

# **Oracle Database: SQL Fundamentals I**

**Activity Guide**

D64258GC11

Edition 1.1

March 2012

D76184

**ORACLE®**

**Authors**

Supriya Ananth  
Salome Clement  
Brian Pottle

**Technical Contributors  
and Reviewers**

Diganta Choudhury  
Bryan Roberts  
Kimseong Loh  
Laszlo Czinkoczki  
Brent Dayley  
Nancy Greenberg  
Manish Pawar  
Clair Bennett  
Zarko Cesljas  
Yanti Chang  
Gerlinde Frenzen  
Helen Robertson  
Joel Goodman  
Pedro Neves  
Hilda Simon

**Editor**

Raj Kumar

**Graphic Designer**

Satish Bettgowda

**Publishers**

Sujatha Nagendra  
Joseph Fernandez

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

**Disclaimer**

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

**Restricted Rights Notice**

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

**U.S. GOVERNMENT RIGHTS**

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

**Trademark Notice**

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Oracle Internal & Oracle Academy  
Use Only

# Table of Contents

<b>Practices for Lesson 1: Introduction</b>	<b>1-1</b>
Practices for Lesson 1	1-2
Practice 1-1: Introduction	1-3
Solution 1-1: Introduction	1-4
<b>Practices for Lesson 2: Retrieving Data Using the SQL SELECT Statement</b>	<b>2-1</b>
Practices for Lesson 2	2-2
Practice 2-1: Retrieving Data Using the SQL SELECT Statement	2-3
Solution 2-1: Retrieving Data Using the SQL SELECT Statement	2-7
<b>Practices for Lesson 3: Restricting and Sorting Data</b>	<b>3-1</b>
Practices for Lesson 3	3-2
Practice 3-1: Restricting and Sorting Data	3-3
Solution 3-1: Restricting and Sorting Data	3-7
<b>Practices for Lesson 4: Using Single-Row Functions to Customize Output</b>	<b>4-1</b>
Practices for Lesson 4	4-2
Practice 4-1: Using Single-Row Functions to Customize Output	4-3
Solution 4-1: Using Single-Row Functions to Customize Output	4-6
<b>Practices for Lesson 5: Using Conversion Functions and Conditional Expressions</b>	<b>5-1</b>
Practices for Lesson 5	5-2
Practice 5-1: Using Conversion Functions and Conditional Expressions	5-3
Solution 5-1: Using Conversion Functions and Conditional Expressions	5-6
<b>Practices for Lesson 6: Reporting Aggregated Data Using the Group Functions</b>	<b>6-1</b>
Practices for Lesson 6	6-2
Practice 6-1: Reporting Aggregated Data Using the Group Functions	6-3
Solution 6-1: Reporting Aggregated Data Using the Group Functions	6-6
<b>Practices for Lesson 7: Displaying Data from Multiple Tables Using Joins</b>	<b>7-1</b>
Practices for Lesson 7	7-2
Practice 7-1: Displaying Data from Multiple Tables Using Joins	7-3
Solution 7-1: Displaying Data from Multiple Tables Using Joins	7-6
<b>Practices for Lesson 8: Using Subqueries to Solve Queries</b>	<b>8-1</b>
Practices for Lesson 8	8-2
Practice 8-1: Using Subqueries to Solve Queries	8-3
Solution 8-1: Using Subqueries to Solve Queries	8-6
<b>Practices for Lesson 9: Using the Set Operators</b>	<b>9-1</b>
Practices for Lesson 9	9-2
Practice 9-1: Using the Set Operators	9-3
Solution 9-1: Using the Set Operators	9-5
<b>Practices for Lesson 10: Manipulating Data</b>	<b>10-1</b>
Practices for Lesson 10	10-2
Practice 10-1: Manipulating Data	10-3
Solution 10-1: Manipulating Data	10-6
<b>Practices for Lesson 11: Using DDL Statements to Create and Manage Tables</b>	<b>11-1</b>
Practices for Lesson 11	11-2
Practice 11-1: Using DDL Statements to Create and Manage Tables	11-3
Solution 11-1: Using DDL Statements to Create and Manage Tables	11-5
<b>Practices for Lesson 12: Creating Other Schema Objects</b>	<b>12-1</b>

Practices for Lesson 12.....	12-2
Practice 12-1: Creating Other Schema Objects .....	12-3
Solution 12-1: Creating Other Schema Objects .....	12-5
<b>Additional Practices and Solutions .....</b>	<b>13-1</b>
Practice 1-1 .....	13-2
Solution 1-1 .....	13-10
Case Study.....	13-15
Practice 2-1 .....	13-16
Solution 2-1 .....	13-24

Oracle Internal & Oracle Academy  
Use Only

# **Practices for Lesson 1: Introduction**

## **Chapter 1**

Oracle Internal & Oracle Academy  
Use Only

## Practices for Lesson 1

---

### Lesson Overview

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 1-1: 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: `oral`
  - c. Password: `oral`
  - 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:

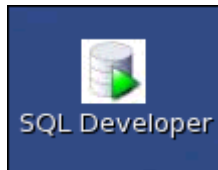
COUNTRIES  
DEPARTMENTS  
EMPLOYEES  
JOB\_GRADES  
JOB\_HISTORY  
JOBS  
  
LOCATIONS  
REGIONS

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

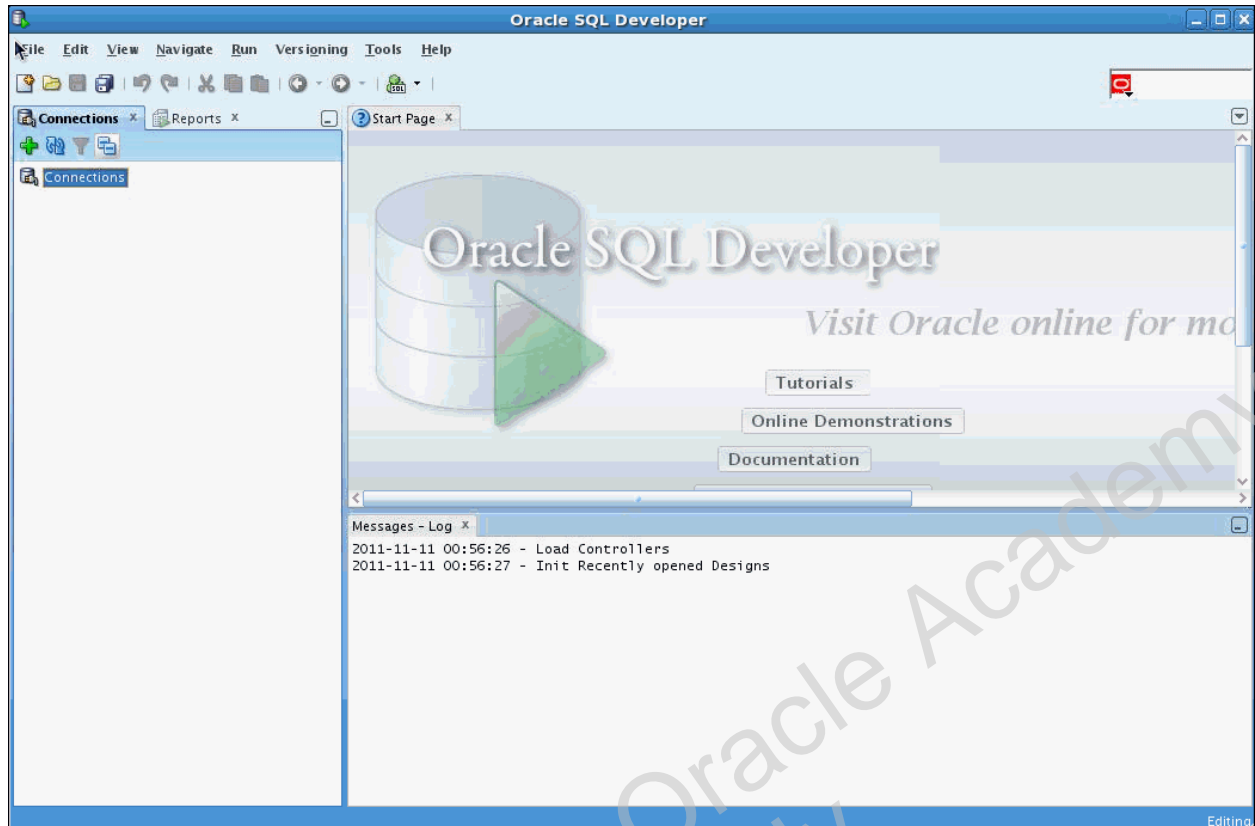
## Solution 1-1: Introduction

### Starting Oracle SQL Developer

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



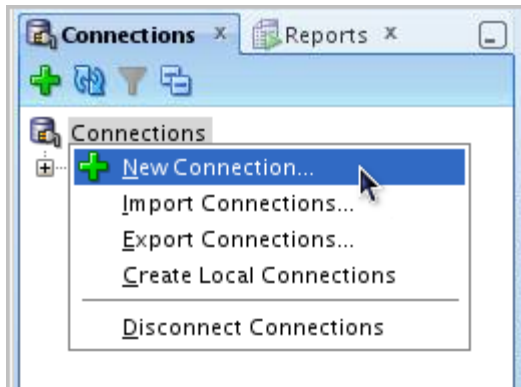
The SQL Developer Interface appears.



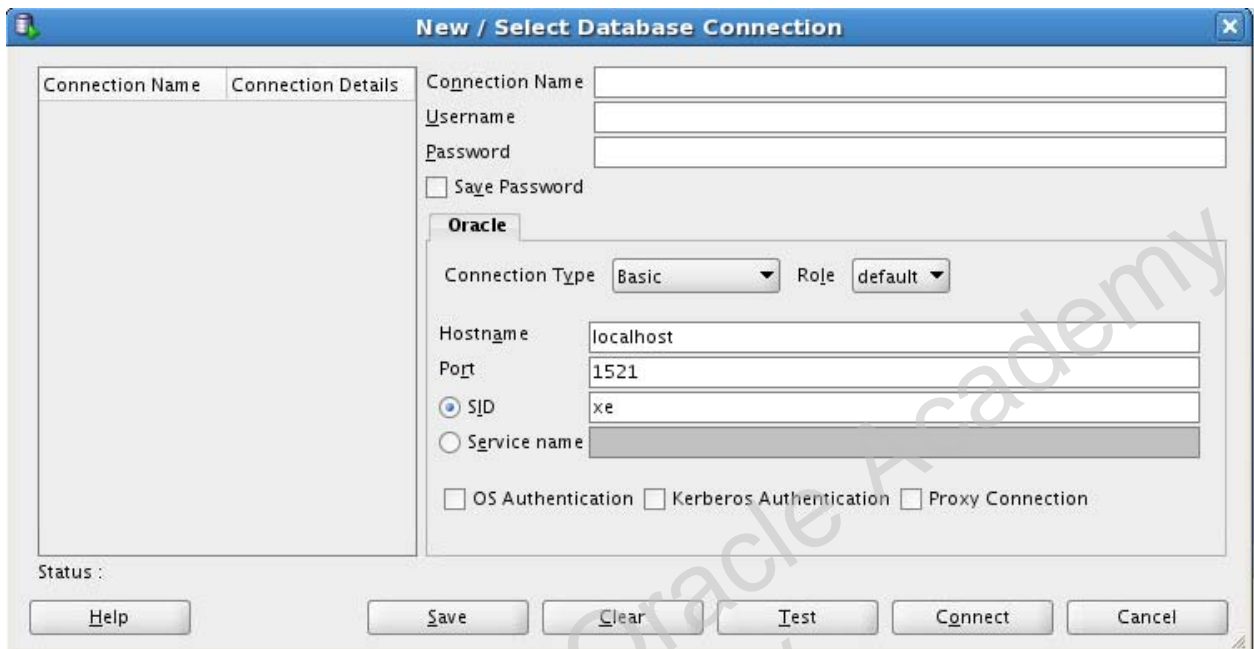


## Creating a New Oracle SQL Developer Database Connection

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.

**New / Select Database Connection**

Connection Name	Connection Details
myconnection	ora1@//localhost...

Connection Name: myconnection  
Username: ora1  
Password: .....  
☒ Save Password

**Oracle**

Connection Type: Basic Role: default

Hostname: localhost  
Port: 1521  
☒ SID: ORCL  
☐ Service name:   
☐ OS Authentication ☐ Kerberos Authentication ☐ Proxy Connection

Status:

Help Save Clear Test Connect Cancel

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

4. Test the new connection.

**New / Select Database Connection**

Connection Name	Connection Details
myconnection	ora1@//localhost...

Connection Name: myconnection  
Username: ora1  
Password: .....  
☒ Save Password

**Oracle**

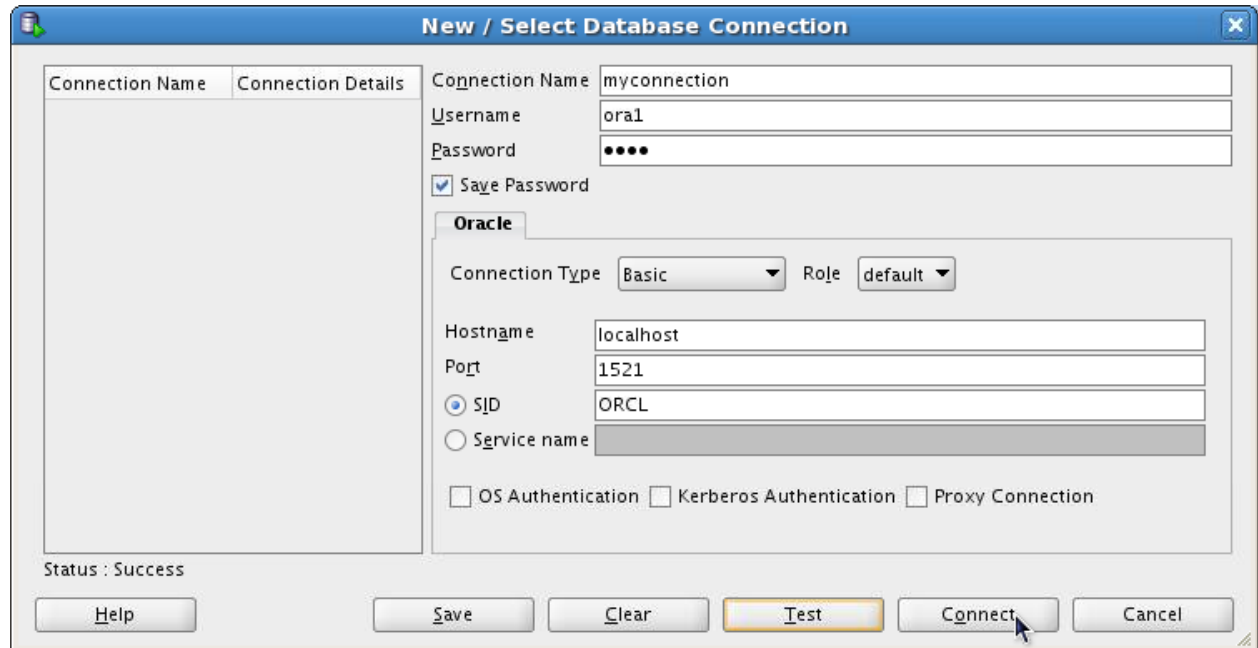
Connection Type: Basic Role: default

Hostname: localhost  
Port: 1521  
☒ SID: ORCL  
☐ Service name:   
☐ OS Authentication ☐ Kerberos Authentication ☐ Proxy Connection

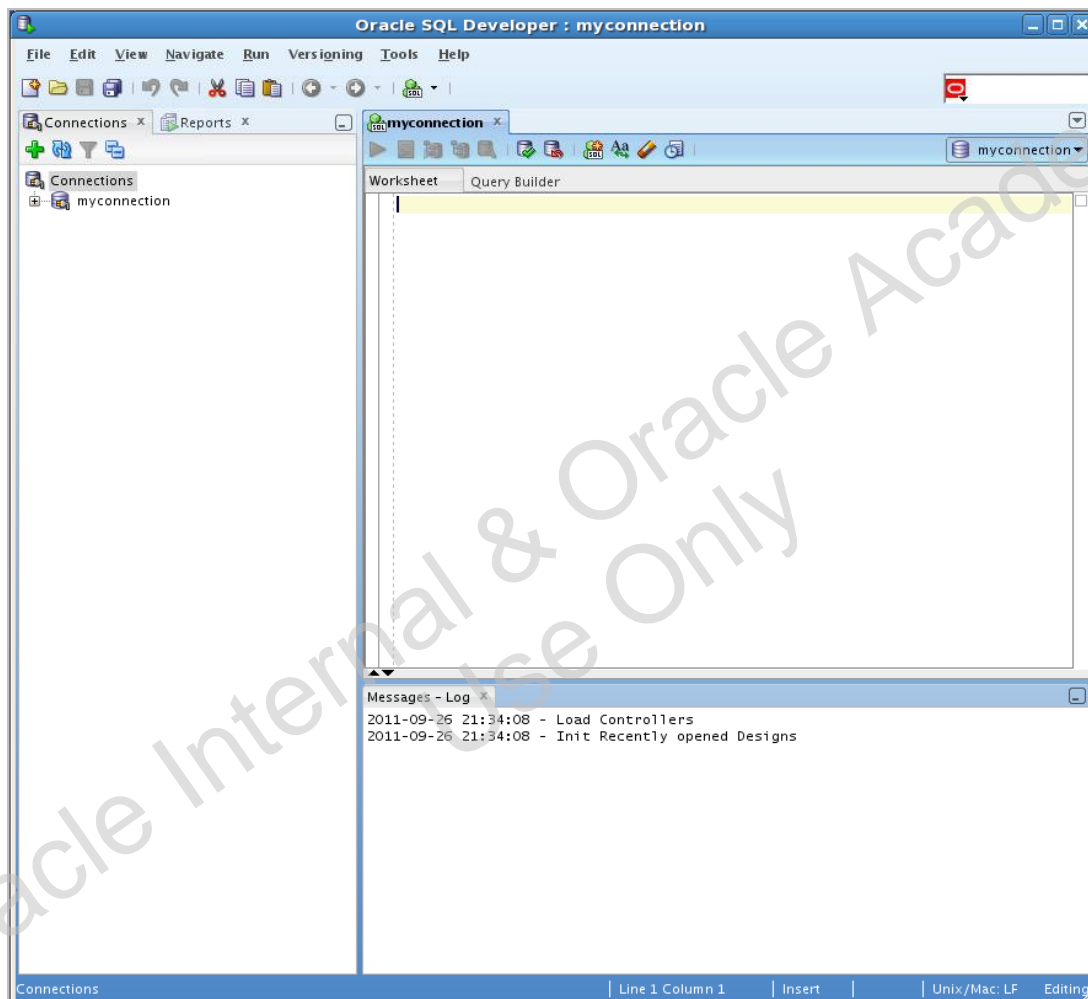
Status: Success

Help Save Clear Test Connect Cancel

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



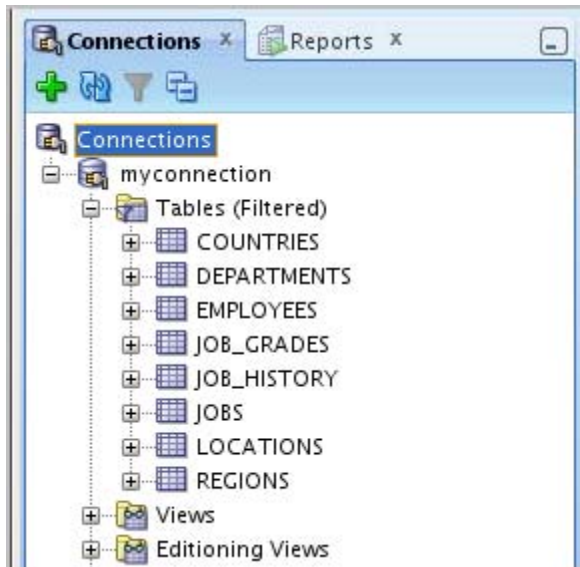
When you create a connection, a SQL Worksheet for that connection opens automatically.



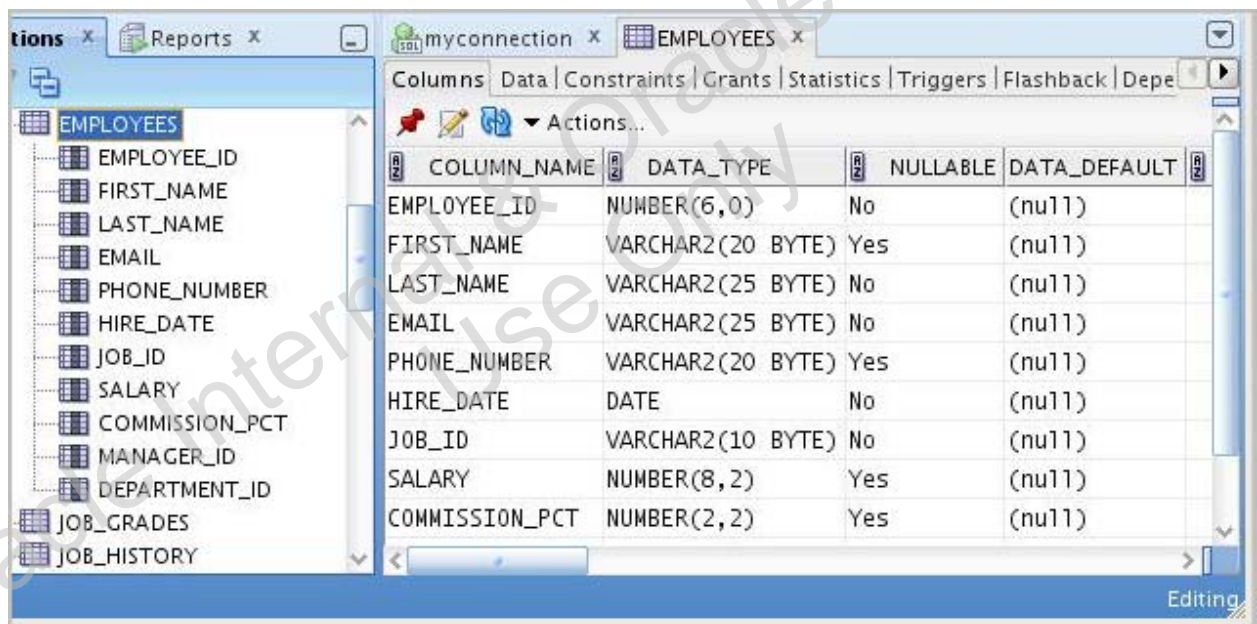
## Browsing the Tables in the Connections Navigator

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.

The screenshot shows the SQL Developer interface with the 'DEPARTMENTS' table selected in the 'Tables (Filtered)' pane on the left. The main pane displays the 'Data' tab for the 'DEPARTMENTS' table, showing a list of departments with their IDs, names, and manager IDs. The table has columns: DEPARTMENT\_ID, DEPARTMENT\_NAME, and MANAGER\_ID. The data is as follows:

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

The status bar at the bottom indicates 'DEPARTMENTS@myconnection' and 'Editing'.

Oracle Internal & Oracle Academy  
Use Only

## **Practices for Lesson 2: Retrieving Data Using the SQL SELECT Statement**

### **Chapter 2**

Oracle Internal & Oracle Academy  
Use Only

## Practices for Lesson 2

---

### Lesson Overview

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

Oracle Internal & Oracle Academy  
Use Only



## Practice 2-1: 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.

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, hiredate, 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\_02\_05.sql so that you can dispatch this file to the HR department.

6. Test your query in the `lab_02_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.

	EMPLOYEE_ID	LAST_NAME	JOB_ID	STARTDATE
1	100	King	AD_PRES	17-JUN-87
2	101	Kochhar	AD_VP	21-SEP-89
3	102	De Haan	AD_VP	13-JAN-93
4	103	Hunold	IT_PROG	03-JAN-90
5	104	Ernst	IT_PROG	21-MAY-91

...

19	205	Higgins	AC_MGR	07-JUN-94
20	206	Gietz	AC_ACCOUNT	07-JUN-94

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_02_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	100	King	AD_PRES	17-JUN-87
2	101	Kochhar	AD_VP	21-SEP-89
3	102	De Haan	AD_VP	13-JAN-93
4	103	Hunold	IT_PROG	03-JAN-90
5	104	Ernst	IT_PROG	21-MAY-91

...

19	205	Higgins	AC_MGR	07-JUN-94
20	206	Gietz	AC_ACCOUNT	07-JUN-94

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`.

	A Z	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`.

	A Z	THE_OUTPUT
1		100,Steven,King,SKING,515.123.4567,AD_PRES,,17-JUN-87,24000,,90
2		101,Neena,Kochhar,NK0CHHAR,515.123.4568,AD_VP,100,21-SEP-89,17000,,90
3		102,Lex,De Haan,LDEHAAN,515.123.4569,AD_VP,100,13-JAN-93,17000,,90
4		103,Alexander,Hunold,AHUNOLD,590.423.4567,IT_PROG,102,03-JAN-90,9000,,60
5		104,Bruce,Ernst,BERNST,590.423.4568,IT_PROG,103,21-MAY-91,6000,,60

...

19		205,Shelley,Higgins,SHIGGINS,515.123.8080,AC_MGR,101,07-JUN-94,12000,,110
20		206,William,Gietz,WGIETZ,515.123.8181,AC_ACCOUNT,205,07-JUN-94,8300,,110

## Solution 2-1: 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
```

- 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\_02\_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\_02\_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\_02\_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;
```

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 3: Restricting and Sorting Data**

### **Chapter 3**

Oracle Internal & Oracle Academy  
Use Only

## Practices for Lesson 3

---

### Lesson Overview

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.

Oracle Internal & Oracle Academy  
Use Only



## Practice 3-1: Restricting and Sorting Data

---

### Task

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_03_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_03_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_03_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

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\_03\_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\_03\_03.sql as lab\_03\_06.sql again. Run the statement in lab\_03\_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 of 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 the 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

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_03_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

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\_03\_06.sql to display the last name, salary, and commission for all employees whose commission is 20%. Save lab\_03\_06.sql as lab\_03\_15.sql again. Rerun the statement in lab\_03\_15.sql.

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

Oracle Internal & Oracle Academy  
Use Only

## Solution 3-1: 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_03_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_03_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_03_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_03_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_03_03.sql` as `lab_03_06.sql` again. Run the statement in `lab_03_06.sql`.

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

7. The HR department needs a report that displays the last name and hire date of 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;
```

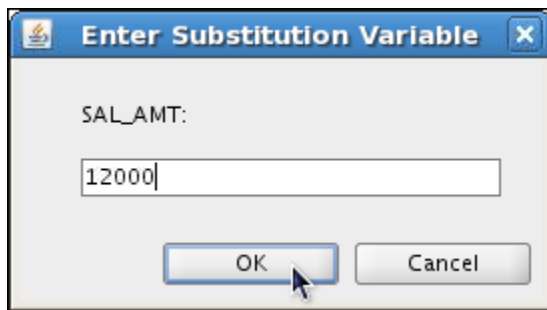
9. Create a report to display the last name, salary, and commission for all employees who earn commissions. Sort data in descending order of salary and commissions. Use the column's numeric position in the ORDER BY clause.

```
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\_03\_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\_03\_06.sql to display the last name, salary, and commission for all employees whose commission amount is 20%. Save lab\_03\_06.sql as lab\_03\_15.sql again. Rerun the statement in lab\_03\_15.sql.

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

Oracle Internal & Oracle Academy  
Use Only



# **Practices for Lesson 4: Using Single-Row Functions to Customize Output**

## **Chapter 4**

Oracle Internal & Oracle Academy  
Use Only

## Practices for Lesson 4

---

### Lesson Overview

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

Oracle Internal & Oracle Academy  
Use Only

## Practice 4-1: 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_04_02.sql`.
3. Run your query in the `lab_04_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_04_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_04_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

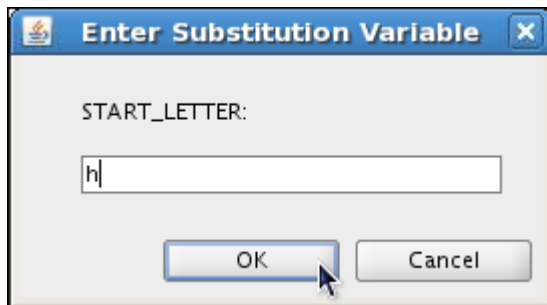
5. Write a query that displays the last name (with the first letter in uppercase and all the other letters in lowercase) and the length of the last name for all employees whose name starts with the letters "J," "A," or "M." Give each column an appropriate label. Sort the results by the employees' last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

	Name	Length
1	Abel	4
2	Matos	5
3	Mourgos	7

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

	Name	Length
1	Hartstein	9
2	Higgins	7
3	Hunold	6

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



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

	Name	Length
1	Hartstein	9
2	Higgins	7
3	Hunold	6

- 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. The number of months must be rounded 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.

	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 ***

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 as 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

## Solution 4-1: 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_04_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_04_02.sql`.

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

4. Modify your query `lab_04_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_04_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."

```
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. The number of months must be rounded 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;

```

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 as 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;

```

Oracle Internal & Oracle Academy  
Use Only



# **Practices for Lesson 5: Using Conversion Functions and Conditional Expressions**

## **Chapter 5**

Oracle Internal & Oracle Academy  
Use Only

## Practices for Lesson 5

---

### Lesson Overview

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.

Oracle Internal & Oracle Academy  
Use Only

## Practice 5-1: Using Conversion Functions and Conditional Expressions

- 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.

- 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

- 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

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

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

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

...

14	SA_REP	D
15	SA_REP	D

...

19	ST_CLERK	E
20	ST_MAN	B

## Solution 5-1: Using Conversion Functions and Conditional Expressions

---

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

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

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

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

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

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

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

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

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

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

```
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;
```

Oracle Internal & Oracle Academy  
Use Only

Oracle Internal & Oracle Academy  
Use Only



# **Practices for Lesson 6: Reporting Aggregated Data Using the Group Functions**

**Chapter 6**

Oracle Internal & Oracle Academy  
Use Only

## Practices for Lesson 6

---

### Lesson Overview

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

Oracle Internal & Oracle Academy  
Use Only

## Practice 6-1: 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_06_04.sql`. Run the query.

	Maximum	Minimum	Sum	Average
1	24000	2500	175500	8775

5. Modify the query in `lab_06_04.sql` to display the minimum, maximum, sum, and average salary for each job type. Save `lab_06_04.sql` as `lab_06_05.sql` again. Run the statement in `lab_06_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

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\_06\_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

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

## Solution 6-1: 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_06_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_06_04.sql` to display the minimum, maximum, sum, and average salary for each job type. Save `lab_06_04.sql` as `lab_06_05.sql` again. Run the statement in `lab_06_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_06_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;
```

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;
```

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;
```

Oracle Internal & Oracle Academy  
Use Only



# **Practices for Lesson 7: Displaying Data from Multiple Tables Using Joins**

**Chapter 7**

Oracle Internal & Oracle Academy  
Use Only

## Practices for Lesson 7

---

### Lesson Overview

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

Oracle Internal & Oracle Academy  
Use Only

## Practice 7-1: Displaying Data from Multiple Tables Using Joins

- 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

- 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

- 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

- 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_07_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

...

18 Taylor	176	Zlotkey	149
19 Abel	174	Zlotkey	149

5. Modify `lab_07_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_07_05.sql`. Run the query in `lab_07_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_07_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	

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.

R2	LAST_NAME	R2	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\_07\_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

## Solution 7-1: 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 with corresponding departments. 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_07_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_07_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_07_05.sql`. Run the query in `lab_07_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;
```

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_07_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_07_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;

```

Oracle Internal & Oracle Academy  
Use Only



## **Practices for Lesson 8: Using Subqueries to Solve Queries**

### **Chapter 8**

Oracle Internal & Oracle Academy  
Use Only

## Practices for Lesson 8

---

### Lesson Overview

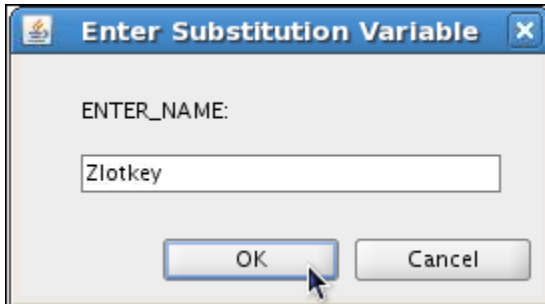
In this practice, you write complex queries using nested `SELECT` statements.

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.

Oracle Internal & Oracle Academy  
Use Only

## Practice 8-1: Using Subqueries to Solve Queries

1. The HR department needs a query that prompts the user for an employee's 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\_08\_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

- 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\_08\_04.sql.

- 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

- 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

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

	LAST_NAME
1	King
2	Kochhar
3	De Haan
4	Hartstein
5	Higgins
6	Abel
7	Zlotkey
8	Hunold

If you have the time, complete the following exercise:

8. Modify the query in `lab_08_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_08_03.sql` as `lab_08_08.sql` again. Run the statement in `lab_08_08.sql`.

	EMPLOYEE_ID	LAST_NAME	SALARY
1	103	Hunold	9000

Oracle Internal & Oracle Academy  
Use Only

## Solution 8-1: Using Subqueries to Solve Queries

---

1. The HR department needs a query that prompts the user for an employee's 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';
```

**Note:** UNDEFINE and SELECT are individual queries, execute them one after the other or press Ctrl + A + F9 to run them together.

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\_08\_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);
```

Modify the query so that the user is prompted for a location ID. Save this to a file named lab\_08\_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\_08\_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\_08\_03.sql to lab\_08\_08.sql again. Run the statement in lab\_08\_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);
```

Oracle Internal & Oracle Academy  
Use Only



## **Practices for Lesson 9: Using the Set Operators**

### **Chapter 9**

Oracle Internal & Oracle Academy  
Use Only

## Practices for Lesson 9

---

### Lesson Overview

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

Oracle Internal & Oracle Academy  
Use Only

## Practice 9-1: 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.

R	LAST_NAME	R	DEPARTMENT_ID	R	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	

## Solution 9-1: 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;
```

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;
```

Oracle Internal & Oracle Academy  
Use Only

## **Practices for Lesson 10: Manipulating Data**

### **Chapter 10**

Oracle Internal & Oracle Academy  
Use Only

## Practices for Lesson 10

---

### Lesson Overview

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.

Oracle Internal & Oracle Academy  
Use Only



## Practice 10-1: 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_10_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)
```

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\_10\_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\_10\_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\_10\_24.sql.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
6	Anthony	Mark	manthony	1230

25. Run the lab\_10\_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

## Solution 10-1: Manipulating Data

---

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

1. Run the statement in the lab\_10\_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\_10\_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;
```

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\_10\_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;
```

If you have time, complete the following exercise:

24. Modify the lab\_10\_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\_10\_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\_10\_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 11: Using DDL Statements to Create and Manage Tables**

**Chapter 11**

Oracle Internal & Oracle Academy  
Use Only

## Practices for Lesson 11

---

### Lesson Overview

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.

Oracle Internal & Oracle Academy  
Use Only



## Practice 11-1: Using DDL Statements to Create and Manage Tables

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

1. Create the DEPT table based on the following table instance chart. Save the statement in a script called lab\_11\_01.sql, and then execute the statement in the script to create the table. Confirm that the table is created.

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\_11\_03.sql, and then execute the statement in the script to create the table. Confirm that the table is created.

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

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

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. Please note that this option is supported in Oracle Database 11g.
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. Please note that this option is supported in Oracle Database 11g.

You should get the following messages:

```
table EMPLOYEES2 altered.
1 rows inserted.
```

8. Drop the EMPLOYEES2 table.

## Solution 11-1: Using DDL Statements to Create and Manage Tables

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

1. Create the DEPT table based on the following table instance chart. Save the statement in a script called lab\_11\_01.sql, and then execute the statement in the script to create the table. Confirm that the table is created.

```
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\_11\_03.sql, and then execute the statement in the script to create the table. Confirm that the table is created.

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

```
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)
```

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;
```

Oracle Internal & Oracle Academy  
Use Only

Oracle Internal & Oracle Academy  
Use Only

# **Practices for Lesson 12: Creating Other Schema Objects**

**Chapter 12**

Oracle Internal & Oracle Academy  
Use Only

## Practices for Lesson 12

---

### Lesson Overview

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.

Oracle Internal & Oracle Academy  
Use Only



## Practice 12-1: 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

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

## Part 2

- 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`.
- To test your sequence, write a script to insert two rows in the `DEPT` table. Name your script `lab_12_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.
- Create a nonunique index on the `NAME` column in the `DEPT` table.
- Create a synonym for your `EMPLOYEES` table. Call it `EMP`.

## Solution 12-1: 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.

## 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_12_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.

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

```
b) 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;
```

# **Additional Practices and Solutions**

## **Chapter 13**

Oracle Internal & Oracle Academy  
Use Only

## Practice 1-1

These exercises can be used for extra practice after you have discussed the following topics: basic SQL `SELECT` statement, basic SQL Developer commands, and SQL functions.

1. The HR department needs to find data for all the clerks who were hired after the year 1997.

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
1	143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98	ST_CLERK	2600
2	144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-98	ST_CLERK	2500


2. The HR department needs a report of employees who earn commission. Show the last name, job, salary, and commission of those employees. Sort the data by salary in descending order.

	LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
1	Abel	SA_REP	11000	0.3
2	Zlotkey	SA_MAN	10500	0.2
3	Taylor	SA_REP	8600	0.2
4	Grant	SA_REP	7000	0.15


3. For budgeting purposes, the HR department needs a report on projected raises. The report should display those employees who have no commission, but who have a 10% raise in salary (round off the salaries).

	New salary
1	The salary of King after a 10% raise is 26400
2	The salary of Kochhar after a 10% raise is 18700
3	The salary of De Haan after a 10% raise is 18700
4	The salary of Hunold after a 10% raise is 9900
5	The salary of Ernst after a 10% raise is 6600
6	The salary of Lorentz after a 10% raise is 4620
7	The salary of Mourgos after a 10% raise is 6380
8	The salary of Rajs after a 10% raise is 3850
9	The salary of Davies after a 10% raise is 3410
10	The salary of Matos after a 10% raise is 2860
11	The salary of Vargas after a 10% raise is 2750
12	The salary of Whalen after a 10% raise is 4840
13	The salary of Hartstein after a 10% raise is 14300
14	The salary of Fay after a 10% raise is 6600
15	The salary of Higgins after a 10% raise is 13200
16	The salary of Gietz after a 10% raise is 9130

4. Create a report of employees and their length of employment. Show the last names of all the employees together with the number of years and the number of completed months that they have been employed. Order the report by the length of their employment. The employee who has been employed the longest should appear at the top of the list.

	 LAST_NAME	 YEARS	 MONTHS
1	King	22	0
2	Whalen	21	9
3	Kochhar	19	9
4	Hunold	19	6
5	Ernst	18	1
6	De Haan	16	6
7	Higgins	15	1
8	Gietz	15	1
9	Rajs	13	8
10	Hartstein	13	4
11	Abel	13	2
12	Davies	12	5
13	Fay	11	10
14	Matos	11	4
15	Taylor	11	3
16	Vargas	11	0
17	Lorentz	10	5
18	Grant	10	1
19	Mourgos	9	7
20	Zlotkey	9	5

5. Show those employees who have a last name starting with the letters "J," "K," "L," or "M."

	 LAST_NAME
1	King
2	Kochhar
3	Lorentz
4	Matos
5	Mourgos

6. Create a report that displays all employees, and indicate with the words *Yes* or *No* whether they receive a commission. Use the `DECODE` expression in your query.

	LAST_NAME	SALARY	COMMISSION
1	King	24000	No
2	Kochhar	17000	No
3	De Haan	17000	No
4	Hunold	9000	No
5	Ernst	6000	No
6	Lorentz	4200	No
7	Mourgos	5800	No
8	Rajs	3500	No
9	Davies	3100	No
10	Matos	2600	No
11	Vargas	2500	No
12	Zlotkey	10500	Yes
13	Abel	11000	Yes
14	Taylor	8600	Yes
15	Grant	7000	Yes
16	Whalen	4400	No
17	Hartstein	13000	No
18	Fay	6000	No
19	Higgins	12000	No
20	Gietz	8300	No

These exercises can be used for extra practice after you have discussed the following topics: basic SQL `SELECT` statement, basic SQL Developer commands, SQL functions, joins, and group functions.

7. Create a report that displays the department name, location ID, last name, job title, and salary of those employees who work in a specific location. Prompt the user for the location. For example, if the user enters 1800, these are the results:

	DEPARTMENT_NAME	LOCATION_ID	LAST_NAME	JOB_ID	SALARY
1	Marketing	1800	Hartstein	MK_MAN	13000
2	Marketing	1800	Fay	MK_REP	6000

8. Find the number of employees who have a last name that ends with the letter "n." Create two possible solutions.

	COUNT(*)
1	3



9. Create a report that shows the name, location, and number of employees for each department. Make sure that the report also includes departments without employees.

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	COUNT(E.EMPLOYEE_ID)
1	80	Sales	2500	3
2	110	Accounting	1700	2
3	10	Administration	1700	1
4	60	IT	1400	3
5	20	Marketing	1800	2
6	90	Executive	1700	3
7	50	Shipping	1500	5
8	190	Contracting	1700	0

10. The HR department needs to find the job titles in departments 10 and 20. Create a report to display the job IDs for those departments.

JOB_ID
AD_ASST
MK_MAN
MK_REP

11. Create a report that displays the jobs that are found in the Administration and Executive departments. Also display the number of employees for these jobs. Show the job with the highest number of employees first.

JOB_ID	FREQUENCY
AD_VP	2
AD PRES	1
AD_ASST	1

These exercises can be used for extra practice after you have discussed the following topics: basic SQL SELECT statements, basic SQL Developer commands, SQL functions, joins, group functions, and subqueries.

12. Show all the employees who were hired in the first half of the month (before the 16th of the month).

LAST_NAME	HIRE_DATE
De Haan	13-JAN-93
Hunold	03-JAN-90
Lorentz	07-FEB-99
Matos	15-MAR-98
Vargas	09-JUL-98
Abel	11-MAY-96
Higgins	07-JUN-94
Gietz	07-JUN-94

13. Create a report that displays the following for all employees: last name, salary, and salary expressed in terms of thousands of dollars.

R	LAST_NAME	R	SALARY	R	THOUSANDS
1	King		24000		24
2	Kochhar		17000		17
3	De Haan		17000		17
4	Hunold		9000		9
5	Ernst		6000		6
6	Lorentz		4200		4
7	Mourgos		5800		5
8	Rajs		3500		3
9	Davies		3100		3
10	Matos		2600		2
11	Vargas		2500		2
12	Zlotkey		10500		10
13	Abel		11000		11
14	Taylor		8600		8
15	Grant		7000		7
16	Whalen		4400		4
17	Hartstein		13000		13
18	Fay		6000		6
19	Higgins		12000		12
20	Gietz		8300		8

14. Show all the employees who have managers with a salary higher than \$15,000. Show the following data: employee name, manager name, manager salary, and salary grade of the manager.

R	LAST_NAME	R	MANAGER	R	SALARY	R	GRADE_LEVEL
1	De Haan		King		24000		E
2	Hartstein		King		24000		E
3	Higgins		Kochhar		17000		E
4	Hunold		De Haan		17000		E
5	Kochhar		King		24000		E
6	Mourgos		King		24000		E
7	Whalen		Kochhar		17000		E
8	Zlotkey		King		24000		E

15. Show the department number, name, number of employees, and average salary of all the departments, together with the names, salaries, and jobs of the employees working in each department.

	DEPARTMENT_ID	DEPARTMENT_NAME	EMPLOYEES	AVG_SAL	LAST_NAME	SALARY	JOB_ID
1	10	Administration	1	4400.00	Whalen	4400	AD_ASST
2	20	Marketing	2	9500.00	Hartstein	13000	MK_MAN
3	20	Marketing	2	9500.00	Fay	6000	MK_REP
4	50	Shipping	5	3500.00	Davies	3100	ST_CLERK
5	50	Shipping	5	3500.00	Matos	2600	ST_CLERK
6	50	Shipping	5	3500.00	Rajs	3500	ST_CLERK
7	50	Shipping	5	3500.00	Mourgos	5800	ST_MAN
8	50	Shipping	5	3500.00	Vargas	2500	ST_CLERK
9	60	IT	3	6400.00	Hunold	9000	IT_PROG
10	60	IT	3	6400.00	Lorentz	4200	IT_PROG
11	60	IT	3	6400.00	Ernst	6000	IT_PROG
12	80	Sales	3	10033.33	Zlotkey	10500	SA_MAN
13	80	Sales	3	10033.33	Taylor	8600	SA_REP
14	80	Sales	3	10033.33	Abel	11000	SA_REP
15	90	Executive	3	19333.33	Kochhar	17000	AD_VP
16	90	Executive	3	19333.33	De Haan	17000	AD_VP
17	90	Executive	3	19333.33	King	24000	AD_PRES
18	110	Accounting	2	10150.00	Gietz	8300	AC_ACCOUNT
19	110	Accounting	2	10150.00	Higgins	12000	AC_MGR
20	(null)	(null)	0	No average	Grant	7000	SA_REP

16. Create a report to display the department number and lowest salary of the department with the highest average salary.

	DEPARTMENT_ID	MIN(SALARY)
1	90	17000

17. Create a report that displays departments where no sales representatives work. Include the department number, department name, manager ID, and the location in the output.

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

18. Create the following statistical reports for the HR department: Include the department number, department name, and the number of employees working in each department that:
- a. Employs fewer than three employees:

	DEPARTMENT_ID	DEPARTMENT_NAME	COUNT(*)
1	10	Administration	1
2	110	Accounting	2
3	20	Marketing	2

- b. Has the highest number of employees:

	DEPARTMENT_ID	DEPARTMENT_NAME	COUNT(*)
1	50	Shipping	5

- c. Has the lowest number of employees:

	DEPARTMENT_ID	DEPARTMENT_NAME	COUNT(*)
1	10	Administration	1

19. Create a report that displays the employee number, last name, salary, department number, and the average salary in their department for all employees.

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	SALARY	AVG(S.SALARY)
1	149	Zlotkey	80	10500	10033.3333333333...
2	174	Abel	80	11000	10033.3333333333...
3	144	Vargas	50	2500	3500
4	101	Kochhar	90	17000	19333.3333333333...
5	100	King	90	24000	19333.3333333333...
6	103	Hunold	60	9000	6400
7	142	Davies	50	3100	3500
8	205	Higgins	110	12000	10150
9	104	Ernst	60	6000	6400
10	143	Matos	50	2600	3500
11	102	De Haan	90	17000	19333.3333333333...
12	107	Lorentz	60	4200	6400
13	141	Rajs	50	3500	3500
14	200	Whalen	10	4400	4400
15	202	Fay	20	6000	9500
16	176	Taylor	80	8600	10033.3333333333...
17	201	Hartstein	20	13000	9500
18	206	Gietz	110	8300	10150
19	124	Mourgos	50	5800	3500

20. Show all the employees who were hired on the day of the week on which the highest number of employees were hired.

	LAST_NAME	DAY
1	Ernst	TUESDAY
2	Mourgos	TUESDAY
3	Rajs	TUESDAY
4	Taylor	TUESDAY
5	Higgins	TUESDAY
6	Gietz	TUESDAY

21. Create an anniversary overview based on the hire date of the employees. Sort the anniversaries in ascending order.

	LAST_NAME	BIRTHDAY
1	Hunold	January 03
2	De Haan	January 13
3	Davies	January 29
4	Zlotkey	January 29
5	Lorentz	February 07
6	Hartstein	February 17
7	Matos	March 15
8	Taylor	March 24
9	Abel	May 11
10	Ernst	May 21
11	Grant	May 24
12	Higgins	June 07
13	Gietz	June 07
14	King	June 17
15	Vargas	July 09
16	Fay	August 17
17	Whalen	September 17
18	Kochhar	September 21
19	Rajs	October 17
20	Mourgos	November 16

## Solution 1-1

---

These exercises can be used for extra practice after you have discussed the following topics: basic SQL `SELECT` statement, basic SQL Developer commands, and SQL functions.

1. The HR department needs to find data for all of the clerks who were hired after the year 1997.

```
SELECT *
FROM   employees
WHERE  job_id = 'ST_CLERK'
AND    hire_date > '31-DEC-1997';
```

2. The HR department needs a report of employees who earn commission. Show the last name, job, salary, and commission of those employees. Sort the data by salary in descending order.

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

3. For budgeting purposes, the HR department needs a report on projected raises. The report should display those employees who do not get a commission but who have a 10% raise in salary (round off the salaries).

```
SELECT 'The salary of '||last_name||' after a 10% raise is '
      || ROUND(salary*1.10) "New salary"
FROM   employees
WHERE  commission_pct IS NULL;
```

4. Create a report of employees and their duration of employment. Show the last names of all employees together with the number of years and the number of completed months that they have been employed. Order the report by the duration of their employment. The employee who has been employed the longest should appear at the top of the list.

```
SELECT last_name,
       TRUNC(MONTHS_BETWEEN(SYSDATE, hire_date) / 12) YEARS,
       TRUNC(MOD(MONTHS_BETWEEN(SYSDATE, hire_date), 12))
       MONTHS
FROM   employees
ORDER BY years DESC, MONTHS desc;
```

5. Show those employees who have a last name starting with the letters "J," "K," "L," or "M."

```
SELECT last_name
FROM   employees
WHERE  SUBSTR(last_name, 1,1) IN ('J', 'K', 'L', 'M');
```

6. Create a report that displays all employees, and indicate with the words Yes or No whether they receive a commission. Use the `DECODE` expression in your query.

```
SELECT last_name, salary,
       decode(commission_pct, NULL, 'No', 'Yes') commission
FROM   employees;
```

These exercises can be used for extra practice after you have discussed the following topics: basic SQL `SELECT` statement, basic SQL Developer commands, SQL functions, joins, and group functions.

7. Create a report that displays the department name, location ID, name, job title, and salary of those employees who work in a specific location. Prompt the user for the location.

a. Enter 1800 for `location_id` when prompted.

```
SELECT d.department_name, d.location_id, e.last_name, e.job_id,
       e.salary
FROM   employees e, departments d
WHERE  e.department_id = d.department_id
AND    d.location_id = &location_id;
```

8. Find the number of employees who have a last name that ends with the letter "n." Create two possible solutions.

```
SELECT COUNT(*)
FROM   employees
WHERE  last_name LIKE '%n';
--or
SELECT COUNT(*)
FROM   employees
WHERE  SUBSTR(last_name, -1) = 'n';
```

9. Create a report that shows the name, location, and number of employees for each department. Make sure that the report also includes departments without employees.

```
SELECT d.department_id, d.department_name,
       d.location_id, COUNT(e.employee_id)
FROM   employees e RIGHT OUTER JOIN departments d
ON     e.department_id = d.department_id
GROUP BY d.department_id, d.department_name, d.location_id;
```

10. The HR department needs to find the job titles in departments 10 and 20. Create a report to display the job IDs for those departments.

```
SELECT DISTINCT job_id
FROM   employees
WHERE  department_id IN (10, 20);
```

11. Create a report that displays the jobs that are found in the Administration and Executive departments. Also display the number of employees for these jobs. Show the job with the highest number of employees first.

```
SELECT e.job_id, count(e.job_id) FREQUENCY
FROM   employees e JOIN departments d
ON     e.department_id = d.department_id
WHERE  d.department_name IN ('Administration', 'Executive')
GROUP BY e.job_id
ORDER BY FREQUENCY DESC;
```

These exercises can be used for extra practice after you have discussed the following topics: basic SQL `SELECT` statements, basic SQL Developer commands, SQL functions, joins, group functions, and subqueries.

12. Show all employees who were hired in the first half of the month (before the 16th of the month).

```
SELECT last_name, hire_date
FROM   employees
WHERE  TO_CHAR(hire_date, 'DD') < 16;
```

13. Create a report that displays the following for all employees: last name, salary, and salary expressed in terms of thousands of dollars.

```
SELECT last_name, salary, TRUNC(salary, -3)/1000 Thousands
FROM   employees;
```

14. Show all employees who have managers with a salary higher than \$15,000. Show the following data: employee name, manager name, manager salary, and salary grade of the manager.

```
SELECT e.last_name, m.last_name manager, m.salary, j.grade_level
FROM   employees e JOIN employees m
ON     e.manager_id = m.employee_id
JOIN   job_grades j
ON     m.salary BETWEEN j.lowest_sal AND j.highest_sal
AND    m.salary > 15000;
```



15. Show the department number, name, number of employees, and average salary of all departments, together with the names, salaries, and jobs of the employees working in each department.

```
SELECT  d.department_id, d.department_name,
        count(e1.employee_id) employees,
        NVL(TO_CHAR(AVG(e1.salary), '99999.99'), 'No average' )
avg_sal,
        e2.last_name, e2.salary, e2.job_id
FROM    departments d RIGHT OUTER JOIN employees e1
ON      d.department_id = e1.department_id
RIGHT OUTER JOIN  employees e2
ON      d.department_id = e2.department_id
GROUP BY d.department_id, d.department_name, e2.last_name,
e2.salary,
        e2.job_id
ORDER BY d.department_id, employees;
```

16. Create a report to display the department number and lowest salary of the department with the highest average salary.

```
SELECT department_id, MIN(salary)
FROM    employees
GROUP BY department_id
HAVING AVG(salary) = (SELECT MAX(AVG(salary))
                     FROM    employees
                     GROUP BY department_id);
```

17. Create a report that displays the departments where no sales representatives work. Include the department number, department name, manager id and location in the output.

```
SELECT *
FROM    departments
WHERE   department_id NOT IN(SELECT department_id
                             FROM employees
                             WHERE job_id = 'SA_REP'
                             AND department_id IS NOT NULL);
```

18. Create the following statistical reports for the HR department: Include the department number, department name, and the number of employees working in each department that:

a. Employs fewer than three employees:

```
SELECT d.department_id, d.department_name, COUNT(*)
FROM    departments d JOIN employees e
ON      d.department_id = e.department_id
GROUP BY d.department_id, d.department_name
HAVING COUNT(*) < 3;
```

b. Has the highest number of employees:

```
SELECT d.department_id, d.department_name, COUNT(*)
FROM   departments d JOIN employees e
ON     d.department_id = e.department_id
GROUP BY d.department_id, d.department_name
HAVING COUNT(*) = (SELECT MAX(COUNT(*))
                   FROM   employees
                   GROUP BY department_id);
```

c. Has the lowest number of employees:

```
SELECT d.department_id, d.department_name, COUNT(*)
FROM   departments d JOIN employees e
ON     d.department_id = e.department_id
GROUP BY d.department_id, d.department_name
HAVING COUNT(*) = (SELECT MIN(COUNT(*))
                   FROM   employees
                   GROUP BY department_id);
```

19. Create a report that displays the employee number, last name, salary, department number, and the average salary in their department for all employees.

```
SELECT e.employee_id, e.last_name, e.department_id, e.salary,
       AVG(s.salary)
FROM   employees e JOIN employees s
ON     e.department_id = s.department_id
GROUP BY e.employee_id, e.last_name, e.department_id,
         e.salary;
```

20. Show all employees who were hired on the day of the week on which the highest number of employees were hired.

```
SELECT last_name, TO_CHAR(hire_date, 'DAY') day
FROM   employees
WHERE  TO_CHAR(hire_date, 'Day') =
       (SELECT TO_CHAR(hire_date, 'Day')
        FROM   employees
        GROUP BY TO_CHAR(hire_date, 'Day')
        HAVING COUNT(*) = (SELECT MAX(COUNT(*))
                           FROM   employees
                           GROUP BY TO_CHAR(hire_date,
                                         'Day')));
```

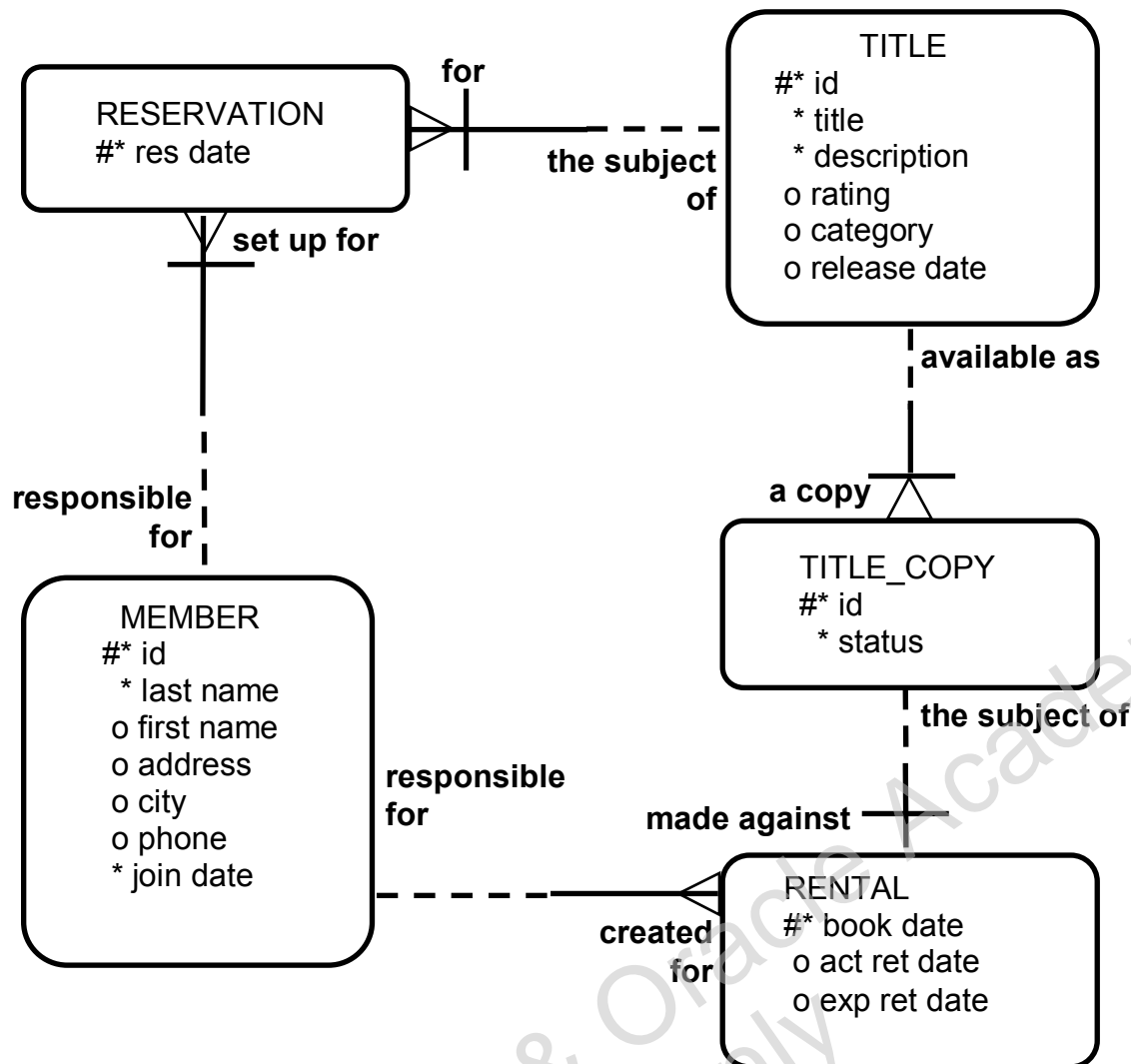
21. Create an anniversary overview based on the hire date of the employees. Sort the anniversaries in ascending order.

```
SELECT last_name, TO_CHAR(hire_date, 'Month DD') BIRTHDAY
FROM   employees
ORDER BY TO_CHAR(hire_date, 'DDD');
```

## Case Study

In this case study, you build a set of database tables for a video application. After you create the tables, you insert, update, and delete records in a video store database and generate a report. The database contains only the essential tables.

The following is a diagram of the entities and attributes for the video application:



**Note:** If you want to build the tables, you can execute the commands in the `buildtab.sql` script in SQL Developer. If you want to drop the tables, you can execute the commands in the `dropvid.sql` script in SQL Developer. Then you can execute the commands in the `buildvid.sql` script in SQL Developer to create and populate the tables.

All the three SQL scripts are present in the `/home/oracle/labs/sql1/labs` folder.

- If you use the `buildtab.sql` script to build the tables, start with step 4.
- If you use the `dropvid.sql` script to remove the video tables, start with step 1.
- If you use the `buildvid.sql` script to build and populate the tables, start with step 6(b)

## Practice 2-1

1. Create the tables based on the following table instance charts. Choose the appropriate data types and be sure to add integrity constraints.
  - a. Table name: MEMBER

Column_ Name	MEMBER_ ID	LAST_ NAME	FIRST_NAME	ADDRESS	CITY	PHONE	JOIN DATE
Key Type	PK						
Null/ Unique	NN,U	NN					NN
Default Value							System Date
Data Type	NUMBER	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	DATE
Length	10	25	25	100	30	15	

- b. Table name: TITLE

Column_ Name	TITLE_ID	TITLE	DESCRIPTION	RATING	CATEGORY	RELEASE DATE
Key Type	PK					
Null/ Unique	NN,U	NN	NN			
Check				G, PG, R, NC17, NR	DRAMA, COMEDY, ACTION, CHILD, SCIFI, DOCUMENTARY	
Data Type	NUMBER	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	DATE
Length	10	60	400	4	20	

c. Table name: TITLE\_COPY

<b>Column Name</b>	COPY_ID	TITLE_ID	STATUS
<b>Key Type</b>	PK	PK,FK	
<b>Null/Unique</b>	NN,U	NN,U	NN
<b>Check</b>			AVAILABLE, DESTROYED, RENTED, RESERVED
<b>FK Ref Table</b>		TITLE	
<b>FK Ref Col</b>		TITLE_ID	
<b>Data Type</b>	NUMBER	NUMBER	VARCHAR2
<b>Length</b>	10	10	15

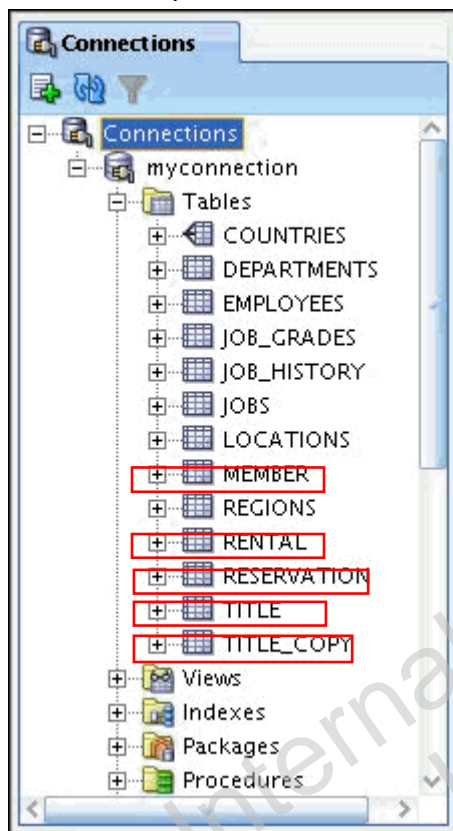
d. Table name: RENTAL

<b>Column Name</b>	BOOK_DATE	MEMBER_ID	COPY_ID	ACT_RET_DATE	EXP_RET_DATE	TITLE_ID
<b>Key Type</b>	PK	PK,FK1	PK,FK2			PK,FK2
<b>Default Value</b>	System Date				System Date + 2 days	
<b>FK Ref Table</b>		MEMBER	TITLE_COPY			TITLE_COPY
<b>FK Ref Col</b>		MEMBER_ID	COPY_ID			TITLE_ID
<b>Data Type</b>	DATE	NUMBER	NUMBER	DATE	DATE	NUMBER
<b>Length</b>		10	10			10

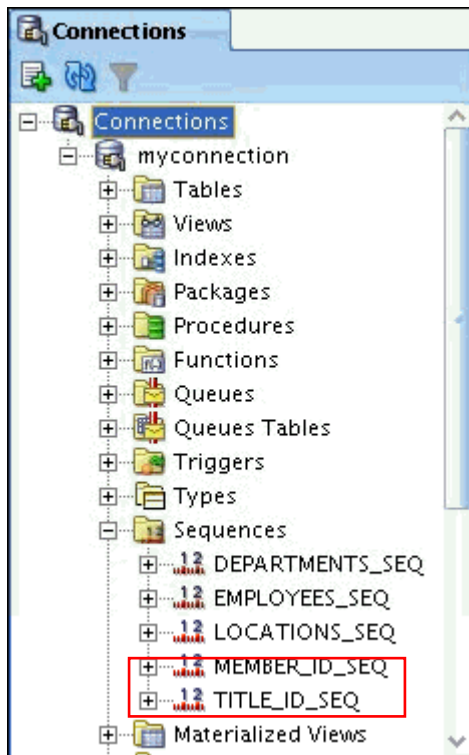
e. Table name: RESERVATION

Column Name	RES_DATE	MEMBER_ID	TITLE_ID
Key Type	PK	PK,FK1	PK,FK2
Null/Unique	NN,U	NN,U	NN
FK Ref Table		MEMBER	TITLE
FK Ref Column		MEMBER_ID	TITLE_ID
Data Type	DATE	NUMBER	NUMBER
Length		10	10

2. Verify that the tables were created properly by checking in the Connections Navigator in SQL Developer.



3. Create sequences to uniquely identify each row in the `MEMBER` table and the `TITLE` table.
  - a. Member number for the `MEMBER` table: Start with 100; do not allow caching of the values. Name the sequence `MEMBER_ID_SEQ`.
  - b. Title number for the `TITLE` table: Start with 91; do not allow caching of the values. Name the sequence `TITLE_ID_SEQ`.
  - c. Verify the existence of the sequences in the Connections Navigator in SQL Developer.



4. Add data to the tables. Create a script for each set of data to be added.
  - a. Add movie titles to the `TITLE` table. Write a script to enter the movie information. Save the statements in a script named `lab_apcs_4a.sql`. Use the sequences to uniquely identify each title. Enter the release dates in the DD-MON-YYYY format. Remember that single quotation marks in a character field must be specially handled. Verify your additions.

	TITLE
1	Willie and Christmas Too
2	Alien Again
3	The Glob
4	My Day Off
5	Miracles on Ice
6	Soda Gang

Title	Description	Rating	Category	Release_date
Willie and Christmas Too	All of Willie's friends make a Christmas list for Santa, but Willie has yet to add his own wish list.	G	CHILD	05-OCT-1995
Alien Again	Yet another installation of science fiction history. Can the heroine save the planet from the alien life form?	R	SCIFI	19-MAY-1995
The Glob	A meteor crashes near a small American town and unleashes carnivorous goo in this classic.	NR	SCIFI	12-AUG-1995
My Day Off	With a little luck and a lot of ingenuity, a teenager skips school for a day in New York.	PG	COMEDY	12-JUL-1995
Miracles on Ice	A six-year-old has doubts about Santa Claus, but she discovers that miracles really do exist.	PG	DRAMA	12-SEP-1995
Soda Gang	After discovering a cache of drugs, a young couple find themselves pitted against a vicious gang.	NR	ACTION	01-JUN-1995

- b. Add data to the `MEMBER` table. Save the insert statements in a script named `lab_apcs_4b.sql`. Execute commands in the script. Be sure to use the sequence to add the member numbers.

First_Name	Last_Name	Address	City	Phone	Join_Date
Carmen	Velasquez	283 King Street	Seattle	206-899-6666	08-MAR-1990
LaDoris	Ngao	5 Modrany	Bratislava	586-355-8882	08-MAR-1990
Midori	Nagayama	68 Via Centrale	Sao Paolo	254-852-5764	17-JUN-1991
Mark	Quick-to-See	6921 King Way	Lagos	63-559-7777	07-APR-1990



Audry	Ropeburn	86 Chu Street	Hong Kong	41-559-87	18-JAN-1991
Molly	Urguhart	3035 Laurier	Quebec	418-542-9988	18-JAN-1991

c. Add the following movie copies in the `TITLE_COPY` table:

**Note:** Have the `TITLE_ID` numbers available for this exercise.

Title	Copy_Id	Status	Title	Copy_Id
Willie and Christmas Too	1	Available	Willie and Christmas Too	1
Alien Again	1	Available	Alien Again	1
	2	Rented		2
The Glob	1	Available	The Glob	1
My Day Off	1	Available	My Day Off	1
	2	Available		2
	3	Rented		3
Miracles on Ice	1	Available	Miracles on Ice	1
Soda Gang	1	Available	Soda Gang	1

d. Add the following rentals to the `RENTAL` table:

**Note:** The title number may be different depending on the sequence number.

Title_Id	Copy_Id	Member_Id	Book_date	Exp_Ret_Date
92	1	101	3 days ago	1 day ago
93	2	101	1 day ago	1 day from now
95	3	102	2 days ago	Today
97	1	106	4 days ago	2 days ago

5. Create a view named `TITLE_AVAIL` to show the movie titles, the availability of each copy, and its expected return date if rented. Query all rows from the view. Order the results by title.

**Note:** Your results may be different.

	TITLE	COPY_ID	STATUS	EXP_RET_DATE
1	Alien Again	1	AVAILABLE	(null)
2	Alien Again	2	RENTED	15-JUL-09
3	Miracles on Ice	1	AVAILABLE	(null)
4	My Day Off	1	AVAILABLE	(null)
5	My Day Off	2	AVAILABLE	(null)
6	My Day Off	3	RENTED	16-JUL-09
7	Soda Gang	1	AVAILABLE	14-JUL-09
8	The Glob	1	AVAILABLE	(null)
9	Willie and Christmas Too	1	AVAILABLE	15-JUL-09

6. Make changes to the data in the tables.
- Add a new title. The movie is "Interstellar Wars," which is rated PG and classified as a science fiction movie. The release date is 07-JUL-77. The description is "Futuristic interstellar action movie. Can the rebels save the humans from the evil empire?" Be sure to add a title copy record for two copies.
  - Enter two reservations. One reservation is for Carmen Velasquez, who wants to rent "Interstellar Wars." The other is for Mark Quick-to-See, who wants to rent "Soda Gang."

7. Make a modification to one of the tables.

- a. Run the `lab_apcs_7a.sql` script located in the `/home/oracle/labs/sql1/labs` folder, to add a `PRICE` column to the `TITLE` table to record the purchase price of the video. Verify your modifications.

DESCRIBE title		
Name	Null	Type
-----		
TITLE_ID	NOT NULL	NUMBER(10)
TITLE	NOT NULL	VARCHAR2(60)
DESCRIPTION	NOT NULL	VARCHAR2(400)
RATING		VARCHAR2(4)
CATEGORY		VARCHAR2(20)
RELEASE_DATE		DATE
PRICE		NUMBER(8,2)

Title	Price
Willie and Christmas Too	25
Alien Again	35
The Glob	35
My Day Off	35
Miracles on Ice	30
Soda Gang	35
Interstellar Wars	29

- b. Create a script named `lab_apcs_7b.sql` that contains update statements that update each video with a price according to the preceding list. Run the commands in the script.

**Note:** Have the `TITLE_ID` numbers available for this exercise.

8. Create a report that contains each customer's history of renting videos. Be sure to include the customer name, movie rented, dates of the rental, and duration of rentals. Total the number of rentals for all customers for the reporting period. Save the commands that generate the report in a script file named `lab_apcs_8.sql`.

**Note:** Your results may be different.

	MEMBER	TITLE	BOOK_DATE	DURATION
1	Carmen Velasquez	Willie and Christmas Too	13-JUL-09	1
2	Carmen Velasquez	Alien Again	15-JUL-09	(null)
3	LaDoris Ngao	My Day Off	14-JUL-09	(null)
4	Molly Urganhart	Soda Gang	12-JUL-09	2

## Solution 2-1

1. Create the tables based on the following table instance charts. Choose the appropriate data types and be sure to add integrity constraints.

- a. Table name: MEMBER

```
CREATE TABLE member
(
    member_id      NUMBER(10)
        CONSTRAINT member_member_id_pk PRIMARY KEY,
    last_name      VARCHAR2(25)
        CONSTRAINT member_last_name_nn NOT NULL,
    first_name     VARCHAR2(25),
    address        VARCHAR2(100),
    city           VARCHAR2(30),
    phone          VARCHAR2(15),
    join_date      DATE DEFAULT SYSDATE
        CONSTRAINT member_join_date_nn NOT NULL);
```

- b. Table name: TITLE

```
CREATE TABLE title
(
    title_id       NUMBER(10)
        CONSTRAINT title_title_id_pk PRIMARY KEY,
    title          VARCHAR2(60)
        CONSTRAINT title_title_nn NOT NULL,
    description    VARCHAR2(400)
        CONSTRAINT title_description_nn NOT NULL,
    rating         VARCHAR2(4)
        CONSTRAINT title_rating_ck CHECK
            (rating IN ('G', 'PG', 'R', 'NC17', 'NR')),
    category       VARCHAR2(20)
        CONSTRAINT title_category_ck CHECK
            (category IN ('DRAMA', 'COMEDY', 'ACTION',
                'CHILD', 'SCIFI', 'DOCUMENTARY')),
    release_date   DATE);
```

- c. Table name: TITLE\_COPY

```
CREATE TABLE title_copy
(
    copy_id        NUMBER(10),
    title_id       NUMBER(10)
        CONSTRAINT title_copy_title_if_fk REFERENCES
            title(title_id),
    status         VARCHAR2(15)
        CONSTRAINT title_copy_status_nn NOT NULL
        CONSTRAINT title_copy_status_ck CHECK (status IN
            ('AVAILABLE', 'DESTROYED', 'RENTED', 'RESERVED')),
    CONSTRAINT title_copy_copy_id_title_id_pk
```

```
PRIMARY KEY (copy_id, title_id));
```

d. Table name: RENTAL

```
CREATE TABLE rental
(
    book_date DATE DEFAULT SYSDATE,
    member_id NUMBER(10)
    CONSTRAINT rental_member_id_fk REFERENCES
member(member_id),
    copy_id NUMBER(10),
    act_ret_date DATE,
    exp_ret_date DATE DEFAULT SYSDATE + 2,
    title_id NUMBER(10),
    CONSTRAINT rental_book_date_copy_title_pk
PRIMARY KEY (book_date, member_id, copy_id, title_id),
    CONSTRAINT rental_copy_id_title_id_fk
FOREIGN KEY (copy_id, title_id)
REFERENCES title_copy(copy_id, title_id));
```

e. Table name: RESERVATION

```
CREATE TABLE reservation
(
    res_date DATE,
    member_id NUMBER(10)
    CONSTRAINT reservation_member_id REFERENCES
member(member_id),
    title_id NUMBER(10)
    CONSTRAINT reservation_title_id REFERENCES
title(title_id),
    CONSTRAINT reservation_resdate_mem_tit_pk PRIMARY KEY
(res_date, member_id, title_id));
```

2. Verify that the tables were created properly by checking in the Connections Navigator in SQL Developer.

a. In the Connections Navigator, expand Connections > myconnection > Tables.

3. Create sequences to uniquely identify each row in the MEMBER table and the TITLE table.

a. Member number for the MEMBER table: Start with 100; do not allow caching of the values. Name the sequence MEMBER\_ID\_SEQ.

```
CREATE SEQUENCE member_id_seq
START WITH 100
NOCACHE;
```

b. Title number for the TITLE table: Start with 91; do not allow caching of the values. Name the sequence TITLE\_ID\_SEQ.

```
CREATE SEQUENCE title_id_seq
START WITH 91
NOCACHE;
```

- c. Verify the existence of the sequences in the Connections Navigator in SQL Developer.

```
SELECT  sequence_name, increment_by, last_number
FROM    user_sequences
WHERE   sequence_name IN ('MEMBER_ID_SEQ', 'TITLE_ID_SEQ');
```

- d. In the Connections Navigator, assuming that the myconnection node is expanded, expand Sequences.

4. Add data to the tables. Create a script for each set of data to be added.

- a. Add movie titles to the `TITLE` table. Write a script to enter the movie information. Save the statements in a script named `lab_apcs_4a.sql`. Use the sequences to uniquely identify each title. Enter the release dates in the DD-MON-YYYY format. Remember that single quotation marks in a character field must be specially handled. Verify your additions.

```
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'Willie and Christmas Too',
        'All of Willie's friends make a Christmas list for
        Santa, but Willie has yet to add his own wish list.',
        'G', 'CHILD', TO_DATE('05-OCT-1995','DD-MON-YYYY'))
/
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'Alien Again', 'Yet another
        installment of science fiction history. Can the
        heroine save the planet from the alien life form?',
        'R', 'SCIFI', TO_DATE('19-MAY-1995','DD-MON-YYYY'))
/
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'The Glob', 'A meteor crashes
        near a small American town and unleashes carnivorous
        goo in this classic.', 'NR', 'SCIFI',
        TO_DATE('12-AUG-1995','DD-MON-YYYY'))
/
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'My Day Off', 'With a little
        luck and a lot ingenuity, a teenager skips school for
        a day in New York.', 'PG', 'COMEDY',
        TO_DATE('12-JUL-1995','DD-MON-YYYY'))
/
INSERT INTO title(title_id, title, description, rating,
```

```

        category, release_date)
VALUES      (title_id_seq.NEXTVAL, 'Miracles on Ice', 'A six-year-old
has        doubts about Santa Claus, but she discovers that miracles
really do exist.', 'PG', 'DRAMA',
        TO_DATE('12-SEP-1995','DD-MON-YYYY'))
/

```

```

INSERT INTO title(title_id, title, description, rating,
        category, release_date)
VALUES      (title_id_seq.NEXTVAL, 'Soda Gang', 'After discovering a
cache of drugs, a young couple find themselves pitted against a
vicious gang.', 'NR', 'ACTION', TO_DATE('01-JUN-1995','DD-MON-
YYYY'))
/
COMMIT
/
SELECT  title
FROM    title;

```

- b. Add data to the MEMBER table. Place the insert statements in a script named lab\_apcs\_4b.sql. Execute the commands in the script. Be sure to use the sequence to add the member numbers.

```

SET VERIFY OFF
INSERT INTO member(member_id, first_name, last_name,
        address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Carmen', 'Velasquez',
        '283 King Street', 'Seattle', '206-899-6666', TO_DATE('08-
MAR-1990',
        'DD-MM-YYYY'))
/

INSERT INTO member(member_id, first_name, last_name,
        address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'LaDoris', 'Ngao',
        '5 Modrany', 'Bratislava', '586-355-8882', TO_DATE('08-MAR-
1990',
        'DD-MM-YYYY'))
/

INSERT INTO member(member_id, first_name, last_name,
        address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Midori', 'Nagayama',

```

```

        '68 Via Centrale', 'Sao Paolo', '254-852-5764',
TO_DATE('17-JUN-1991',
        'DD-MM-YYYY'))
/

INSERT INTO member(member_id, first_name, last_name,
        address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Mark', 'Quick-to-See',
        '6921 King Way', 'Lagos', '63-559-7777', TO_DATE('07-APR-
1990',
        'DD-MM-YYYY'))
/

INSERT INTO member(member_id, first_name, last_name,
        address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Audry', 'Ropeburn',
        '86 Chu Street', 'Hong Kong', '41-559-87', TO_DATE('18-JAN-
1991',
        'DD-MM-YYYY'))
/

INSERT INTO member(member_id, first_name, last_name,
        address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Molly', 'Urguhart',
        '3035 Laurier', 'Quebec', '418-542-9988', TO_DATE('18-JAN-
1991',
        'DD-MM-YYYY'));
/

COMMIT
SET VERIFY ON

```

c. Add the following movie copies in the TITLE\_COPY table:

**Note:** Have the TITLE\_ID numbers available for this exercise.

```

INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 92, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 93, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (2, 93, 'RENTED')

```



```

/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 94, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 95, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (2, 95, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (3, 95, 'RENTED')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 96, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 97, 'AVAILABLE')
/

```

- d. Add the following rentals to the RENTAL table:

**Note:** The title number may be different depending on the sequence number.

```

INSERT INTO rental(title_id, copy_id, member_id,
                  book_date, exp_ret_date, act_ret_date)
VALUES (92, 1, 101, sysdate-3, sysdate-1, sysdate-2)
/
INSERT INTO rental(title_id, copy_id, member_id,
                  book_date, exp_ret_date, act_ret_date)
VALUES (93, 2, 101, sysdate-1, sysdate-1, NULL)
/
INSERT INTO rental(title_id, copy_id, member_id,
                  book_date, exp_ret_date, act_ret_date)
VALUES (95, 3, 102, sysdate-2, sysdate, NULL)
/
INSERT INTO rental(title_id, copy_id, member_id,
                  book_date, exp_ret_date, act_ret_date)
VALUES (97, 1, 106, sysdate-4, sysdate-2, sysdate-2)
/
COMMIT
/

```

5. Create a view named TITLE\_AVAIL to show the movie titles, the availability of each copy and its expected return date if rented. Query all rows from the view. Order the results by title.

**Note:** Your results may be different.

```

CREATE VIEW title_avail AS
  SELECT  t.title, c.copy_id, c.status, r.exp_ret_date
  FROM    title t JOIN title_copy c
  ON      t.title_id = c.title_id
  FULL OUTER JOIN rental r
  ON      c.copy_id = r.copy_id
  AND     c.title_id = r.title_id;

SELECT  *
FROM    title_avail
ORDER BY title, copy_id;

```

6. Make changes to data in the tables.

- a. Add a new title. The movie is “Interstellar Wars,” which is rated PG and classified as a science fiction movie. The release date is 07-JUL-77. The description is “Futuristic interstellar action movie. Can the rebels save the humans from the evil empire?” Be sure to add a title copy record for two copies.

```

INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'Interstellar Wars',
        'Futuristic interstellar action movie. Can the
        rebels save the humans from the evil empire?',
        'PG', 'SCIFI', '07-JUL-77')
/
INSERT INTO title_copy (copy_id, title_id, status)
VALUES (1, 98, 'AVAILABLE')
/
INSERT INTO title_copy (copy_id, title_id, status)
VALUES (2, 98, 'AVAILABLE')
/

```

- b. Enter two reservations. One reservation is for Carmen Velasquez, who wants to rent “Interstellar Wars.” The other is for Mark Quick-to-See, who wants to rent “Soda Gang.”

```

INSERT INTO reservation (res_date, member_id, title_id)
VALUES (SYSDATE, 101, 98)
/
INSERT INTO reservation (res_date, member_id, title_id)
VALUES (SYSDATE, 104, 97)
/

```

7. Make a modification to one of the tables.

- a. Run the lab\_apcs\_7a.sql script located in the /home/oracle/labs/sql1/labs folder, to add a PRICE column to the TITLE table to record the purchase price of the video. Verify your modifications.

```

ALTER TABLE title

```

```
ADD (price NUMBER(8,2));
```

```
DESCRIBE title
```

- b. Create a script named `lab_apcs_7b.sql` that contains update statements that update each video with a price according to the list provided. Run the commands in the script.

**Note:** Have the `TITLE_ID` numbers available for this exercise.

```
SET ECHO OFF
```

```
SET VERIFY OFF
```

```
UPDATE title
```

```
SET price = &price
```

```
WHERE title_id = &title_id;
```

```
SET VERIFY OFF
```

```
SET ECHO OFF
```

8. Create a report that contains each customer's history of renting videos. Be sure to include the customer name, movie rented, dates of the rental, and duration of rentals. Total the number of rentals for all customers for the reporting period. Save the commands that generate the report in a script file named `lab_apcs_8.sql`.

**Note:** Your results may be different.

```
SELECT m.first_name||' '|m.last_name MEMBER, t.title,  
       r.book_date, r.act_ret_date - r.book_date DURATION
```

```
FROM member m
```

```
JOIN rental r
```

```
ON r.member_id = m.member_id
```

```
JOIN title t
```

```
ON r.title_id = t.title_id
```

```
ORDER BY member;
```

Oracle Internal & Oracle Academy  
Use Only