOB_ID
1	10	Administration	1	4400.00	Whalen	4400	AD_ASST
2	20	Marketing	2	9500.00	Hartstein	13000	MK_MAN
3	20	Marketing	2	9500.00	Fay	6000	MK_REP
4	50	Shipping	5	3500.00	Davies	3100	ST_CLERK
5	50	Shipping	5	3500.00	Matos	2600	ST_CLERK
6	50	Shipping	5	3500.00	Rajs	3500	ST_CLERK
7	50	Shipping	5	3500.00	Mourgos	5800	ST_MAN
8	50	Shipping	5	3500.00	Vargas	2500	ST_CLERK
9	60	IT	3	6400.00	Hunold	9000	IT_PROG
10	60	IT	3	6400.00	Lorentz	4200	IT_PROG
11	60	IT	3	6400.00	Ernst	6000	IT_PROG
12	80	Sales	3	10033.33	Zlotkey	10500	SA_MAN
13	80	Sales	3	10033.33	Taylor	8600	SA_REP
14	80	Sales	3	10033.33	Abel	11000	SA_REP
15	90	Executive	3	19333.33	Kochhar	17000	AD_VP
16	90	Executive	3	19333.33	De Haan	17000	AD_VP
17	90	Executive	3	19333.33	King	24000	AD PRES
18	110	Accounting	2	10150.00	Gietz	8300	AC_ACCOUNT
19	110	Accounting	2	10150.00	Higgins	12000	AC_MGR
20	(null)	(null)	0	No average	Grant	7000	SA_REP

- 16) Create a report to display the department number and lowest salary of the department with the highest average salary.

	DEPARTMENT_ID	MIN(SALARY)
1	90	17000

- 17) Create a report that displays departments where no sales representatives work. Include the department number, department name, manager ID, and the location in the output.

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	90	Executive	100	1700
6	110	Accounting	205	1700
7	190	Contracting	(null)	1700

## Practice 1-1 (continued)

18) Create the following statistical reports for the HR department: Include the department number, department name, and the number of employees working in each department that:

a) Employs fewer than three employees:

	DEPARTMENT_ID	DEPARTMENT_NAME	COUNT(*)
1	10	Administration	1
2	110	Accounting	2
3	20	Marketing	2

b) Has the highest number of employees:

	DEPARTMENT_ID	DEPARTMENT_NAME	COUNT(*)
1	50	Shipping	5

c) Has the lowest number of employees:



	DEPARTMENT_ID	DEPARTMENT_NAME	COUNT(*)
1	10	Administration	1

19) Create a report that displays the employee number, last name, salary, department number, and the average salary in their department for all employees.


	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	SALARY	AVG(S.SALARY)
1	149	Zlotkey	80	10500	10033.3333333333...
2	174	Abel	80	11000	10033.3333333333...
3	144	Vargas	50	2500	3500
4	101	Kochhar	90	17000	19333.3333333333...
5	100	King	90	24000	19333.3333333333...
6	103	Hunold	60	9000	6400
7	142	Davies	50	3100	3500
8	205	Higgins	110	12000	10150
9	104	Ernst	60	6000	6400
10	143	Matos	50	2600	3500
11	102	De Haan	90	17000	19333.3333333333...
12	107	Lorentz	60	4200	6400
13	141	Rajs	50	3500	3500
14	200	Whalen	10	4400	4400
15	202	Fay	20	6000	9500
16	176	Taylor	80	8600	10033.3333333333...
17	201	Hartstein	20	13000	9500
18	206	Gietz	110	8300	10150
19	124	Mourgos	50	5800	3500

### **Practice 1-1 (continued)**

20) Show all the employees who were hired on the day of the week on which the highest number of employees were hired.

	 LAST_NAME	 DAY
1	Ernst	TUESDAY
2	Mourgos	TUESDAY
3	Rajs	TUESDAY
4	Taylor	TUESDAY
5	Higgins	TUESDAY
6	Gietz	TUESDAY

21) Create an anniversary overview based on the hire date of the employees. Sort the anniversaries in ascending order.

	 LAST_NAME	 BIRTHDAY
1	Hunold	January 03
2	De Haan	January 13
3	Davies	January 29
4	Zlotkey	January 29
5	Lorentz	February 07
6	Hartstein	February 17
7	Matos	March 15
8	Taylor	March 24
9	Abel	May 11
10	Ernst	May 21
11	Grant	May 24
12	Higgins	June 07
13	Gietz	June 07
14	King	June 17
15	Vargas	July 09
16	Fay	August 17
17	Whalen	September 17
18	Kochhar	September 21
19	Rajs	October 17
20	Mourgos	November 16

## Practice Solutions 1-1

These exercises can be used for extra practice after you have discussed the following topics: basic SQL SELECT statement, basic SQL Developer commands, and SQL functions.

- 1) The HR department needs to find data for all of the clerks who were hired after the year 1997.

```
SELECT *
FROM   employees
WHERE  job_id = 'ST_CLERK'
AND    hire_date > '31-DEC-1997';
```

- 2) The HR department needs a report of employees who earn commission. Show the last name, job, salary, and commission of those employees. Sort the data by salary in descending order.

```
SELECT last_name, job_id, salary, commission_pct
FROM   employees
WHERE  commission_pct IS NOT NULL
ORDER BY salary DESC;
```

- 3) For budgeting purposes, the HR department needs a report on projected raises. The report should display those employees who do not get a commission but who have a 10% raise in salary (round off the salaries).

```
SELECT 'The salary of '||last_name||' after a 10% raise is '
      || ROUND(salary*1.10) "New salary"
FROM   employees
WHERE  commission_pct IS NULL;
```

- 4) Create a report of employees and their duration of employment. Show the last names of all employees together with the number of years and the number of completed months that they have been employed. Order the report by the duration of their employment. The employee who has been employed the longest should appear at the top of the list.

```
SELECT last_name,
       TRUNC(MONTHS_BETWEEN(SYSDATE, hire_date) / 12) YEARS,
       TRUNC(MOD(MONTHS_BETWEEN(SYSDATE, hire_date), 12))
       MONTHS
FROM   employees
ORDER BY years DESC, MONTHS desc;
```

- 5) Show those employees who have a last name starting with the letters “J,” “K,” “L,” or “M.”

```
SELECT last_name
FROM   employees
WHERE  SUBSTR(last_name, 1,1) IN ('J', 'K', 'L', 'M');
```

## Practice Solutions 1-1 (continued)

- 6) Create a report that displays all employees, and indicate with the words *Yes* or *No* whether they receive a commission. Use the DECODE expression in your query.

```
SELECT last_name, salary,
       decode(commission_pct, NULL, 'No', 'Yes') commission
FROM   employees;
```

These exercises can be used for extra practice after you have discussed the following topics: basic SQL SELECT statement, basic SQL Developer commands, SQL functions, joins, and group functions.

- 7) Create a report that displays the department name, location ID, name, job title, and salary of those employees who work in a specific location. Prompt the user for the location.

a) Enter 1800 for location\_id when prompted.

```
SELECT d.department_name, d.location_id, e.last_name,
       e.job_id, e.salary
FROM   employees e, departments d
WHERE  e.department_id = d.department_id
AND    d.location_id = &location_id;
```

- 8) Find the number of employees who have a last name that ends with the letter “n.” Create two possible solutions.

```
SELECT COUNT(*)
FROM   employees
WHERE  last_name LIKE '%n';
--or
SELECT COUNT(*)
FROM   employees
WHERE  SUBSTR(last_name, -1) = 'n';
```

- 9) Create a report that shows the name, location, and number of employees for each department. Make sure that the report also includes departments without employees.

```
SELECT d.department_id, d.department_name,
       d.location_id, COUNT(e.employee_id)
FROM   employees e RIGHT OUTER JOIN departments d
ON     e.department_id = d.department_id
GROUP BY d.department_id, d.department_name, d.location_id;
```

- 10) The HR department needs to find the job titles in departments 10 and 20. Create a report to display the job IDs for those departments.

```
SELECT DISTINCT job_id
FROM   employees
WHERE  department_id IN (10, 20);
```



## Practice Solutions 1-1 (continued)

- 11) Create a report that displays the jobs that are found in the Administration and Executive departments. Also display the number of employees for these jobs. Show the job with the highest number of employees first.

```
SELECT e.job_id, count(e.job_id) FREQUENCY
FROM   employees e JOIN departments d
ON     e.department_id = d.department_id
WHERE  d.department_name IN ('Administration', 'Executive')
GROUP BY e.job_id
ORDER BY FREQUENCY DESC;
```

These exercises can be used for extra practice after you have discussed the following topics: basic SQL SELECT statements, basic SQL Developer commands, SQL functions, joins, group functions, and subqueries.

- 12) Show all employees who were hired in the first half of the month (before the 16th of the month).

```
SELECT last_name, hire_date
FROM   employees
WHERE  TO_CHAR(hire_date, 'DD') < 16;
```

- 13) Create a report that displays the following for all employees: last name, salary, and salary expressed in terms of thousands of dollars.

```
SELECT last_name, salary, TRUNC(salary, -3)/1000 Thousands
FROM   employees;
```

- 14) Show all employees who have managers with a salary higher than \$15,000. Show the following data: employee name, manager name, manager salary, and salary grade of the manager.

```
SELECT e.last_name, m.last_name manager, m.salary,
       j.grade_level
FROM   employees e JOIN employees m
ON     e.manager_id = m.employee_id
JOIN   job_grades j
ON     m.salary BETWEEN j.lowest_sal AND j.highest_sal
AND    m.salary > 15000;
```

## Practice Solutions 1-1 (continued)

- 15) Show the department number, name, number of employees, and average salary of all departments, together with the names, salaries, and jobs of the employees working in each department.

```
SELECT  d.department_id, d.department_name,
        count(e1.employee_id) employees,
        NVL(TO_CHAR(AVG(e1.salary), '99999.99'), 'No average'
) avg_sal,
        e2.last_name, e2.salary, e2.job_id
FROM    departments d RIGHT OUTER JOIN employees e1
ON      d.department_id = e1.department_id
RIGHT OUTER JOIN employees e2
ON      d.department_id = e2.department_id
GROUP BY d.department_id, d.department_name, e2.last_name,
        e2.salary,
        e2.job_id
ORDER BY d.department_id, employees;
```

- 16) Create a report to display the department number and lowest salary of the department with the highest average salary.

```
SELECT department_id, MIN(salary)
FROM    employees
GROUP BY department_id
HAVING AVG(salary) = (SELECT MAX(AVG(salary))
                     FROM    employees
                     GROUP BY department_id);
```

- 17) Create a report that displays the departments where no sales representatives work. Include the department number, department name, and location in the output.

```
SELECT *
FROM    departments
WHERE   department_id NOT IN(SELECT department_id
                             FROM employees
                             WHERE job_id = 'SA_REP'
                             AND department_id IS NOT NULL);
```

- 18) Create the following statistical reports for the HR department: Include the department number, department name, and the number of employees working in each department that:

- a) Employs fewer than three employees:

```
SELECT d.department_id, d.department_name, COUNT(*)
FROM    departments d JOIN employees e
ON      d.department_id = e.department_id
GROUP BY d.department_id, d.department_name
HAVING COUNT(*) < 3;
```

## Practice Solutions 1-1 (continued)

b) Has the highest number of employees:

```
SELECT d.department_id, d.department_name, COUNT(*)
FROM   departments d JOIN employees e
ON     d.department_id = e.department_id
GROUP BY d.department_id, d.department_name
HAVING COUNT(*) = (SELECT MAX(COUNT(*))
                  FROM   employees
                  GROUP BY department_id);
```

c) Has the lowest number of employees:

```
SELECT d.department_id, d.department_name, COUNT(*)
FROM   departments d JOIN employees e
ON     d.department_id = e.department_id
GROUP BY d.department_id, d.department_name
HAVING COUNT(*) = (SELECT MIN(COUNT(*))
                  FROM   employees
                  GROUP BY department_id);
```

19) Create a report that displays the employee number, last name, salary, department number, and the average salary in their department for all employees.

```
SELECT e.employee_id, e.last_name, e.department_id, e.salary,
       AVG(s.salary)
FROM   employees e JOIN employees s
ON     e.department_id = s.department_id
GROUP BY e.employee_id, e.last_name, e.department_id,
       e.salary;
```

20) Show all employees who were hired on the day of the week on which the highest number of employees were hired.

```
SELECT last_name, TO_CHAR(hire_date, 'DAY') day
FROM   employees
WHERE  TO_CHAR(hire_date, 'Day') =
       (SELECT TO_CHAR(hire_date, 'Day')
        FROM   employees
        GROUP BY TO_CHAR(hire_date, 'Day')
        HAVING COUNT(*) = (SELECT MAX(COUNT(*))
                          FROM   employees
                          GROUP BY TO_CHAR(hire_date,
'Day')));
```

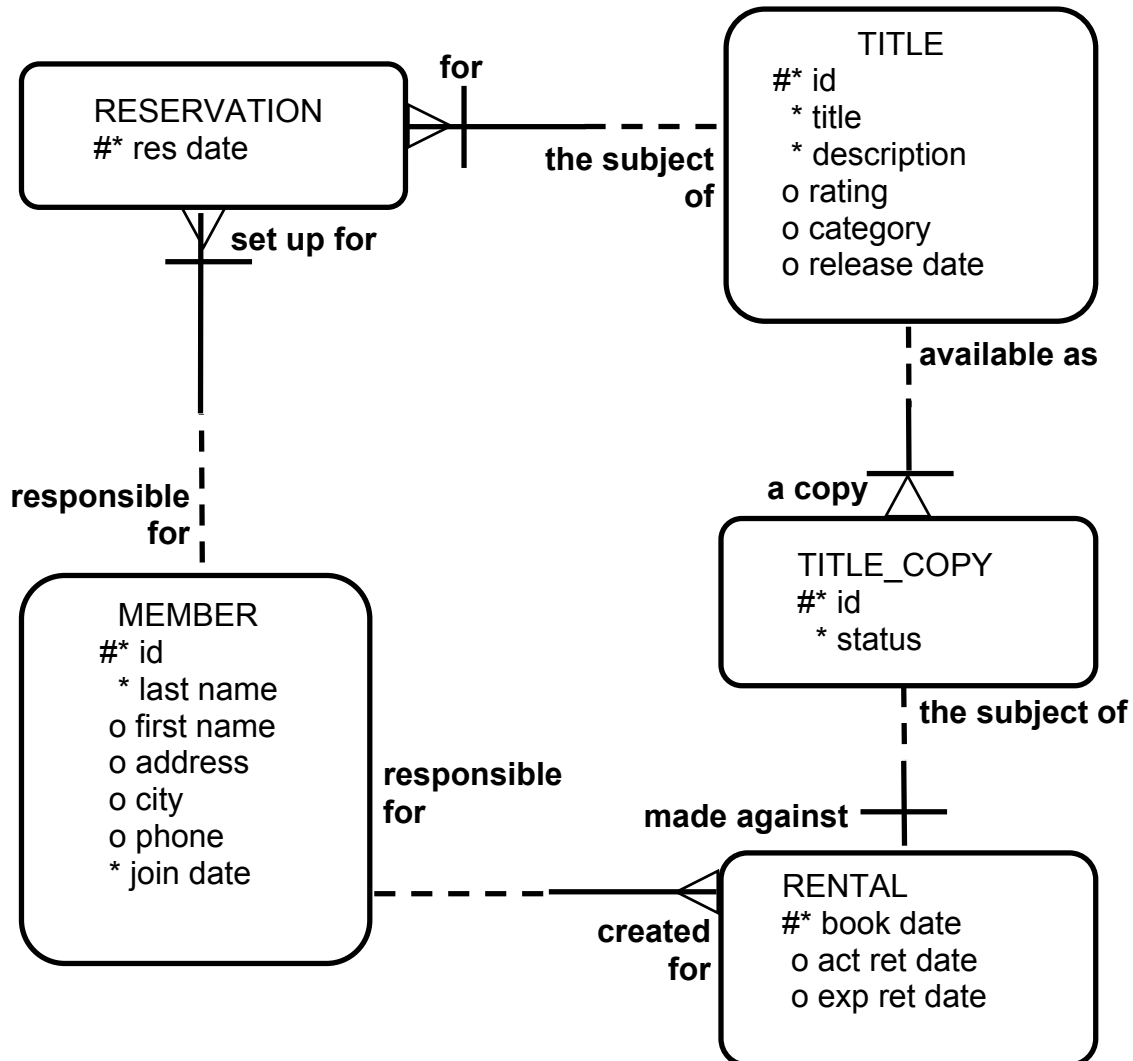
21) Create an anniversary overview based on the hire date of the employees. Sort the anniversaries in ascending order.

```
SELECT last_name, TO_CHAR(hire_date, 'Month DD') BIRTHDAY
FROM   employees
ORDER BY TO_CHAR(hire_date, 'DDD');
```

## Case Study

In this case study, you build a set of database tables for a video application. After you create the tables, you insert, update, and delete records in a video store database and generate a report. The database contains only the essential tables.

The following is a diagram of the entities and attributes for the video application:



**Note:** If you want to build the tables, you can execute the commands in the `buildtab.sql` script in SQL Developer. If you want to drop the tables, you can execute the commands in the `dropvid.sql` script in SQL Developer. Then you can execute the commands in the `buildvid.sql` script in SQL Developer to create and populate the tables.

All the three SQL scripts are present in the `/home/oracle/labs/sql1/labs` folder.

- If you use the `buildtab.sql` script to build the tables, start with step 4.

### ***Practice Solutions 1-1 (continued)***

- If you use the `dropvid.sql` script to remove the video tables, start with step 1.
- If you use the `buildvid.sql` script to build and populate the tables, start with step 6(b).

## Practice 2-1

- 1) Create the tables based on the following table instance charts. Choose the appropriate data types and be sure to add integrity constraints.

a) Table name: MEMBER

Column_ Name	MEMBER_ ID	LAST_ NAME	FIRST_NAME	ADDRESS	CITY	PHONE	JOIN _DATE
Key Type	PK						
Null/ Unique	NN,U	NN					NN
Default Value							System Date
Data Type	NUMBER	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	DATE
Length	10	25	25	100	30	15	

b) Table name: TITLE

Column_ Name	TITLE_ID	TITLE	DESCRIPTION	RATING	CATEGORY	RELEASE_ DATE
Key Type	PK					
Null/ Unique	NN,U	NN	NN			
Check				G, PG, R, NC17, NR	DRAMA, COMEDY, ACTION, CHILD, SCIFI, DOCUMENTARY	
Data Type	NUMBER	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	DATE
Length	10	60	400	4	20	

## Practice 2-1 (continued)

c) Table name: TITLE\_COPY

<b>Column Name</b>	COPY_ID	TITLE_ID	STATUS
<b>Key Type</b>	PK	PK,FK	
<b>Null/Unique</b>	NN,U	NN,U	NN
<b>Check</b>			AVAILABLE, DESTROYED, RENTED, RESERVED
<b>FK Ref Table</b>		TITLE	
<b>FK Ref Col</b>		TITLE_ID	
<b>Data Type</b>	NUMBER	NUMBER	VARCHAR2
<b>Length</b>	10	10	15

d) Table name: RENTAL

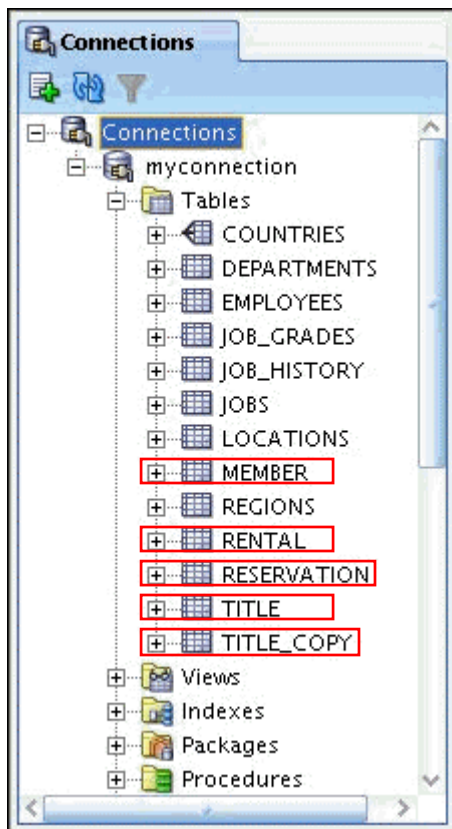
<b>Column Name</b>	BOOK_DATE	MEMBER_ID	COPY_ID	ACT_RET_DATE	EXP_RET_DATE	TITLE_ID
<b>Key Type</b>	PK	PK,FK1	PK,FK2			PK,FK2
<b>Default Value</b>	System Date				System Date + 2 days	
<b>FK Ref Table</b>		MEMBER	TITLE_COPY			TITLE_COPY
<b>FK Ref Col</b>		MEMBER_ID	COPY_ID			TITLE_ID
<b>Data Type</b>	DATE	NUMBER	NUMBER	DATE	DATE	NUMBER
<b>Length</b>		10	10			10

## Practice 2-1 (continued)

e) Table name: RESERVATION

Column Name	RES_DATE	MEMBER_ID	TITLE_ID
Key Type	PK	PK,FK1	PK,FK2
Null/Unique	NN,U	NN,U	NN
FK Ref Table		MEMBER	TITLE
FK Ref Column		MEMBER_ID	TITLE_ID
Data Type	DATE	NUMBER	NUMBER
Length		10	10

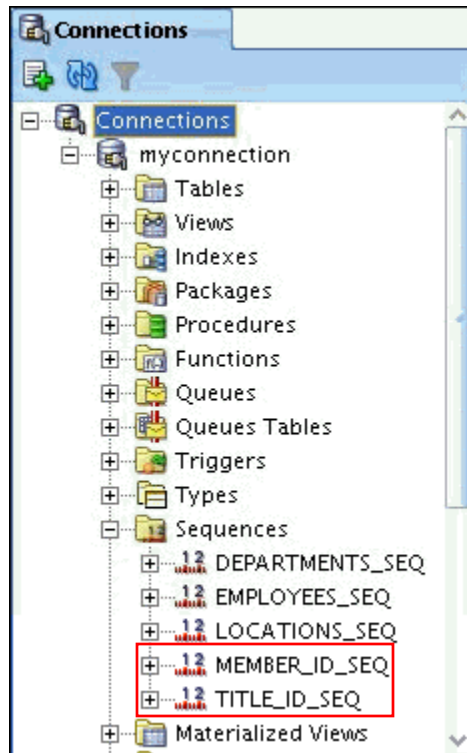
- 2) Verify that the tables were created properly by checking in the Connections Navigator in SQL Developer.





### Practice 2-1 (continued)

- 3) Create sequences to uniquely identify each row in the MEMBER table and the TITLE table.
  - a) Member number for the MEMBER table: Start with 101; do not allow caching of the values. Name the sequence MEMBER\_ID\_SEQ.
  - b) Title number for the TITLE table: Start with 92; do not allow caching of the values. Name the sequence TITLE\_ID\_SEQ.
  - c) Verify the existence of the sequences in the Connections Navigator in SQL Developer.



- 4) Add data to the tables. Create a script for each set of data to be added.
  - a) Add movie titles to the TITLE table. Write a script to enter the movie information. Save the statements in a script named lab\_apcs\_4a.sql. Use the sequences to uniquely identify each title. Enter the release dates in the DD-MON-YYYY format. Remember that single quotation marks in a character field must be specially handled. Verify your additions.

	TITLE
1	Willie and Christmas Too
2	Alien Again
3	The Glob
4	My Day Off
5	Miracles on Ice
6	Soda Gang

## Practice 2-1 (continued)

Title	Description	Rating	Category	Release_date
Willie and Christmas Too	All of Willie's friends make a Christmas list for Santa, but Willie has yet to add his own wish list.	G	CHILD	05-OCT-1995
Alien Again	Yet another installation of science fiction history. Can the heroine save the planet from the alien life form?	R	SCIFI	19-MAY-1995
The Glob	A meteor crashes near a small American town and unleashes carnivorous goo in this classic.	NR	SCIFI	12-AUG-1995
My Day Off	With a little luck and a lot of ingenuity, a teenager skips school for a day in New York.	PG	COMEDY	12-JUL-1995
Miracles on Ice	A six-year-old has doubts about Santa Claus, but she discovers that miracles really do exist.	PG	DRAMA	12-SEP-1995
Soda Gang	After discovering a cache of drugs, a young couple find themselves pitted against a vicious gang.	NR	ACTION	01-JUN-1995

- b) Add data to the MEMBER table. Save the insert statements in a script named lab\_apcs\_4b.sql. Execute commands in the script. Be sure to use the sequence to add the member numbers.

First_Name	Last_Name	Address	City	Phone	Join_Date
Carmen	Velasquez	283 King Street	Seattle	206-899-6666	08-MAR-1990
LaDoris	Ngao	5 Modrany	Bratislava	586-355-8882	08-MAR-1990
Midori	Nagayama	68 Via Centrale	Sao Paolo	254-852-5764	17-JUN-1991
Mark	Quick-to-See	6921 King Way	Lagos	63-559-7777	07-APR-1990
Audry	Ropeburn	86 Chu Street	Hong Kong	41-559-87	18-JAN-1991
Molly	Urguhart	3035 Laurier	Quebec	418-542-9988	18-JAN-1991

### Practice 2-1 (continued)

c) Add the following movie copies in the TITLE\_COPY table:

**Note:** Have the TITLE\_ID numbers available for this exercise.

Title	Copy_Id	Status	Title	Copy_Id
Willie and Christmas Too	1	AVAILABLE	Willie and Christmas Too	1
Alien Again	1	AVAILABLE	Alien Again	1
	2	RENTED		2
The Glob	1	AVAILABLE	The Glob	1
My Day Off	1	AVAILABLE	My Day Off	1
	2	AVAILABLE		2
	3	RENTED		3
Miracles on Ice	1	AVAILABLE	Miracles on Ice	1
Soda Gang	1	AVAILABLE	Soda Gang	1

d) Add the following rentals to the RENTAL table:





**Note:** The title number may be different depending on the sequence number.

Title_Id	Copy_Id	Member_Id	Book_date	Exp_Ret_Date
92	1	101	3 days ago	1 day ago
93	2	101	1 day ago	1 day from now
95	3	102	2 days ago	Today
97	1	106	4 days ago	2 days ago

### Practice 2-1 (continued)

- 5) Create a view named `TITLE_AVAIL` to show the movie titles, the availability of each copy, and its expected return date if rented. Query all rows from the view. Order the results by title.

**Note:** Your results may be different.

 TITLE	 COPY_ID	 STATUS	 EXP_RET_DATE
1 Alien Again	1	AVAILABLE	(null)
2 Alien Again	2	RENTED	15-JUL-09
3 Miracles on Ice	1	AVAILABLE	(null)
4 My Day Off	1	AVAILABLE	(null)
5 My Day Off	2	AVAILABLE	(null)
6 My Day Off	3	RENTED	16-JUL-09
7 Soda Gang	1	AVAILABLE	14-JUL-09
8 The Glob	1	AVAILABLE	(null)
9 Willie and Christmas Too	1	AVAILABLE	15-JUL-09

- 6) Make changes to the data in the tables.
- Add a new title. The movie is “Interstellar Wars,” which is rated PG and classified as a science fiction movie. The release date is 07-JUL-77. The description is “Futuristic interstellar action movie. Can the rebels save the humans from the evil empire?” Be sure to add a title copy record for two copies.
  - Enter two reservations. One reservation is for Carmen Velasquez, who wants to rent “Interstellar Wars.” The other is for Mark Quick-to-See, who wants to rent “Soda Gang.”

## Practice 2-1 (continued)

7) Make a modification to one of the tables.

- a) Run the `lab_apcs_7a.sql` script located in the `/home/oracle/labs/sql1/labs` folder, to add a `PRICE` column to the `TITLE` table to record the purchase price of the video. Verify your modifications.

DESCRIBE title		
Name	Null	Type
-----		
TITLE_ID	NOT NULL	NUMBER(10)
TITLE	NOT NULL	VARCHAR2(60)
DESCRIPTION	NOT NULL	VARCHAR2(400)
RATING		VARCHAR2(4)
CATEGORY		VARCHAR2(20)
RELEASE_DATE		DATE
PRICE		NUMBER(8,2)

Title	Price
Willie and Christmas Too	25
Alien Again	35
The Glob	35
My Day Off	35
Miracles on Ice	30
Soda Gang	35
Interstellar Wars	29

- b) Create a script named `lab_apcs_7b.sql` that contains update statements that update each video with a price according to the preceding list. Run the commands in the script.

**Note:** Have the `TITLE_ID` numbers available for this exercise.

- 8) Create a report that contains each customer's history of renting videos. Be sure to include the customer name, movie rented, dates of the rental, and duration of rentals. Total the number of rentals for all customers for the reporting period. Save the commands that generate the report in a script file named `lab_apcs_8.sql`.

**Note:** Your results may be different.

	MEMBER	TITLE	BOOK_DATE	DURATION
1	Carmen Velasquez	Willie and Christmas Too	13-JUL-09	1
2	Carmen Velasquez	Alien Again	15-JUL-09	(null)
3	LaDoris Ngao	My Day Off	14-JUL-09	(null)
4	Molly Uguhart	Soda Gang	12-JUL-09	2

## Practice Solutions 2-1

- 1) Create the tables based on the following table instance charts. Choose the appropriate data types and be sure to add integrity constraints.

a) Table name: MEMBER

```
CREATE TABLE member
(member_id          NUMBER(10)
 CONSTRAINT member_member_id_pk PRIMARY KEY,
 last_name         VARCHAR2(25)
 CONSTRAINT member_last_name_nn NOT NULL,
 first_name        VARCHAR2(25),
 address           VARCHAR2(100),
 city              VARCHAR2(30),
 phone             VARCHAR2(15),
 join_date         DATE DEFAULT SYSDATE
 CONSTRAINT member_join_date_nn NOT NULL);
```

b) Table name: TITLE

```
CREATE TABLE title
(title_id          NUMBER(10)
 CONSTRAINT title_title_id_pk PRIMARY KEY,
 title             VARCHAR2(60)
 CONSTRAINT title_title_nn NOT NULL,
 description       VARCHAR2(400)
 CONSTRAINT title_description_nn NOT NULL,
 rating            VARCHAR2(4)
 CONSTRAINT title_rating_ck CHECK
 (rating IN ('G', 'PG', 'R', 'NC17', 'NR')),
 category          VARCHAR2(20)
 CONSTRAINT title_category_ck CHECK
 (category IN ('DRAMA', 'COMEDY', 'ACTION',
 'CHILD', 'SCIFI', 'DOCUMENTARY')),
 release_date      DATE);
```

c) Table name: TITLE\_COPY

```
CREATE TABLE title_copy
(copy_id          NUMBER(10),
 title_id         NUMBER(10)
 CONSTRAINT title_copy_title_id_fk REFERENCES
 title(title_id),
 status           VARCHAR2(15)
 CONSTRAINT title_copy_status_nn NOT NULL
 CONSTRAINT title_copy_status_ck CHECK (status IN
 ('AVAILABLE', 'DESTROYED', 'RENTED', 'RESERVED')),
 CONSTRAINT title_copy_copy_id_title_id_pk
 PRIMARY KEY (copy_id, title_id));
```

## Practice Solutions 2-1 (continued)

d) Table name: RENTAL

```
CREATE TABLE rental
  (book_date      DATE DEFAULT SYSDATE,
   member_id      NUMBER(10)
   CONSTRAINT rental_member_id_fk REFERENCES
member(member_id),
   copy_id        NUMBER(10),
   act_ret_date   DATE,
   exp_ret_date   DATE DEFAULT SYSDATE + 2,
   title_id       NUMBER(10),
   CONSTRAINT rental_book_date_copy_title_pk
     PRIMARY KEY (book_date, member_id, copy_id, title_id),
   CONSTRAINT rental_copy_id_title_id_fk
     FOREIGN KEY (copy_id, title_id)
     REFERENCES title_copy(copy_id, title_id));
```

e) Table name: RESERVATION

```
CREATE TABLE reservation
  (res_date       DATE,
   member_id      NUMBER(10)
   CONSTRAINT reservation_member_id REFERENCES
member(member_id),
   title_id       NUMBER(10)
   CONSTRAINT reservation_title_id REFERENCES
title(title_id),
   CONSTRAINT reservation_resdate_mem_tit_pk PRIMARY KEY
   (res_date, member_id, title_id));
```

2) Verify that the tables were created properly by checking in the Connections Navigator in SQL Developer.

a) In the Connections Navigator, expand Connections > myconnection > Tables.

3) Create sequences to uniquely identify each row in the MEMBER table and the TITLE table.

a) Member number for the MEMBER table: Start with 101; do not allow caching of the values. Name the sequence MEMBER\_ID\_SEQ.

```
CREATE SEQUENCE member_id_seq
START WITH 101
NOCACHE;
```

b) Title number for the TITLE table: Start with 92; do not allow caching of the values. Name the sequence TITLE\_ID\_SEQ.

```
CREATE SEQUENCE title_id_seq
START WITH 92
NOCACHE;
```

## Practice Solutions 2-1 (continued)

- c) Verify the existence of the sequences in the Connections Navigator in SQL Developer.
  - i) In the Connections Navigator, assuming that the myconnection node is expanded, expand Sequences.
- 4) Add data to the tables. Create a script for each set of data to be added.
  - a) Add movie titles to the TITLE table. Write a script to enter the movie information. Save the statements in a script named lab\_apcs\_4a.sql. Use the sequences to uniquely identify each title. Enter the release dates in the DD-MON-YYYY format. Remember that single quotation marks in a character field must be specially handled. Verify your additions.

```
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'Willie and Christmas Too',
        'All of Willie's friends make a Christmas list for
        Santa, but Willie has yet to add his own wish list.',
        'G', 'CHILD', TO_DATE('05-OCT-1995','DD-MON-YYYY'))
/
INSERT INTO title(title_id , title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'Alien Again', 'Yet another
        installment of science fiction history. Can the
        heroine save the planet from the alien life form?',
        'R', 'SCIFI', TO_DATE( '19-MAY-1995','DD-MON-YYYY'))
/
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'The Glob', 'A meteor crashes
        near a small American town and unleashes carnivorous
        goo in this classic.', 'NR', 'SCIFI',
        TO_DATE( '12-AUG-1995','DD-MON-YYYY'))
/
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'My Day Off', 'With a little
        luck and a lot ingenuity, a teenager skips school
        for
        a day in New York.', 'PG', 'COMEDY',
        TO_DATE( '12-JUL-1995','DD-MON-YYYY'))
/
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'Miracles on Ice', 'A six-
        year-old has doubts about Santa Claus, but she discovers
        that miracles really do exist.', 'PG', 'DRAMA',
        TO_DATE('12-SEP-1995','DD-MON-YYYY'))
/
```



## Practice Solutions 2-1 (continued)

```
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES          (title_id_seq.NEXTVAL, 'Soda Gang', 'After
discovering a cache of drugs, a young couple find themselves
pitted against a vicious gang.', 'NR', 'ACTION', TO_DATE('01-
JUN-1995','DD-MON-YYYY'))
/
COMMIT
/
SELECT  title
FROM    title;
```

- b) Add data to the MEMBER table. Place the insert statements in a script named lab\_apcs\_4b.sql. Execute the commands in the script. Be sure to use the sequence to add the member numbers.

```
SET VERIFY OFF
INSERT INTO member(member_id, first_name, last_name,
                  address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Carmen', 'Velasquez',
        '283 King Street', 'Seattle', '206-899-6666',
        TO_DATE('08-MAR-1990',
        'DD-MM-YYYY'))
/

INSERT INTO member(member_id, first_name, last_name,
                  address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'LaDoris', 'Ngao',
        '5 Modrany', 'Bratislava', '586-355-8882',
        TO_DATE('08-MAR-1990',
        'DD-MM-YYYY'))
/

INSERT INTO member(member_id, first_name, last_name,
                  address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Midori', 'Nagayama',
        '68 Via Centrale', 'Sao Paolo', '254-852-5764',
        TO_DATE('17-JUN-1991',
        'DD-MM-YYYY'))
/

INSERT INTO member(member_id, first_name, last_name,
                  address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Mark', 'Quick-to-See',
        '6921 King Way', 'Lagos', '63-559-7777', TO_DATE('07-
APR-1990',
        'DD-MM-YYYY'))
/

INSERT INTO member(member_id, first_name, last_name,
```

## Practice Solutions 2-1 (continued)

```
        address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Audry', 'Ropeburn',
        '86 Chu Street', 'Hong Kong', '41-559-87',
TO_DATE('18-JAN-1991',
        'DD-MM-YYYY'))
/

INSERT INTO member(member_id, first_name, last_name,
        address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Molly', 'Urguhart',
        '3035 Laurier', 'Quebec', '418-542-9988', TO_DATE('18-
JAN-1991',
        'DD-MM-YYYY'));
/

COMMIT
SET VERIFY ON
```

c) Add the following movie copies in the TITLE\_COPY table:

**Note:** Have the TITLE\_ID numbers available for this exercise.

```
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 92, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 93, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (2, 93, 'RENTED')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 94, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 95, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (2, 95, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (3, 95, 'RENTED')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 96, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 97, 'AVAILABLE')
/
```

## Practice Solutions 2-1 (continued)

d) Add the following rentals to the RENTAL table:

**Note:** The title number may be different depending on the sequence number.

```
INSERT INTO rental(title_id, copy_id, member_id,
                  book_date, exp_ret_date, act_ret_date)
VALUES (92, 1, 101, sysdate-3, sysdate-1, sysdate-2)
/
INSERT INTO rental(title_id, copy_id, member_id,
                  book_date, exp_ret_date, act_ret_date)
VALUES (93, 2, 101, sysdate-1, sysdate-1, NULL)
/
INSERT INTO rental(title_id, copy_id, member_id,
                  book_date, exp_ret_date, act_ret_date)
VALUES (95, 3, 102, sysdate-2, sysdate, NULL)
/
INSERT INTO rental(title_id, copy_id, member_id,
                  book_date, exp_ret_date, act_ret_date)
VALUES (97, 1, 106, sysdate-4, sysdate-2, sysdate-2)
/
COMMIT
/
```

5) Create a view named TITLE\_AVAIL to show the movie titles, the availability of each copy, and its expected return date if rented. Query all rows from the view. Order the results by title.

**Note:** Your results may be different.

```
CREATE VIEW title_avail AS
  SELECT  t.title, c.copy_id, c.status, r.exp_ret_date
  FROM    title t JOIN title_copy c
  ON      t.title_id = c.title_id
  FULL OUTER JOIN rental r
  ON      c.copy_id = r.copy_id
  AND     c.title_id = r.title_id;

SELECT  *
FROM    title_avail
ORDER BY title, copy_id;
```

## Practice Solutions 2-1 (continued)

6) Make changes to data in the tables.

- a) Add a new title. The movie is “Interstellar Wars,” which is rated PG and classified as a science fiction movie. The release date is 07-JUL-77. The description is “Futuristic interstellar action movie. Can the rebels save the humans from the evil empire?” Be sure to add a title copy record for two copies.

```
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'Interstellar Wars',
        'Futuristic interstellar action movie. Can the
        rebels save the humans from the evil empire?',
        'PG', 'SCIFI', '07-JUL-77')
/
INSERT INTO title_copy (copy_id, title_id, status)
VALUES (1, 98, 'AVAILABLE')
/
INSERT INTO title_copy (copy_id, title_id, status)
VALUES (2, 98, 'AVAILABLE')
/
```

- b) Enter two reservations. One reservation is for Carmen Velasquez, who wants to rent “Interstellar Wars.” The other is for Mark Quick-to-See, who wants to rent “Soda Gang.”

```
INSERT INTO reservation (res_date, member_id, title_id)
VALUES (SYSDATE, 101, 98)
/
INSERT INTO reservation (res_date, member_id, title_id)
VALUES (SYSDATE, 104, 97)
/
```

7) Make a modification to one of the tables.

- a) Run the lab\_apcs\_7a.sql script located in the /home/oracle/labs/sql1/labs folder, to add a PRICE column to the TITLE table to record the purchase price of the video. Verify your modifications.

```
ALTER TABLE title
ADD (price NUMBER(8,2));

DESCRIBE title
```

## Practice Solutions 2-1 (continued)

- b) Create a script named `lab_apcs_7b.sql` that contains update statements that update each video with a price according to the list provided. Run the commands in the script.

**Note:** Have the `TITLE_ID` numbers available for this exercise.

```
SET ECHO OFF
SET VERIFY OFF
UPDATE title
SET    price = &price
WHERE  title_id = &title_id;
SET VERIFY OFF
SET ECHO OFF
```

- 8) Create a report that contains each customer's history of renting videos. Be sure to include the customer name, movie rented, dates of the rental, and duration of rentals. Total the number of rentals for all customers for the reporting period. Save the commands that generate the report in a script file named `lab_apcs_8.sql`.

**Note:** Your results may be different.

```
SELECT  m.first_name||' '||m.last_name MEMBER, t.title,
        r.book_date, r.act_ret_date - r.book_date DURATION
FROM    member m
JOIN    rental r
ON      r.member_id = m.member_id
JOIN    title t
ON      r.title_id = t.title_id
ORDER BY member;
```