Appendix A
Practices and Solutions

Table of Contents

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.

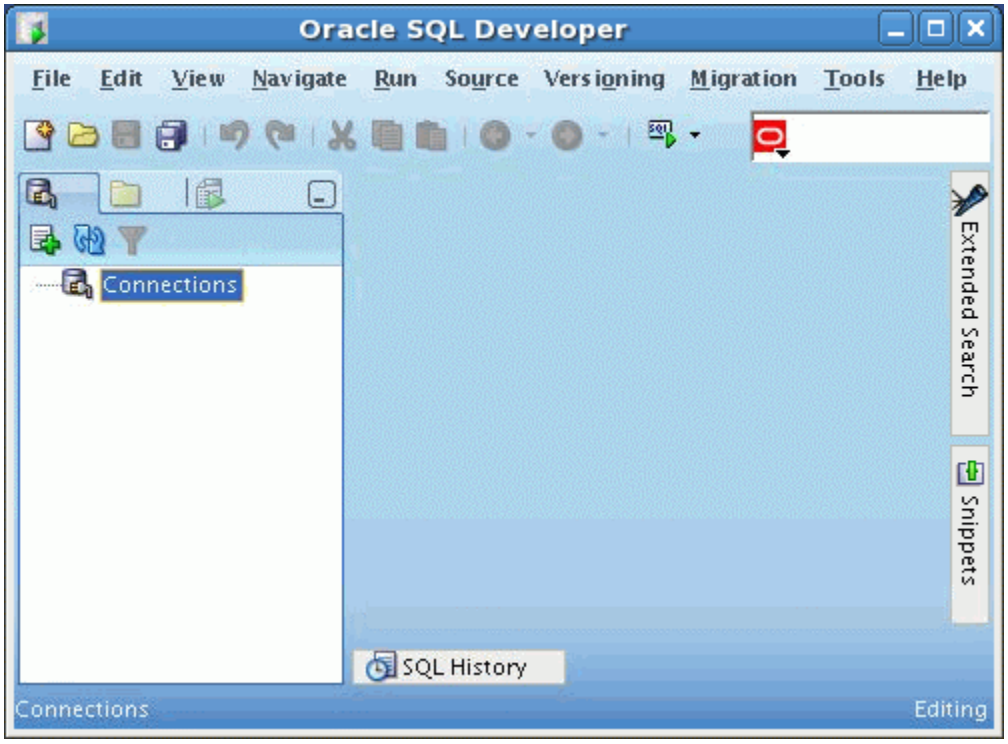
Practice Solutions I-1: Introduction

Starting Oracle SQL Developer

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

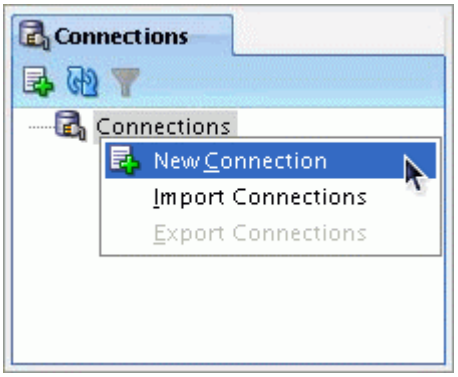
   a) Double-click the SQL Developer desktop icon.

   The SQL Developer Interface appears.

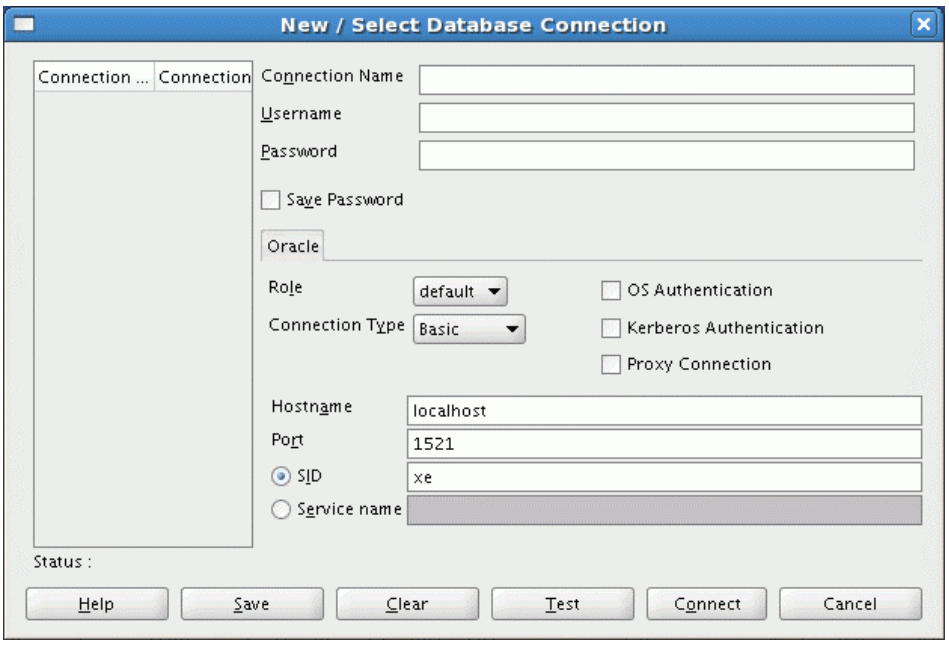


Creating a New Oracle SQL Developer Database Connection

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

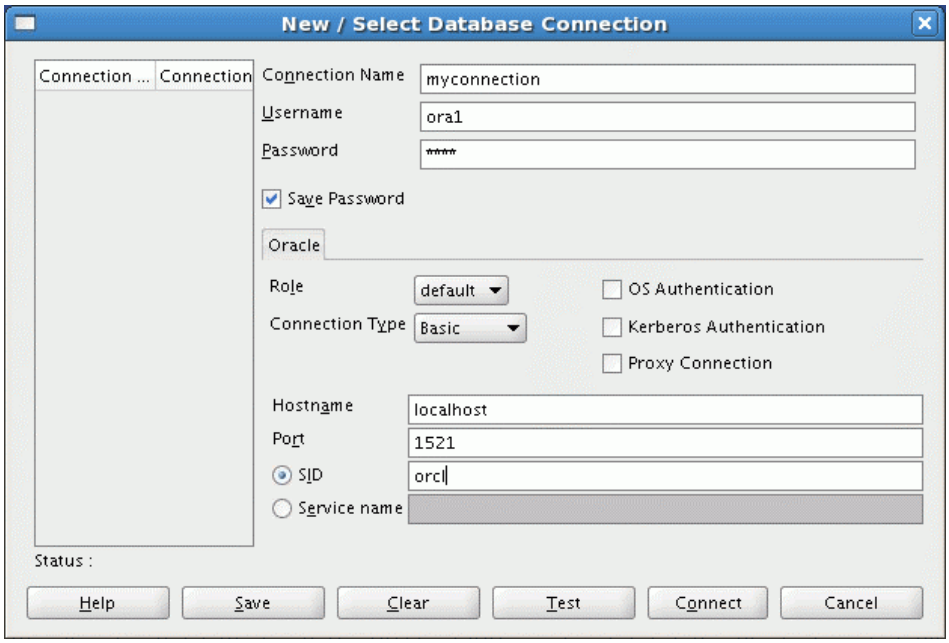


The New / Select Database Connection dialog box appears.



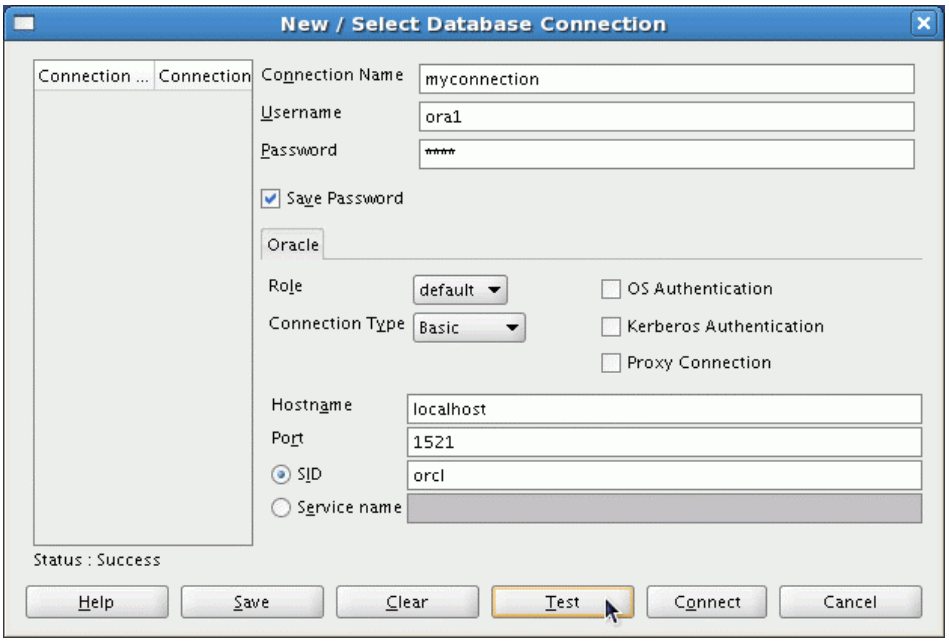
3) Create a database connection using the following information:
   a) Connection Name: `myconnection`
   b) Username: `ora1`
   c) Password: `ora1`
   d) Hostname: `localhost`
   e) Port: `1521`
   f) SID: `ORCL`

Ensure that you select the Save Password check box.

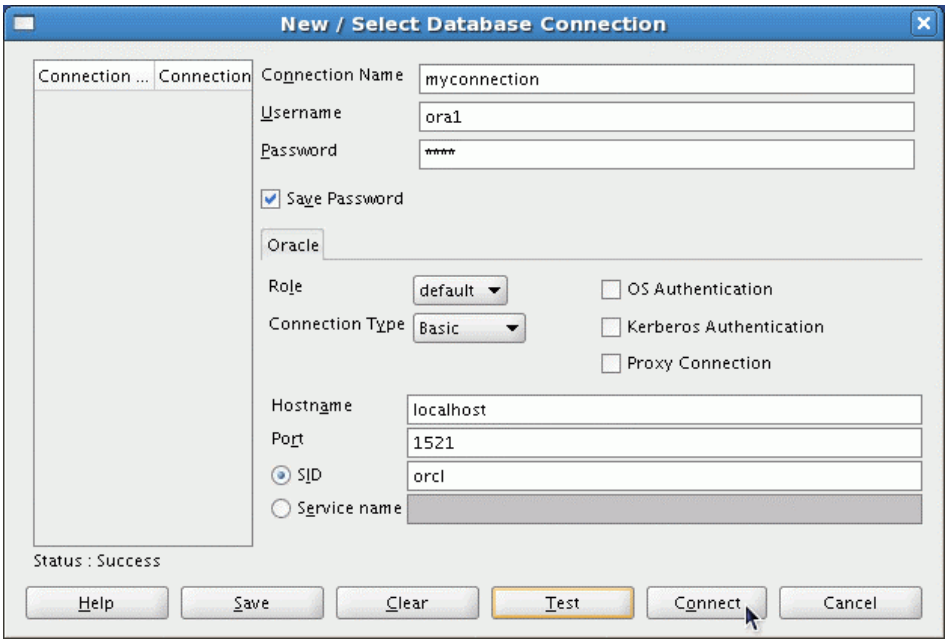


Testing and Connecting Using the Oracle SQL Developer Database Connection

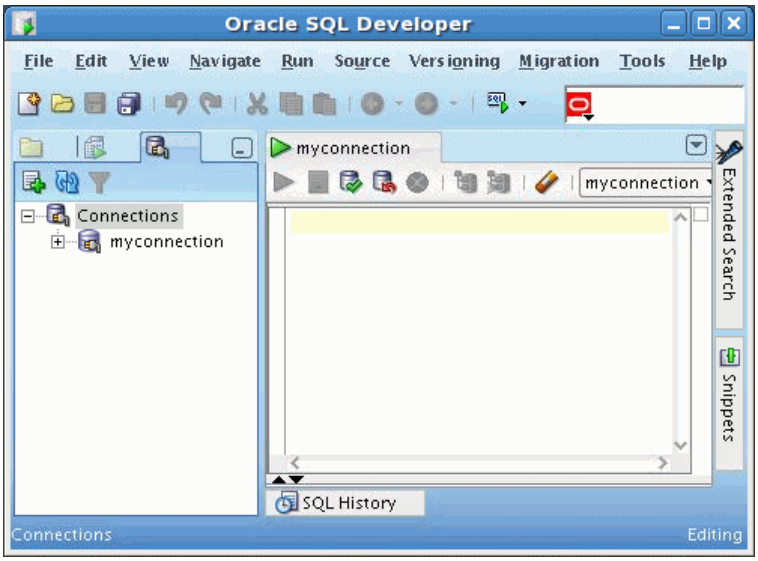
4) Test the new connection.



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



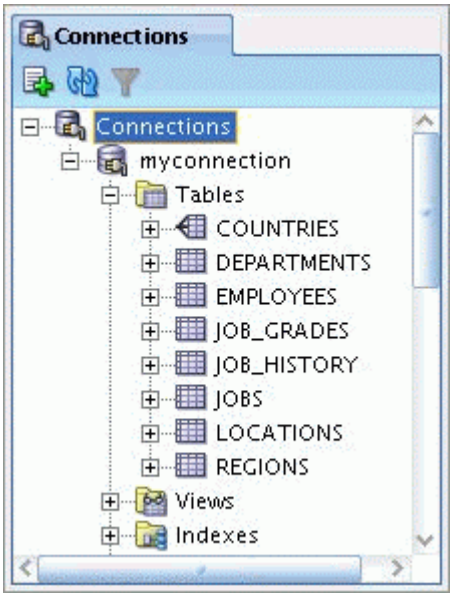
When you create a connection, a SQL Worksheet for that connection opens automatically.



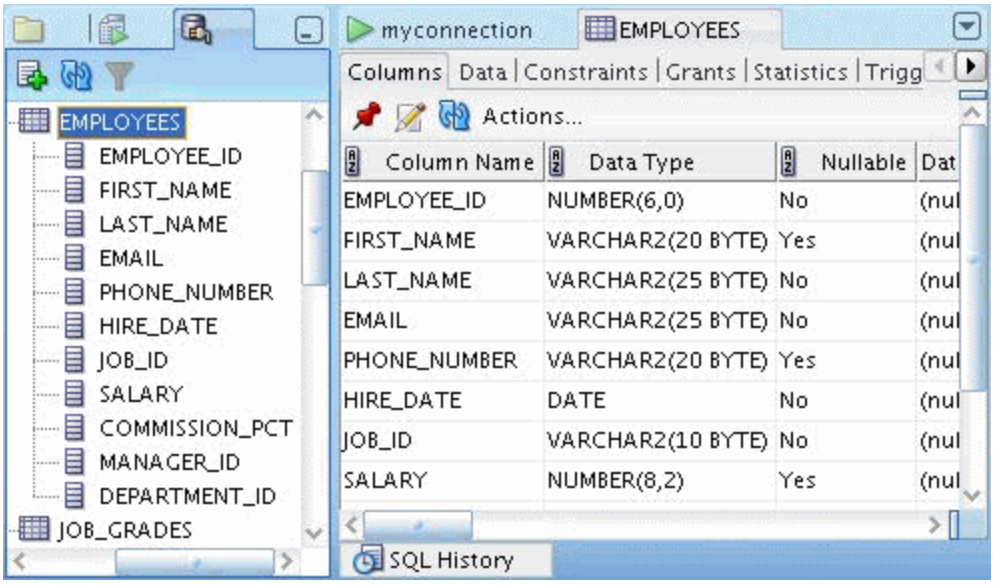
Browsing the Tables in the Connections Navigator

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
```

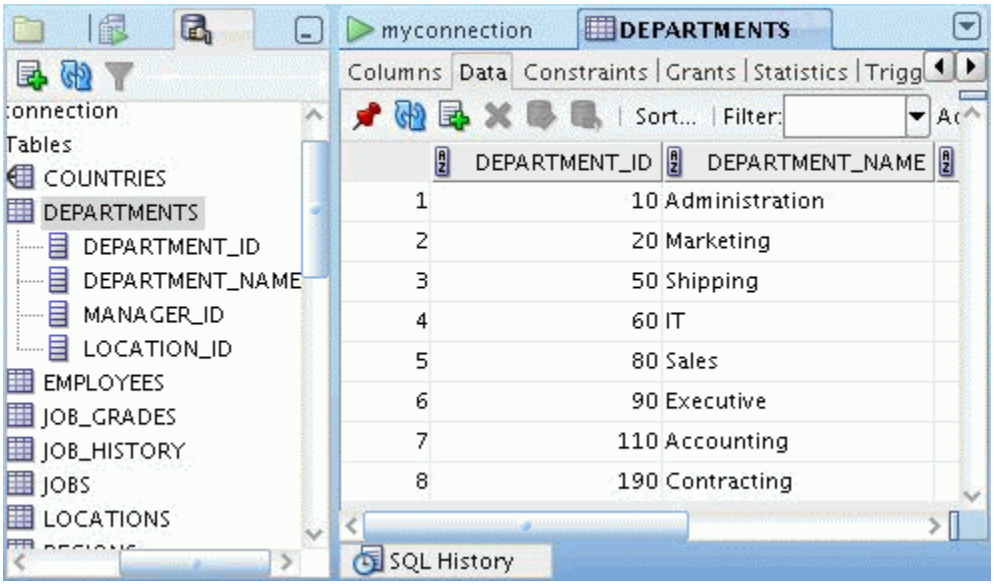


7) Browse the structure of the `EMPLOYEES` table.



8) View the data of the `DEPARTMENTS` table.

Practice Solutions I-1: Introduction (continued)



	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

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_PRES
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.

	LAST_NAME	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.

	LAST_NAME	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`.

	LAST_NAME	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.

	LAST_NAME	JOB_ID	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.

	LAST_NAME	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`.

	Employee	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.

	LAST_NAME	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.

	LAST_NAME	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.

	LAST_NAME	SALARY	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:

	LAST_NAME	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:

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

manager_id = 201, sorted by salary:

	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	202	Fay	6000	20

manager_id = 124, sorted by employee_id:

	EMPLOYEE_ID	LAST_NAME	SALARY	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."

	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;
```

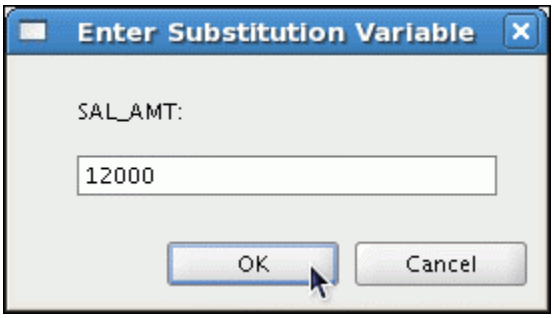
9) Create a report to display the last name, salary, and commission for all employees who earn commissions. Sort data in descending order of salary and commissions. Use the column's numeric position in the `ORDER BY` clause.

```
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

```
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)

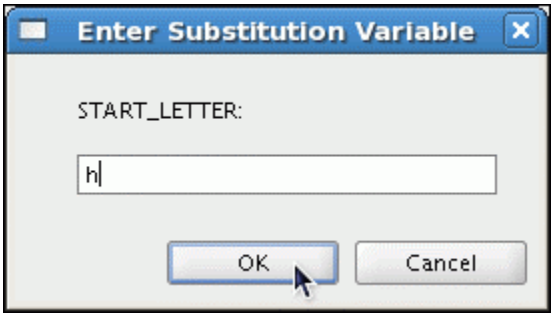
5) Write a query that displays the last name (with the first letter in uppercase and all the other letters in lowercase) and the length of the last name for all employees whose name starts with the letters "J," "A," or "M." Give each column an appropriate label. Sort the results by the employees' last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



	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_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 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:

Job	***Grade***
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.

	TOTAL	1995	1996	1997	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.

	Job	Dept 20	Dept 50	Dept 80	Dept 90	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`.

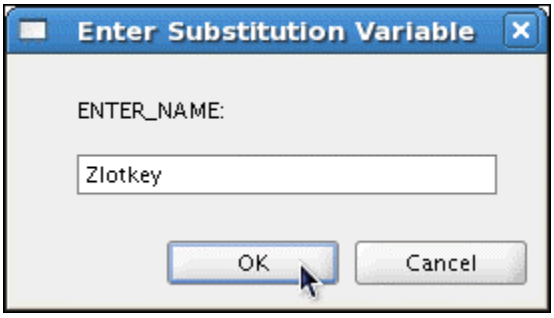
```
SELECT w.last_name, w.hire_date, m.last_name, m.hire_date
FROM   employees w JOIN employees m
ON    (w.manager_id = m.employee_id)
WHERE    w.hire_date <  m.hire_date;
```

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.

	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

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`.

	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;
```

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.

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

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

```
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)
```

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;
```