ORACLE
UNIVERSITY

Integrated Cloud Applications & Platform Services

Oracle Database 12*c*: Introduction to SQL - Cloud Edition (WDP only)

Activity Guide

D99738GC20

Edition 2.0 | March 2017 | D99777

Learn more from Oracle University at **education.oracle.com**

ORACLE

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.

Author

Apoorva Srinivas

Technical Contributors and Reviewers

Nancy Greenberg, Suresh Rajan, Peckkwai Yap, Bryan Roberts, Sharath Bhujani

This book was published using: Oracle Tutor

Table of Contents

Practices for Lesson 1: Introduction

Chapter 1

Practices for Lesson 1: Overview

Practice Overview

In this practice, you start SQL Developer, create a new database connection, and browse your HR tables. You also set some SQL Developer preferences.

In some of the practices, 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 an additional 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

- 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.
- For any query, the sequence of rows retrieved from the database may differ from the screenshots shown.

Practice 1-1: Introduction

Overview

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. The practices are intended to cover most of the topics that are presented in the corresponding lesson.

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 the data objects in the `ora1` account. The `ora1` account contains the `HR` schema tables.

Note the following location for the practice files:

For labs 1 - 11 : `/home/oracle/labs/sql1/labs`

For labs 12 - 21: `/home/oracle/labs/sql2/labs`

If you are asked to save any practice files, save them in the preceding location.

Tasks

1. Start Oracle SQL Developer by using the SQL Developer Desktop icon.
2. Create a New Oracle SQL Developer Database Connection
   a. To create a new database connection, in the Connections Navigator, right-click Connections and select New Connection from the context menu. The New/Select Database Connection dialog box appears.
   b. Create a database connection by using the following information:

      Connection Name: `myconnection`

      Username: `ora1`

      Password: `ora1`

      Hostname: `localhost`

      Port: `1521`

      SID: `ORCL`

   Ensure that you select the Save Password check box.
3. Testing the Oracle SQL Developer Database Connection and Connecting to the Database
   a. Test the new connection.
   b. If the status is Success, connect to the database by using this new connection.

4. Browsing the Tables in the Connections Navigator
   a. In the Connections Navigator, view the objects that are available to you in the Tables node. Verify that the following tables are present:

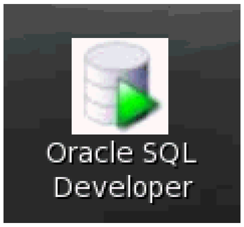
      ```
      COUNTRIES
      DEPARTMENTS
      EMPLOYEES
      JOB_GRADES
      JOB_HISTORY
      JOBS
      LOCATIONS
      REGIONS
      ```

   b. Browse the structure of the `EMPLOYEES` table.
   c. View the data of the `DEPARTMENTS` table.

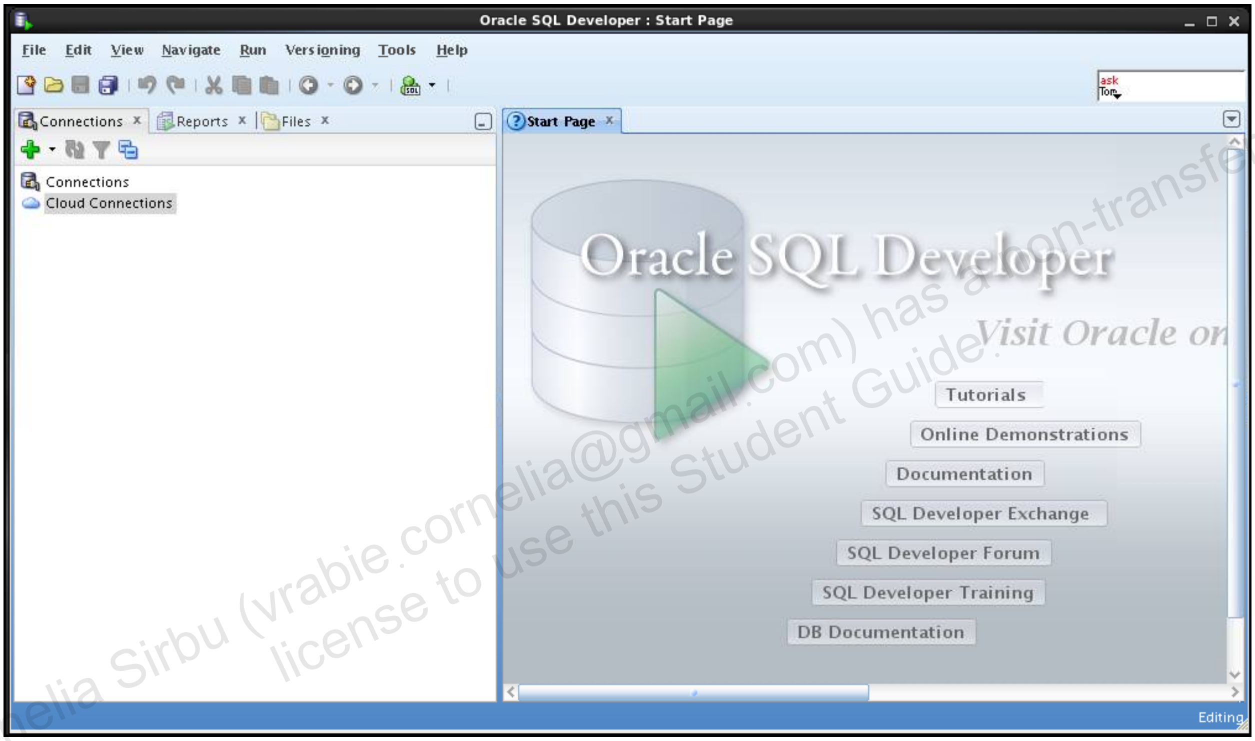
Solution 1-1: Introduction

1. Starting Oracle SQL Developer Using the SQL Developer Desktop Icon

   Double-click the Oracle SQL Developer desktop icon.

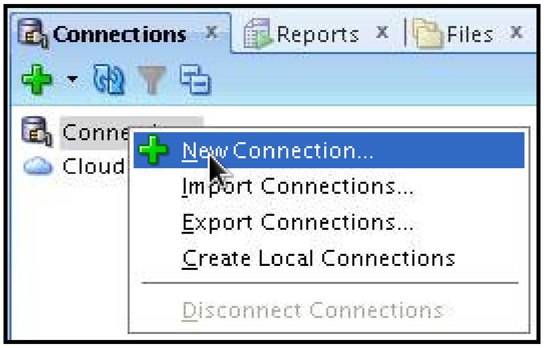


The SQL Developer Interface appears.

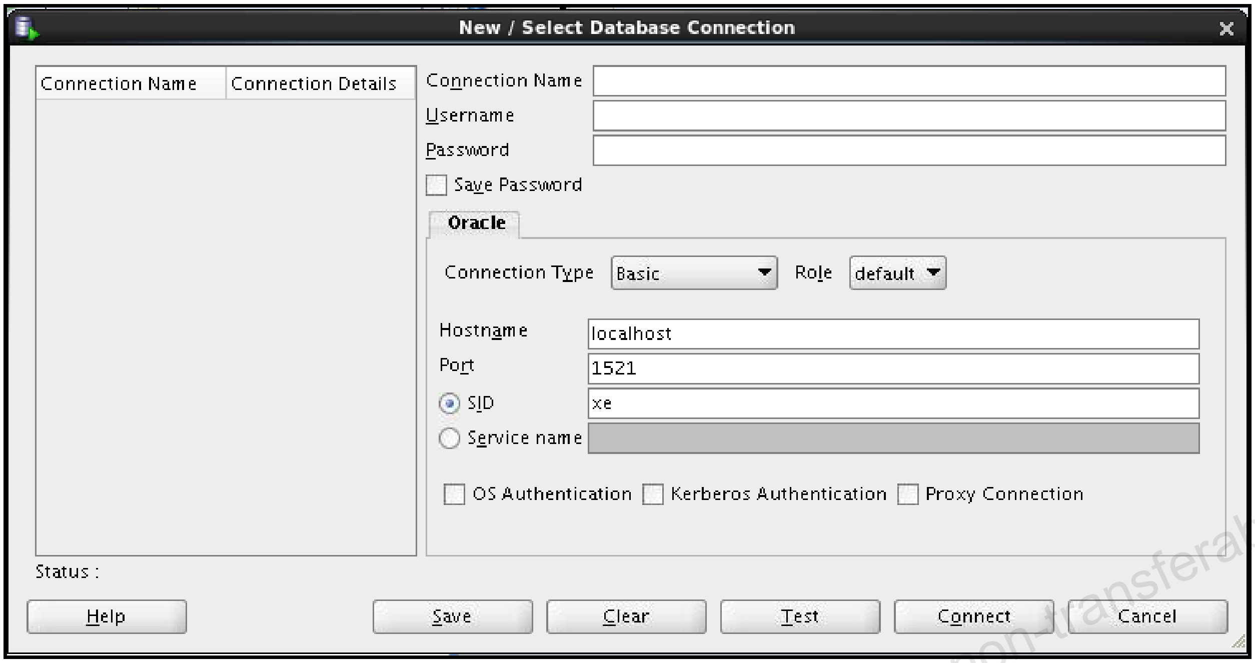


2. Creating a New Oracle SQL Developer Database Connection

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



The New / Select Database Connection dialog box appears.



New / Select Database Connection

Connection Name	Connection Details

Connection Name

Username

Password

□ Save Password

Oracle

Connection Type: Basic   Role: default

Hostname: localhost

Port: 1521

● SID: xe

○ Service name

□ OS Authentication □ Kerberos Authentication □ Proxy Connection

Status :

b. Create a database connection by using the following information:

i. Connection Name: `myconnection`

ii. Username: `ora1`

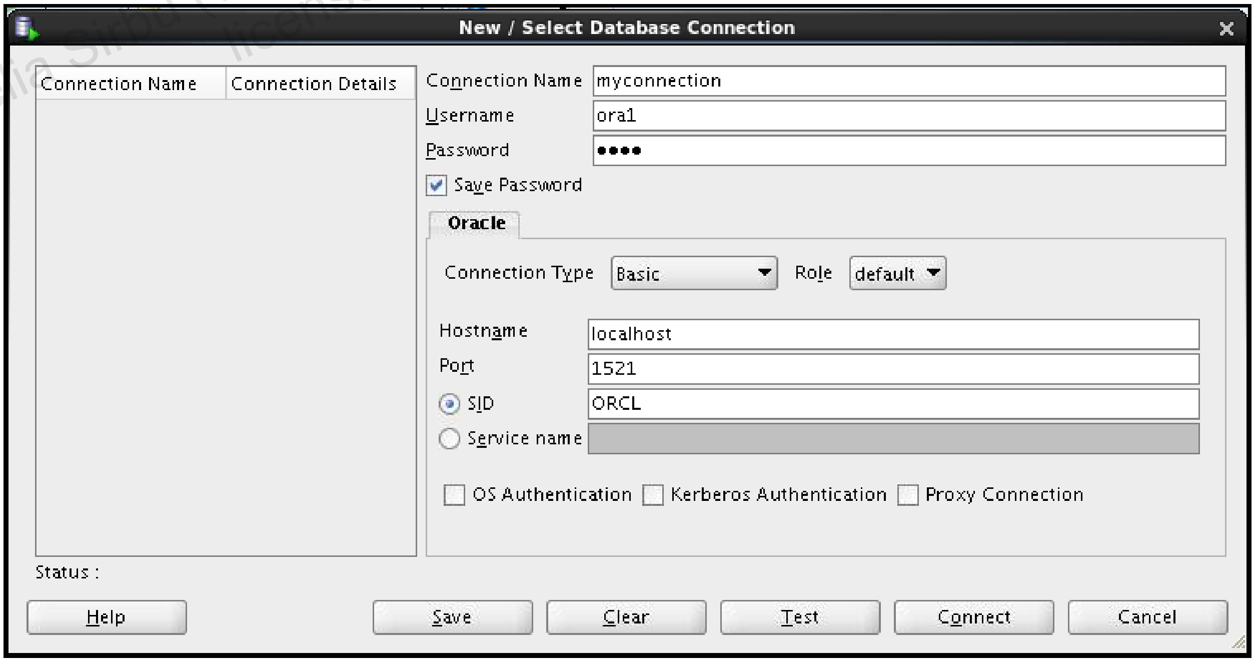
iii. Password: `ora1`

iv. Hostname: `localhost`

v. Port: `1521`

vi. SID: `ORCL`

Ensure that you select the Save Password check box.



New / Select Database Connection

Connection Name	Connection Details

Connection Name: myconnection

Username: ora1

Password: ••••

☑ Save Password

Oracle

Connection Type: Basic   Role: default

Hostname: localhost

Port: 1521

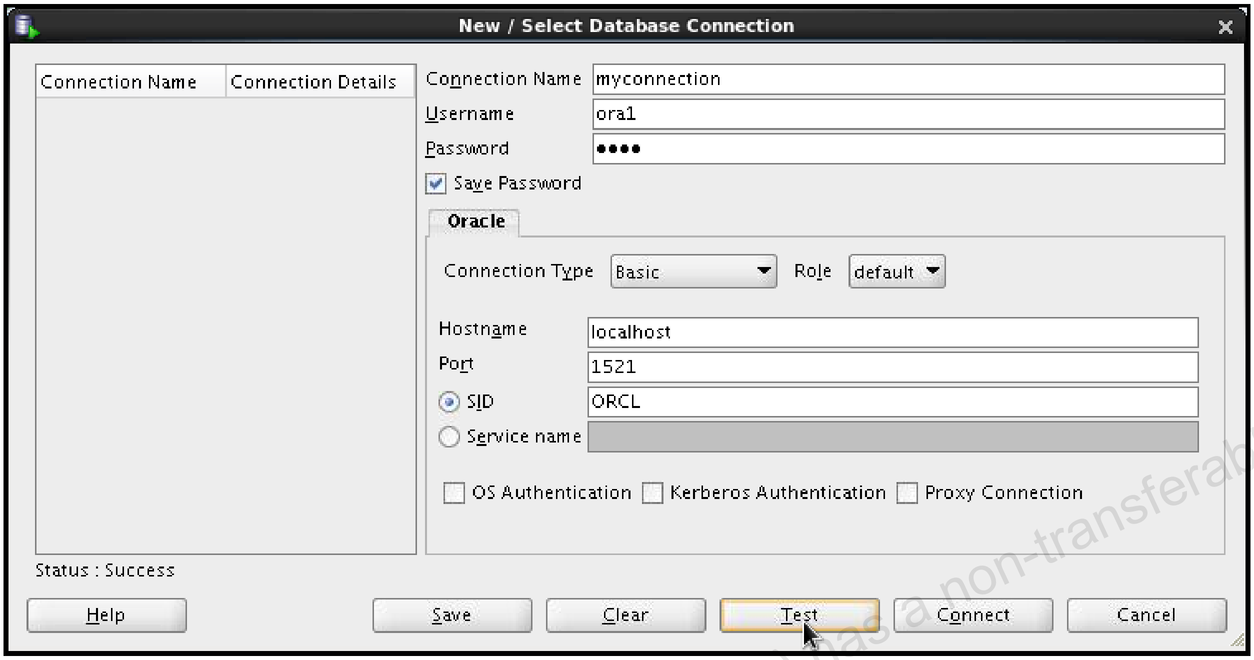
● SID: ORCL

○ Service name

□ OS Authentication □ Kerberos Authentication □ Proxy Connection

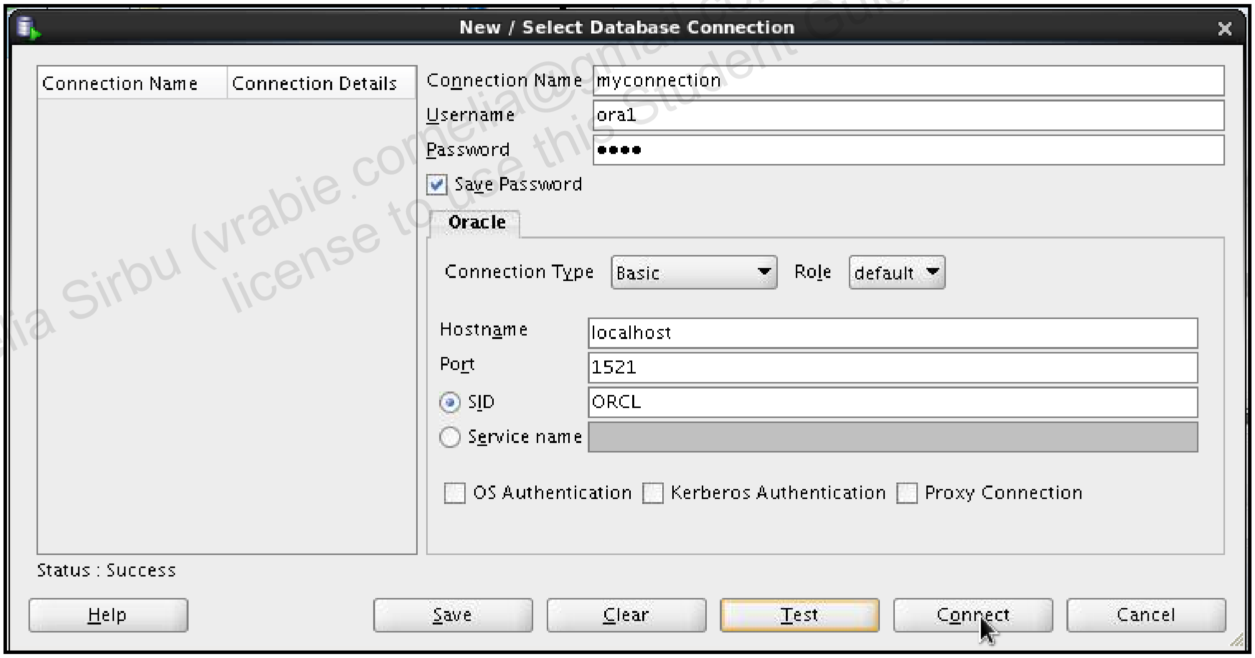
Status :

3. Testing and Connecting Using the Oracle SQL Developer Database Connection
   a. Test the new connection.



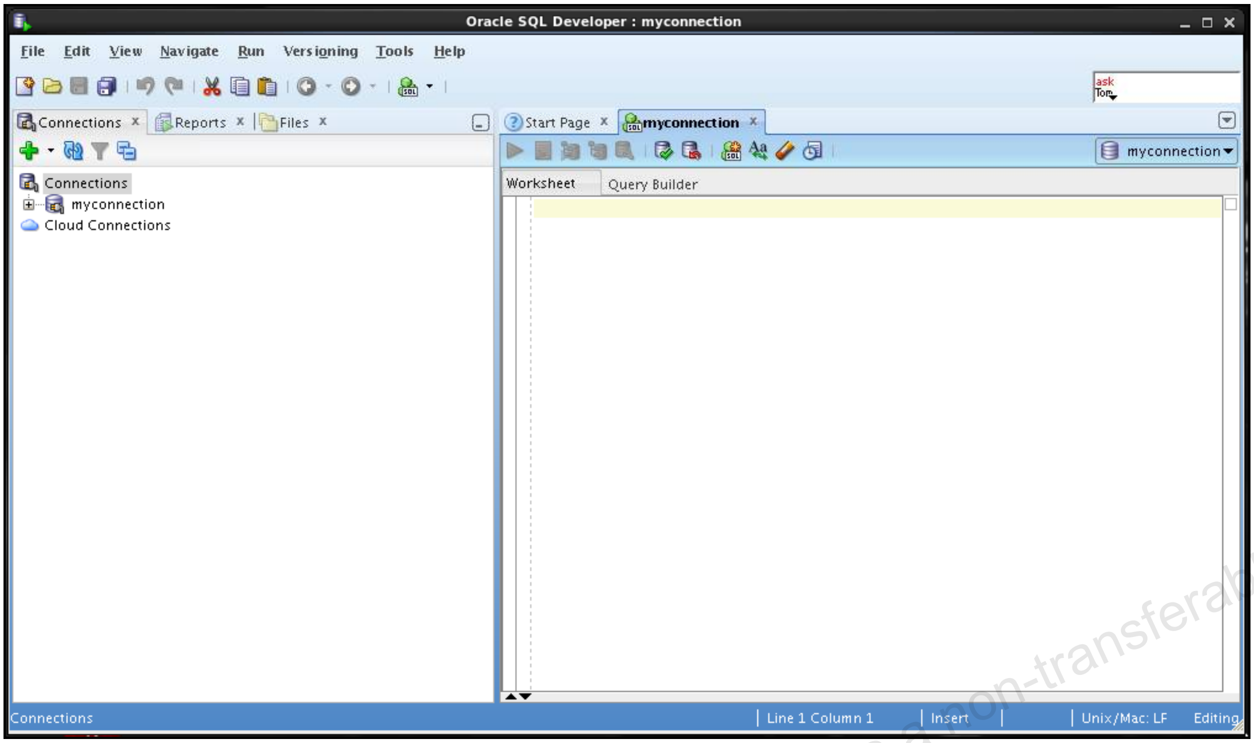
Connection Name: myconnection
Username: ora1
Password: ●●●●
☑ Save Password
Connection Type: Basic
Role: default
Hostname: localhost
Port: 1521
● SID: ORCL
○ Service name
□ OS Authentication □ Kerberos Authentication □ Proxy Connection
Status : Success

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



Connection Name: myconnection
Username: ora1
Password: ●●●●
☑ Save Password
Connection Type: Basic
Role: default
Hostname: localhost
Port: 1521
● SID: ORCL
○ Service name
□ OS Authentication □ Kerberos Authentication □ Proxy Connection
Status : Success

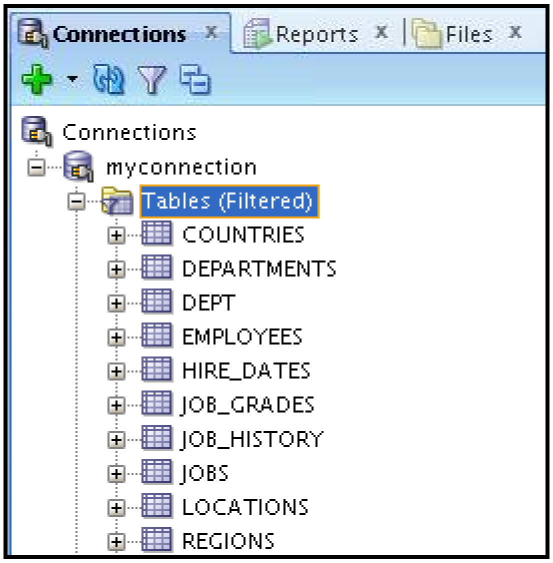
When you create a connection, a SQL Worksheet for that connection opens automatically.


Oracle SQL Developer : myconnection
File Edit View Navigate Run Versioning Tools Help
Connections Reports Files
Connections
myconnection
Cloud Connections
Start Page myconnection
myconnection
Worksheet Query Builder
Connections Line 1 Column 1 Insert Unix/Mac: LF Editing

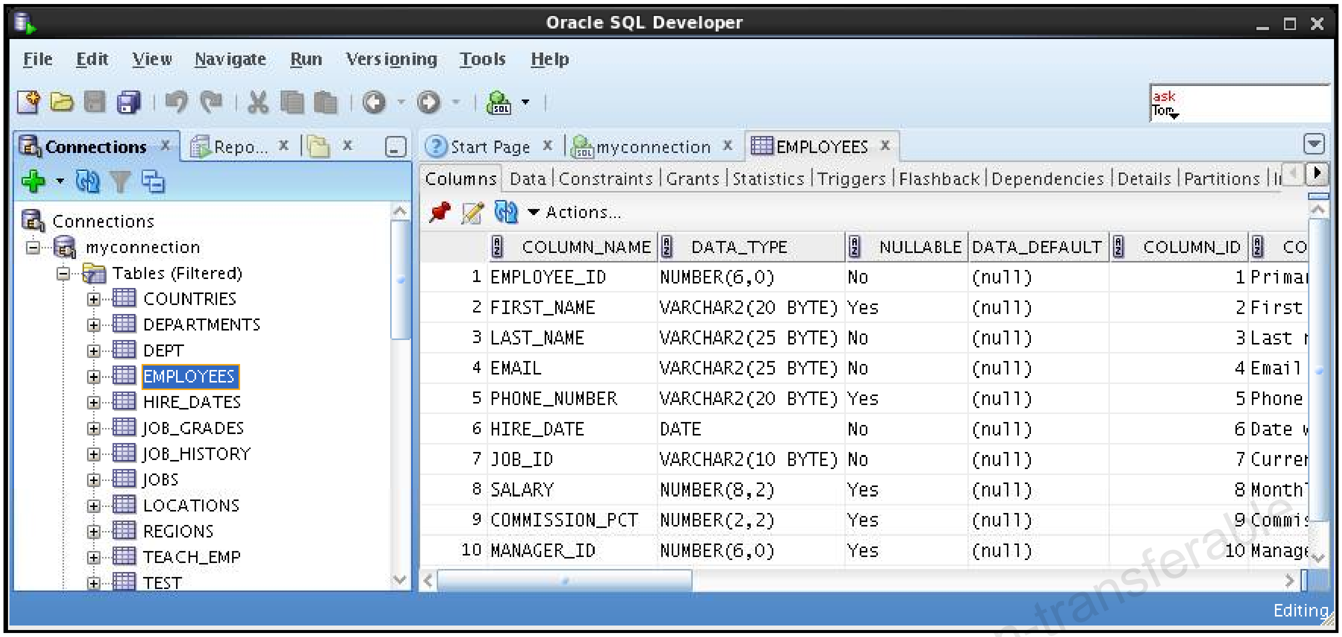
4. Browsing the Tables in the Connections Navigator

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

```
COUNTRIES
DEPARTMENTS
EMPLOYEES
JOB_GRADES
JOB_HISTORY
JOBS
LOCATIONS
REGIONS
```

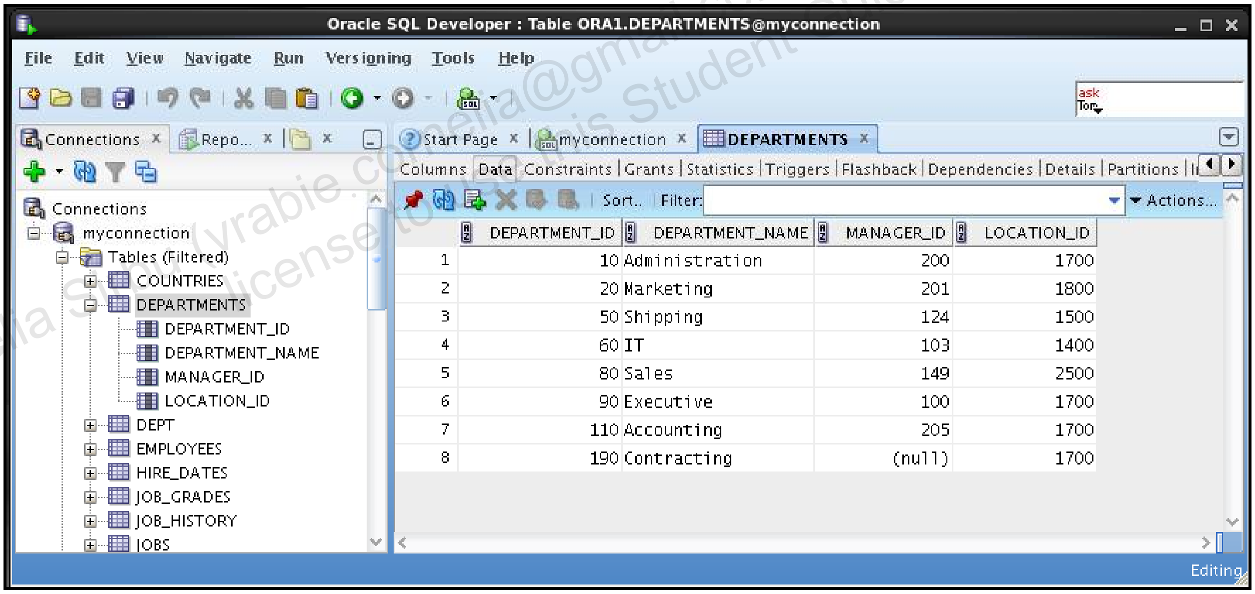


b. Browse the structure of the `EMPLOYEES` table.



	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	CO
1	EMPLOYEE_ID	NUMBER(6,0)	No	(null)	1	Primar
2	FIRST_NAME	VARCHAR2(20 BYTE)	Yes	(null)	2	First
3	LAST_NAME	VARCHAR2(25 BYTE)	No	(null)	3	Last r
4	EMAIL	VARCHAR2(25 BYTE)	No	(null)	4	Email
5	PHONE_NUMBER	VARCHAR2(20 BYTE)	Yes	(null)	5	Phone
6	HIRE_DATE	DATE	No	(null)	6	Date w
7	JOB_ID	VARCHAR2(10 BYTE)	No	(null)	7	Curren
8	SALARY	NUMBER(8,2)	Yes	(null)	8	Month
9	COMMISSION_PCT	NUMBER(2,2)	Yes	(null)	9	Commis
10	MANAGER_ID	NUMBER(6,0)	Yes	(null)	10	Manage

c. View the data of the `DEPARTMENTS` table.



	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

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

Chapter 2

Practices for Lesson 2: Overview

Practice Overview

This practice covers the following topics:

- Selecting all data from different tables
- Describing the structure of tables
- Performing arithmetic calculations and specifying column names

Note the following location for the practice files: `/home/oracle/labs/sql1/labs`

Practice 2-1: Retrieving Data Using the SQL `SELECT` Statement

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.

Task 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;
```

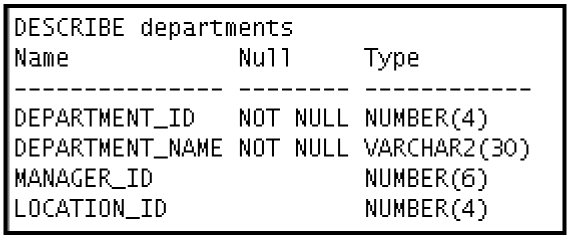
Task 2

Note the following points before you begin with the practices:

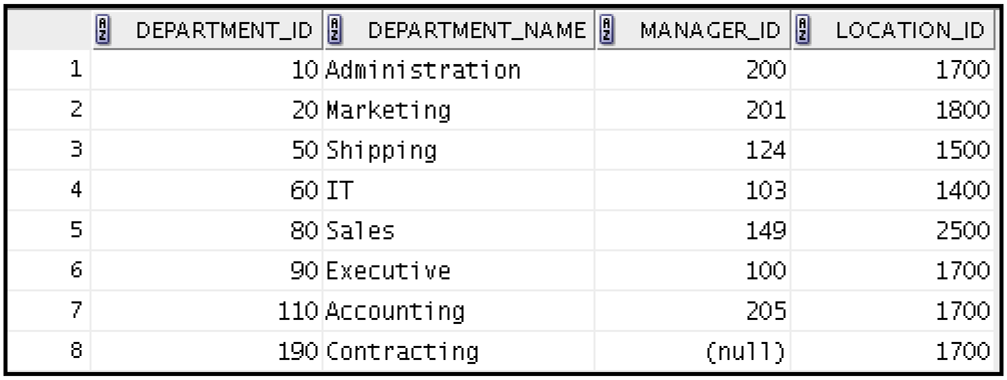
- Save all your practice files at the following location:
  `/home/oracle/labs/sql1/labs`
- Enter your SQL statements in a SQL Worksheet. To save a script in SQL Developer, make sure that the required SQL Worksheet is active, and then from the File menu, select Save As to save your SQL statement as a `lab_<lessonno>_<stepno>.sql` script. When you modify 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 the data from the Human Resources tables.

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



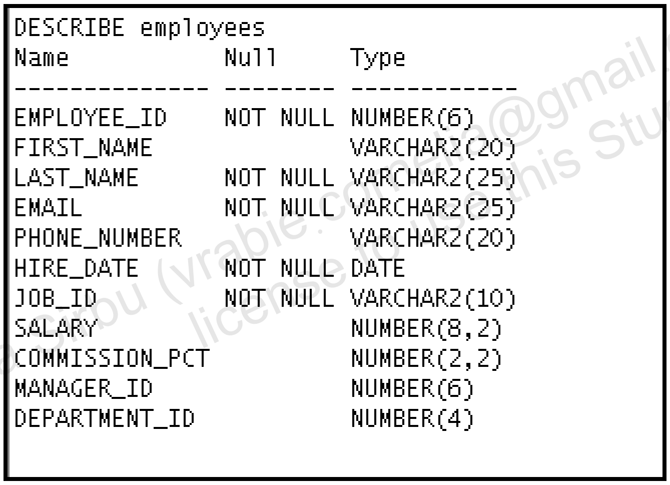
```
DESCRIBE departments
Name            Null     Type
--------------- -------- ------------
DEPARTMENT_ID   NOT NULL NUMBER(4)
DEPARTMENT_NAME NOT NULL VARCHAR2(30)
MANAGER_ID               NUMBER(6)
LOCATION_ID              NUMBER(4)
```



	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. Your task is to determine the structure of the EMPLOYEES table and its contents.

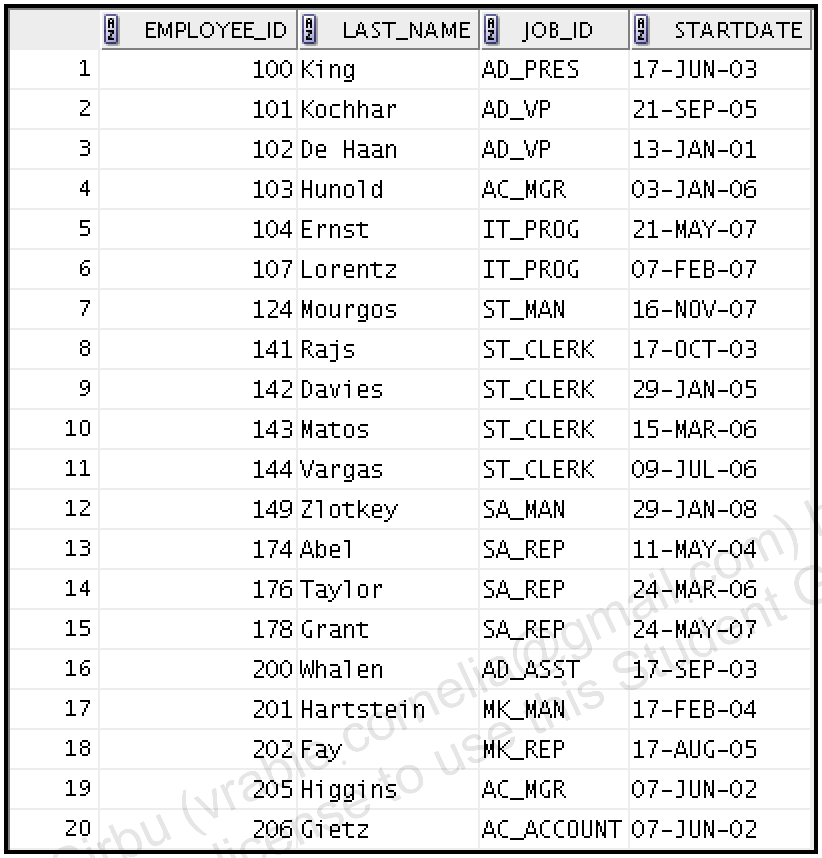
   a. Determine the structure of the EMPLOYEES table.



```
DESCRIBE employees
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)
```

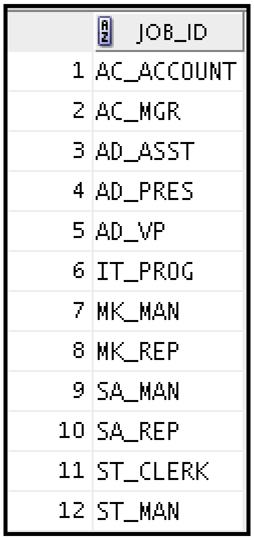
b. The HR department wants a query to display the last name, job ID, hire date, and employee ID for each employee, with the employee ID appearing first. Provide an alias `STARTDATE` for the `HIRE_DATE` column. Save your SQL statement to a file named `lab_02_5b.sql` so that you can dispatch this file to the HR department. Test your query in the `lab_02_5b.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-03
2	101	Kochhar	AD_VP	21-SEP-05
3	102	De Haan	AD_VP	13-JAN-01
4	103	Hunold	AC_MGR	03-JAN-06
5	104	Ernst	IT_PROG	21-MAY-07
6	107	Lorentz	IT_PROG	07-FEB-07
7	124	Mourgos	ST_MAN	16-NOV-07
8	141	Rajs	ST_CLERK	17-OCT-03
9	142	Davies	ST_CLERK	29-JAN-05
10	143	Matos	ST_CLERK	15-MAR-06
11	144	Vargas	ST_CLERK	09-JUL-06
12	149	Zlotkey	SA_MAN	29-JAN-08
13	174	Abel	SA_REP	11-MAY-04
14	176	Taylor	SA_REP	24-MAR-06
15	178	Grant	SA_REP	24-MAY-07
16	200	Whalen	AD_ASST	17-SEP-03
17	201	Hartstein	MK_MAN	17-FEB-04
18	202	Fay	MK_REP	17-AUG-05
19	205	Higgins	AC_MGR	07-JUN-02
20	206	Gietz	AC_ACCOUNT	07-JUN-02

6. The HR department wants a query to display all unique job IDs from the `EMPLOYEES` table.

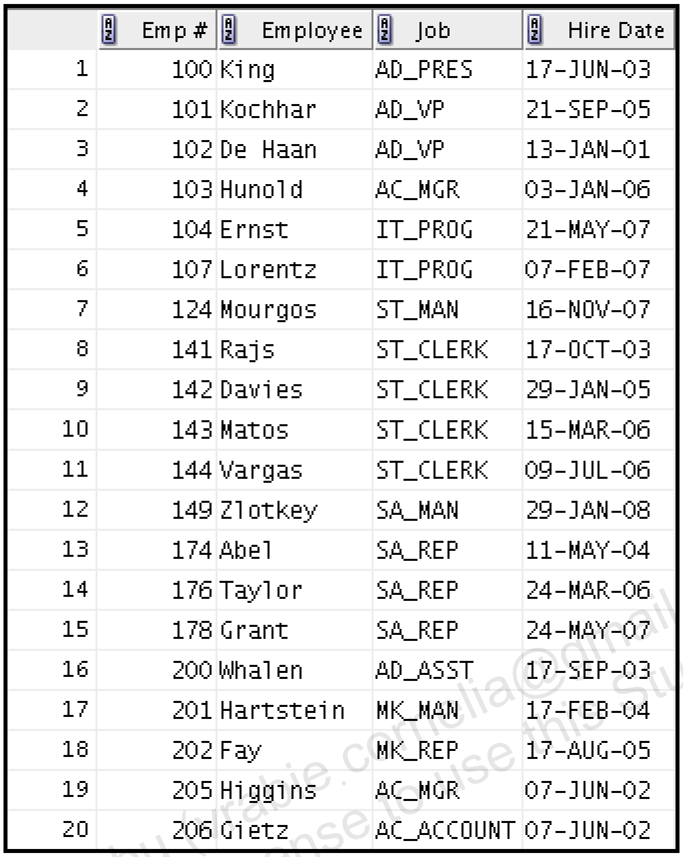


	JOB_ID
1	AC_ACCOUNT
2	AC_MGR
3	AD_ASST
4	AD_PRES
5	AD_VP
6	IT_PROG
7	MK_MAN
8	MK_REP
9	SA_MAN
10	SA_REP
11	ST_CLERK
12	ST_MAN

Task 3

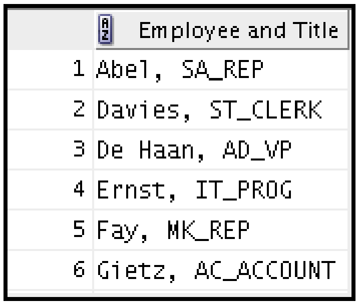
If you have time, complete the following exercises:

7. The HR department wants more descriptive column headings for its report on employees. Copy the statement from `lab_02_5b.sql` to a new SQL Worksheet. Name the columns `Emp #`, `Employee`, `Job`, and `Hire Date`, respectively. Then run the query again.



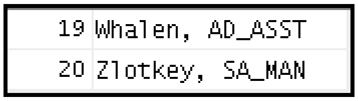
	Emp #	Employee	Job	Hire Date
1	100	King	AD_PRES	17-JUN-03
2	101	Kochhar	AD_VP	21-SEP-05
3	102	De Haan	AD_VP	13-JAN-01
4	103	Hunold	AC_MGR	03-JAN-06
5	104	Ernst	IT_PROG	21-MAY-07
6	107	Lorentz	IT_PROG	07-FEB-07
7	124	Mourgos	ST_MAN	16-NOV-07
8	141	Rajs	ST_CLERK	17-OCT-03
9	142	Davies	ST_CLERK	29-JAN-05
10	143	Matos	ST_CLERK	15-MAR-06
11	144	Vargas	ST_CLERK	09-JUL-06
12	149	Zlotkey	SA_MAN	29-JAN-08
13	174	Abel	SA_REP	11-MAY-04
14	176	Taylor	SA_REP	24-MAR-06
15	178	Grant	SA_REP	24-MAY-07
16	200	Whalen	AD_ASST	17-SEP-03
17	201	Hartstein	MK_MAN	17-FEB-04
18	202	Fay	MK_REP	17-AUG-05
19	205	Higgins	AC_MGR	07-JUN-02
20	206	Gietz	AC_ACCOUNT	07-JUN-02

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



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

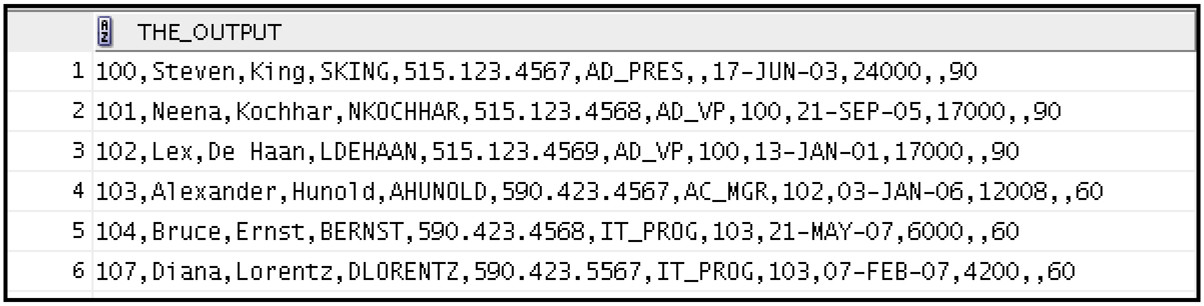
...



19	Whalen, AD_ASST
20	Zlotkey, SA_MAN

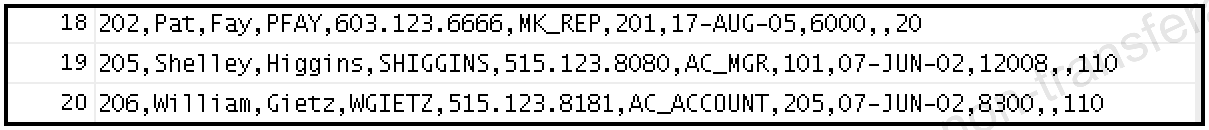
If you want an extra challenge, complete the following exercise:

9. 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 `THE_OUTPUT`.



	THE_OUTPUT
1	100,Steven,King,SKING,515.123.4567,AD_PRES,,17-JUN-03,24000,,90
2	101,Neena,Kochhar,NKOCHHAR,515.123.4568,AD_VP,100,21-SEP-05,17000,,90
3	102,Lex,De Haan,LDEHAAN,515.123.4569,AD_VP,100,13-JAN-01,17000,,90
4	103,Alexander,Hunold,AHUNOLD,590.423.4567,AC_MGR,102,03-JAN-06,12008,,60
5	104,Bruce,Ernst,BERNST,590.423.4568,IT_PROG,103,21-MAY-07,6000,,60
6	107,Diana,Lorentz,DLORENTZ,590.423.5567,IT_PROG,103,07-FEB-07,4200,,60

...



18	202,Pat,Fay,PFAY,603.123.6666,MK_REP,201,17-AUG-05,6000,,20
19	205,Shelley,Higgins,SHIGGINS,515.123.8080,AC_MGR,101,07-JUN-02,12008,,110
20	206,William,Gietz,WGIETZ,515.123.8181,AC_ACCOUNT,205,07-JUN-02,8300,,110

Solution 2-1: Retrieving Data Using the SQL SELECT Statement

Task 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.**

Task 2

You have been hired as a SQL programmer for Acme Corporation. Your first task is to create some reports based on the 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. Your task is to determine the structure of the EMPLOYEES table and its contents.

   a. Determine the structure of the EMPLOYEES table.

```
DESCRIBE employees
```

   b. 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_5b.sql so that you can dispatch this file to the HR department. Test your query in the lab_02_5b.sql file to ensure that it runs correctly.

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

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

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

Task 3

If you have time, complete the following exercises:

7. The HR department wants more descriptive column headings for its report on employees. Copy the statement from lab_02_5b.sql to a new SQL Worksheet. Name the columns 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;
```

8. 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:

9. 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 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

Practices for Lesson 3: Overview

Practices Overview

This practice covers the following topics:

- Selecting data and changing the order of the rows that are displayed
- Restricting rows by using the `WHERE` clause
- Sorting rows by using the `ORDER BY` clause
- Using substitution variables to add flexibility to your SQL `SELECT` statements

Note the following location for the practice files: `/home/oracle/labs/sql1/labs`

Practice 3-1: Restricting and Sorting Data

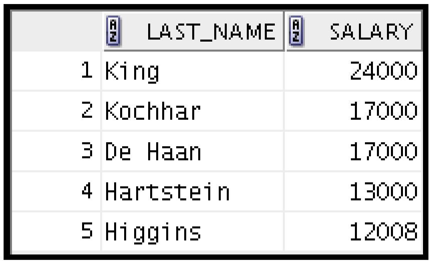
Overview

In this practice, you build more reports by using 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.

Task

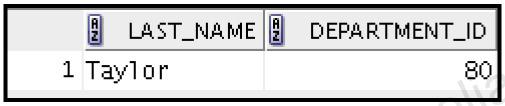
The HR department needs your assistance in creating some queries.

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.



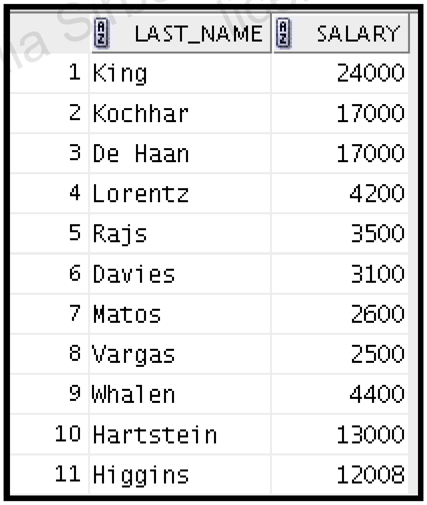
	LAST_NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000
4	Hartstein	13000
5	Higgins	12008

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



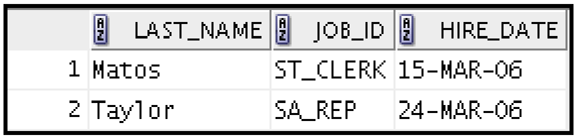
	LAST_NAME	DEPARTMENT_ID
1	Taylor	80

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 $5,000 through $12,000. Save your SQL statement as `lab_03_03.sql`.



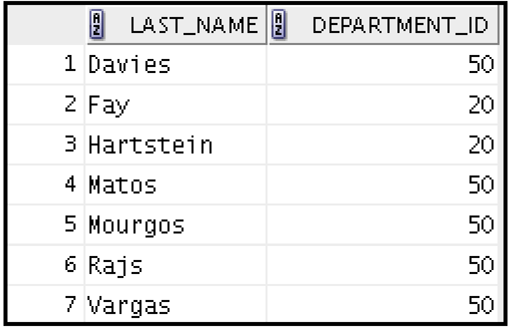
	LAST_NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000
4	Lorentz	4200
5	Rajs	3500
6	Davies	3100
7	Matos	2600
8	Vargas	2500
9	Whalen	4400
10	Hartstein	13000
11	Higgins	12008

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.



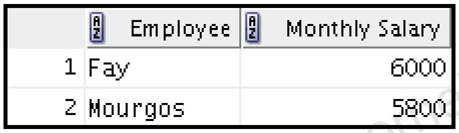
	LAST_NAME	JOB_ID	HIRE_DATE
1	Matos	ST_CLERK	15-MAR-06
2	Taylor	SA_REP	24-MAR-06

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



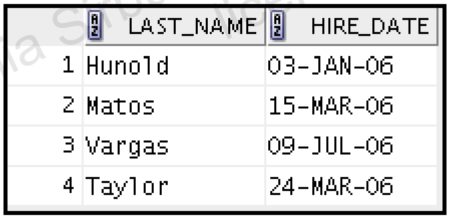
	LAST_NAME	DEPARTMENT_ID
1	Davies	50
2	Fay	20
3	Hartstein	20
4	Matos	50
5	Mourgos	50
6	Rajs	50
7	Vargas	50

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`.



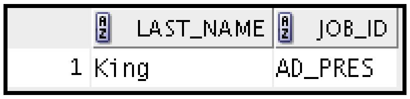
	Employee	Monthly Salary
1	Fay	6000
2	Mourgos	5800

7. The HR department needs a report that displays the last name and hire date of all employees who were hired in 2006.



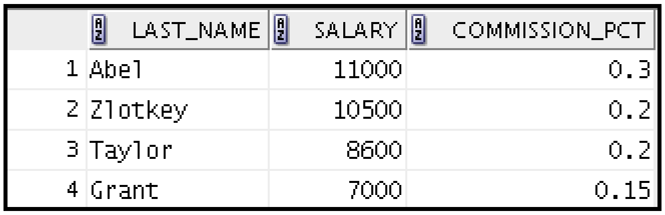
	LAST_NAME	HIRE_DATE
1	Hunold	03-JAN-06
2	Matos	15-MAR-06
3	Vargas	09-JUL-06
4	Taylor	24-MAR-06

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



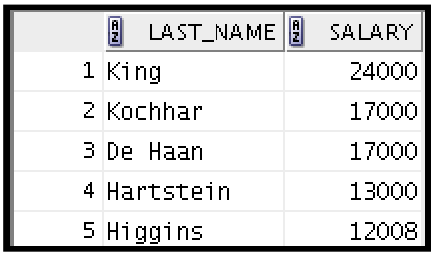
	LAST_NAME	JOB_ID
1	King	AD_PRES

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

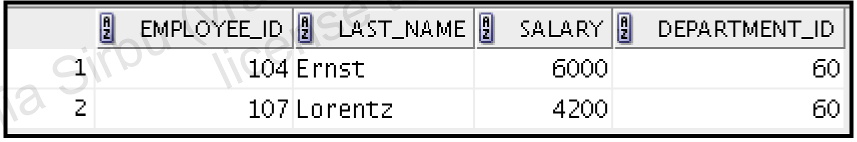
10. Members of the HR department want to have more flexibility with the queries that you are writing. They would like a report that displays the last name and salary of employees who earn more than an amount that the user specifies after a prompt. Save this query to a file named `lab_03_10.sql`. (You can use the query created in Task 1 and modify it.) If you enter 12000 when prompted, the report displays the following results:



	LAST_NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000
4	Hartstein	13000
5	Higgins	12008

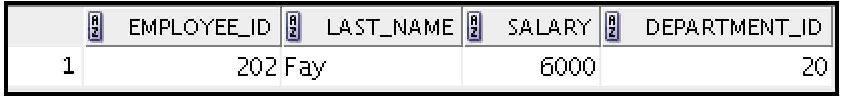
11. The HR department wants to run reports based on a manager. Create a query that prompts the user for a manager ID, and generates the employee ID, last name, salary, and department for that manager's employees. The HR department wants the ability to sort the report on a selected column. You can test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

    manager_id = 103, sorted by last_name:



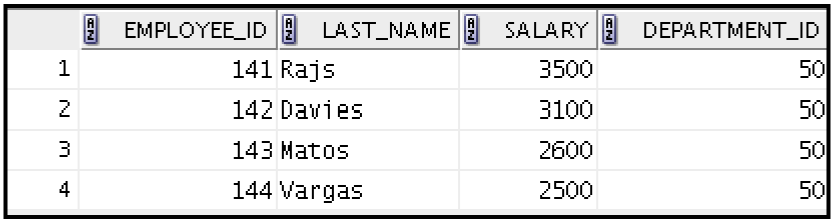
	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	104	Ernst	6000	60
2	107	Lorentz	4200	60

    manager_id = 201, sorted by salary:



	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	202	Fay	6000	20

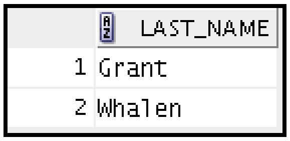
    manager_id = 124, sorted by employee_id:



	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

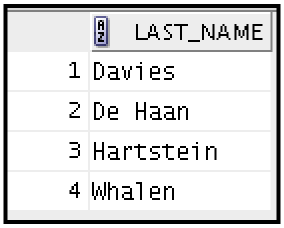
If you have time, complete the following exercises:

12. Display the last names of all employees where the third letter of the name is “a.”



	LAST_NAME
1	Grant
2	Whalen

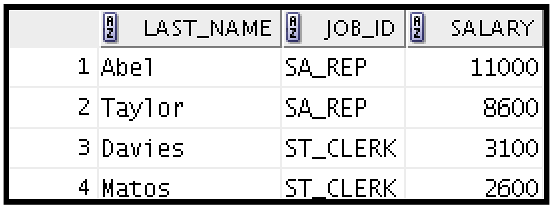
13. Display the last names of all employees who have both an “a” and an “e” in their last name.



	LAST_NAME
1	Davies
2	De Haan
3	Hartstein
4	Whalen

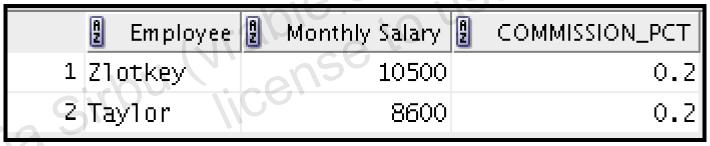
If you want an extra challenge, complete the following exercises:

14. Display the last name, job, and salary for all employees whose jobs are either that of a sales representative or 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

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 `last_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 2006.

```
SELECT   last_name, hire_date
FROM     employees
WHERE    hire_date >= '01-JAN-06' AND hire_date < '01-JAN-07';
```

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.

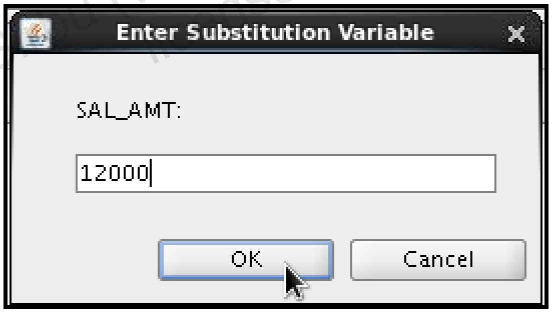
```
SELECT   last_name, salary, commission_pct
FROM     employees
WHERE    commission_pct IS NOT NULL
ORDER BY 2 DESC, 3 DESC;
```

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 Task 1 and modify it.) Save this query to a file named lab_03_10.sql.

Enter 12000 when prompted:

```
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 the last names of all employees where 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;
```

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

Chapter 4

Practices for Lesson 4: Overview

Practice Overview

This practice covers the following topics:

- Writing a query that displays the current date
- Creating queries that require the use of numeric, character, and date functions
- Performing calculations of years and months of service for an employee

Note the following location for the practice files: `/home/oracle/labs/sql1/labs`

Practice 4-1: Using Single-Row Functions to Customize Output

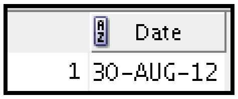
Overview

This practice provides a variety of exercises using the different functions that are available for character, number, and date data types. Remember that for nested functions, the results are evaluated from the innermost function to the outermost function.

Tasks

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

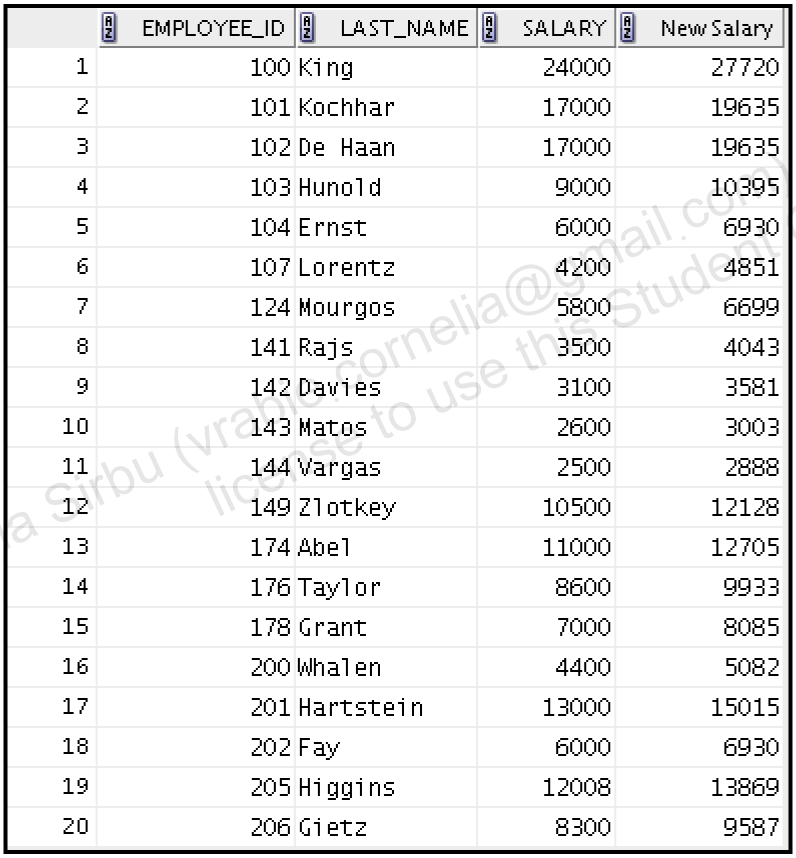
   Note: If your database is remotely located in a different time zone, the output will be the date for the operating system on which the database resides.



	Date
1	30-AUG-12

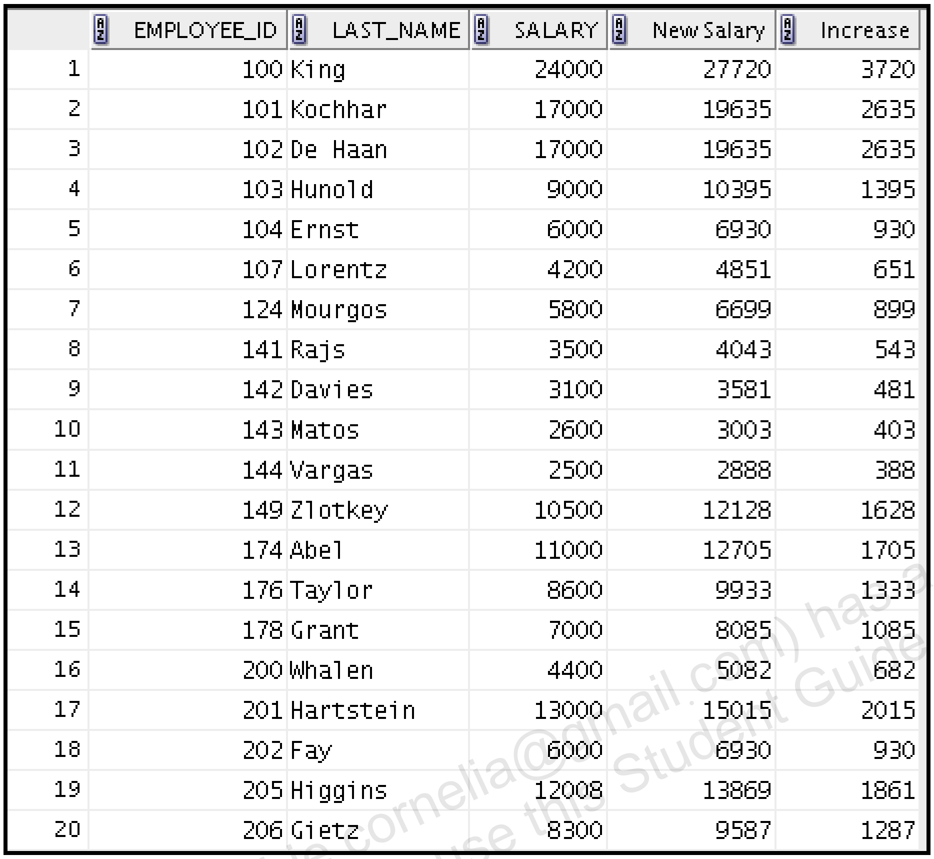
2. The HR department needs a report to display the employee number, last name, salary, and salary increased by 15.5% (expressed as a whole number) for each employee. Label the column `New Salary`. Save your SQL statement in a file named `lab_04_02.sql`.

3. Run your query in the `lab_04_02.sql` file.



	EMPLOYEE_ID	LAST_NAME	SALARY	New Salary
1	100	King	24000	27720
2	101	Kochhar	17000	19635
3	102	De Haan	17000	19635
4	103	Hunold	9000	10395
5	104	Ernst	6000	6930
6	107	Lorentz	4200	4851
7	124	Mourgos	5800	6699
8	141	Rajs	3500	4043
9	142	Davies	3100	3581
10	143	Matos	2600	3003
11	144	Vargas	2500	2888
12	149	Zlotkey	10500	12128
13	174	Abel	11000	12705
14	176	Taylor	8600	9933
15	178	Grant	7000	8085
16	200	Whalen	4400	5082
17	201	Hartstein	13000	15015
18	202	Fay	6000	6930
19	205	Higgins	12008	13869
20	206	Gietz	8300	9587

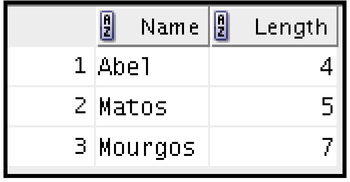
4. Modify your query in `lab_04_02.sql` to add a column that subtracts the old salary from the new salary. Label the column `Increase`. Save the contents of the file as `lab_04_04.sql`. Run the revised query.



	EMPLOYEE_ID	LAST_NAME	SALARY	New Salary	Increase
1	100	King	24000	27720	3720
2	101	Kochhar	17000	19635	2635
3	102	De Haan	17000	19635	2635
4	103	Hunold	9000	10395	1395
5	104	Ernst	6000	6930	930
6	107	Lorentz	4200	4851	651
7	124	Mourgos	5800	6699	899
8	141	Rajs	3500	4043	543
9	142	Davies	3100	3581	481
10	143	Matos	2600	3003	403
11	144	Vargas	2500	2888	388
12	149	Zlotkey	10500	12128	1628
13	174	Abel	11000	12705	1705
14	176	Taylor	8600	9933	1333
15	178	Grant	7000	8085	1085
16	200	Whalen	4400	5082	682
17	201	Hartstein	13000	15015	2015
18	202	Fay	6000	6930	930
19	205	Higgins	12008	13869	1861
20	206	Gietz	8300	9587	1287

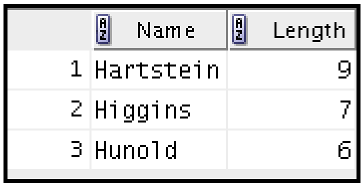
5. Perform the following tasks:

   a. 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.



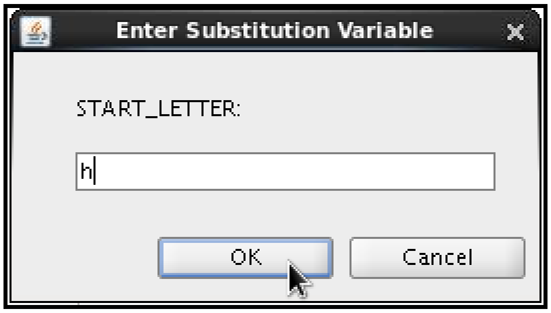
	Name	Length
1	Abel	4
2	Matos	5
3	Mourgos	7

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



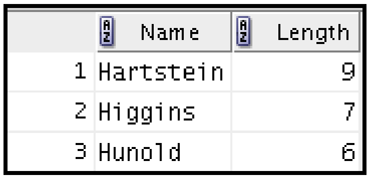
	Name	Length
1	Hartstein	9
2	Higgins	7
3	Hunold	6

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



Enter Substitution Variable

START_LETTER: h

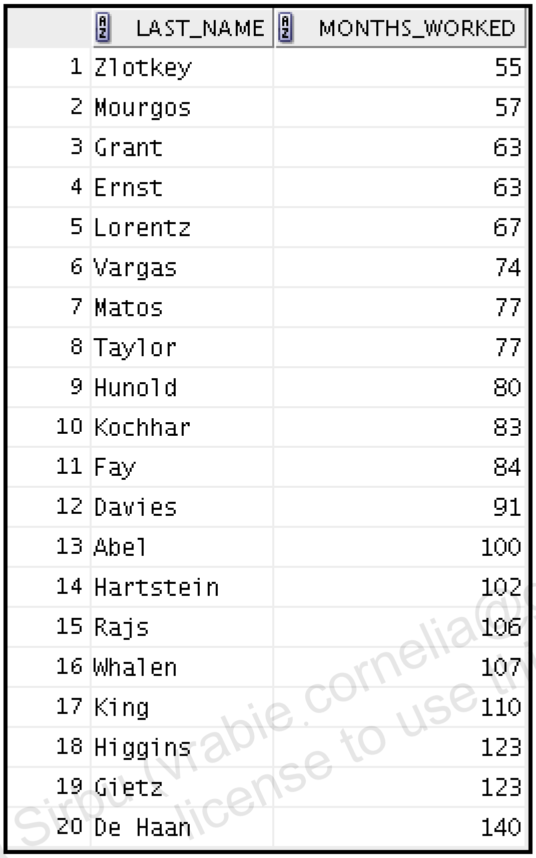


	Name	Length
1	Hartstein	9
2	Higgins	7
3	Hunold	6

If you have time, complete the following exercises:

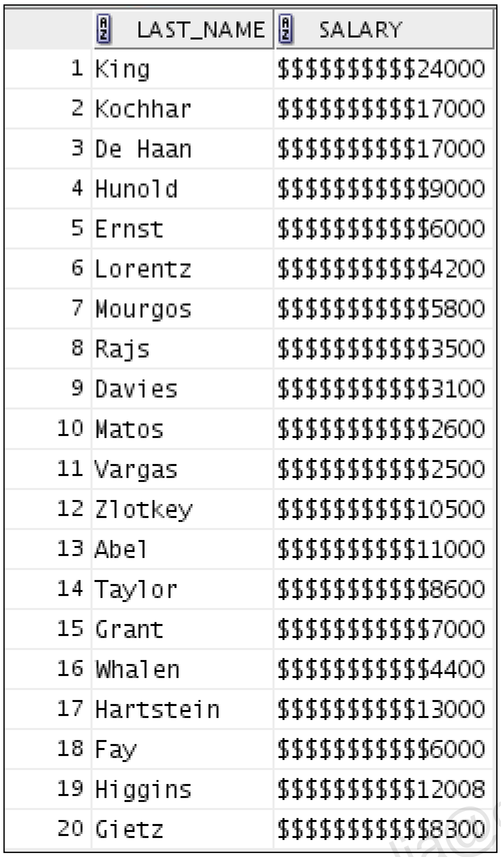
6. The HR department wants to find the duration of employment for each employee. For each employee, display the last name and calculate the number of months between today and the date on which the employee was hired. Label the column as `MONTHS_WORKED`. Order your results by the number of months employed.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



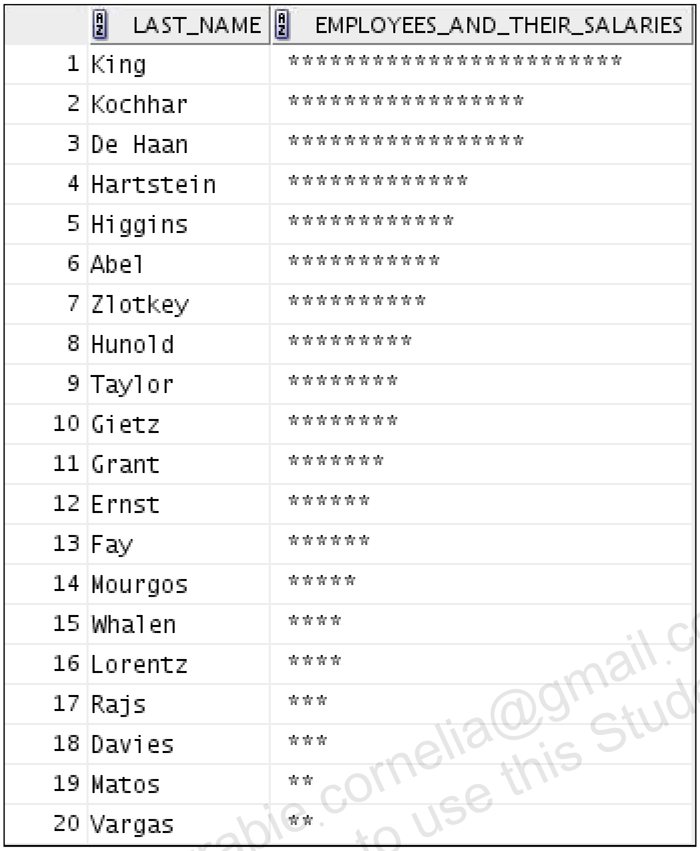
	LAST_NAME	MONTHS_WORKED
1	Zlotkey	55
2	Mourgos	57
3	Grant	63
4	Ernst	63
5	Lorentz	67
6	Vargas	74
7	Matos	77
8	Taylor	77
9	Hunold	80
10	Kochhar	83
11	Fay	84
12	Davies	91
13	Abel	100
14	Hartstein	102
15	Rajs	106
16	Whalen	107
17	King	110
18	Higgins	123
19	Gietz	123
20	De Haan	140

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	King	$$$$$$$$$$24000
2	Kochhar	$$$$$$$$$$17000
3	De Haan	$$$$$$$$$$17000
4	Hunold	$$$$$$$$$$$9000
5	Ernst	$$$$$$$$$$$6000
6	Lorentz	$$$$$$$$$$$4200
7	Mourgos	$$$$$$$$$$$5800
8	Rajs	$$$$$$$$$$$3500
9	Davies	$$$$$$$$$$$3100
10	Matos	$$$$$$$$$$$2600
11	Vargas	$$$$$$$$$$$2500
12	Zlotkey	$$$$$$$$$$10500
13	Abel	$$$$$$$$$$11000
14	Taylor	$$$$$$$$$$$8600
15	Grant	$$$$$$$$$$$7000
16	Whalen	$$$$$$$$$$$4400
17	Hartstein	$$$$$$$$$$13000
18	Fay	$$$$$$$$$$$6000
19	Higgins	$$$$$$$$$$12008
20	Gietz	$$$$$$$$$$$8300

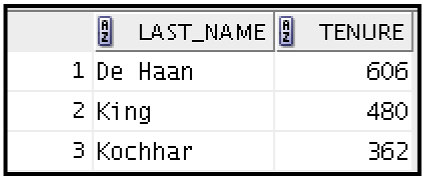
8. Create a query that displays the employees' last names, and indicates the amounts of their salaries with asterisks. Each asterisk signifies a thousand dollars. Sort the data in descending order of salary. Label the column `EMPLOYEES_AND_THEIR_SALARIES`.



	LAST_NAME	EMPLOYEES_AND_THEIR_SALARIES
1	King	************************
2	Kochhar	*****************
3	De Haan	*****************
4	Hartstein	*************
5	Higgins	************
6	Abel	***********
7	Zlotkey	**********
8	Hunold	*********
9	Taylor	********
10	Gietz	********
11	Grant	*******
12	Ernst	******
13	Fay	******
14	Mourgos	*****
15	Whalen	****
16	Lorentz	****
17	Rajs	***
18	Davies	***
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 because it depends on the date on which you run the query.



	LAST_NAME	TENURE
1	De Haan	606
2	King	480
3	Kochhar	362

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 in the `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. Perform the following tasks:
   a. 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;
```

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

c. Modify the query such that the case of the letter that is entered 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;
```

If you have time, complete the following exercises:

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;
```

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 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 last_name,
       rpad(' ', salary/1000, '*')
               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 because it depends on the date when you run the query.

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

Practices for Lesson 5: Using Conversion Functions and Conditional Expressions

Chapter 5

Practices for Lesson 5: Overview

Practice Overview

This practice covers the following topics:

- Creating queries that use the `TO_CHAR` and `TO_DATE` functions.
- Creating queries that use conditional expressions such as `CASE` , `SEARCHED CASE`, and `DECODE`

Note the following location for the practice files: `/home/oracle/labs/sql1/labs`

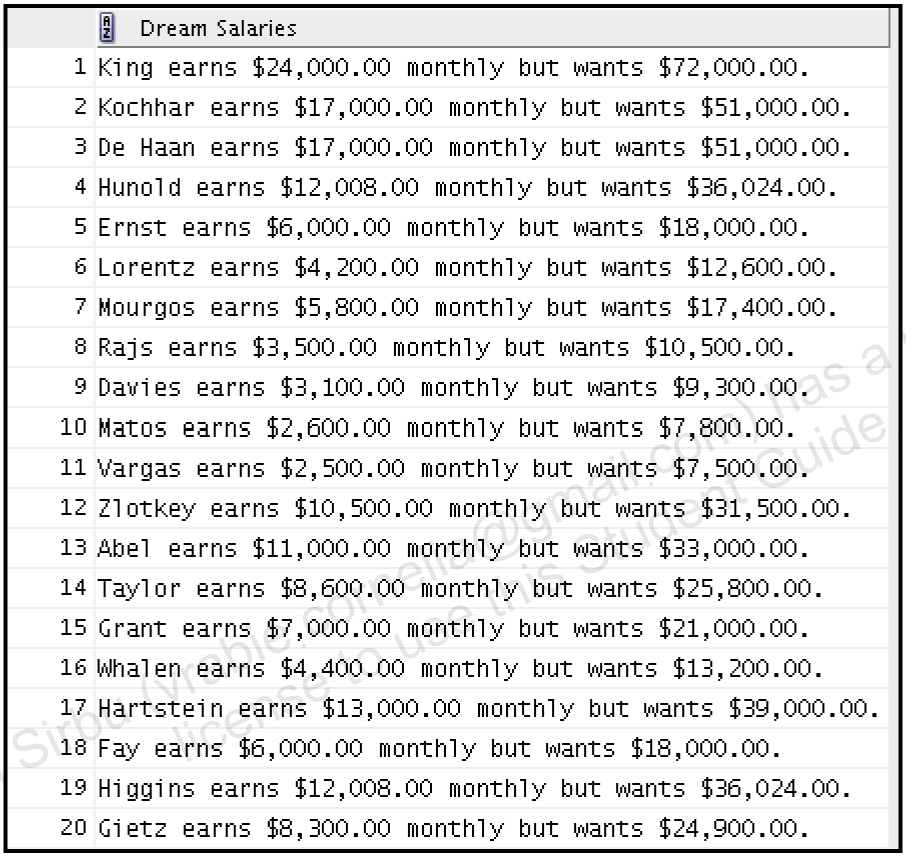
Practice 5-1: Using Conversion Functions and Conditional Expressions

Overview

This practice provides a variety of exercises using the `TO_CHAR` and `TO_DATE` functions, and conditional expressions such as `CASE`, searched `CASE`, and `DECODE`.

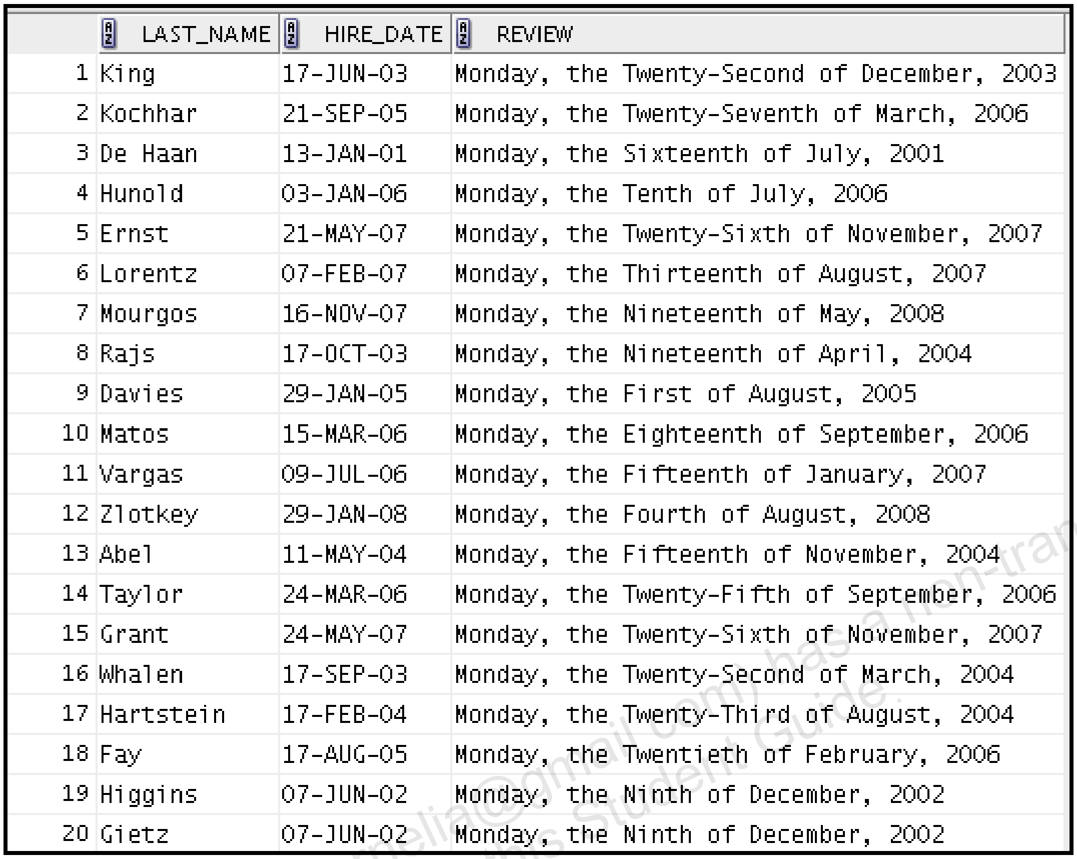
Tasks

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



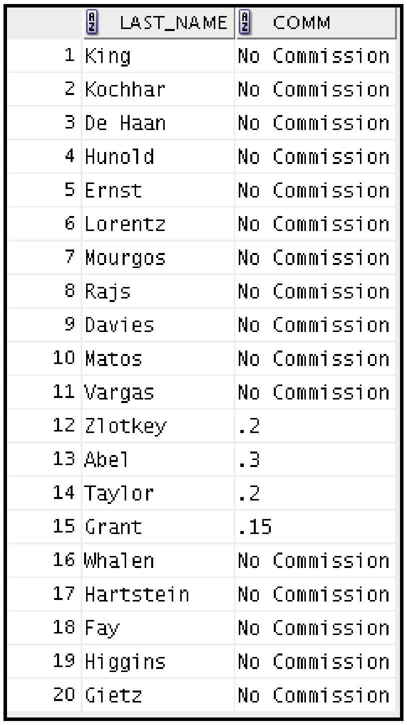
	Dream Salaries
1	King earns $24,000.00 monthly but wants $72,000.00.
2	Kochhar earns $17,000.00 monthly but wants $51,000.00.
3	De Haan earns $17,000.00 monthly but wants $51,000.00.
4	Hunold earns $12,008.00 monthly but wants $36,024.00.
5	Ernst earns $6,000.00 monthly but wants $18,000.00.
6	Lorentz earns $4,200.00 monthly but wants $12,600.00.
7	Mourgos earns $5,800.00 monthly but wants $17,400.00.
8	Rajs earns $3,500.00 monthly but wants $10,500.00.
9	Davies earns $3,100.00 monthly but wants $9,300.00.
10	Matos earns $2,600.00 monthly but wants $7,800.00.
11	Vargas earns $2,500.00 monthly but wants $7,500.00.
12	Zlotkey earns $10,500.00 monthly but wants $31,500.00.
13	Abel earns $11,000.00 monthly but wants $33,000.00.
14	Taylor earns $8,600.00 monthly but wants $25,800.00.
15	Grant earns $7,000.00 monthly but wants $21,000.00.
16	Whalen earns $4,400.00 monthly but wants $13,200.00.
17	Hartstein earns $13,000.00 monthly but wants $39,000.00.
18	Fay earns $6,000.00 monthly but wants $18,000.00.
19	Higgins earns $12,008.00 monthly but wants $36,024.00.
20	Gietz earns $8,300.00 monthly but wants $24,900.00.

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



	LAST_NAME	HIRE_DATE	REVIEW
1	King	17-JUN-03	Monday, the Twenty-Second of December, 2003
2	Kochhar	21-SEP-05	Monday, the Twenty-Seventh of March, 2006
3	De Haan	13-JAN-01	Monday, the Sixteenth of July, 2001
4	Hunold	03-JAN-06	Monday, the Tenth of July, 2006
5	Ernst	21-MAY-07	Monday, the Twenty-Sixth of November, 2007
6	Lorentz	07-FEB-07	Monday, the Thirteenth of August, 2007
7	Mourgos	16-NOV-07	Monday, the Nineteenth of May, 2008
8	Rajs	17-OCT-03	Monday, the Nineteenth of April, 2004
9	Davies	29-JAN-05	Monday, the First of August, 2005
10	Matos	15-MAR-06	Monday, the Eighteenth of September, 2006
11	Vargas	09-JUL-06	Monday, the Fifteenth of January, 2007
12	Zlotkey	29-JAN-08	Monday, the Fourth of August, 2008
13	Abel	11-MAY-04	Monday, the Fifteenth of November, 2004
14	Taylor	24-MAR-06	Monday, the Twenty-Fifth of September, 2006
15	Grant	24-MAY-07	Monday, the Twenty-Sixth of November, 2007
16	Whalen	17-SEP-03	Monday, the Twenty-Second of March, 2004
17	Hartstein	17-FEB-04	Monday, the Twenty-Third of August, 2004
18	Fay	17-AUG-05	Monday, the Twentieth of February, 2006
19	Higgins	07-JUN-02	Monday, the Ninth of December, 2002
20	Gietz	07-JUN-02	Monday, the Ninth of December, 2002

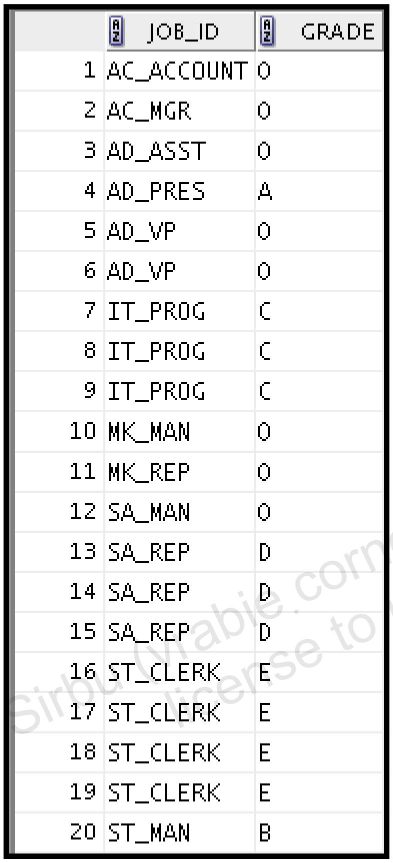
3. Create a query that displays employees’ last names and commission amounts. If an employee does not earn commission, show “No Commission.” Label the column `COMM`.



	LAST_NAME	COMM
1	King	No Commission
2	Kochhar	No Commission
3	De Haan	No Commission
4	Hunold	No Commission
5	Ernst	No Commission
6	Lorentz	No Commission
7	Mourgos	No Commission
8	Rajs	No Commission
9	Davies	No Commission
10	Matos	No Commission
11	Vargas	No Commission
12	Zlotkey	.2
13	Abel	.3
14	Taylor	.2
15	Grant	.15
16	Whalen	No Commission
17	Hartstein	No Commission
18	Fay	No Commission
19	Higgins	No Commission
20	Gietz	No Commission

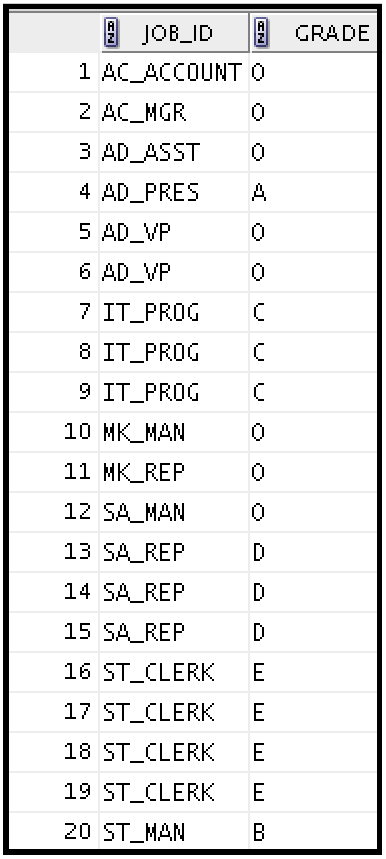
4. Using the `CASE` function, write a query that displays the grade of all employees based on the value of the `JOB_ID` column, using the following data:

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



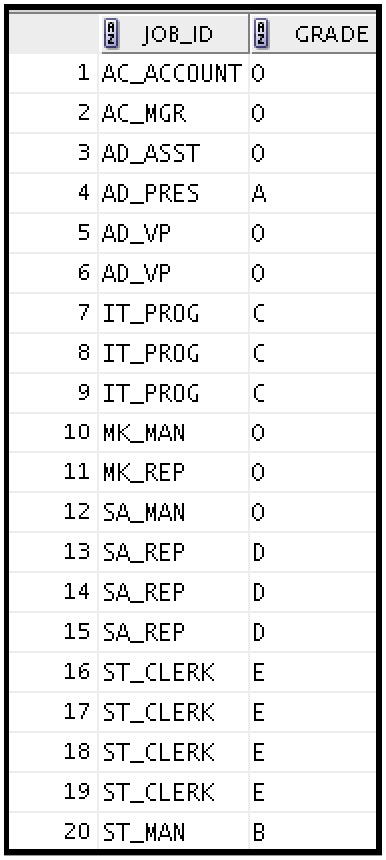
	JOB_ID	GRADE
1	AC_ACCOUNT	O
2	AC_MGR	O
3	AD_ASST	O
4	AD_PRES	A
5	AD_VP	O
6	AD_VP	O
7	IT_PROG	C
8	IT_PROG	C
9	IT_PROG	C
10	MK_MAN	O
11	MK_REP	O
12	SA_MAN	O
13	SA_REP	D
14	SA_REP	D
15	SA_REP	D
16	ST_CLERK	E
17	ST_CLERK	E
18	ST_CLERK	E
19	ST_CLERK	E
20	ST_MAN	B

5. Rewrite the statement in the preceding exercise by using the searched CASE syntax.



	JOB_ID	GRADE
1	AC_ACCOUNT	O
2	AC_MGR	O
3	AD_ASST	O
4	AD_PRES	A
5	AD_VP	O
6	AD_VP	O
7	IT_PROG	C
8	IT_PROG	C
9	IT_PROG	C
10	MK_MAN	O
11	MK_REP	O
12	SA_MAN	O
13	SA_REP	D
14	SA_REP	D
15	SA_REP	D
16	ST_CLERK	E
17	ST_CLERK	E
18	ST_CLERK	E
19	ST_CLERK	E
20	ST_MAN	B

6. Rewrite the statement in the preceding exercise by using the searched DECODE syntax.



	JOB_ID	GRADE
1	AC_ACCOUNT	O
2	AC_MGR	O
3	AD_ASST	O
4	AD_PRES	A
5	AD_VP	O
6	AD_VP	O
7	IT_PROG	C
8	IT_PROG	C
9	IT_PROG	C
10	MK_MAN	O
11	MK_REP	O
12	SA_MAN	O
13	SA_REP	D
14	SA_REP	D
15	SA_REP	D
16	ST_CLERK	E
17	ST_CLERK	E
18	ST_CLERK	E
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 a format that is 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. Create a query that displays 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;
```

4. Using the `CASE` function, write a query that displays the grade of all employees based on the value of the `JOB_ID` column, using the following data:

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

```
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;
```

5. Rewrite the statement in the preceding exercise by using the searched `CASE` syntax.

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

6. Rewrite the statement in the preceding exercise by using the searched `DECODE` syntax.

```
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;
```

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

Chapter 6

Practices for Lesson 6: Overview

Practice Overview

This practice covers the following topics:

- Writing queries that use group functions
- Grouping by rows to achieve multiple results
- Restricting groups by using the `HAVING` clause

Note the following location for the practice files: `/home/oracle/labs/sql1/labs`

Practice 6-1: Reporting Aggregated Data by Using Group Functions

Overview

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

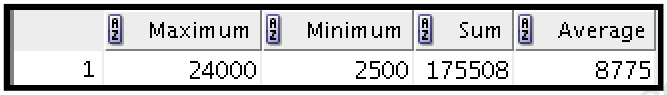
Tasks

Determine the validity of the following 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

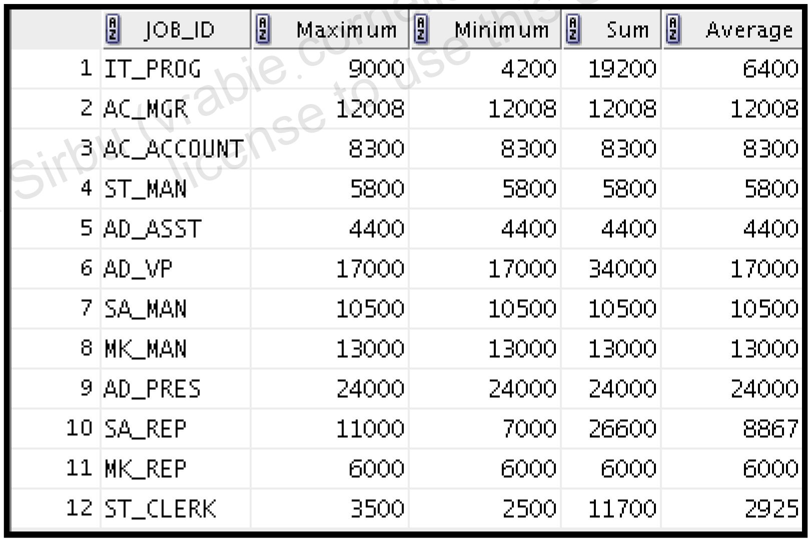
The HR department needs the following reports:

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.



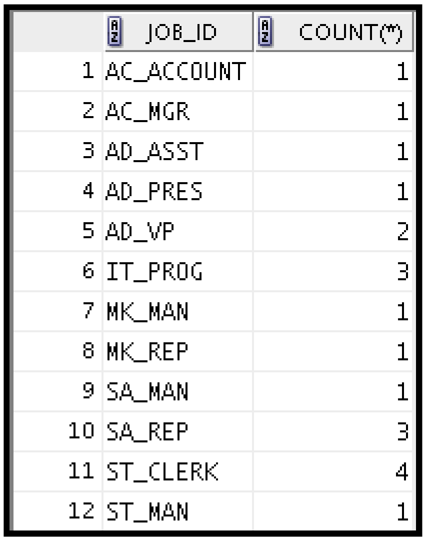
	Maximum	Minimum	Sum	Average
1	24000	2500	175508	8775

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`.



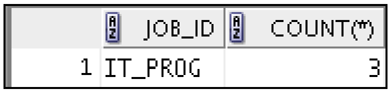
	JOB_ID	Maximum	Minimum	Sum	Average
1	IT_PROG	9000	4200	19200	6400
2	AC_MGR	12008	12008	12008	12008
3	AC_ACCOUNT	8300	8300	8300	8300
4	ST_MAN	5800	5800	5800	5800
5	AD_ASST	4400	4400	4400	4400
6	AD_VP	17000	17000	34000	17000
7	SA_MAN	10500	10500	10500	10500
8	MK_MAN	13000	13000	13000	13000
9	AD_PRES	24000	24000	24000	24000
10	SA_REP	11000	7000	26600	8867
11	MK_REP	6000	6000	6000	6000
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

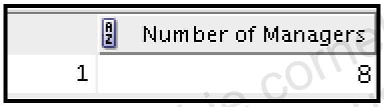
Generalize the query so that the user in the HR department is prompted for a job title. Save the script to a file named `lab_06_06.sql`. Run the query. Enter `IT_PROG` when prompted.



	JOB_ID	COUNT(*)
1	IT_PROG	3

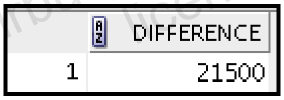
7. Determine the number of managers without listing them. Label the column `Number of Managers`.

Hint: Use the `MANAGER_ID` column to determine the number of managers.



	Number of Managers
1	8

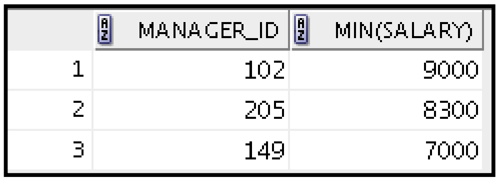
8. Find the difference between the highest and lowest salaries. Label the column `DIFFERENCE`.



	DIFFERENCE
1	21500

If you have time, complete the following exercises:

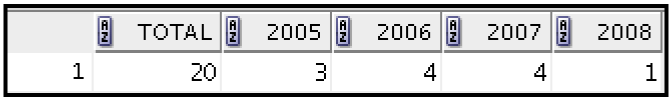
9. Create a report to display the manager number and the salary of the lowest-paid employee for that manager. Exclude anyone whose manager is not known. Exclude any groups where the minimum salary is $6,000 or less. Sort the output in descending order of salary.



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

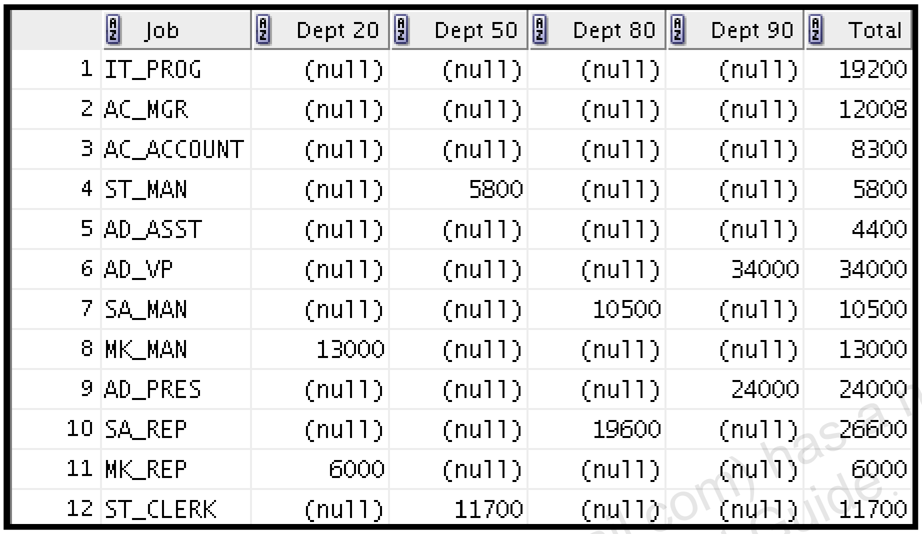
If you want an extra challenge, complete the following exercises:

10. Create a query to display the total number of employees and, of that total, the number of employees hired in 2005, 2006, 2007, and 2008. Create appropriate column headings.



	TOTAL	2005	2006	2007	2008
1	20	3	4	4	1

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.



	Job	Dept 20	Dept 50	Dept 80	Dept 90	Total
1	IT_PROG	(null)	(null)	(null)	(null)	19200
2	AC_MGR	(null)	(null)	(null)	(null)	12008
3	AC_ACCOUNT	(null)	(null)	(null)	(null)	8300
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	SA_MAN	(null)	(null)	10500	(null)	10500
8	MK_MAN	13000	(null)	(null)	(null)	13000
9	AD_PRES	(null)	(null)	(null)	24000	24000
10	SA_REP	(null)	(null)	19600	(null)	26600
11	MK_REP	6000	(null)	(null)	(null)	6000
12	ST_CLERK	(null)	11700	(null)	(null)	11700

Solution 6-1: Reporting Aggregated Data by Using Group Functions

Determine the validity of the following 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 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 displays the total number of employees and, of that total, the number of employees hired in 2005, 2006, 2007, and 2008. Create appropriate column headings.

```
SELECT  COUNT(*) total,
        SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),2005,1,0))"2005",
        SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),2006,1,0))"2006",
        SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),2007,1,0))"2007",
        SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),2008,1,0))"2008"
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;
```

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

Chapter 7

Practices for Lesson 7: Overview

Practice Overview

This practice covers the following topics:

- Joining tables using an equijoin
- Performing outer and self-joins
- Adding conditions

Note the following location for the practice files: `/home/oracle/labs/sql1/labs`

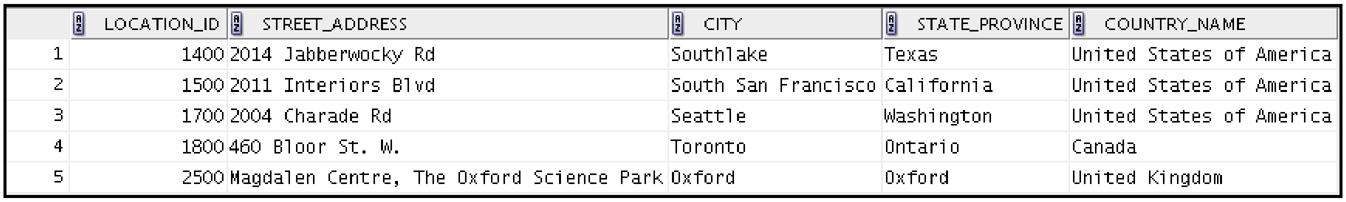
Practice 7-1: Displaying Data from Multiple Tables by Using Joins

Overview

This practice is intended to give you experience in extracting data from multiple tables using the SQL:1999–compliant joins.

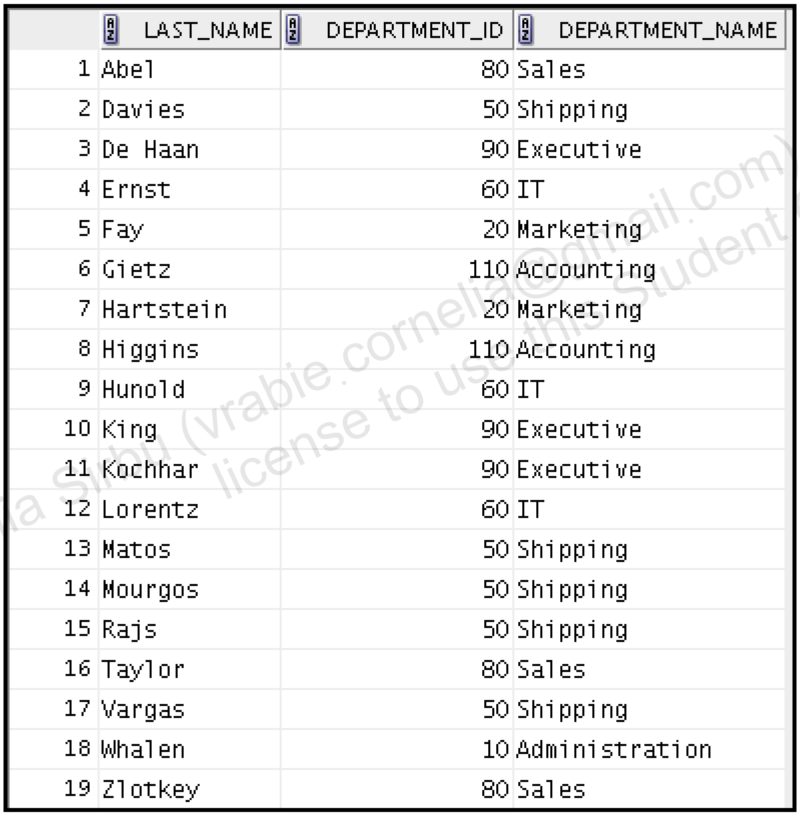
Tasks

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



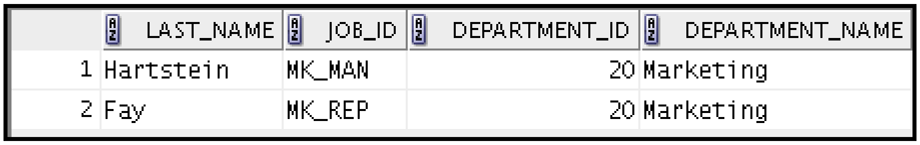
	LOCATION_ID	STREET_ADDRESS	CITY	STATE_PROVINCE	COUNTRY_NAME
1	1400	2014 Jabberwocky Rd	Southlake	Texas	United States of America
2	1500	2011 Interiors Blvd	South San Francisco	California	United States of America
3	1700	2004 Charade Rd	Seattle	Washington	United States of America
4	1800	460 Bloor St. W.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

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



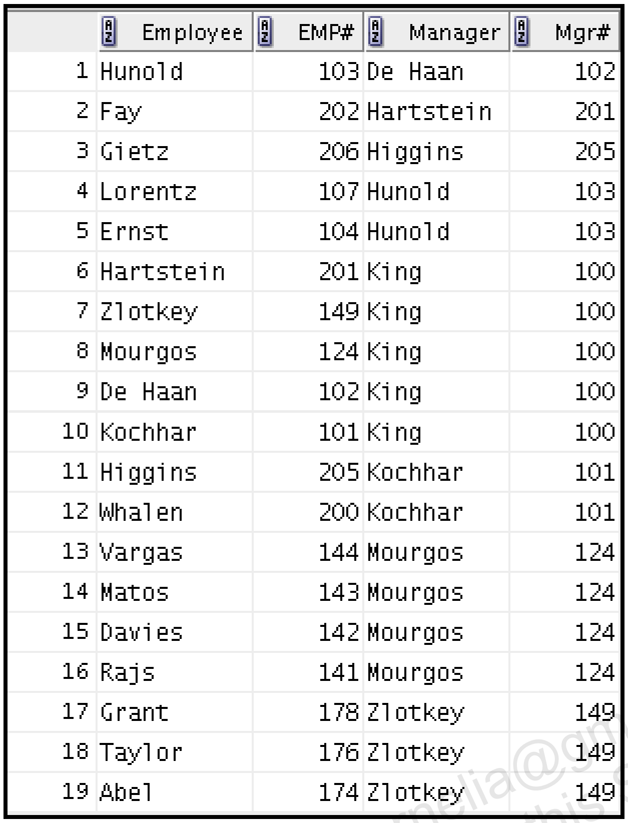
	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Abel	80	Sales
2	Davies	50	Shipping
3	De Haan	90	Executive
4	Ernst	60	IT
5	Fay	20	Marketing
6	Gietz	110	Accounting
7	Hartstein	20	Marketing
8	Higgins	110	Accounting
9	Hunold	60	IT
10	King	90	Executive
11	Kochhar	90	Executive
12	Lorentz	60	IT
13	Matos	50	Shipping
14	Mourgos	50	Shipping
15	Rajs	50	Shipping
16	Taylor	80	Sales
17	Vargas	50	Shipping
18	Whalen	10	Administration
19	Zlotkey	80	Sales

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



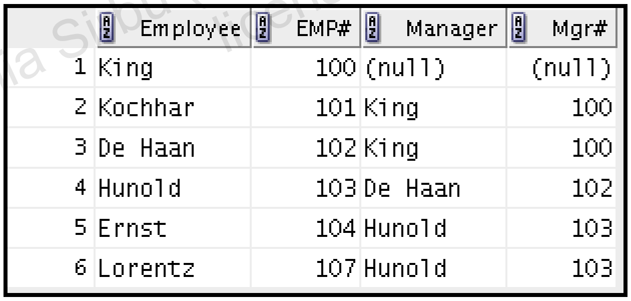
	LAST_NAME	JOB_ID	DEPARTMENT_ID	DEPARTMENT_NAME
1	Hartstein	MK_MAN	20	Marketing
2	Fay	MK_REP	20	Marketing

4. Create a report to display employees' last names and employee numbers along with their managers' last names and manager numbers. Label the columns `Employee`, `Emp#`, `Manager`, and `Mgr#`, respectively. Save your SQL statement as `lab_07_04.sql`. Run the query.



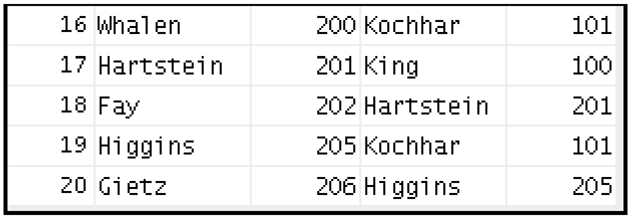
	Employee	EMP#	Manager	Mgr#
1	Hunold	103	De Haan	102
2	Fay	202	Hartstein	201
3	Gietz	206	Higgins	205
4	Lorentz	107	Hunold	103
5	Ernst	104	Hunold	103
6	Hartstein	201	King	100
7	Zlotkey	149	King	100
8	Mourgos	124	King	100
9	De Haan	102	King	100
10	Kochhar	101	King	100
11	Higgins	205	Kochhar	101
12	Whalen	200	Kochhar	101
13	Vargas	144	Mourgos	124
14	Matos	143	Mourgos	124
15	Davies	142	Mourgos	124
16	Rajs	141	Mourgos	124
17	Grant	178	Zlotkey	149
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 employee number. Save your SQL statement as `lab_07_05.sql`. Run the query in `lab_07_05.sql`.



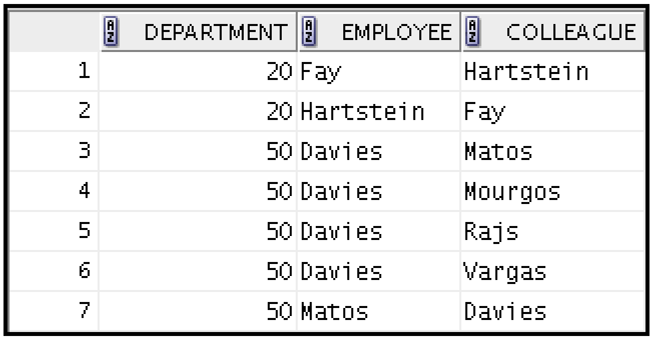
	Employee	EMP#	Manager	Mgr#
1	King	100	(null)	(null)
2	Kochhar	101	King	100
3	De Haan	102	King	100
4	Hunold	103	De Haan	102
5	Ernst	104	Hunold	103
6	Lorentz	107	Hunold	103

...



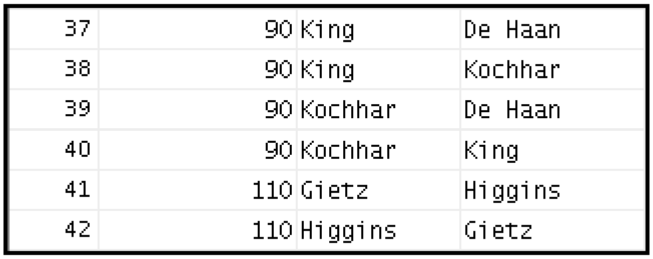
16	Whalen	200	Kochhar	101
17	Hartstein	201	King	100
18	Fay	202	Hartstein	201
19	Higgins	205	Kochhar	101
20	Gietz	206	Higgins	205

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`.



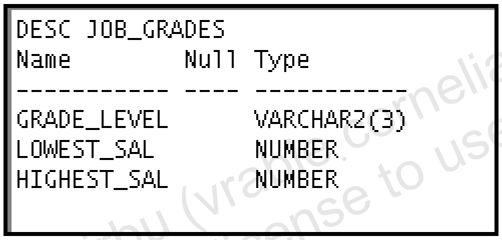
	DEPARTMENT	EMPLOYEE	COLLEAGUE
1	20	Fay	Hartstein
2	20	Hartstein	Fay
3	50	Davies	Matos
4	50	Davies	Mourgos
5	50	Davies	Rajs
6	50	Davies	Vargas
7	50	Matos	Davies

...

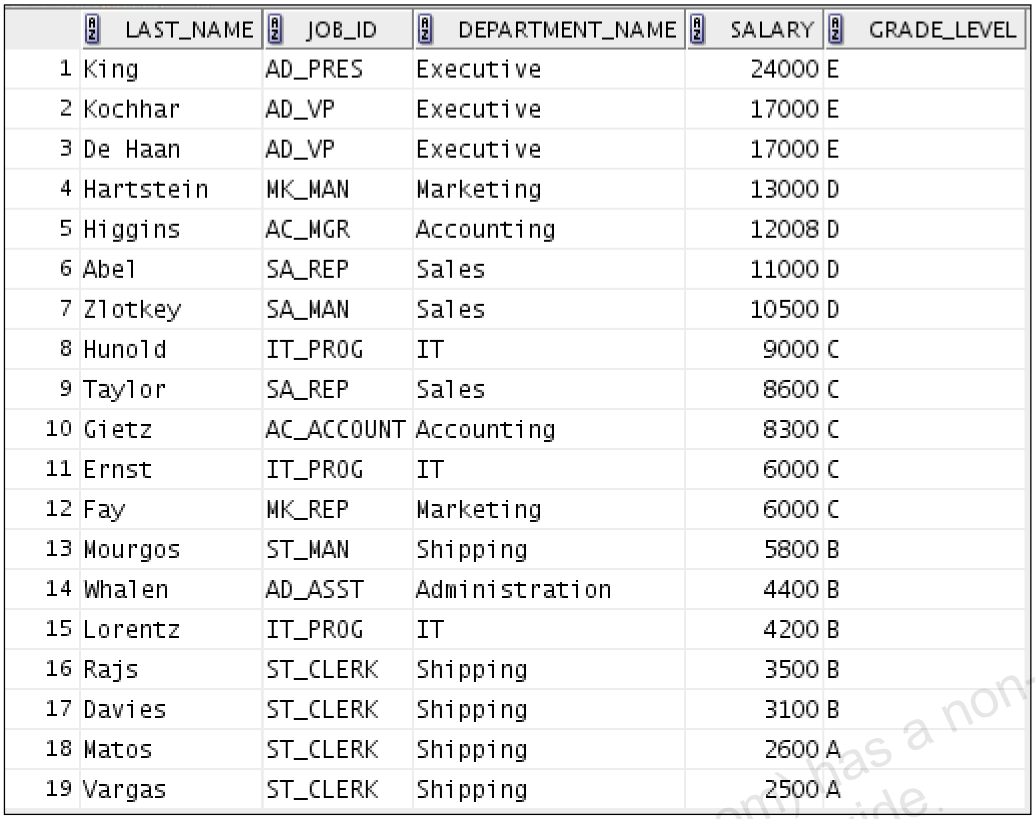


37	90	King	De Haan
38	90	King	Kochhar
39	90	Kochhar	De Haan
40	90	Kochhar	King
41	110	Gietz	Higgins
42	110	Higgins	Gietz

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



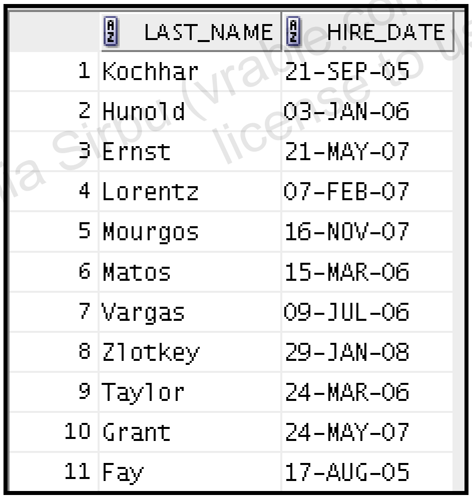
```
DESC JOB_GRADES
Name        Null Type
----------- ---- -----------
GRADE_LEVEL      VARCHAR2(3)
LOWEST_SAL       NUMBER
HIGHEST_SAL      NUMBER
```



	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	12008	D
6	Abel	SA_REP	Sales	11000	D
7	Zlotkey	SA_MAN	Sales	10500	D
8	Hunold	IT_PROG	IT	9000	C
9	Taylor	SA_REP	Sales	8600	C
10	Gietz	AC_ACCOUNT	Accounting	8300	C
11	Ernst	IT_PROG	IT	6000	C
12	Fay	MK_REP	Marketing	6000	C
13	Mourgos	ST_MAN	Shipping	5800	B
14	Whalen	AD_ASST	Administration	4400	B
15	Lorentz	IT_PROG	IT	4200	B
16	Rajs	ST_CLERK	Shipping	3500	B
17	Davies	ST_CLERK	Shipping	3100	B
18	Matos	ST_CLERK	Shipping	2600	A
19	Vargas	ST_CLERK	Shipping	2500	A

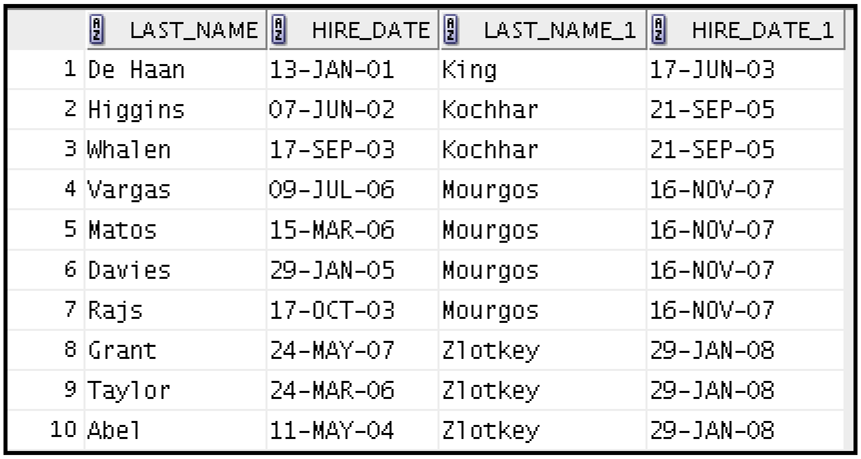
If you want an extra challenge, complete the following exercises:

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



	LAST_NAME	HIRE_DATE
1	Kochhar	21-SEP-05
2	Hunold	03-JAN-06
3	Ernst	21-MAY-07
4	Lorentz	07-FEB-07
5	Mourgos	16-NOV-07
6	Matos	15-MAR-06
7	Vargas	09-JUL-06
8	Zlotkey	29-JAN-08
9	Taylor	24-MAR-06
10	Grant	24-MAY-07
11	Fay	17-AUG-05

9. The HR department needs to find the names and hire dates of 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`.



	LAST_NAME	HIRE_DATE	LAST_NAME_1	HIRE_DATE_1
1	De Haan	13-JAN-01	King	17-JUN-03
2	Higgins	07-JUN-02	Kochhar	21-SEP-05
3	Whalen	17-SEP-03	Kochhar	21-SEP-05
4	Vargas	09-JUL-06	Mourgos	16-NOV-07
5	Matos	15-MAR-06	Mourgos	16-NOV-07
6	Davies	29-JAN-05	Mourgos	16-NOV-07
7	Rajs	17-OCT-03	Mourgos	16-NOV-07
8	Grant	24-MAY-07	Zlotkey	29-JAN-08
9	Taylor	24-MAR-06	Zlotkey	29-JAN-08
10	Abel	11-MAY-04	Zlotkey	29-JAN-08

Solution 7-1: Displaying Data from Multiple Tables by 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
USING  (location_id)
WHERE LOWER(l.city) = 'toronto';
```

4. Create a report to display employees' last names and employee numbers along with their managers' last names and manager numbers. 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 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 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 of 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;
```

Practices for Lesson 8: Using Subqueries to Solve Queries

Chapter 8

Practices for Lesson 8: Overview

Practice Overview

This practice covers the following topics:

- Creating subqueries to query values based on unknown criteria
- Using subqueries to find values that exist in one set of data and not in another

Note the following location for the practice files: `/home/oracle/labs/sql1/labs`

Practice 8-1: Using Subqueries to Solve Queries

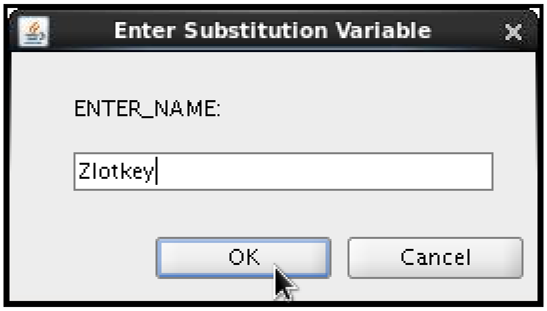
Overview

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

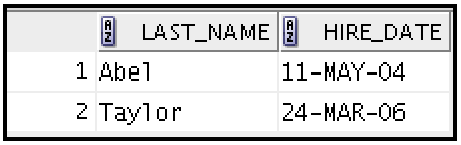
For practice questions, you may want to create the inner query first. Make sure that it runs and produces the data that you anticipate before you code the outer query.

Tasks

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 the user supplies (excluding that employee). For example, if the user enters `Zlotkey`, find all employees who work with Zlotkey (excluding Zlotkey).

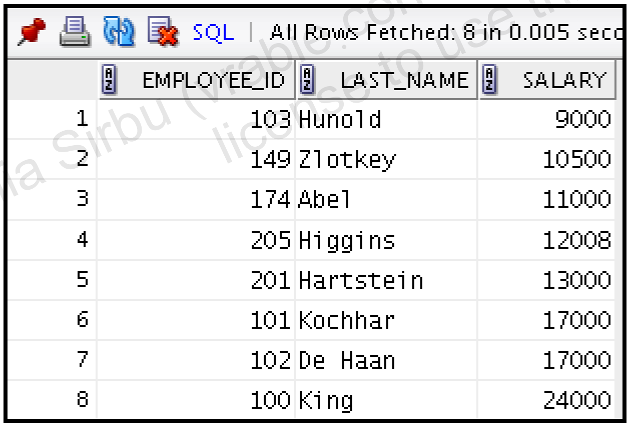

Enter Substitution Variable

ENTER_NAME: Zlotkey



	LAST_NAME	HIRE_DATE
1	Abel	11-MAY-04
2	Taylor	24-MAR-06

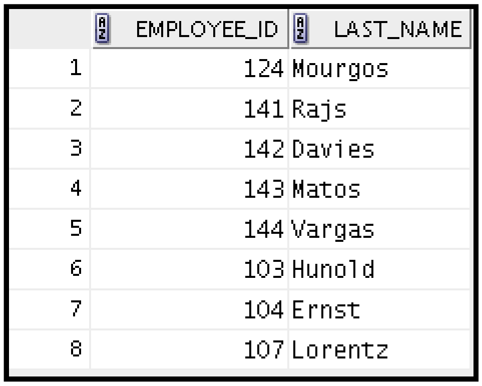
2. Create a report that displays the employee number, last name, and salary of all employees who earn more than the average salary. Sort the results in ascending order by salary.



All Rows Fetched: 8 in 0.005 seco

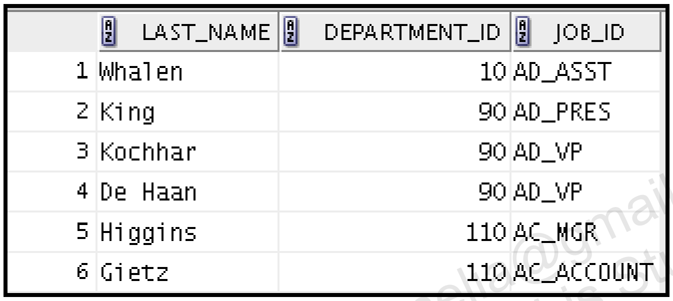
	EMPLOYEE_ID	LAST_NAME	SALARY
1	103	Hunold	9000
2	149	Zlotkey	10500
3	174	Abel	11000
4	205	Higgins	12008
5	201	Hartstein	13000
6	101	Kochhar	17000
7	102	De Haan	17000
8	100	King	24000

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

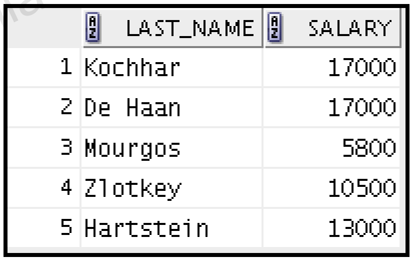
4. The HR department needs a report that displays the last name, department number, and job ID of all employees whose department location ID is 1700.



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

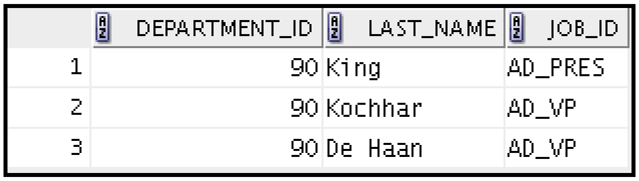
Modify the query so that the user is prompted for a location ID. Save this to a file named `lab_08_04.sql`.

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



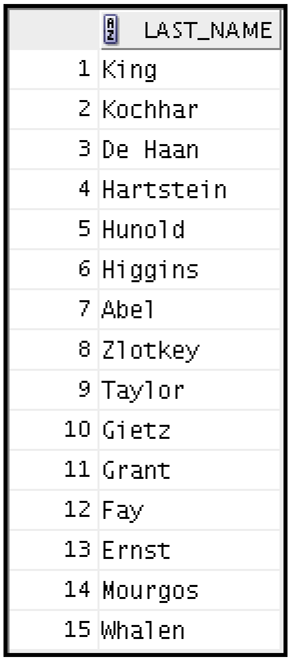
	LAST_NAME	SALARY
1	Kochhar	17000
2	De Haan	17000
3	Mourgos	5800
4	Zlotkey	10500
5	Hartstein	13000

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



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

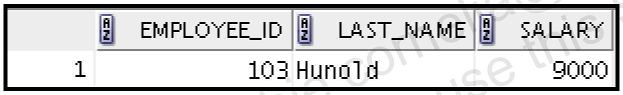
7. Create a report that displays a list of all employees whose salary is more than the salary of any employee from department 60.



	LAST_NAME
1	King
2	Kochhar
3	De Haan
4	Hartstein
5	Hunold
6	Higgins
7	Abel
8	Zlotkey
9	Taylor
10	Gietz
11	Grant
12	Fay
13	Ernst
14	Mourgos
15	Whalen

If you have 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 the letter “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

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 the user supplies (excluding that employee). For example, if the user enters `Zlotkey`, find all employees who work with Zlotkey (excluding Zlotkey).

```
--Execute the UNDEFINE command to remove a variable

UNDEFINE Enter_name

-- Execute the below SELECT statements to retrieve the values
from employees table

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 ascending order by 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 the letter "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 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 the letter “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);
```

Practices for Lesson 9: Using the Set Operators

Chapter 9

Practices for Lesson 9: Overview

Practice Overview

In this practice, you create reports by using the following:

- `UNION` operator
- `INTERSECT` operator
- `MINUS` operator

Note the following location for the practice files: `/home/oracle/labs/sql1/labs`

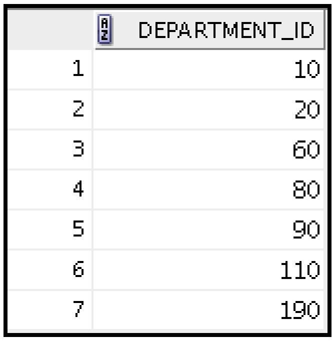
Practice 9-1: Using Set Operators

Overview

In this practice, you write queries using the set operators UNION, INTERSECT, and MINUS.

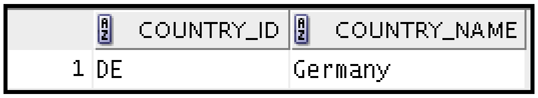
Tasks

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.



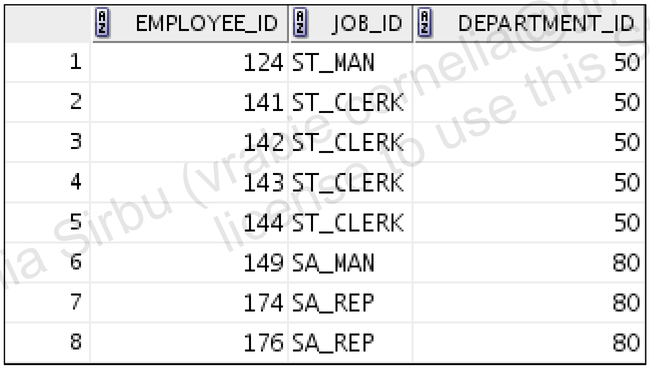
	DEPARTMENT_ID
1	10
2	20
3	60
4	80
5	90
6	110
7	190

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



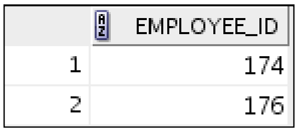
	COUNTRY_ID	COUNTRY_NAME
1	DE	Germany

3. Produce a list of all the employees who work in departments 50 and 80. Display the employee ID, job ID, and department ID by using the set operators.



	EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
1	124	ST_MAN	50
2	141	ST_CLERK	50
3	142	ST_CLERK	50
4	143	ST_CLERK	50
5	144	ST_CLERK	50
6	149	SA_MAN	80
7	174	SA_REP	80
8	176	SA_REP	80

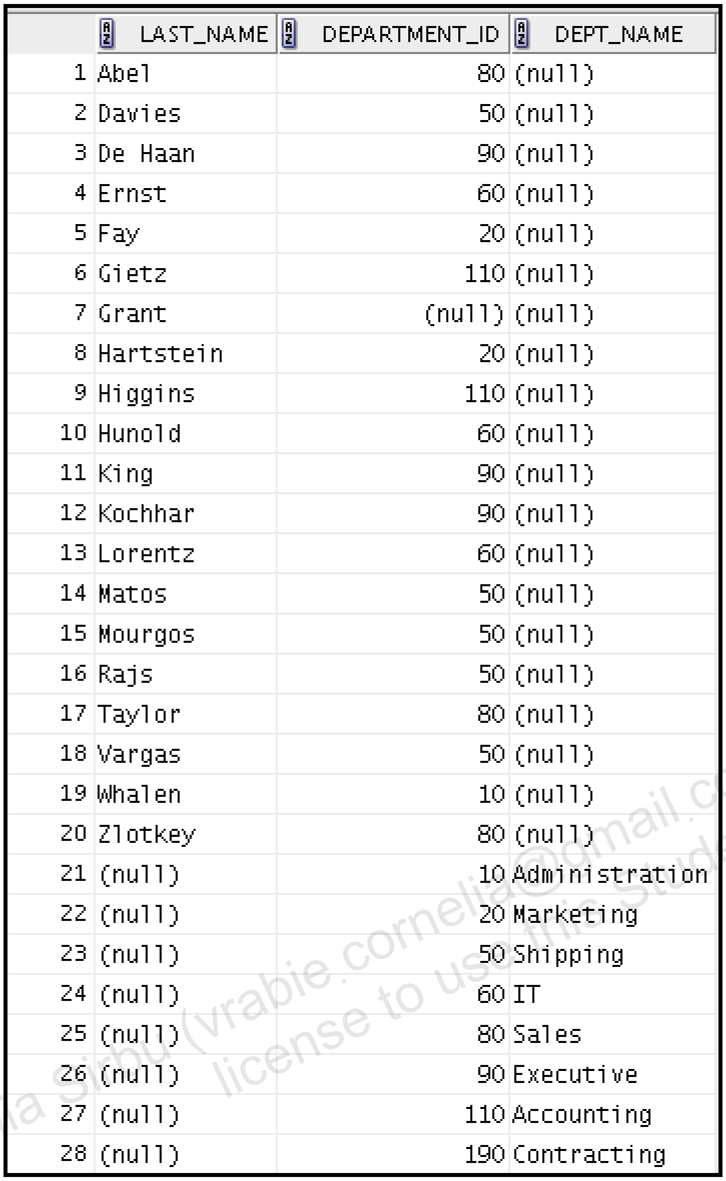
4. Create a report that lists the detail of all employees who are sales representatives and are currently working in the sales department.



	EMPLOYEE_ID
1	174
2	176

5. The HR department needs a report with the following specifications:
   - Last names and department IDs of all employees from the EMPLOYEES table, regardless of whether or not they belong to a department
   - Department IDs and department names 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 report.



	LAST_NAME	DEPARTMENT_ID	DEPT_NAME
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 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 IDs and the names 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 all the employees who work in departments 50 and 80. Display the employee ID, job ID, and department ID by using the set operators.

```
SELECT employee_id, job_id, department_id
FROM EMPLOYEES
WHERE department_id=50
UNION ALL
SELECT employee_id, job_id, department_id
FROM EMPLOYEES
WHERE department_id=80;
```

4. Create a report that lists the detail of all employees who are sales representatives and are currently working in the sales department.

```
SELECT EMPLOYEE_ID
FROM EMPLOYEES
WHERE JOB_ID='SA_REP'
INTERSECT
SELECT EMPLOYEE_ID
FROM EMPLOYEES
WHERE DEPARTMENT_ID=80;
```

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

   - Last names and department IDs of all employees from the EMPLOYEES table, regardless of whether or not they belong to a department
   - Department IDs and department names 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 report.

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

Practices for Lesson 10: Managing Tables by Using DML Statements

Chapter 10

Practices for Lesson 10: Overview

Lesson Overview

This practice covers the following topics:

- Inserting rows into tables
- Updating and deleting rows in a table
- Controlling transactions

Note: Before starting this practice, execute `/home/oracle/labs/sql1/code_ex /cleanup_scripts/cleanup_10.sql script.`

Note the following location for the practice files: `/home/oracle/labs/sql1/labs`

Practice 10-1: Managing Tables by Using DML Statements

Overview

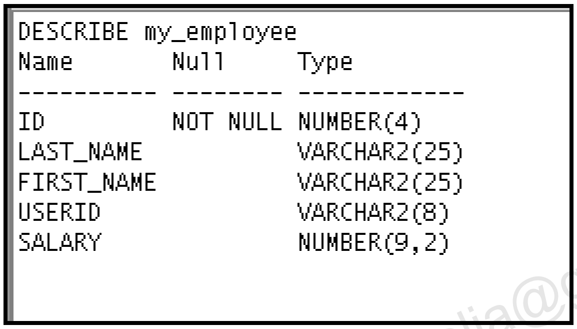
The HR department wants you to create SQL statements to insert, update, and delete employee data. As a prototype, you use the MY_EMPLOYEE table before giving the statements to the HR department.

Note

- For all the DML statements, use the Run Script icon (or press F5) to execute the query. Thus, 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.
- Execute cleanup_10.sql script from /home/oracle/labs/sql1/code_ex /cleanup_scripts/ before performing the following tasks.

Tasks

1. Create a table called MY_EMPLOYEE.
2. Describe the structure of the MY_EMPLOYEE table to identify the column names.



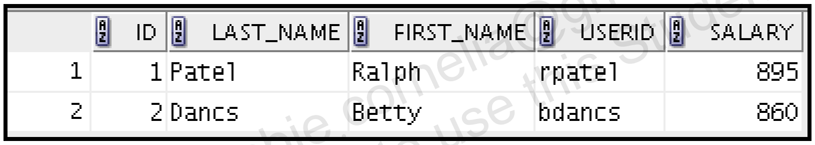
```
DESCRIBE my_employee
Name       Null     Type
---------- -------- ------------
ID         NOT NULL NUMBER(4)
LAST_NAME           VARCHAR2(25)
FIRST_NAME          VARCHAR2(25)
USERID              VARCHAR2(8)
SALARY              NUMBER(9,2)
```

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.

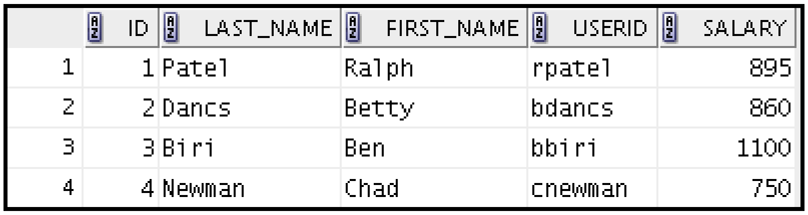


	ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	1	Patel	Ralph	rpatel	895
2	2	Dancs	Betty	bdancs	860

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

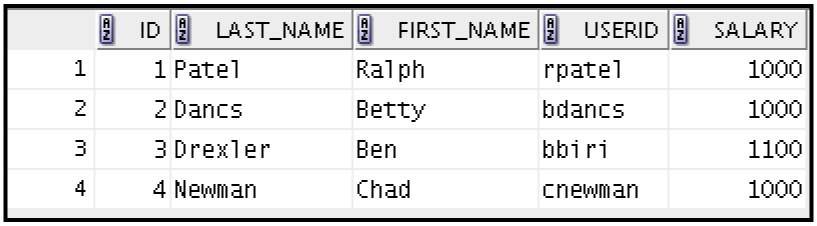
9. Make the data additions permanent.

Update and delete data in the MY_EMPLOYEE table.

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

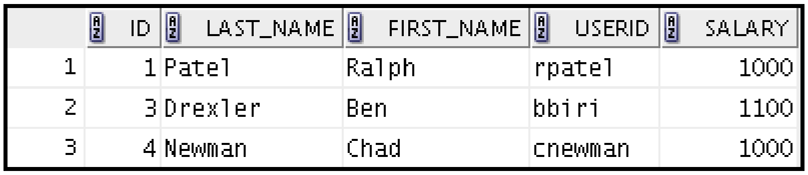
11. Change the salary to $1,000 for all employees who have a salary less than $900.

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.



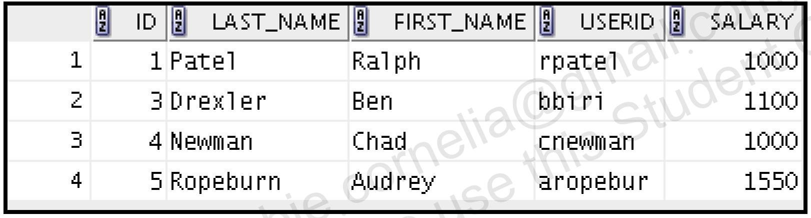
	ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	1	Patel	Ralph	rpatel	1000
2	3	Drexler	Ben	bbiri	1100
3	4	Newman	Chad	cnewman	1000

15. Commit all pending changes.

Control the data transaction to the MY_EMPLOYEE table.

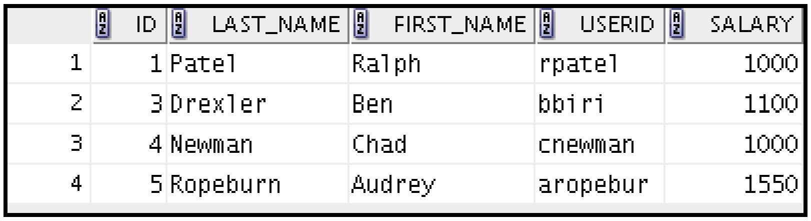
16. Populate the table with the last row of the sample data listed in step 3 by using the statements in the script that you created in step 6. Run the statements in the script.

    Note: Perform the steps (17-23) in one session only.
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.

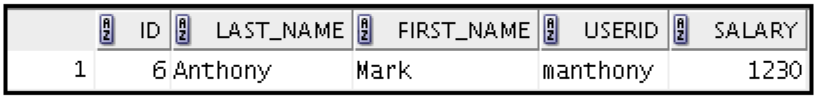
If you have time, complete the following exercise:

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.

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`.



	ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	6	Anthony	Mark	manthony	1230

Solution 10-1: Managing Tables by Using DML Statements

Insert data into the MY_EMPLOYEE table.

1. Create a table called MY_EMPLOYEE.

```
CREATE TABLE my_employee
  (id  NUMBER(4) CONSTRAINT my_employee_id_pk PRIMARY Key,
   last_name VARCHAR2(25),
   first_name VARCHAR2(25),
   userid  VARCHAR2(8),
   salary  NUMBER(9,2));
```

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 the 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 the 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);
```

Note: Perform the steps (17-23) in one session only.

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. The script should, therefore, 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: Introduction to Data Definition Language

Chapter 11

Practices for Lesson 11: Overview

Lesson Overview

This practice covers the following topics:

- Creating new tables
- Creating a new table by using the `CREATE TABLE AS` syntax
- Verifying that tables exist
- Altering tables
- Adding columns
- Dropping columns
- Setting a table to read-only status
- Dropping tables

Note: Before starting this practice, execute the `/home/oracle/labs/sql1/code_ex/cleanup_scripts/cleanup_11.sql` script.

Note the following location for the practice files: `/home/oracle/labs/sql1/labs`

Practice 11-1: Introduction to Data Definition Language

Overview

In this practice, you 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`. You use the `ALTER TABLE` command to modify table columns.

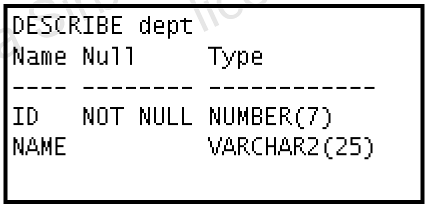
Note

- For all the DDL and DML statements, click the Run Script icon (or press F5) to execute the query in SQL Developer. Thus, 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.
- Execute the `cleanup_11.sql` script from `/home/oracle/labs/sql1/code_ex/cleanup_scripts/cleanup_11.sql` before performing the following tasks.

Tasks

1. Create the `DEPT` table based on the following table instance chart. Save the statement in the `lab_11_01.sql` script, and then execute the statement in the script to create the table. Confirm that the table is created.

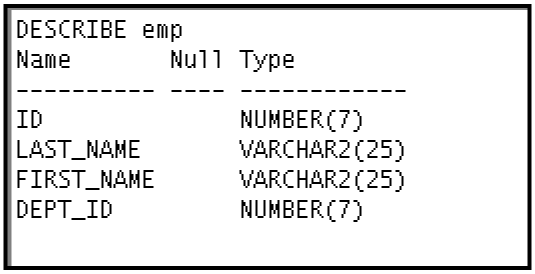
Column Name	ID	NAME
Key Type	Primary key	
Nulls/Unique		
FK Table		
FK Column		
Data type	NUMBER	VARCHAR2
Length	7	25



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

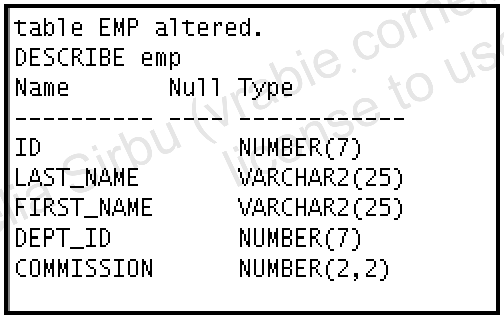
2. Create the `EMP` table based on the following table instance chart. Save the statement in the `lab_11_02.sql` script, and then execute the statement in the script to create the table. Confirm that the table is created.

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



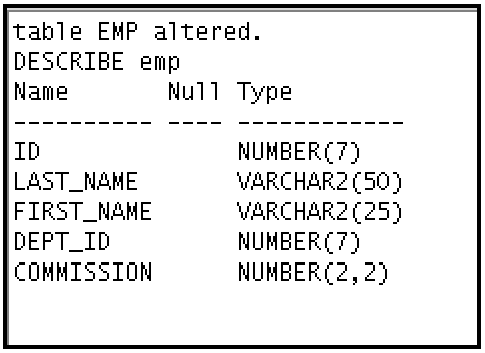
```
DESCRIBE emp
Name       Null Type
---------- ---- ------------
ID              NUMBER(7)
LAST_NAME       VARCHAR2(25)
FIRST_NAME      VARCHAR2(25)
DEPT_ID         NUMBER(7)
```

3. Modify the `EMP` table. Add a `COMMISSION` column of the `NUMBER` data type, with precision 2 and scale 2. Confirm your modification.



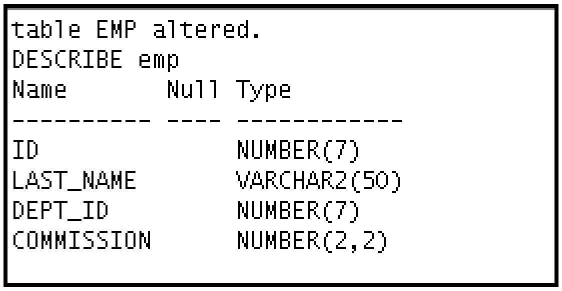
```
table EMP altered.
DESCRIBE emp
Name       Null Type
---------- ---- ------------
ID              NUMBER(7)
LAST_NAME       VARCHAR2(25)
FIRST_NAME      VARCHAR2(25)
DEPT_ID         NUMBER(7)
COMMISSION      NUMBER(2,2)
```

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



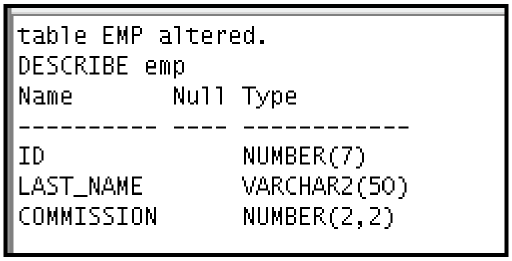
```
table EMP altered.
DESCRIBE emp
Name       Null Type
---------- ---- ------------
ID              NUMBER(7)
LAST_NAME       VARCHAR2(50)
FIRST_NAME      VARCHAR2(25)
DEPT_ID         NUMBER(7)
COMMISSION      NUMBER(2,2)
```

5. Drop the FIRST_NAME column from the EMP table. Confirm your modification by checking the description of the table.



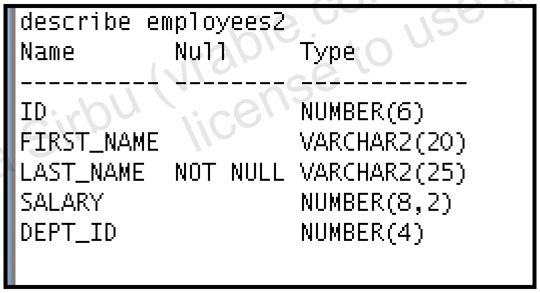
```
table EMP altered.
DESCRIBE emp
Name       Null Type
---------- ---- ------------
ID              NUMBER(7)
LAST_NAME       VARCHAR2(50)
DEPT_ID         NUMBER(7)
COMMISSION      NUMBER(2,2)
```

6. In the EMP table, mark the DEPT_ID column as UNUSED. Confirm your modification by checking the description of the table.



```
table EMP altered.
DESCRIBE emp
Name       Null Type
---------- ---- ------------
ID              NUMBER(7)
LAST_NAME       VARCHAR2(50)
COMMISSION      NUMBER(2,2)
```

7. Drop all the UNUSED columns from the EMP table.
8. 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.

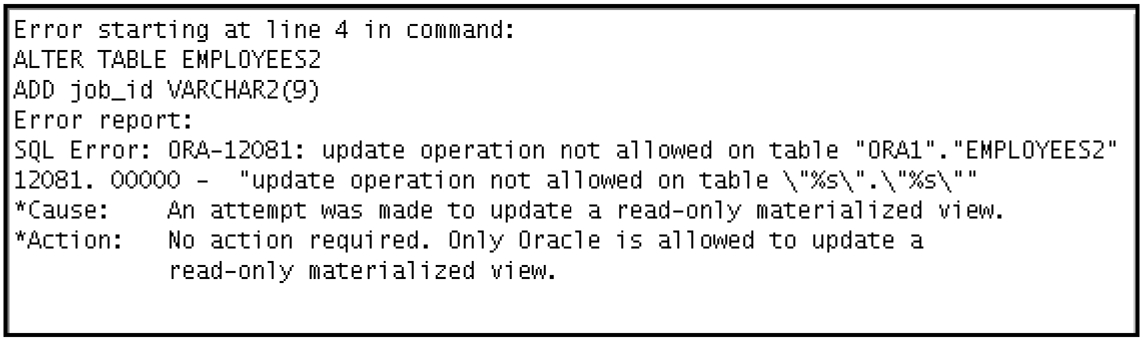


```
describe employees2
Name       Null     Type
---------- -------- ------------
ID                  NUMBER(6)
FIRST_NAME          VARCHAR2(20)
LAST_NAME  NOT NULL VARCHAR2(25)
SALARY              NUMBER(8,2)
DEPT_ID             NUMBER(4)
```

9. Alter the status of the EMPLOYEES2 table to read-only.

10. Try to add a column `JOB_ID` in the `EMPLOYEES2` table.

    Note: You will get the “Update operation not allowed on table” error message. You will not be allowed to add any column to the table because it is assigned a read-only status.

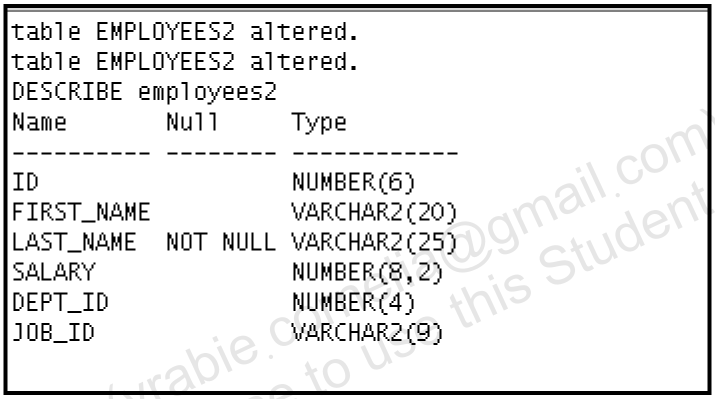


```
Error starting at line 4 in command:
ALTER TABLE EMPLOYEES2
ADD job_id VARCHAR2(9)
Error report:
SQL Error: ORA-12081: update operation not allowed on table "ORA1"."EMPLOYEES2"
12081. 00000 -  "update operation not allowed on table \"%s\".\"%s\""
*Cause:    An attempt was made to update a read-only materialized view.
*Action:   No action required. Only Oracle is allowed to update a
           read-only materialized view.
```

11. Revert the `EMPLOYEES2` table to read/write status. Now try to add the same column again.

    Now, because the table is assigned a `READ WRITE` status, you will be allowed to add a column to the table.

    You should get the following messages:



```
table EMPLOYEES2 altered.
table EMPLOYEES2 altered.
DESCRIBE employees2
Name       Null     Type
---------- -------- ------------
ID                  NUMBER(6)
FIRST_NAME          VARCHAR2(20)
LAST_NAME  NOT NULL VARCHAR2(25)
SALARY              NUMBER(8,2)
DEPT_ID             NUMBER(4)
JOB_ID              VARCHAR2(9)
```

12. Drop the `EMP`, `DEPT`, and `EMPLOYEES2` table.

Solution 11-1: Introduction to Data Definition Language

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.

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

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

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

```
DESCRIBE dept;
```

2. Create the EMP table based on the following table instance chart. Save the statement in a script called lab_11_02.sql, and then execute the statement in the script to create the table. Confirm that the table is created.

```
CREATE TABLE  emp
```

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

```
  (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
```

3. Modify the `EMP` table. Add a `COMMISSION` column of the `NUMBER` data type, with precision 2 and scale 2. Confirm your modification.

```
ALTER TABLE emp
      ADD commission NUMBER(2,2);

DESCRIBE emp
```

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

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

DESCRIBE emp
```

5. Drop the `FIRST_NAME` column from the `EMP` table. Confirm your modification by checking the description of the table.

```
ALTER TABLE emp
     DROP COLUMN first_name;
 DESCRIBE emp
```

6. In the `EMP` table, mark the `DEPT_ID` column as `UNUSED`. Confirm your modification by checking the description of the table.

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

   DESCRIBE emp
```

7. Drop all the `UNUSED` columns from the `EMP` table.

```
   ALTER TABLE emp
      DROP UNUSED COLUMNS;
```

8. 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. Confirm that the table is created.

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

9. Alter the EMPLOYEES2 table status to read-only.

```
ALTER TABLE employees2 READ ONLY;
```

10. Try to add a column JOB_ID in the EMPLOYEES2 table.

Note: You will get the “Update operation not allowed on table” error message. You will not be allowed to add any column to the table because it is assigned a read-only status.

```
ALTER TABLE employees2
ADD job_id VARCHAR2(9);
```

11. Revert the EMPLOYEES2 table to the read/write status. Now try to add the same column again.

Now, because the table is assigned a READ WRITE status, you will be allowed to add a column to the table.

```
ALTER TABLE employees2 READ WRITE;

ALTER TABLE employees2
ADD job_id VARCHAR2(9);
DESCRIBE employees2
```

12. Drop the EMP, DEPT, and EMPLOYEES2 table.

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

```
DROP TABLE emp;
DROP TABLE dept;
DROP TABLE employees2;
```

Practices for Lesson 12: Introduction to Data Dictionary Views

Chapter 12

Practices for Lesson 12: Overview

Practice overview

This practice covers the following topics:

- Querying the dictionary views for table and column information
- Querying the dictionary views for constraint information
- Adding a comment to a table and querying the dictionary views for comment information

Note the following location for the practice files: `/home/oracle/labs/sql2/labs`

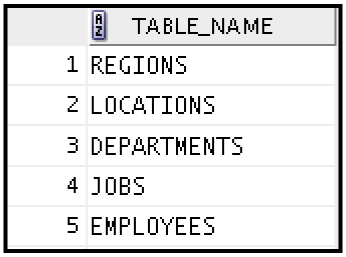
Practice 12-1: Introduction to Data Dictionary Views

Overview

In this practice, you query the dictionary views to find information about objects in your schema.

Tasks

1. Query the `USER_TABLES` data dictionary view to see information about the tables that you own.

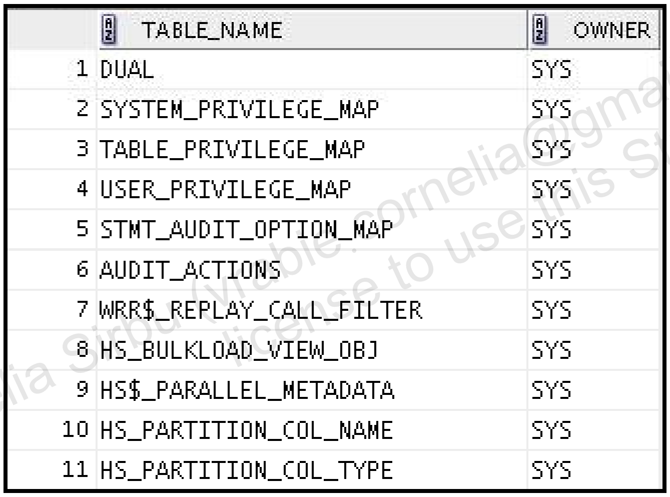


	TABLE_NAME
1	REGIONS
2	LOCATIONS
3	DEPARTMENTS
4	JOBS
5	EMPLOYEES

…

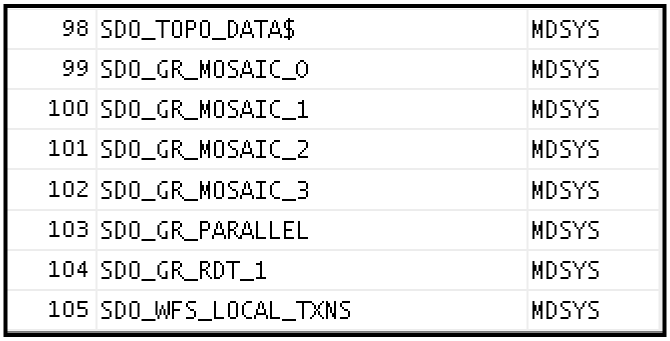
2. Query the `ALL_TABLES` data dictionary view to see information about all the tables that you can access. Exclude the tables that you own.

   Note: Your list may not exactly match the following list:



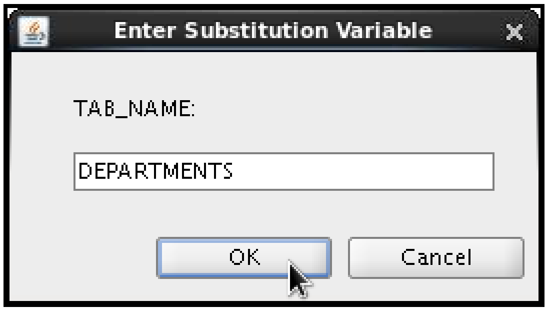
	TABLE_NAME	OWNER
1	DUAL	SYS
2	SYSTEM_PRIVILEGE_MAP	SYS
3	TABLE_PRIVILEGE_MAP	SYS
4	USER_PRIVILEGE_MAP	SYS
5	STMT_AUDIT_OPTION_MAP	SYS
6	AUDIT_ACTIONS	SYS
7	WRR$_REPLAY_CALL_FILTER	SYS
8	HS_BULKLOAD_VIEW_OBJ	SYS
9	HS$_PARALLEL_METADATA	SYS
10	HS_PARTITION_COL_NAME	SYS
11	HS_PARTITION_COL_TYPE	SYS

…

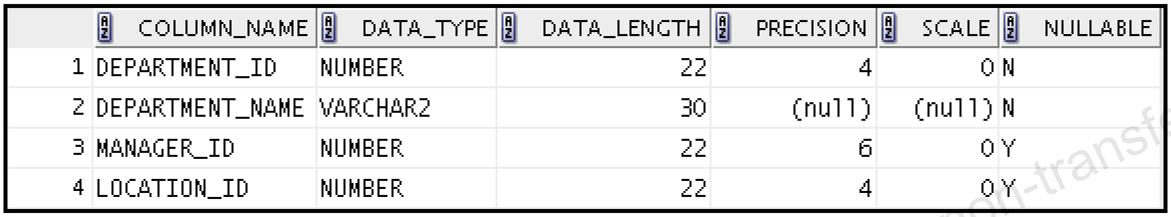


98	SDO_TOPO_DATA$	MDSYS
99	SDO_GR_MOSAIC_0	MDSYS
100	SDO_GR_MOSAIC_1	MDSYS
101	SDO_GR_MOSAIC_2	MDSYS
102	SDO_GR_MOSAIC_3	MDSYS
103	SDO_GR_PARALLEL	MDSYS
104	SDO_GR_RDT_1	MDSYS
105	SDO_WFS_LOCAL_TXNS	MDSYS

3. For a specified table, create a script that reports the column names, data types, and data types' lengths, as well as whether nulls are allowed. Prompt the user to enter the table name. Give appropriate aliases to the `DATA_PRECISION and DATA_SCALE` columns. Save this script in a file named `lab_02_03.sql`.
   For example, if the user enters `DEPARTMENTS`, the following output results:

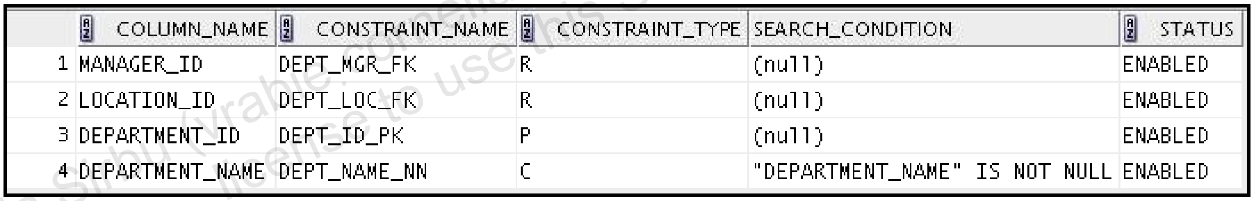


TAB_NAME: DEPARTMENTS



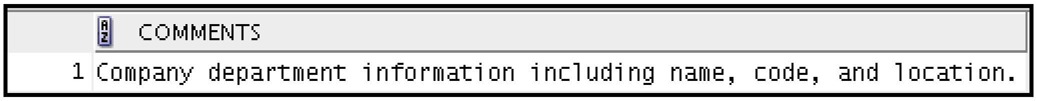
	COLUMN_NAME	DATA_TYPE	DATA_LENGTH	PRECISION	SCALE	NULLABLE
1	DEPARTMENT_ID	NUMBER	22	4	0	N
2	DEPARTMENT_NAME	VARCHAR2	30	(null)	(null)	N
3	MANAGER_ID	NUMBER	22	6	0	Y
4	LOCATION_ID	NUMBER	22	4	0	Y

4. Create a script that reports the column name, constraint name, constraint type, search condition, and status for a specified table. You must join the `USER_CONSTRAINTS` and `USER_CONS_COLUMNS` tables to obtain all this information. Prompt the user to enter the table name. Save the script in a file named `lab_02_04.sql`.
   For example, if the user enters `DEPARTMENTS`, the following output results:



	COLUMN_NAME	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	STATUS
1	MANAGER_ID	DEPT_MGR_FK	R	(null)	ENABLED
2	LOCATION_ID	DEPT_LOC_FK	R	(null)	ENABLED
3	DEPARTMENT_ID	DEPT_ID_PK	P	(null)	ENABLED
4	DEPARTMENT_NAME	DEPT_NAME_NN	C	"DEPARTMENT_NAME" IS NOT NULL	ENABLED

5. Add a comment to the `DEPARTMENTS` table. Then query the `USER_TAB_COMMENTS` view to verify that the comment is present.

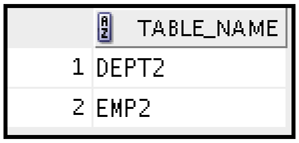


	COMMENTS
1	Company department information including name, code, and location.

6. Run the `lab_02_06_tab.sql` script as a prerequisite for exercises 16 through 19. Alternatively, open the script file to copy the code and paste it into your SQL Worksheet. Then execute the script. This script:
   - Drops the existing `DEPT2` and `EMP2` tables
   - Creates the `DEPT2` and `EMP2` tables

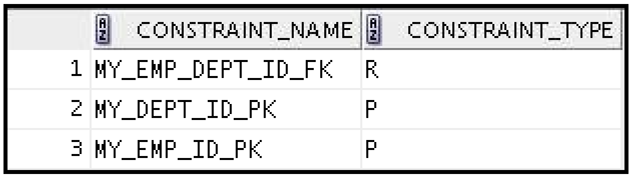
   Note: In Practice 2, you should have already dropped the `DEPT2` and `EMP2` tables so that they cannot be restored.

7. Confirm that both the `DEPT2` and `EMP2` tables are stored in the data dictionary.



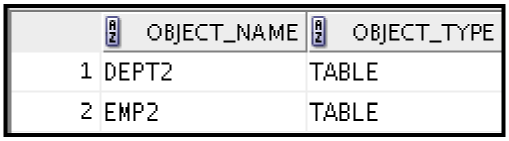
	TABLE_NAME
1	DEPT2
2	EMP2

8. Confirm that the constraints were added, by querying the `USER_CONSTRAINTS` view. Note the types and names of the constraints.



	CONSTRAINT_NAME	CONSTRAINT_TYPE
1	MY_EMP_DEPT_ID_FK	R
2	MY_DEPT_ID_PK	P
3	MY_EMP_ID_PK	P

9. Display the object names and types from the `USER_OBJECTS` data dictionary view for the `EMP2` and `DEPT2` tables.



	OBJECT_NAME	OBJECT_TYPE
1	DEPT2	TABLE
2	EMP2	TABLE

Solution 12-1: Introduction to Data Dictionary Views

Solution

1. Query the data dictionary to see information about the tables you own.

```
SELECT table_name
   FROM   user_tables;
```

2. Query the dictionary view to see information about all the tables that you can access. Exclude tables that you own.

```
SELECT table_name, owner
  FROM   all_tables
  WHERE  owner <>'ORAxx';
```

3. For a specified table, create a script that reports the column names, data types, and data types' lengths, as well as whether nulls are allowed. Prompt the user to enter the table name. Give appropriate aliases to the `DATA_PRECISION` and `DATA_SCALE` columns. Save this script in a file named `lab_02_03.sql`.

```
SELECT column_name, data_type, data_length,
       data_precision PRECISION, data_scale SCALE, nullable
FROM   user_tab_columns
WHERE  table_name = UPPER('&tab_name');
```

To test, run the script and enter `DEPARTMENTS` as the table name.

4. Create a script that reports the column name, constraint name, constraint type, search condition, and status for a specified table. You must join the `USER_CONSTRAINTS` and `USER_CONS_COLUMNS` tables to obtain all this information. Prompt the user to enter the table name. Save the script in a file named `lab_02_04.sql`.

```
SELECT ucc.column_name, uc.constraint_name, uc.constraint_type,
       uc.search_condition, uc.status
FROM   user_constraints uc JOIN user_cons_columns ucc
ON     uc.table_name = ucc.table_name
AND    uc.constraint_name = ucc.constraint_name
AND    uc.table_name = UPPER('&tab_name');
```

To test, run the script and enter `DEPARTMENTS` as the table name.

5. Add a comment to the DEPARTMENTS table. Then query the USER_TAB_COMMENTS view to verify that the comment is present.

```
COMMENT ON TABLE departments IS
  'Company department information including name, code, and
location.';

SELECT COMMENTS
FROM   user_tab_comments
WHERE  table_name = 'DEPARTMENTS';
```

6. Run the lab_02_06_tab.sql script as a prerequisite for exercises 16 through 19. Alternatively, open the script file to copy the code and paste it into your SQL Worksheet. Then execute the script. This script:
   - Drops the DEPT2 and EMP2 tables
   - Creates the DEPT2 and EMP2 tables

7. Confirm that both the DEPT2 and EMP2 tables are stored in the data dictionary.

```
SELECT   table_name
FROM     user_tables
WHERE    table_name IN ('DEPT2', 'EMP2');
```

8. Query the data dictionary to find out the constraint names and types for both the tables.

```
SELECT   constraint_name, constraint_type
FROM     user_constraints
WHERE    table_name IN ('EMP2', 'DEPT2');
```

9. Display the object names and types from the USER_OBJECTS data dictionary view for the EMP2 and DEPT2 tables.

```
SELECT   object_name, object_type
FROM     user_objects
WHERE    object_name= 'EMP2'
OR       object_name= 'DEPT2';
```

Practices for Lesson 13: Creating Sequences, Synonyms, and Indexes

Chapter 13

Practices for Lesson 13: Overview

Practices Overview

This practice covers the following topics:

- Creating sequences
- Using sequences
- Querying the dictionary views for sequence information
- Creating synonyms
- Querying the dictionary views for synonyms information
- Creating indexes
- Querying the dictionary views for indexes information

Note: Before starting this practice, execute /home/oracle/sql2/code_ex/code_ex_scripts/clean_up_scripts/cleanup_03.sql script.

Note the following location for the practice files: `/home/oracle/labs/sql2/labs`

Practice 13-1: Creating Sequences, Synonyms, and Indexes

Overview

This practice provides you with a variety of exercises in creating and using a sequence, an index, and a synonym.

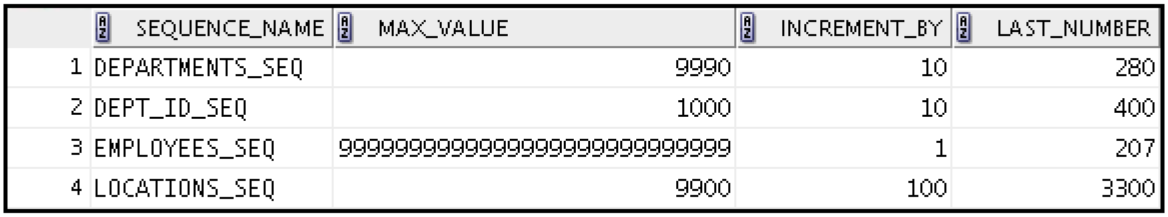
Note: Execute `cleanup_03.sql` script from /home/oracle/sql2/code_ex/code_ex_scripts/clean_up_scripts/ before performing the following tasks.

Tasks

1. Create the `DEPT` table based on the following table instance chart. Confirm that the table is created.

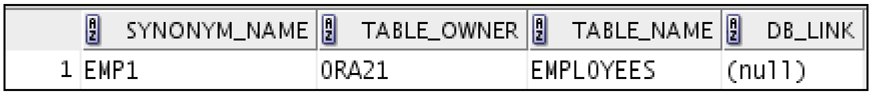
Column Name	`ID`	`NAME`
Key Type	`Primary key`	
Null/Unique		
FK Table		
FK Column		
Data Type	`NUMBER`	`VARCHAR2`
Length	7	25

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`.
3. To test your sequence, write a script to insert two rows in the `DEPT` table. Name your script `lab_03_03.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.
4. Find the names of your sequences. Write a query in a script to display the following information about your sequences: sequence name, maximum value, increment size, and last number. Name the script `lab_03_04.sql`. Run the statement in your script.



	SEQUENCE_NAME	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
1	DEPARTMENTS_SEQ	9990	10	280
2	DEPT_ID_SEQ	1000	10	400
3	EMPLOYEES_SEQ	9999999999999999999999999999	1	207
4	LOCATIONS_SEQ	9900	100	3300

5. Create a synonym for your `EMPLOYEES` table. Call it `EMP1`. Then find the names of all synonyms that are in your schema.

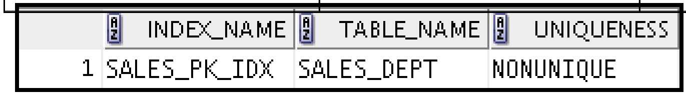


	SYNONYM_NAME	TABLE_OWNER	TABLE_NAME	DB_LINK
1	EMP1	ORA21	EMPLOYEES	(null)

6. Drop the `EMP1` synonym.
7. Create a nonunique index on the `NAME` column in the `DEPT` table.

8. Create the SALES_DEPT table based on the following table instance chart. Name the index for the PRIMARY KEY column SALES_PK_IDX. Then query the data dictionary view to find the index name, table name, and whether the index is unique.

Column Name	Team_Id	Location
Primary Key	Yes	
Data Type	Number	VARCHAR2
Length	3	30



```
     INDEX_NAME     TABLE_NAME     UNIQUENESS
   1 SALES_PK_IDX   SALES_DEPT     NONUNIQUE
```

9. Drop the tables and sequences created in this practice.

Solution 13-1: Creating Sequences, Synonyms, and Indexes

1. Create the DEPT table based on the following table instance chart. Confirm that the table is created.

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

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

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

```
DESCRIBE dept;
```

2. 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;
```

3. To test your sequence, write a script to insert two rows in the DEPT table. Name your script lab_03_03.sql. Be sure to use the sequence that you created for the ID column. Add two departments: Education and Administration. Confirm your additions. Run the commands in your script.

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

4. Find the names of your sequences. Write a query in a script to display the following information about your sequences: sequence name, maximum value, increment size, and last number. Name the script lab_03_04.sql. Run the statement in your script.

```
SELECT   sequence_name, max_value, increment_by, last_number
FROM user_sequences;
```

5. Create a synonym for your EMPLOYEES table. Call it EMP1. Then find the names of all synonyms that are in your schema.

```
CREATE SYNONYM emp1 FOR EMPLOYEES;
SELECT *
FROM   user_synonyms;
```

6. Drop the EMP1 synonym.

```
DROP SYNONYM emp1;
```

7. Create a nonunique index on the NAME column in the DEPT table.

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

8. Create the SALES_DEPT table based on the following table instance chart. Name the index for the PRIMARY KEY column SALES_PK_IDX. Then query the data dictionary view to find the index name, table name, and whether the index is unique.

Column Name	Team_Id	Location
Primary Key	Yes	
Data Type	Number	VARCHAR2
Length	3	30

```
CREATE TABLE SALES_DEPT
       (team_id NUMBER(3)
       PRIMARY KEY USING INDEX
       (CREATE INDEX sales_pk_idx ON
        SALES_DEPT(team_id)),
        location VARCHAR2(30));
SELECT INDEX_NAME, TABLE_NAME, UNIQUENESS
FROM USER_INDEXES
WHERE TABLE_NAME = 'SALES_DEPT';
```

9. Drop the tables and sequences created in this practice.

```
DROP TABLE DEPT;
DROP TABLE SALES_DEPT;
DROP SEQUENCE dept_id_seq;
```

Practices for Lesson 14: Creating Views

Chapter 14

Practices for Lesson 14: Overview

Practices Overview

This practice covers the following topics:

- Creating a simple view
- Creating a complex view
- Creating a view with a check constraint
- Attempting to modify data in the view
- Querying the dictionary views for view information
- Removing views

Note the following location for the practice files: `/home/oracle/labs/sql2/labs`

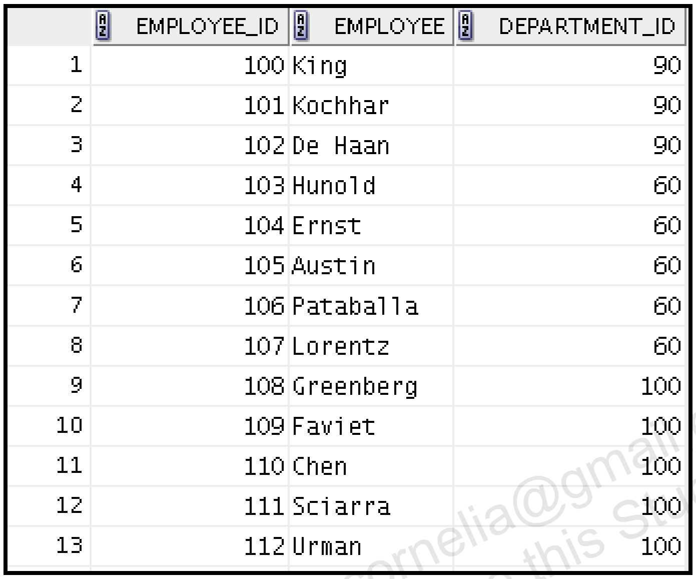
Practice 14-1: Creating Views

Overview:

This lesson's practice provides you with a variety of exercises in creating, using, querying data dictionary views for view information, and removing views.

Tasks:

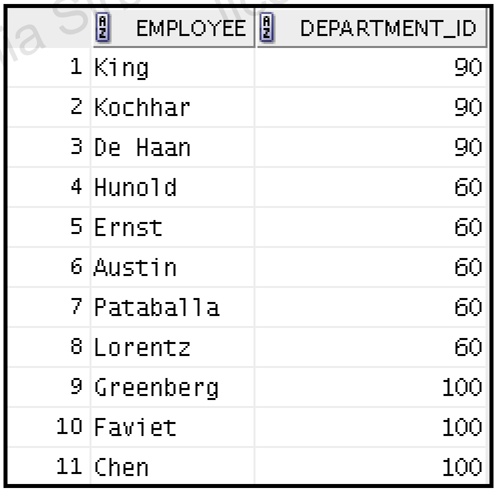
1. The staff in the HR department wants to hide some of the data in the EMPLOYEES table. Create a view called EMPLOYEES_VU based on the employee numbers, employee last names, and department numbers from the EMPLOYEES table. The heading for the employee name should be EMPLOYEE.
2. Confirm that the view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



	EMPLOYEE_ID	EMPLOYEE	DEPARTMENT_ID
1	100	King	90
2	101	Kochhar	90
3	102	De Haan	90
4	103	Hunold	60
5	104	Ernst	60
6	105	Austin	60
7	106	Pataballa	60
8	107	Lorentz	60
9	108	Greenberg	100
10	109	Faviet	100
11	110	Chen	100
12	111	Sciarra	100
13	112	Urman	100

…

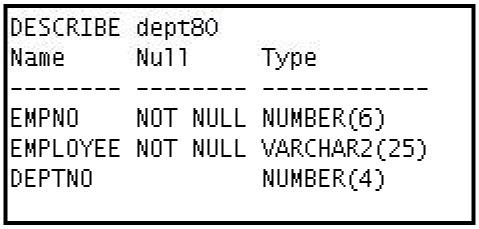
3. Using your EMPLOYEES_VU view, write a query for the HR department to display all employee names and department numbers.



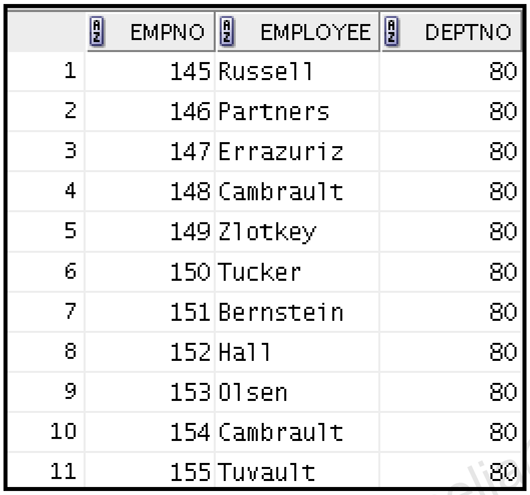
	EMPLOYEE	DEPARTMENT_ID
1	King	90
2	Kochhar	90
3	De Haan	90
4	Hunold	60
5	Ernst	60
6	Austin	60
7	Pataballa	60
8	Lorentz	60
9	Greenberg	100
10	Faviet	100
11	Chen	100

…

4. Department 80 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 80. 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 `DEPT80`view.



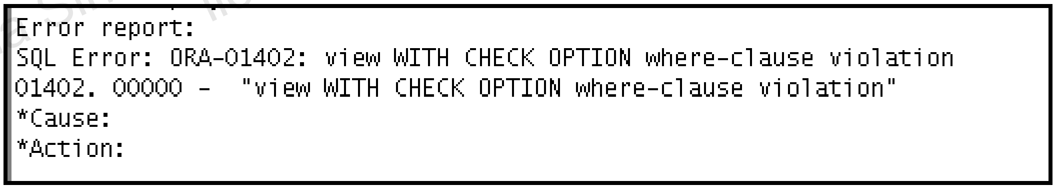
```
DESCRIBE dept80
Name     Null     Type
-------- -------- ------------
EMPNO    NOT NULL NUMBER(6)
EMPLOYEE NOT NULL VARCHAR2(25)
DEPTNO            NUMBER(4)
```



	EMPNO	EMPLOYEE	DEPTNO
1	145	Russell	80
2	146	Partners	80
3	147	Errazuriz	80
4	148	Cambrault	80
5	149	Zlotkey	80
6	150	Tucker	80
7	151	Bernstein	80
8	152	Hall	80
9	153	Olsen	80
10	154	Cambrault	80
11	155	Tuvault	80

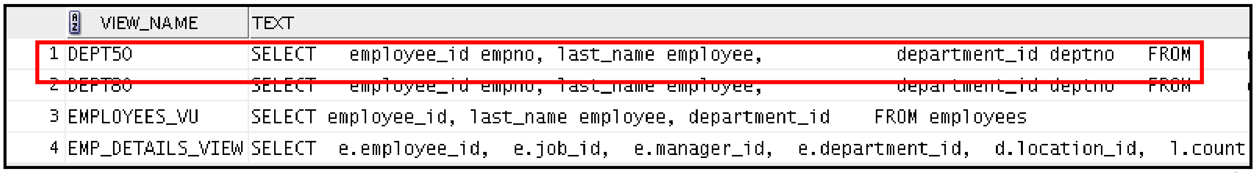
...

6. Test your view. Attempt to reassign Abel to department 80.



```
Error report:
SQL Error: ORA-01402: view WITH CHECK OPTION where-clause violation
01402. 00000 -  "view WITH CHECK OPTION where-clause violation"
*Cause:
*Action:
```

7. Run `lab_04_07.sql` to create the `dept50` view for this exercise.
   You need to determine the names and definitions of all the views in your schema.
   Create a report that retrieves view information: the view name and text from the `USER_VIEWS` data dictionary view.
   Note: The `EMP_DETAILS_VIEW` was created as part of your schema.
   Note: You can see the complete definition of the view if you use Run Script (or press F5) in SQL Developer. If you use Execute Statement (or press F9) in SQL Developer, scroll horizontally in the result pane. If you use SQL*Plus, to see more contents of a `LONG` column, use the `SET LONG` *n* command, where *n* is the value of the number of characters of the `LONG` column that you want to see.



	VIEW_NAME	TEXT
1	DEPT50	SELECT employee_id empno, last_name employee, department_id deptno FROM
2	DEPT80	SELECT employee_id empno, last_name employee, department_id deptno FROM
3	EMPLOYEES_VU	SELECT employee_id, last_name employee, department_id FROM employees
4	EMP_DETAILS_VIEW	SELECT e.employee_id, e.job_id, e.manager_id, e.department_id, d.location_id, l.count

8. Remove the views created in this practice.

Solution 14-1: Creating Views

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 80 needs access to its employee data. Create a view named DEPT80 that contains the employee numbers, employee last names, and department numbers for all employees in department 80. 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 dept80 AS
   SELECT   employee_id empno, last_name employee,
            department_id deptno
   FROM     employees
   WHERE    department_id = 80
   WITH CHECK OPTION CONSTRAINT emp_dept_80;
```

5. Display the structure and contents of the DEPT80 view.

```
DESCRIBE dept80

SELECT   *
FROM     dept80;
```

6. Test your view. Attempt to reassign Abel to department 50.

```
UPDATE   dept80
SET      deptno = 50
WHERE    employee = 'Abel';
```

   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.

7. Run lab_04_07.sql to create the dept50 view for this exercise. You need to determine the names and definitions of all the views in your schema. Create a report that retrieves view information: the view name and text from the USER_VIEWS data dictionary view.

   Note: The EMP_DETAILS_VIEW was created as part of your schema.

   Note: You can see the complete definition of the view if you use Run Script (or press F5) in SQL Developer. If you use Execute Statement (or press F9) in SQL Developer, scroll horizontally in the result pane. If you use SQL*Plus to see more contents of a LONG column, use the SET LONG *n* command, where *n* is the value of the number of characters of the LONG column that you want to see.

```
SELECT   view_name, text
FROM     user_views;
```

8. Remove the views created in this practice.

```
DROP VIEW employees_vu;
DROP VIEW dept80;
DROP VIEW dept50;
```

Practices for Lesson 15: Managing Schema Objects

Chapter 15

Practices for Lesson 15: Overview

Practice Overview

This practice covers the following topics:

- Adding and dropping constraints
- Deferring constraints
- Creating external tables

Note: Before starting this practice, execute **`/home/oracle/labs/sql2/code_ex/ /cleanup_scripts/cleanup_05.sql`** script.

Note the following location for the practice files: `/home/oracle/labs/sql2/labs`

Practice 15-1: Managing Schema Objects

Overview

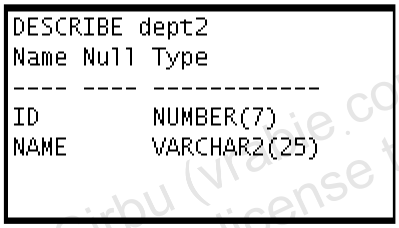
In this practice, you add, drop, and defer constraints. You create external tables.

Note: Execute `cleanup_05.sql` script from **`/home/oracle/labs/sql2/code_ex/`** **`/cleanup_scripts/`** before performing the following tasks.

Tasks

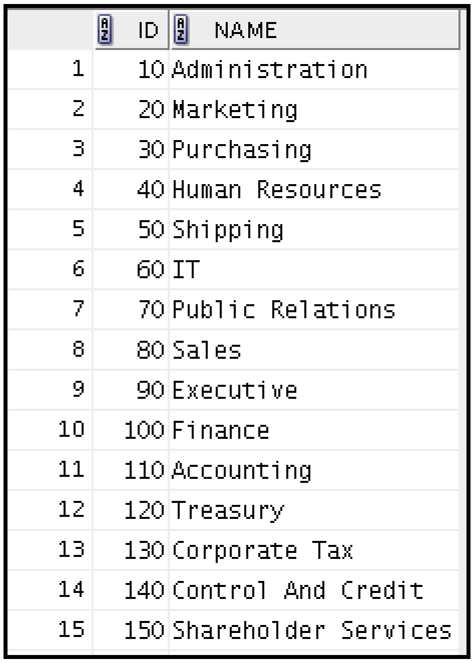
1. Create the `DEPT2` table based on the following table instance chart. Enter the syntax in the SQL Worksheet. Then, execute the statement to create the table. Confirm that the table is created.

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



```
DESCRIBE dept2
Name Null Type
---- ---- ------------
ID        NUMBER(7)
NAME      VARCHAR2(25)
```

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

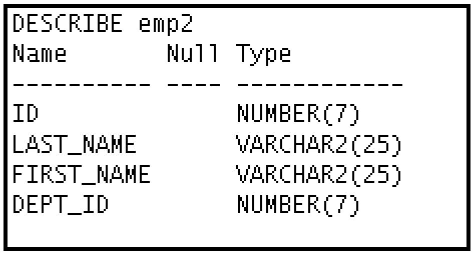


	ID	NAME
1	10	Administration
2	20	Marketing
3	30	Purchasing
4	40	Human Resources
5	50	Shipping
6	60	IT
7	70	Public Relations
8	80	Sales
9	90	Executive
10	100	Finance
11	110	Accounting
12	120	Treasury
13	130	Corporate Tax
14	140	Control And Credit
15	150	Shareholder Services

...

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

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



```
DESCRIBE emp2
Name       Null Type
---------- ---- ------------
ID              NUMBER(7)
LAST_NAME       VARCHAR2(25)
FIRST_NAME      VARCHAR2(25)
DEPT_ID         NUMBER(7)
```

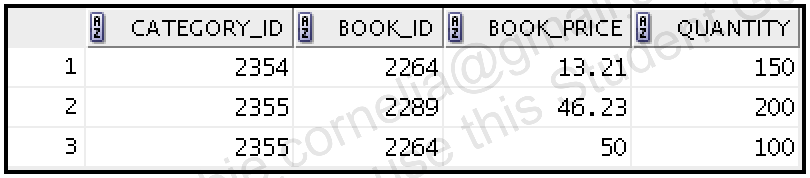
4. Add a table-level PRIMARY KEY constraint to the EMP2 table on the ID column. The constraint should be named at creation. Name the constraint my_emp_id_pk.
5. Create a PRIMARY KEY constraint to the DEPT2 table using the ID column. The constraint should be named at creation. Name the constraint my_dept_id_pk.
6. Add a foreign key reference on the EMP2 table that ensures that the employee is not assigned to a nonexistent department. Name the constraint my_emp_dept_id_fk.
7. Modify the EMP2 table. Add a COMMISSION column of the NUMBER data type, precision 2, scale 2. Add a constraint to the COMMISSION column that ensures that a commission value is greater than zero.
8. Drop the EMP2 and DEPT2 tables so that they cannot be restored.
9. Create an external table library_items_ext. Use the ORACLE_LOADER access driver.

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

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

   a. Open the lab_05_09.sql file. Observe the code snippet to create the library_items_ext external table. Then replace **<*TODO1*>**, **<*TODO2*>**, **<*TODO3*>**, and <*TODO4*> as appropriate and save the file as lab_05_09_soln.sql. Run the script to create the external table.

   b. Query the library_items_ext table.



	CATEGORY_ID	BOOK_ID	BOOK_PRICE	QUANTITY
1	2354	2264	13.21	150
2	2355	2289	46.23	200
3	2355	2264	50	100

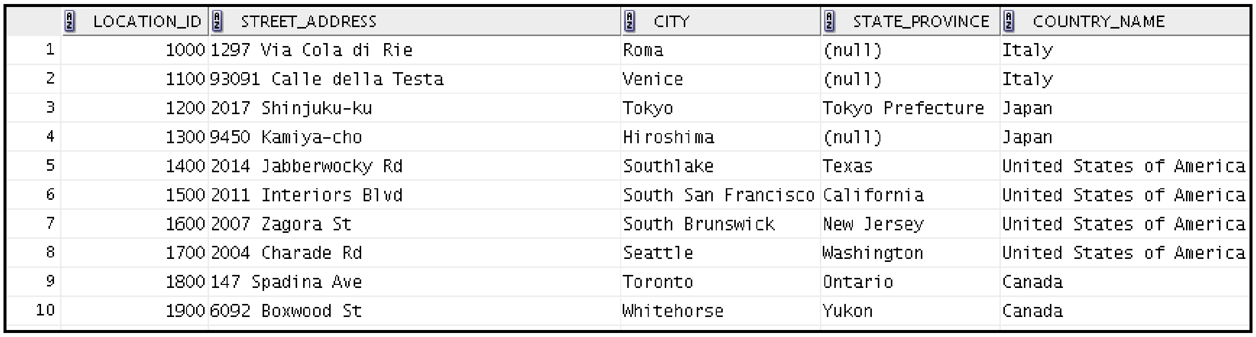
10. The HR department needs a report of the addresses of all departments. Create an external table as dept_add_ext using the ORACLE_DATAPUMP access driver. The report should show the location ID, street address, city, state or province, and country in the output. Use a NATURAL JOIN to produce the results.

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

    a. Open the lab_05_10.sql file. Observe the code snippet to create the dept_add_ext external table. Then, replace <*TODO1*>, <*TODO2*>, and <*TODO3*> with the appropriate code. Replace <*oraxx_emp4.exp*> and <*oraxx_emp5.exp*> with the appropriate file names. For example, if you are the ora21 user, your file names are ora21_emp4.exp and ora21_emp5.exp. Save the script as lab_05_10_soln.sql.

    b. Run the lab_05_10_soln.sql script to create the external table.

c. Query the dept_add_ext table.

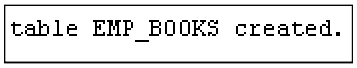


	LOCATION_ID	STREET_ADDRESS	CITY	STATE_PROVINCE	COUNTRY_NAME
1	1000	1297 Via Cola di Rie	Roma	(null)	Italy
2	1100	93091 Calle della Testa	Venice	(null)	Italy
3	1200	2017 Shinjuku-ku	Tokyo	Tokyo Prefecture	Japan
4	1300	9450 Kamiya-cho	Hiroshima	(null)	Japan
5	1400	2014 Jabberwocky Rd	Southlake	Texas	United States of America
6	1500	2011 Interiors Blvd	South San Francisco	California	United States of America
7	1600	2007 Zagora St	South Brunswick	New Jersey	United States of America
8	1700	2004 Charade Rd	Seattle	Washington	United States of America
9	1800	147 Spadina Ave	Toronto	Ontario	Canada
10	1900	6092 Boxwood St	Whitehorse	Yukon	Canada

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

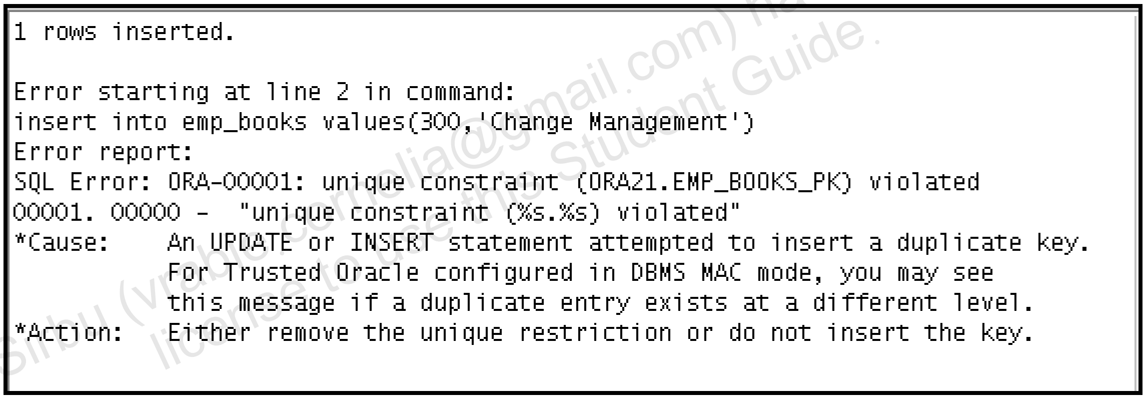
11. Create the emp_books table and populate it with data. Set the primary key as deferred and observe what happens at the end of the transaction.

a. Run the lab_05_11_a.sql file to create the emp_books table. Observe that the emp_books_pk primary key is not created as deferrable.



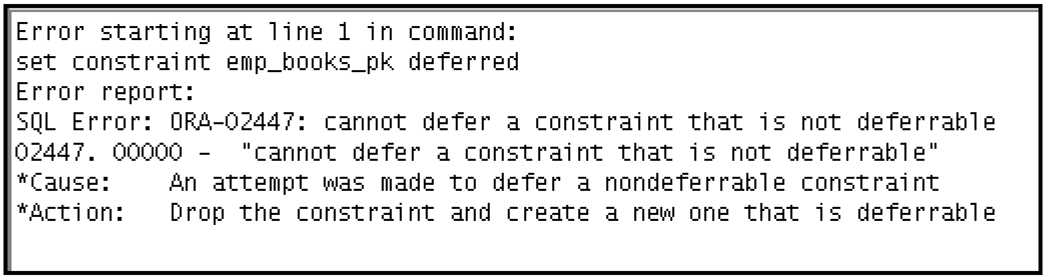
```
table EMP_BOOKS created.
```

b. Run the lab_05_11_b.sql file to populate data into the emp_books table. What do you observe?



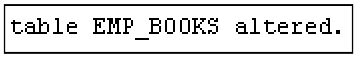
```
1 rows inserted.

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

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



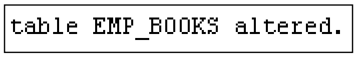
```
Error starting at line 1 in command:
set constraint emp_books_pk deferred
Error report:
SQL Error: ORA-02447: cannot defer a constraint that is not deferrable
02447. 00000 -  "cannot defer a constraint that is not deferrable"
*Cause:    An attempt was made to defer a nondeferrable constraint
*Action: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

d. Drop the emp_books_pk constraint.



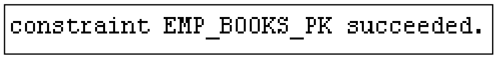
```
table EMP_BOOKS altered.
```

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



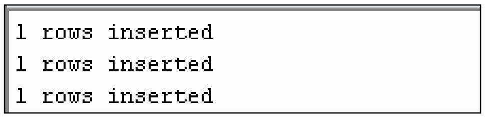
```
table EMP_BOOKS altered.
```

f. Set the `emp_books_pk` constraint as deferred.



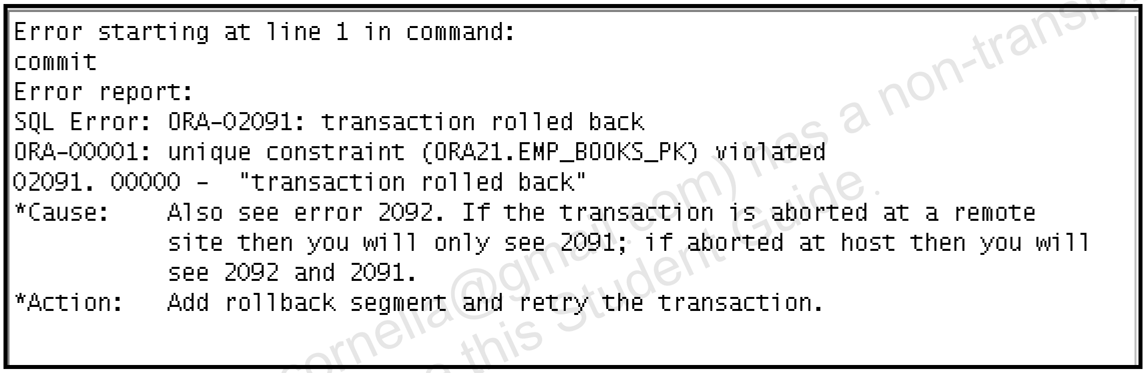
```
constraint EMP_BOOKS_PK succeeded.
```

g. Run the `lab_05_11_g.sql` file to populate data into the `emp_books` table. What do you observe?



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

h. Commit the transaction. What do you observe?



```
Error starting at line 1 in command:
commit
Error report:
SQL Error: ORA-02091: transaction rolled back
ORA-00001: unique constraint (ORA21.EMP_BOOKS_PK) violated
02091. 00000 -  "transaction rolled back"
*Cause:    Also see error 2092. If the transaction is aborted at a remote
           site then you will only see 2091; if aborted at host then you will
           see 2092 and 2091.
*Action:   Add rollback segment and retry the transaction.
```

Solution 15-1: Managing Schema Objects

Solution

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

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

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

DESCRIBE dept2
```

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

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

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

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

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

DESCRIBE emp2
```

4. Add a table-level PRIMARY KEY constraint to the EMP2 table on the ID column. The constraint should be named at creation. Name the constraint my_emp_id_pk.

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

5. Create a PRIMARY KEY constraint to the DEPT2 table using the ID column. The constraint should be named at creation. Name the constraint my_dept_id_pk.

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

6. Add a foreign key reference on the EMP2 table that ensures that the employee is not assigned to a nonexistent department. Name the constraint my_emp_dept_id_fk.

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

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

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

8. Drop the EMP2 and DEPT2 tables so that they cannot be restored.

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

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

   Note: The `emp_dir` directory and `library_items.dat` are already created for this exercise. Ensure that the external file and the database are on the same machine.

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

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

   a. Open the `lab_05_09.sql` file. Observe the code snippet to create the `library_items_ext` external table. Then, replace ***<TODO1>***, ***<TODO2>***, ***<TODO3>***, and <***TODO4***> as appropriate and save the file as `lab_05_09_soln.sql`. Run the script to create the external table.

   ```
   CREATE TABLE library_items_ext ( category_id   number(12)
                                  , book_id number(6)
                                  , book_price number(8,2)
                                  , quantity   number(8)
                                  )
   ORGANIZATION EXTERNAL
    (TYPE ORACLE_LOADER
     DEFAULT DIRECTORY emp_dir
     ACCESS PARAMETERS (RECORDS DELIMITED BY NEWLINE
                        FIELDS TERMINATED BY ',')
    LOCATION ('library_items.dat')
    )
   REJECT LIMIT UNLIMITED;
   ```

   b. Query the `library_items_ext` table.

   ```
   SELECT * FROM library_items_ext;
   ```

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

    Note: The `emp_dir` directory is already created for this exercise. Ensure that the external file and the database are on the same machine.

a. Open the lab_05_10.sql file. Observe the code snippet to create the dept_add_ext external table. Then, replace <*TODO1*>, <*TODO2*>, and <*TODO3*> with appropriate code. Replace <*oraxx_emp4.exp*> and <*oraxx_emp5.exp*> with appropriate file names. For example, if you are user ora21, your file names are ora21_emp4.exp and ora21_emp5.exp. Save the script as lab_5_10_soln.sql.

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

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

b. Run the lab_05_10_soln.sql script to create the external table.

c. Query the dept_add_ext table.

```
SELECT * FROM dept_add_ext;
```

11. Create the emp_books table and populate it with data. Set the primary key as deferred and observe what happens at the end of the transaction.

a. Run the lab_05_11_a.sql script to create the emp_books table. Observe that the emp_books_pk primary key is not created as deferrable.

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

b. Run the lab_05_11_b.sql script to populate data into the emp_books table. What do you observe?

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

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

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

```
SET CONSTRAINT emp_books_pk DEFERRED;
```

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

d. Drop the emp_books_pk constraint.

```
ALTER TABLE emp_books DROP CONSTRAINT emp_books_pk;
```

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

```
ALTER TABLE emp_books ADD (CONSTRAINT emp_books_pk PRIMARY KEY
(book_id) DEFERRABLE);
```

f. Set the emp_books_pk constraint as deferred.

```
SET CONSTRAINT emp_books_pk DEFERRED;
```

g. Run the lab_05_11_g.sql script to populate data into the emp_books table. What do you observe?

```
INSERT INTO emp_books VALUES (300,'Change Management');
INSERT INTO emp_books VALUES (300,'Personality');
INSERT INTO emp_books VALUES (350,'Creativity');
```

You see that all the rows are inserted.

h. Commit the transaction. What do you observe?

```
COMMIT;
```

You see that the transaction is rolled back by the database at this point, because the COMMIT failed due to the constraint violation.

Practices for Lesson 16: Retrieving Data by Using Subqueries

Chapter 16

Practices for Lesson 16: Overview

Practice Overview

This practice covers the following topics:

- Creating multiple-column subqueries
- Writing correlated subqueries
- Using the `EXISTS` operator
- Using scalar subqueries
- Using the `WITH` clause

Note the following location for the practice files: `/home/oracle/labs/sql2/labs`

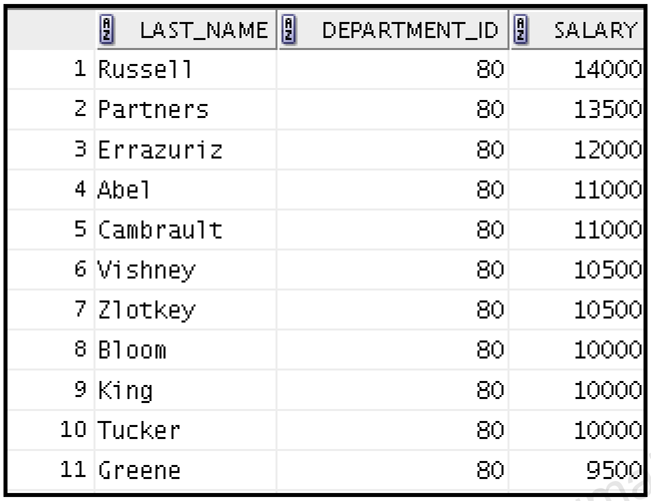
Practice 16-1: Retrieving Data by Using Subqueries

Overview

In this practice, you write multiple-column subqueries, and correlated and scalar subqueries. You also solve problems by writing the WITH clause.

Tasks

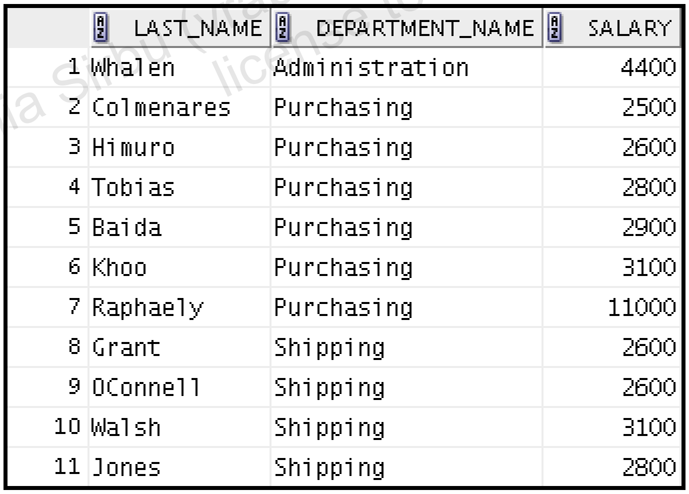
1. Write a query to display the last name, department number, and salary of any employee whose department number and salary both match the department number and salary of any employee who earns a commission.



	LAST_NAME	DEPARTMENT_ID	SALARY
1	Russell	80	14000
2	Partners	80	13500
3	Errazuriz	80	12000
4	Abel	80	11000
5	Cambrault	80	11000
6	Vishney	80	10500
7	Zlotkey	80	10500
8	Bloom	80	10000
9	King	80	10000
10	Tucker	80	10000
11	Greene	80	9500

...

2. Display the last name, department name, and salary of any employee whose salary and job_ID match the salary and job_ID of any employee located in location ID 1700.

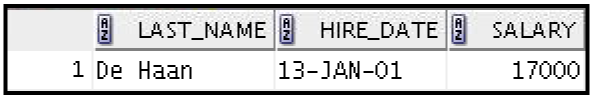


	LAST_NAME	DEPARTMENT_NAME	SALARY
1	Whalen	Administration	4400
2	Colmenares	Purchasing	2500
3	Himuro	Purchasing	2600
4	Tobias	Purchasing	2800
5	Baida	Purchasing	2900
6	Khoo	Purchasing	3100
7	Raphaely	Purchasing	11000
8	Grant	Shipping	2600
9	OConnell	Shipping	2600
10	Walsh	Shipping	3100
11	Jones	Shipping	2800

...

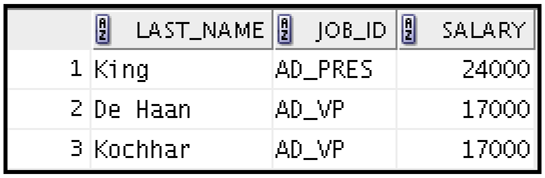
3. Create a query to display the last name, hire date, and salary for all employees who have the same salary and `manager_ID` as Kochhar.

   Note: Do not display Kochhar in the result set.



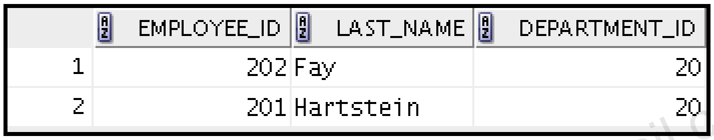
	LAST_NAME	HIRE_DATE	SALARY
1	De Haan	13-JAN-01	17000

4. Create a query to display the employees who earn a salary that is higher than the salary of all the sales managers (`JOB_ID = 'SA_MAN'`). Sort the results from the highest to the lowest.



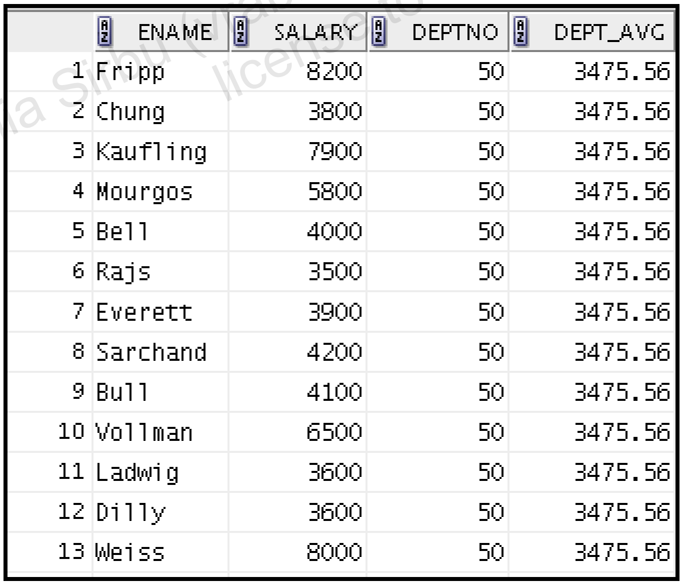
	LAST_NAME	JOB_ID	SALARY
1	King	AD_PRES	24000
2	De Haan	AD_VP	17000
3	Kochhar	AD_VP	17000

5. Display details such as the employee `ID`, last name, and department `ID` of those employees who live in cities the names of which begin with *T*.



	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	202	Fay	20
2	201	Hartstein	20

6. Write a query to find all employees who earn more than the average salary in their departments. Display last name, salary, department `ID`, and the average salary for the department. Sort by average salary and round to two decimals. Use aliases for the columns retrieved by the query as shown in the sample output.

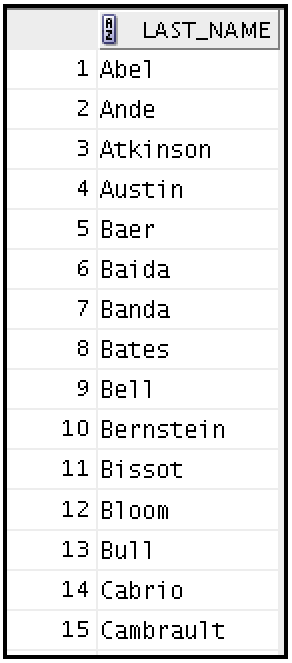


	ENAME	SALARY	DEPTNO	DEPT_AVG
1	Fripp	8200	50	3475.56
2	Chung	3800	50	3475.56
3	Kaufling	7900	50	3475.56
4	Mourgos	5800	50	3475.56
5	Bell	4000	50	3475.56
6	Rajs	3500	50	3475.56
7	Everett	3900	50	3475.56
8	Sarchand	4200	50	3475.56
9	Bull	4100	50	3475.56
10	Vollman	6500	50	3475.56
11	Ladwig	3600	50	3475.56
12	Dilly	3600	50	3475.56
13	Weiss	8000	50	3475.56

…

7. Find all employees who are not supervisors.

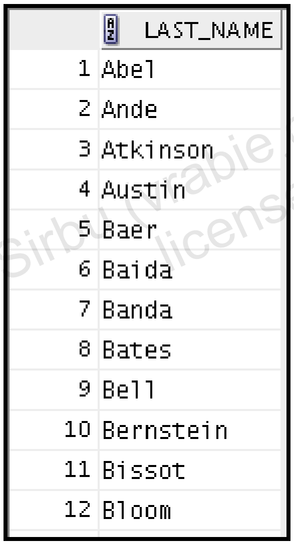
   a. First, do this by using the `NOT EXISTS` operator.



	LAST_NAME
1	Abel
2	Ande
3	Atkinson
4	Austin
5	Baer
6	Baida
7	Banda
8	Bates
9	Bell
10	Bernstein
11	Bissot
12	Bloom
13	Bull
14	Cabrio
15	Cambrault

…

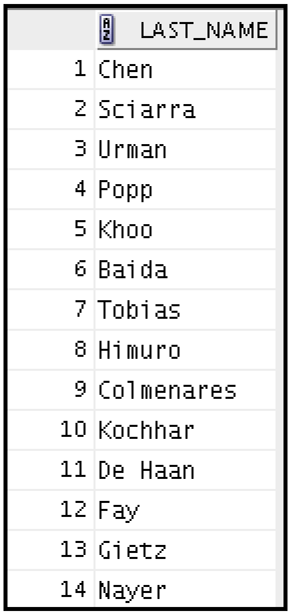
   b. Can this be done by using the `NOT IN` operator? How, or why not? If not, try out using another solution.



	LAST_NAME
1	Abel
2	Ande
3	Atkinson
4	Austin
5	Baer
6	Baida
7	Banda
8	Bates
9	Bell
10	Bernstein
11	Bissot
12	Bloom

…

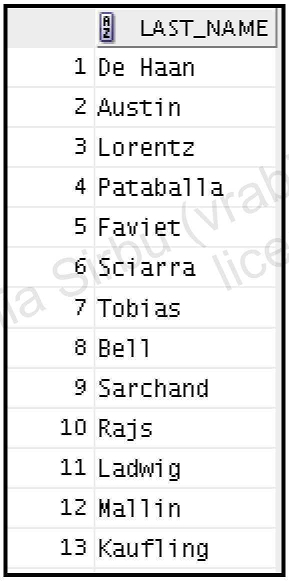
8. Write a query to display the last names of the employees who earn less than the average salary in their departments.



	LAST_NAME
1	Chen
2	Sciarra
3	Urman
4	Popp
5	Khoo
6	Baida
7	Tobias
8	Himuro
9	Colmenares
10	Kochhar
11	De Haan
12	Fay
13	Gietz
14	Nayer

...

9. Write a query to display the last names of the employees who have one or more coworkers in their departments with later hire dates but higher salaries.

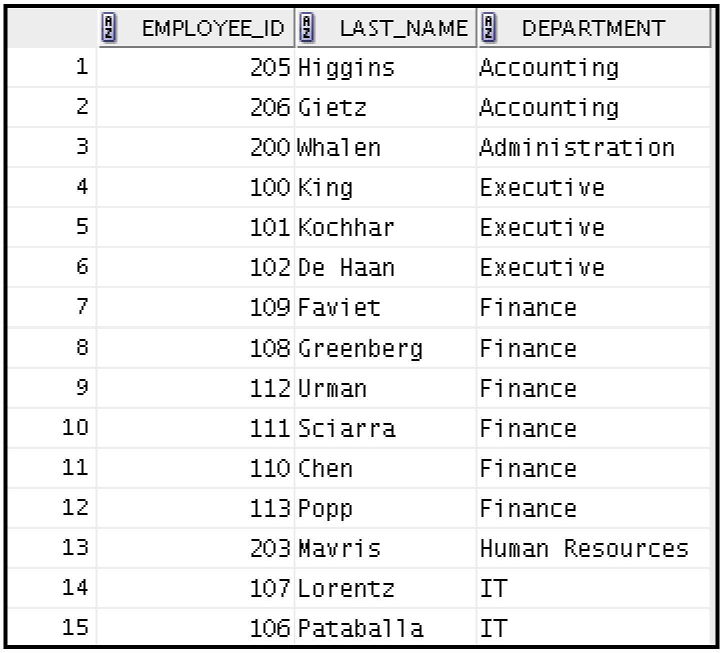


	LAST_NAME
1	De Haan
2	Austin
3	Lorentz
4	Pataballa
5	Faviet
6	Sciarra
7	Tobias
8	Bell
9	Sarchand
10	Rajs
11	Ladwig
12	Mallin
13	Kaufling

...

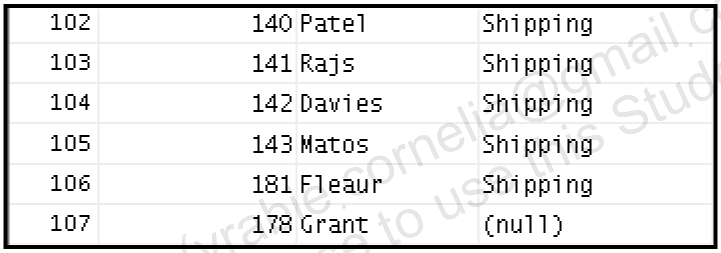
10. Write a query to display the employee ID, last names, and department names of all the employees.

    Note: Use a scalar subquery to retrieve the department name in the SELECT statement.



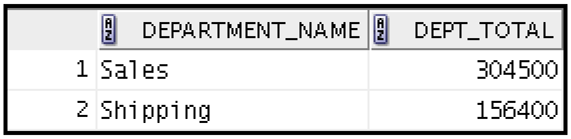
	EMPLOYEE_ID	LAST_NAME	DEPARTMENT
1	205	Higgins	Accounting
2	206	Gietz	Accounting
3	200	Whalen	Administration
4	100	King	Executive
5	101	Kochhar	Executive
6	102	De Haan	Executive
7	109	Faviet	Finance
8	108	Greenberg	Finance
9	112	Urman	Finance
10	111	Sciarra	Finance
11	110	Chen	Finance
12	113	Popp	Finance
13	203	Mavris	Human Resources
14	107	Lorentz	IT
15	106	Pataballa	IT

...



102	140	Patel	Shipping
103	141	Rajs	Shipping
104	142	Davies	Shipping
105	143	Matos	Shipping
106	181	Fleaur	Shipping
107	178	Grant	(null)

11. Write a query to display the department names of those departments whose total salary cost is above one-eighth (1/8) of the total salary cost of the whole company. Use the WITH clause to write this query. Name the query SUMMARY.



	DEPARTMENT_NAME	DEPT_TOTAL
1	Sales	304500
2	Shipping	156400

Solution 16-1: Retrieving Data by Using Subqueries

Solution

1. Write a query to display the last name, department number, and salary of any employee whose department number and salary match the department number and salary of any employee who earns a commission.

```
     SELECT last_name, department_id, salary
     FROM   employees
     WHERE  (salary, department_id) IN
               (SELECT  salary, department_id
                FROM    employees
                WHERE   commission_pct IS NOT NULL);
```

2. Display the last name, department name, and salary of any employee whose salary and `job_ID` match the salary and `job_ID` of any employee located in location `ID` 1700.

```
SELECT e.last_name, d.department_name, e.salary
FROM   employees e JOIN departments d
ON  e.department_id = d.department_id
AND  (salary, job_id) IN
                          (SELECT e.salary, e.job_id
                             FROM    employees e JOIN
departments d
                          ON   e.department_id =
d.department_id
                          AND  d.location_id = 1700);
```

3. Create a query to display the last name, hire date, and salary for all employees who have the same salary and `manager_ID` as Kochhar.
   Note: Do not display Kochhar in the result set.

```
SELECT last_name, hire_date, salary
FROM   employees
WHERE  (salary, manager_id) IN
             (SELECT salary, manager_id
              FROM     employees
              WHERE  last_name = 'Kochhar')
AND last_name != 'Kochhar';
```

4. Create a query to display the employees who earn a salary that is higher than the salary of all the sales managers (`JOB_ID = 'SA_MAN'`). Sort the results on salary from the highest to the lowest.

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  salary > ALL
             (SELECT salary
              FROM   employees
WHERE  job_id = 'SA_MAN')
ORDER BY salary DESC;
```

5. Display details such as the employee `ID`, last name, and department `ID` of those employees who live in cities the names of which begin with *T*.

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  department_id IN (SELECT department_id
                         FROM departments
                         WHERE location_id IN
                              (SELECT  location_id
                               FROM locations
                               WHERE city LIKE 'T%'));
```

6. Write a query to find all employees who earn more than the average salary in their departments. Display last name, salary, department `ID`, and the average salary for the department. Sort by average salary and round to two decimals. Use aliases for the columns retrieved by the query as shown in the sample output.

```
     SELECT e.last_name ename, e.salary salary,
             e.department_id deptno, ROUND(AVG(a.salary),2)
dept_avg
       FROM   employees e, employees a
       WHERE  e.department_id = a.department_id
       AND    e.salary > (SELECT AVG(salary)
                    FROM   employees
                    WHERE  department_id = e.department_id )
       GROUP BY e.last_name, e.salary, e.department_id
       ORDER BY AVG(a.salary);
```

7. Find all employees who are not supervisors.
   a. First, do this by using the NOT EXISTS operator.

```
   SELECT outer.last_name
   FROM    employees outer
   WHERE  NOT EXISTS (SELECT 'X'
                      FROM employees inner
                       WHERE inner.manager_id =
                             outer.employee_id);
```

   b. Can this be done by using the NOT IN operator? How, or why not?

```
SELECT outer.last_name
FROM   employees outer
WHERE  outer.employee_id
NOT IN (SELECT inner.manager_id
        FROM   employees inner);
```

This alternative solution is not a good one. The subquery picks up a NULL value, so the entire query returns no rows. The reason is that all conditions that compare a NULL value result in NULL. Whenever NULL values are likely to be part of the value set, *do not* use NOT IN as a substitute for NOT EXISTS. A much better solution would be a subquery like the following:

```
SELECT last_name
FROM employees
WHERE employee_id NOT IN (SELECT manager_id
                          FROM employees WHERE manager_id IS NOT
NULL);
```

8. Write a query to display the last names of the employees who earn less than the average salary in their departments.

```
         SELECT last_name
         FROM    employees outer
         WHERE outer.salary < (SELECT AVG(inner.salary)
                               FROM employees inner
                               WHERE inner.department_id
                                     = outer.department_id);
```

9. Write a query to display the last names of employees who have one or more coworkers in their departments with later hire dates but higher salaries.

```
        SELECT  last_name
         FROM    employees outer
         WHERE EXISTS (SELECT 'X'
                       FROM employees inner
                       WHERE inner.department_id =
                             outer.department_id
                       AND inner.hire_date > outer.hire_date
                       AND inner.salary > outer.salary);
```

10. Write a query to display the employee `ID`, last names, and department names of all employees.

    Note: Use a scalar subquery to retrieve the department name in the `SELECT` statement.

```
        SELECT employee_id, last_name,
              (SELECT department_name
               FROM departments d
              WHERE   e.department_id =
                  d.department_id ) department
      FROM employees e
      ORDER BY department;
```

11. Write a query to display the department names of those departments whose total salary cost is above one-eighth (1/8) of the total salary cost of the whole company. Use the `WITH` clause to write this query. Name the query `SUMMARY`.

```
WITH
summary AS (
  SELECT d.department_name, SUM(e.salary) AS dept_total
  FROM employees e JOIN departments d
  ON e.department_id = d.department_id
  GROUP BY d.department_name)
SELECT department_name, dept_total
FROM summary
WHERE dept_total > ( SELECT SUM(dept_total) * 1/8
                    FROM summary  )
ORDER BY dept_total DESC;
```

Practices for Lesson 17: Manipulating Data by Using Subqueries

Chapter 17

Practices for Lesson 17: Overview

Practices Overview

This practice covers the following topics:

- Using subqueries to manipulate data
- Inserting by using a subquery as a target
- Using the `WITH CHECK OPTION` keyword on DML statements
- Using correlated subqueries to update and delete rows

Note the following location for the practice files: `/home/oracle/labs/sql2/labs`

Practice 17-1: Manipulating Data by Using Subqueries

Overview

In this practice, you test your knowledge about using subqueries to manipulate data, using the `WITH CHECK OPTION` keyword on DML statements, and correlated subqueries to update and delete rows.

Tasks

1. Which of the following statements are true?
   a. Subqueries are used to retrieve data by using an inline view.
   b. Subqueries cannot be used to copy data from one table to another.
   c. Subqueries update data in one table based on the values of another table.
   d. Subqueries delete rows from one table based on rows in another table.

2. Fill in the blanks:
   a. You can use a subquery in place of the table name in the ______ clause of the `INSERT` statement.

   Options:
   1) `FROM`
   2) `INTO`
   3) `FOR UPDATE`
   4) `VALUES`

3. The `WITH CHECK OPTION` keyword prohibits you from changing rows that are not in the subquery.
   a. `TRUE`
   b. `FALSE`

4. The `SELECT` list of this subquery must have the same number of columns as the column list of the `VALUES` clause.
   a. `TRUE`
   b. `FALSE`

5. You can use a correlated subquery to delete only those rows that also exist in another table.
   a. `TRUE`
   b. `FALSE`

6. To understand the concepts of `WITH CHECK OPTION` and correlated subqueries, run the demo files for this practice.

Solution 17-1: Manipulating Data by Using Subqueries

1. Which of the following statements are true?
   a. Subqueries are used to retrieve data by using an inline view.
   b. Subqueries cannot be used to copy data from one table to another.
   c. Subqueries update data in one table based on the values of another table.
   d. Subqueries delete rows from one table based on rows in another table.

Answer: a, c, and d

2. Fill in the blanks:
   a. You can use a subquery in place of the table name in the ______ clause of the `INSERT` statement.

Options:

1) `FROM`
2) `INTO`
3) `FOR UPDATE`
4) `VALUES`

Answer: 2

3. The `WITH CHECK OPTION` keyword prohibits you from changing rows that are not in the subquery.
   a. `TRUE`
   b. `FALSE`

Answer: a

4. The `SELECT` list of this subquery must have the same number of columns as the column list of the `VALUES` clause.
   a. `TRUE`
   b. `FALSE`

Answer: a

5. You can use a correlated subquery to delete only those rows that also exist in another table.
   a. `TRUE`
   b. `FALSE`

Answer: a

6. To understand the concepts of `WITH CHECK OPTION` and correlated subqueries, run the demo files for this practice.

Practices for Lesson 18: Controlling User Access

Chapter 18

Practices for Lesson 18: Overview

Practice Overview:

This practice covers the following topics:

- Granting other users privileges to your table
- Modifying another user’s table through the privileges granted to you

Note the following location for the practice files: `/home/oracle/labs/sql2/labs`

Practice 18-1: Controlling User Access

Overview

You grant query privilege on your table to another user. You learn how to control access to database objects.

Tasks

1. What privilege should a user be given to log on to the Oracle server? Is this a system privilege or an object privilege?

   ____

2. What privilege should a user be given to create tables?

   ____

3. If you create a table, who can pass along privileges to other users in your table?

   ____

4. You are the DBA. You create many users who require the same system privileges. What should you use to make your job easier?

   ____

5. What command do you use to change your password?

   ____

6. `User21` is the owner of the `EMP` table and grants the `DELETE` privilege to `User22` by using the `WITH GRANT OPTION` clause. `User22` then grants the `DELETE` privilege on `EMP` to `User23.` `User21` now finds that `User23` has the privilege and revokes it from `User22`. Which user can now delete from the `EMP` table?

   ____

7. You want to grant `SCOTT` the privilege to update data in the `DEPARTMENTS` table. You also want to enable `SCOTT` to grant this privilege to other users. What command do you use?

   ____

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

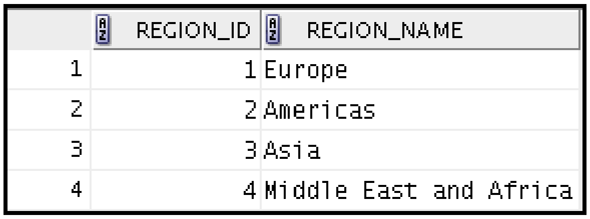
1. **Click the SQL Developer desktop icon.**
2. **In the Connections Navigator, use the *`ora21`* account and the corresponding password provided by your instructor to log on to the database.**
3. **Open another SQL Developer session and connect as *`ora22`*.**

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

   Note: For this exercise, open another SQL Developer session and connect as a different user. For example, if you are currently using `ora21`, open another SQL Developer session and connect as `ora22`. Here onwards we would refer the first SQL Developer session as Team 1 and the second SQL Developer session as Team 2.

   a. Grant another user (for example, `ora22`) privilege to view records in your `REGIONS` table. Include an option for this user to further grant this privilege to other users.

b. Have the user query your REGIONS table.



	REGION_ID	REGION_NAME
1	1	Europe
2	2	Americas
3	3	Asia
4	4	Middle East and Africa

c. Have the user pass on the query privilege to a third user, ora23.

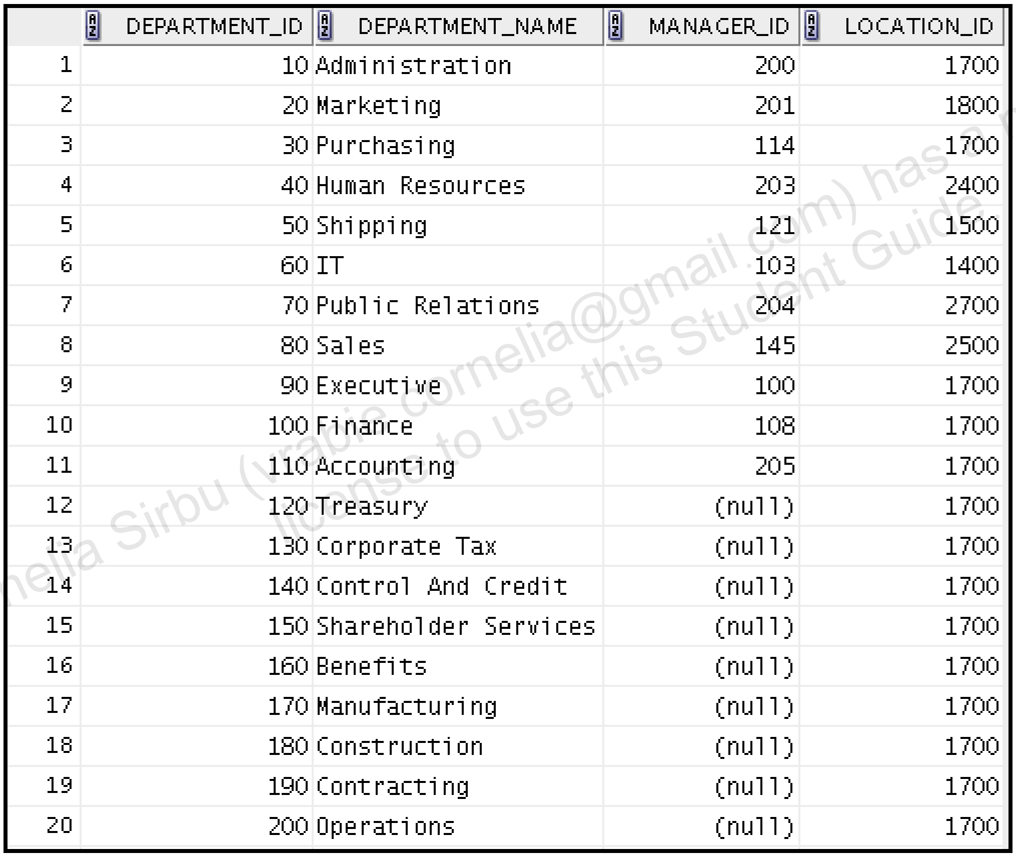
d. Take back the privilege from the user who performs step b.

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

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

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

12. Query all the rows in your DEPARTMENTS table.



	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	30	Purchasing	114	1700
4	40	Human Resources	203	2400
5	50	Shipping	121	1500
6	60	IT	103	1400
7	70	Public Relations	204	2700
8	80	Sales	145	2500
9	90	Executive	100	1700
10	100	Finance	108	1700
11	110	Accounting	205	1700
12	120	Treasury	(null)	1700
13	130	Corporate Tax	(null)	1700
14	140	Control And Credit	(null)	1700
15	150	Shareholder Services	(null)	1700
16	160	Benefits	(null)	1700
17	170	Manufacturing	(null)	1700
18	180	Construction	(null)	1700
19	190	Contracting	(null)	1700
20	200	Operations	(null)	1700

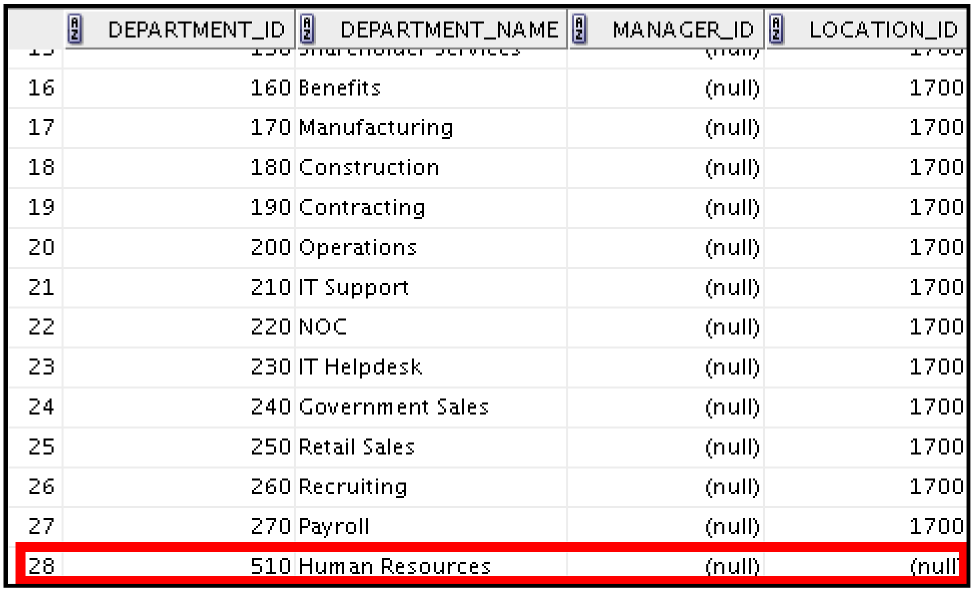
. . .

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

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

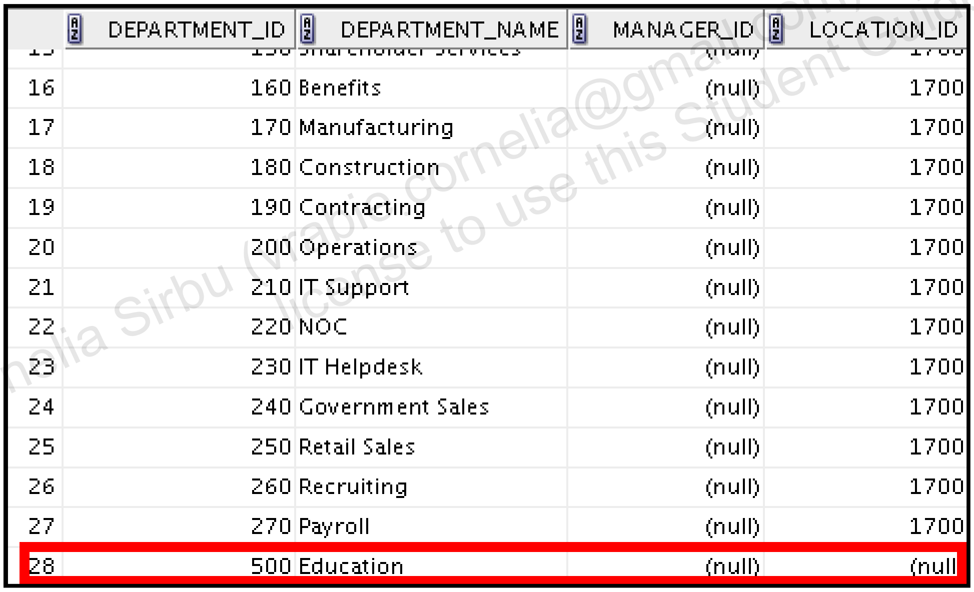
15. Query all the rows in the other team's DEPARTMENTS table by using your synonym.

Team 1 SELECT statement results:



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

Team 2 SELECT statement results:



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

16. Revoke the SELECT privilege from the other team.
17. Remove the row that you inserted into the DEPARTMENTS table in step 13 and save the changes.
18. Drop the synonyms team 1 and team 2.

Solution 18-1: Controlling User Access

1. What privilege should a user be given to log on to the Oracle server? Is this a system or an object privilege?
   The `CREATE SESSION` system privilege

2. What privilege should a user be given to create tables?
   The `CREATE TABLE` privilege

3. If you create a table, who can pass along privileges to other users in your table?
   You can, or anyone you have given those privileges to, by using `WITH GRANT OPTION`

4. You are the DBA. You create many users who require the same system privileges. What should you use to make your job easier?
   Create a role containing the system privileges and grant the role to the users.

5. What command do you use to change your password?
   The `ALTER USER` statement

6. `User21` is the owner of the `EMP` table and grants `DELETE` privileges to `User22` by using the `WITH GRANT OPTION` clause. `User22` then grants `DELETE` privileges on `EMP` to `User23`. `User21` now finds that `User23` has the privilege and revokes it from `User22`. Which user can now delete data from the `EMP` table?

   Only `User21`

7. You want to grant `SCOTT` the privilege to update data in the `DEPARTMENTS` table. You also want to enable `SCOTT` to grant this privilege to other users. What command do you use?

```
GRANT UPDATE ON departments TO scott WITH GRANT OPTION;
```

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

Note: For this exercise, open another SQL Developer session and connect as a different user. For example, if you are currently using `ora21`, open another SQL Developer session and connect as `ora22`. Here onwards we would refer the first SQL Developer session as Team 1 and the second SQL Developer session as Team 2.

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

Note: Replace *<team2_oraxx>* with **`ora22`**, *<team1_oraxx>* with **`ora21`**, and *<team3_oraxx>* with **`ora23`**.

Team 1 executes this statement:

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

   b. Have the user query your `REGIONS` table.

Team 2 executes this statement:

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

   c. Have the user pass on the query privilege to a third user, `ora23` .

Team 2 executes this statement.

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

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

Team 1 executes this statement.

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

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

Team 1 executes this statement.

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

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

Team 1 executes this statement.

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

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

   a. Team 2 executes the `GRANT` statement.

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

b. Team 1 executes the GRANT statement.

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

Here, *<team1_oraxx>* is the username of Team 1 and *<team2_oraxx>* is the username of Team 2.

12. Query all the rows in your DEPARTMENTS table.

```
       SELECT  *
       FROM    departments;
```

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

a. Team 1 executes this INSERT statement.

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

b. Team 2 executes this INSERT statement.

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

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

a. Team 1 creates a synonym named team 2.

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

b. Team 2 creates a synonym named team 1.

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

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

a. Team 1 executes this SELECT statement.

```
SELECT  *
   FROM    team2;
```

b. Team 2 executes this SELECT statement.

```
SELECT  *
    FROM    team1;
```

16. Revoke the SELECT privilege from the other team.
    a. Team 1 revokes the privilege.

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

    b. Team 2 revokes the privilege.

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

17. Remove the row that you inserted into the DEPARTMENTS table in step 13 and save the changes.
    a. Team 1 executes this DELETE statement.

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

    b. Team 2 executes this DELETE statement.

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

18. Drop the synonyms team 1 and team 2.

```
DROP SYNONYM team1;
DROP SYNONYM team2;
```

Practices for Lesson 19: Manipulating Data Using Advanced Queries

Chapter 19

Practices for Lesson 19: Overview

Practice overview:

This practice covers the following topics:

- Performing multitable `INSERTs`
- Performing `MERGE` operations
- Performing flashback operations
- Tracking row versions

Note: Before starting this practice, execute **`/home/oracle/labs/sql2/code_ex/cleanup_scripts/cleanup_09.sql`** script.

Note the following location for the practice files: `/home/oracle/labs/sql2/labs`

Practice 19-1: Manipulating Data

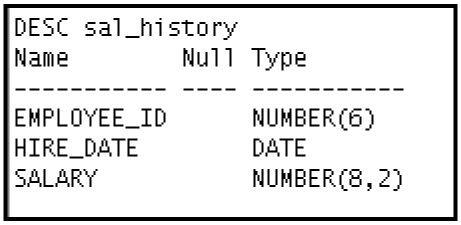
Overview

In this practice, you perform multitable INSERT and MERGE operations, flashback operation, and track row versions.

Note: Execute cleanup_09.sql script from **/home/oracle/labs/sql2/code_ex/cleanup_scripts/** before performing the following tasks.

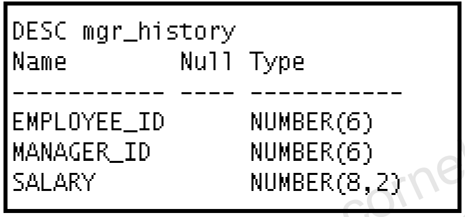
Tasks

1. Run the lab_09_01.sql script in the lab folder to create the SAL_HISTORY table.
2. Display the structure of the SAL_HISTORY table.



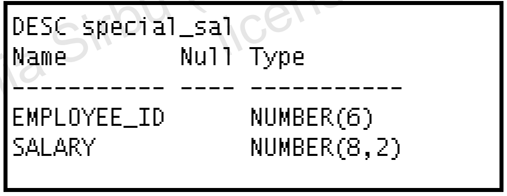
```
DESC sal_history
Name        Null Type
----------- ---- -----------
EMPLOYEE_ID      NUMBER(6)
HIRE_DATE        DATE
SALARY           NUMBER(8,2)
```

3. Run the lab_09_03.sql script in the lab folder to create the MGR_HISTORY table.
4. Display the structure of the MGR_HISTORY table.



```
DESC mgr_history
Name        Null Type
----------- ---- -----------
EMPLOYEE_ID      NUMBER(6)
MANAGER_ID       NUMBER(6)
SALARY           NUMBER(8,2)
```

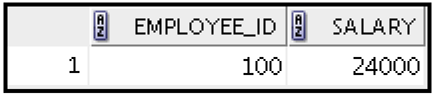
5. Run the lab_09_05.sql script in the lab folder to create the SPECIAL_SAL table.
6. Display the structure of the SPECIAL_SAL table.



```
DESC special_sal
Name        Null Type
----------- ---- -----------
EMPLOYEE_ID      NUMBER(6)
SALARY           NUMBER(8,2)
```

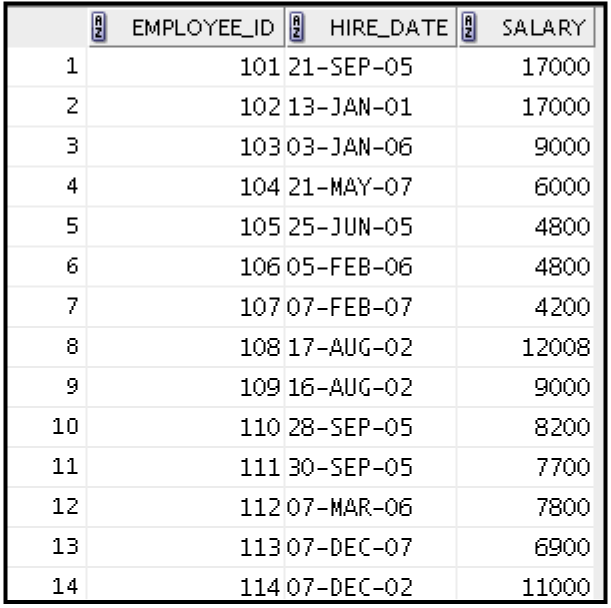
7.
   a. Write a query to do the following:
      - Retrieve details such as the employee ID, hire date, salary, and manager ID of those employees whose employee ID is less than 125 from the EMPLOYEES table.
      - If the salary is more than $20,000, insert details such as the employee ID and salary into the SPECIAL_SAL table.
      - If the salary is less than $20,000:
        - Insert details such as the employee ID, hire date, and salary into the SAL_HISTORY table
        - Insert details such as the employee ID, manager ID, and salary into the MGR_HISTORY table

b. Display the records from the `SPECIAL_SAL` table.



	EMPLOYEE_ID	SALARY
1	100	24000

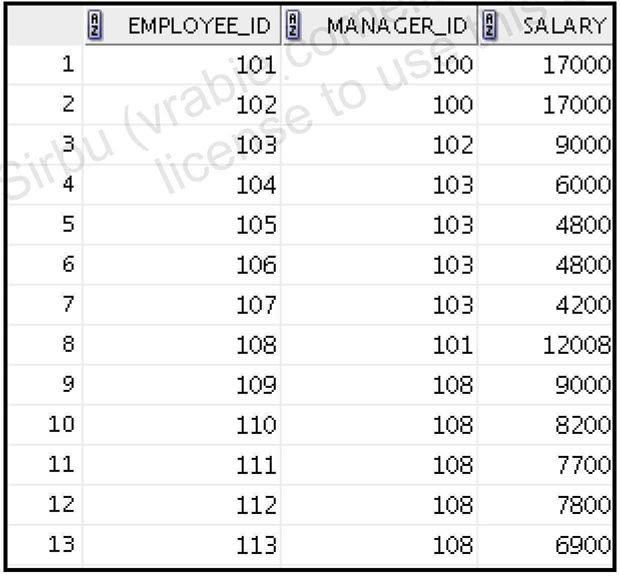
c. Display the records from the `SAL_HISTORY` table.



	EMPLOYEE_ID	HIRE_DATE	SALARY
1	101	21-SEP-05	17000
2	102	13-JAN-01	17000
3	103	03-JAN-06	9000
4	104	21-MAY-07	6000
5	105	25-JUN-05	4800
6	106	05-FEB-06	4800
7	107	07-FEB-07	4200
8	108	17-AUG-02	12008
9	109	16-AUG-02	9000
10	110	28-SEP-05	8200
11	111	30-SEP-05	7700
12	112	07-MAR-06	7800
13	113	07-DEC-07	6900
14	114	07-DEC-02	11000

...

d. Display the records from the `MGR_HISTORY` table.



	EMPLOYEE_ID	MANAGER_ID	SALARY
1	101	100	17000
2	102	100	17000
3	103	102	9000
4	104	103	6000
5	105	103	4800
6	106	103	4800
7	107	103	4200
8	108	101	12008
9	109	108	9000
10	110	108	8200
11	111	108	7700
12	112	108	7800
13	113	108	6900

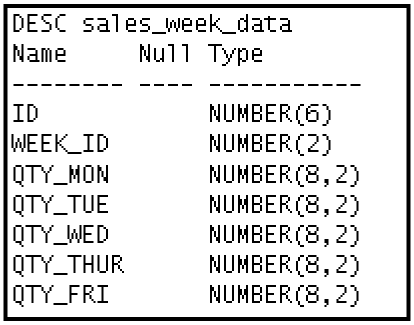
...

8.

a. Run the `lab_09_08_a.sql` script in the lab folder to create the `SALES_WEEK_DATA` table.

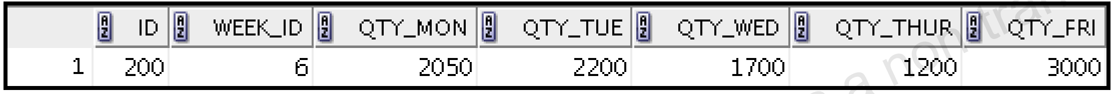
b. Run the `lab_09_08_b.sql` script in the `lab` folder to insert records into the `SALES_WEEK_DATA` table.

c. Display the structure of the `SALES_WEEK_DATA` table.



```
DESC sales_week_data
Name     Null Type
-------- ---- -----------
ID            NUMBER(6)
WEEK_ID       NUMBER(2)
QTY_MON       NUMBER(8,2)
QTY_TUE       NUMBER(8,2)
QTY_WED       NUMBER(8,2)
QTY_THUR      NUMBER(8,2)
QTY_FRI       NUMBER(8,2)
```

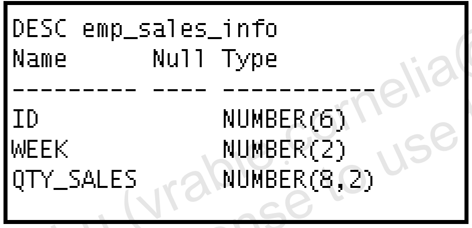
d. Display the records from the `SALES_WEEK_DATA` table.



	ID	WEEK_ID	QTY_MON	QTY_TUE	QTY_WED	QTY_THUR	QTY_FRI
1	200	6	2050	2200	1700	1200	3000

e. Run the `lab_09_08_e.sql` script in the lab folder to create the `EMP_SALES_INFO` table.

f. Display the structure of the `EMP_SALES_INFO` table.

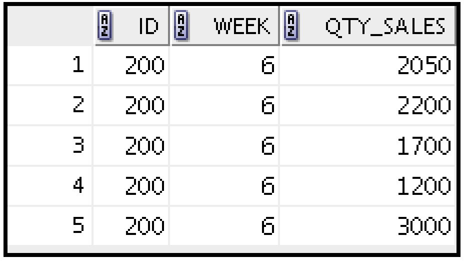


```
DESC emp_sales_info
Name      Null Type
--------- ---- -----------
ID             NUMBER(6)
WEEK           NUMBER(2)
QTY_SALES      NUMBER(8,2)
```

g. Write a query to do the following:

- Retrieve details such as `ID`, week `ID`, sales quantity on Monday, sales quantity on Tuesday, sales quantity on Wednesday, sales quantity on Thursday, and sales quantity on Friday from the `SALES_WEEK_DATA` table.
- Build a transformation such that each record retrieved from the `SALES_WEEK_DATA` table is converted into multiple records for the `EMP_SALES_INFO` table.
  Hint: Use a pivoting `INSERT` statement.

h. Display the records from the `EMP_SALES_INFO` table.



	ID	WEEK	QTY_SALES
1	200	6	2050
2	200	6	2200
3	200	6	1700
4	200	6	1200
5	200	6	3000

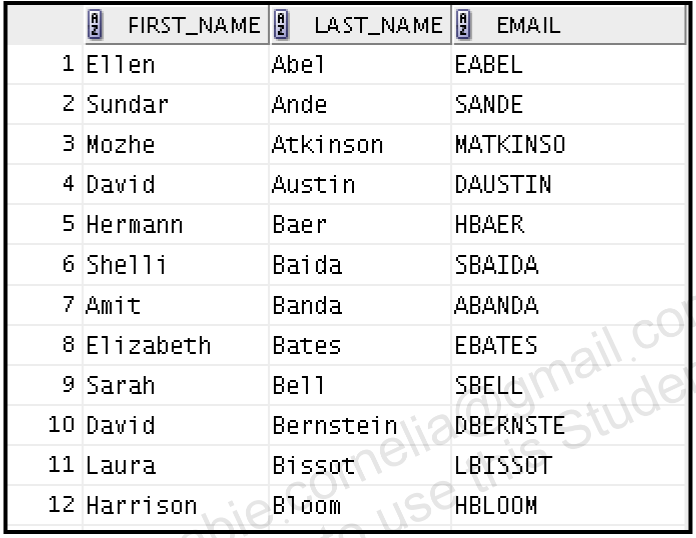
9. You have the data of past employees stored in a flat file called `emp.data`. You want to store the names and email `ID`s of all employees, past and present, in a table. To do this, first create an external table called `EMP_DATA` using the `emp.dat` source file in the `emp_dir` directory. Use the `lab_09_09.sql` script to do this.

10. Run `the lab_09_10.sql` script to create the `EMP_HIST` table.

    a. Increase the size of the email column to 45.

    b. Merge the data in the `EMP_DATA` table created in the last lab into the data in the `EMP_HIST` table. Assume that the data in the external `EMP_DATA` table is the most up-to-date. If a row in the `EMP_DATA` table matches the `EMP_HIST` table, update the email column of the `EMP_HIST` table to match the `EMP_DATA` table row. If a row in the `EMP_DATA` table does not match, insert it into the `EMP_HIST` table. Rows are considered matching when the employee's first and last names are identical.

    c. Retrieve the rows from `EMP_HIST` after the merge.



	FIRST_NAME	LAST_NAME	EMAIL
1	Ellen	Abel	EABEL
2	Sundar	Ande	SANDE
3	Mozhe	Atkinson	MATKINSO
4	David	Austin	DAUSTIN
5	Hermann	Baer	HBAER
6	Shelli	Baida	SBAIDA
7	Amit	Banda	ABANDA
8	Elizabeth	Bates	EBATES
9	Sarah	Bell	SBELL
10	David	Bernstein	DBERNSTE
11	Laura	Bissot	LBISSOT
12	Harrison	Bloom	HBLOOM

...

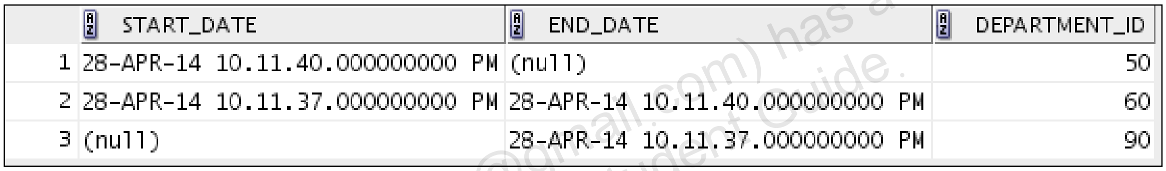
11. Create the `EMP2` table based on the following table instance chart. Enter the syntax in the SQL Worksheet. Then execute the statement to create the table. Confirm that the table is created.

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

12. Drop the `EMP2` table.

13. Query the recycle bin to see whether the table is present.
14. Restore the EMP2 table to a state before the DROP statement.
15. Create the EMP3 table using the lab_09_11.sql script. In the EMP3 table, change the department for Kochhar to 60 and commit your change. Next, change the department for Kochhar to 50 and commit your change. Track the changes to Kochhar using the Row Versions feature.

```
    UPDATE emp3 SET department_id = 60
    WHERE last_name = 'Kochhar';
    COMMIT;
    UPDATE emp3 SET department_id = 50
    WHERE last_name = 'Kochhar';
    COMMIT;

SELECT VERSIONS_STARTTIME "START_DATE",
   VERSIONS_ENDTIME "END_DATE",  DEPARTMENT_ID
FROM EMP3
   VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
WHERE LAST_NAME ='Kochhar';
```



	START_DATE	END_DATE	DEPARTMENT_ID
1	28-APR-14 10.11.40.000000000 PM	(null)	50
2	28-APR-14 10.11.37.000000000 PM	28-APR-14 10.11.40.000000000 PM	60
3	(null)	28-APR-14 10.11.37.000000000 PM	90

16. Drop the EMP2 and EMP3 tables so that they cannot be restored. Check in the recycle bin.

Solution 19-1: Manipulating Data

Solution

1. Run the `lab_09_01.sql` script in the lab folder to create the `SAL_HISTORY` table.
2. Display the structure of the `SAL_HISTORY` table.

```
DESC sal_history
```

3. Run the `lab_09_03.sql` script in the lab folder to create the `MGR_HISTORY` table.
4. Display the structure of the `MGR_HISTORY` table.

```
DESC mgr_history
```

5. Run the `lab_09_05.sql` script in the lab folder to create the `SPECIAL_SAL` table.
6. Display the structure of the `SPECIAL_SAL` table.

```
DESC special_sal
```

7.
   a. Write a query to do the following:
   - Retrieve details such as the employee `ID`, hire date, salary, and manager `ID` of those employees whose employee `ID` is less than `125` from the `EMPLOYEES` table.
   - If the salary is more than $20,000, insert details such as the employee `ID` and salary into the `SPECIAL_SAL` table.
   - If the salary is less than $20,000:
     - Insert details such as the employee `ID`, hire date, and salary into the `SAL_HISTORY` table
     - Insert details such as the employee `ID`, manager `ID`, and salary into the `MGR_HISTORY` table

```
INSERT ALL
WHEN SAL > 20000 THEN
INTO  special_sal VALUES (EMPID, SAL)
ELSE
INTO sal_history VALUES(EMPID,HIREDATE,SAL)
INTO mgr_history VALUES(EMPID,MGR,SAL)
SELECT employee_id EMPID, hire_date HIREDATE,
salary SAL, manager_id MGR
FROM employees
WHERE employee_id < 125;
```

   b. Display the records from the `SPECIAL_SAL` table.

```
   SELECT * FROM  special_sal;
```

c. Display the records from the SAL_HISTORY table.

```
    SELECT * FROM  sal_history;
```

d. Display the records from the MGR_HISTORY table.

```
    SELECT * FROM mgr_history;
```

8.

a. Run the lab_09_08_a.sql script in the lab folder to create the SALES_WEEK_DATA table.

b. Run the lab_09_08_b.sql script in the lab folder to insert records into the SALES_WEEK_DATA table.

c. Display the structure of the SALES_WEEK_DATA table.

```
DESC sales_week_data
```

d. Display the records from the SALES_WEEK_DATA table.

```
SELECT * FROM SALES_WEEK_DATA;
```

e. Run the lab_09_08_e.sql script in the lab folder to create the EMP_SALES_INFO table.

f. Display the structure of the EMP_SALES_INFO table.

```
DESC emp_sales_info
```

g. Write a query to do the following:

- Retrieve details such as the employee ID, week ID, sales quantity on Monday, sales quantity on Tuesday, sales quantity on Wednesday, sales quantity on Thursday, and sales quantity on Friday from the SALES_WEEK_DATA table.
- Build a transformation such that each record retrieved from the SALES_WEEK_DATA table is converted into multiple records for the EMP_SALES_INFO table.

Hint: Use a pivoting INSERT statement.

```
 INSERT ALL
      INTO emp_sales_info VALUES (id, week_id, QTY_MON)
      INTO emp_sales_info VALUES (id, week_id, QTY_TUE)
      INTO emp_sales_info VALUES (id, week_id, QTY_WED)
      INTO emp_sales_info VALUES (id, week_id, QTY_THUR)
      INTO emp_sales_info VALUES (id, week_id, QTY_FRI)
 SELECT ID, week_id, QTY_MON, QTY_TUE, QTY_WED,
      QTY_THUR,QTY_FRI FROM sales_week_data;
```

h. Display the records from the SALES_INFO table.

```
    SELECT * FROM emp_sales_info;
```

9. You have the data of past employees stored in a flat file called emp.data. You want to store the names and email IDs of all employees past and present in a table. To do this, first create an external table called EMP_DATA using the emp.dat source file in the emp_dir directory. You can use the script in lab_09_09.sql to do this.

```
CREATE TABLE emp_data
  (first_name  VARCHAR2(20)
  ,last_name   VARCHAR2(20)
  , email     VARCHAR2(30)
  )
ORGANIZATION EXTERNAL
(
 TYPE oracle_loader
 DEFAULT DIRECTORY emp_dir
 ACCESS PARAMETERS
 (
  RECORDS DELIMITED BY NEWLINE CHARACTERSET US7ASCII
  NOBADFILE
  NOLOGFILE
  FIELDS
  ( first_name POSITION ( 1:20) CHAR
  , last_name POSITION (22:41) CHAR
  ,  email    POSITION (43:72) CHAR )
 )
 LOCATION ('emp.dat') ) ;
```

10. Run the lab_09_10.sql script to create the EMP_HIST table.

a. Increase the size of the email column to 45.

```
ALTER TABLE emp_hist MODIFY email varchar(45);
```

b. Merge the data in the EMP_DATA table created in the last lab into the data in the EMP_HIST table. Assume that the data in the external EMP_DATA table is the most up-to-date. If a row in the EMP_DATA table matches the EMP_HIST table, update the email column of the EMP_HIST table to match the EMP_DATA table row. If a row in the EMP_DATA table does not match, insert it into the EMP_HIST table. Rows are considered matching when the employee’s first and last names are identical.

```
MERGE INTO EMP_HIST f USING EMP_DATA h
 ON (f.first_name = h.first_name
 AND f.last_name = h.last_name)
WHEN MATCHED THEN
 UPDATE SET f.email = h.email
WHEN NOT MATCHED THEN
 INSERT (f.first_name
    , f.last_name
    , f.email)
 VALUES (h.first_name
    , h.last_name
    , h.email);
```

c. Retrieve the rows from EMP_HIST after the merge.

```
SELECT * FROM emp_hist;
```

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

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

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

      DESCRIBE emp2
```

12. Drop the `EMP2` table.

```
     DROP TABLE emp2;
```

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

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

14. Restore the `EMP2` table to a state before the `DROP` statement.

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

15. Create the `EMP3` table using the `lab_09_11.sql` script. In the `EMP3` table, change the department for `Kochhar` to `60` and commit your change. Next, change the department for `Kochhar` to `50` and commit your change. Track the changes to `Kochhar` using the Row Versions feature.

```
    UPDATE emp3 SET department_id = 60
    WHERE last_name = 'Kochhar';
    COMMIT;
    UPDATE emp3 SET department_id = 50
    WHERE last_name = 'Kochhar';
    COMMIT;

SELECT VERSIONS_STARTTIME "START_DATE",
   VERSIONS_ENDTIME "END_DATE",  DEPARTMENT_ID
FROM EMP3
   VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
WHERE LAST_NAME ='Kochhar';
```

16. Drop the `EMP2` and `EMP3` tables so that they cannot be restored. Check in the recycle bin.

```
DROP TABLE emp2 PURGE;
DROP TABLE emp3 PURGE;

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

Practices for Lesson 20: Managing Data in Different Time Zones

Chapter 20

Practices for Lesson 20: Overview

Practice Overview:

This practice covers using the datetime functions.

Note: Before starting this practice, execute `/home/oracle/labs/sql2/code_ex/cleanup_scripts/cleanup_10.sql` script.

Note the following location for the practice files: `/home/oracle/labs/sql2/labs`

Practice 20-1: Managing Data in Different Time Zones

Overview

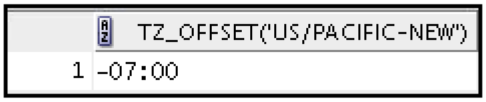
In this practice, you display time zone offsets, `CURRENT_DATE`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP`. You also set time zones and use the `EXTRACT` function.

Note: Execute `cleanup_10.sql` script from `/home/oracle/labs/sql2/code_ex/cleanup_scripts/cleanup_10.sql` before performing the following tasks.

Tasks

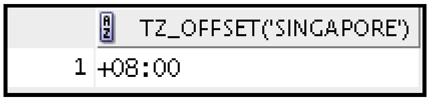
1. Alter the session to set `NLS_DATE_FORMAT` to `DD-MON-YYYY HH24:MI:SS`.
2.
   a. Write queries to display the time zone offsets (`TZ_OFFSET`) for the following time zones.

   - US/Pacific-New



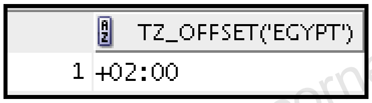
	TZ_OFFSET('US/PACIFIC-NEW')
1	-07:00

   - Singapore



	TZ_OFFSET('SINGAPORE')
1	+08:00

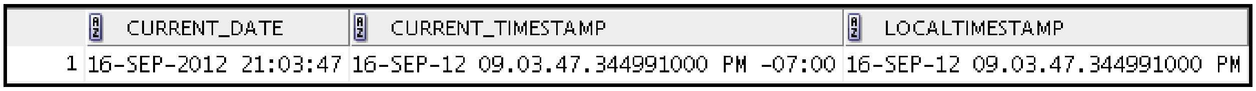
   - Egypt



	TZ_OFFSET('EGYPT')
1	+02:00

   b. Alter the session to set the `TIME_ZONE` parameter value to the time zone offset of US/Pacific-New.

   c. Display `CURRENT_DATE, CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP` for this session.

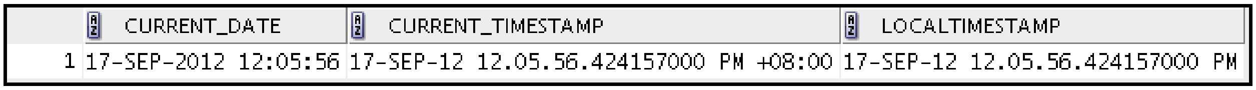


	CURRENT_DATE	CURRENT_TIMESTAMP	LOCALTIMESTAMP
1	16-SEP-2012 21:03:47	16-SEP-12 09.03.47.344991000 PM -07:00	16-SEP-12 09.03.47.344991000 PM

   d. Alter the session to set the `TIME_ZONE` parameter value to the time zone offset of Singapore.

   e. Display `CURRENT_DATE, CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP` for this session.

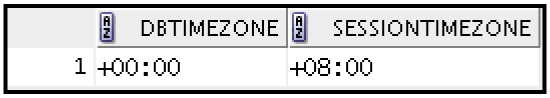
Note: The output might be different based on the date when the command is executed.



	CURRENT_DATE	CURRENT_TIMESTAMP	LOCALTIMESTAMP
1	17-SEP-2012 12:05:56	17-SEP-12 12.05.56.424157000 PM +08:00	17-SEP-12 12.05.56.424157000 PM

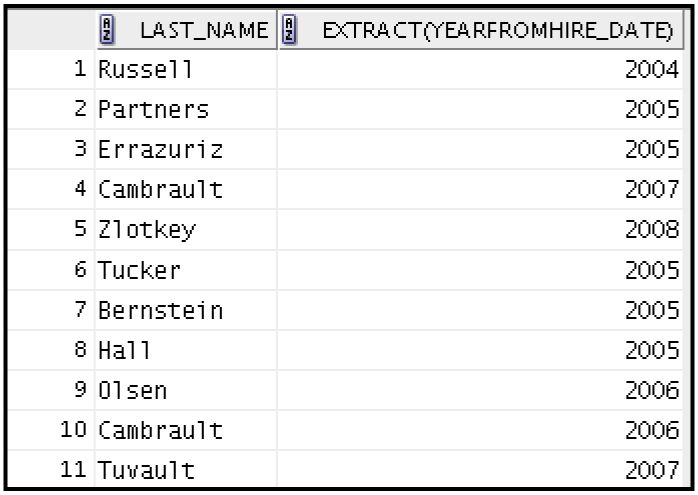
Note: Observe in the preceding practice that `CURRENT_DATE`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP` are sensitive to the session time zone.

3. Write a query to display DBTIMEZONE and SESSIONTIMEZONE.



	DBTIMEZONE	SESSIONTIMEZONE
1	+00:00	+08:00

4. Write a query to extract the YEAR from the HIRE_DATE column of the EMPLOYEES table for those employees who work in department 80.



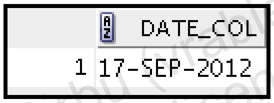
	LAST_NAME	EXTRACT(YEARFROMHIRE_DATE)
1	Russell	2004
2	Partners	2005
3	Errazuriz	2005
4	Cambrault	2007
5	Zlotkey	2008
6	Tucker	2005
7	Bernstein	2005
8	Hall	2005
9	Olsen	2006
10	Cambrault	2006
11	Tuvault	2007

...

5. Alter the session to set NLS_DATE_FORMAT to DD-MON-YYYY.
6. Examine and run the lab_10_06.sql script to create the SAMPLE_DATES table and populate it.

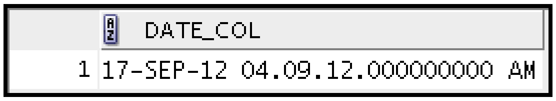
Note: The screenshot dates will change according to the sysdate.

a. Select from the table and view the data.



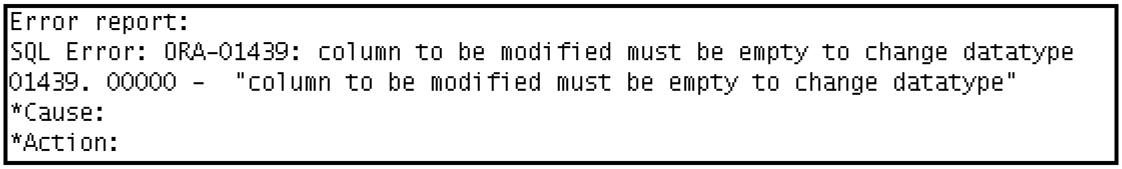
	DATE_COL
1	17-SEP-2012

b. Modify the data type of the DATE_COL column and change it to TIMESTAMP. Select from the table to view the data.



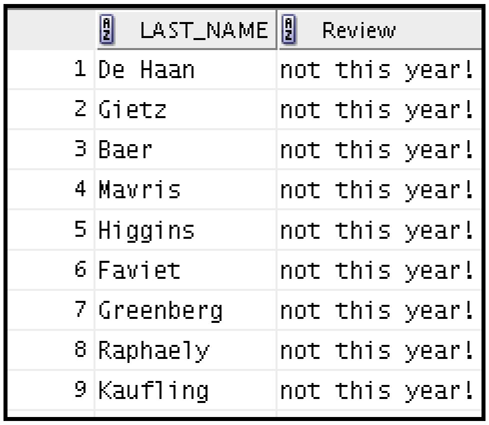
	DATE_COL
1	17-SEP-12 04.09.12.000000000 AM

c. Try to modify the data type of the DATE_COL column and change it to TIMESTAMP WITH TIME ZONE. What happens?



```
Error report:
SQL Error: ORA-01439: column to be modified must be empty to change datatype
01439. 00000 -  "column to be modified must be empty to change datatype"
*Cause:
*Action:
```

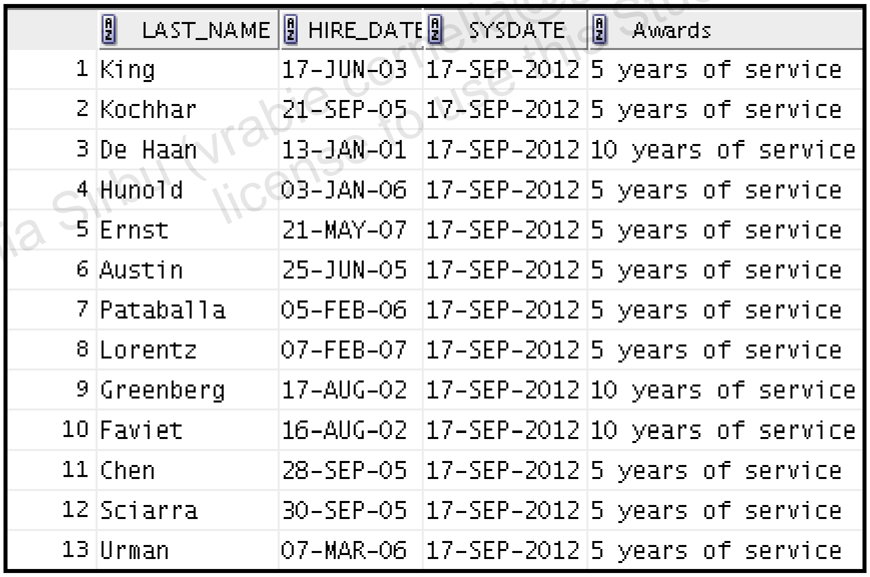
7. Create a query to retrieve last names from the EMPLOYEES table and calculate the review status. If the year hired was 2008, display Needs Review for the review status; otherwise, display not this year! Name the review status column Review. Sort the results by the HIRE_DATE column.
   Hint: Use a CASE expression with the EXTRACT function to calculate the review status.



	LAST_NAME	Review
1	De Haan	not this year!
2	Gietz	not this year!
3	Baer	not this year!
4	Mavris	not this year!
5	Higgins	not this year!
6	Faviet	not this year!
7	Greenberg	not this year!
8	Raphaely	not this year!
9	Kaufling	not this year!

...

8. Create a query to print the last names and the number of years of service for each employee. If the employee has been employed for five or more years, print 5 years of service. If the employee has been employed for 10 or more years, print 10 years of service. If the employee has been employed for 15 or more years, print 15 years of service. If none of these conditions matches, print maybe next year! Sort the results by the HIRE_DATE column. Use the EMPLOYEES table.
   Hint: Use CASE expressions and TO_YMINTERVAL.



	LAST_NAME	HIRE_DATE	SYSDATE	Awards
1	King	17-JUN-03	17-SEP-2012	5 years of service
2	Kochhar	21-SEP-05	17-SEP-2012	5 years of service
3	De Haan	13-JAN-01	17-SEP-2012	10 years of service
4	Hunold	03-JAN-06	17-SEP-2012	5 years of service
5	Ernst	21-MAY-07	17-SEP-2012	5 years of service
6	Austin	25-JUN-05	17-SEP-2012	5 years of service
7	Pataballa	05-FEB-06	17-SEP-2012	5 years of service
8	Lorentz	07-FEB-07	17-SEP-2012	5 years of service
9	Greenberg	17-AUG-02	17-SEP-2012	10 years of service
10	Faviet	16-AUG-02	17-SEP-2012	10 years of service
11	Chen	28-SEP-05	17-SEP-2012	5 years of service
12	Sciarra	30-SEP-05	17-SEP-2012	5 years of service
13	Urman	07-MAR-06	17-SEP-2012	5 years of service

...

Solution 20-1: Managing Data in Different Time Zones

Solution

1. Alter the session to set `NLS_DATE_FORMAT` to `DD-MON-YYYY HH24:MI:SS`.

```
   ALTER SESSION SET NLS_DATE_FORMAT =
   'DD-MON-YYYY HH24:MI:SS';
```

2.

   a. Write queries to display the time zone offsets (`TZ_OFFSET`) for the following time zones: *US/Pacific-New*, *Singapore*, and *Egypt*.

```
          US/Pacific-New
         SELECT TZ_OFFSET ('US/Pacific-New') from dual;
          Singapore
          SELECT TZ_OFFSET ('Singapore') from dual;
          Egypt
          SELECT TZ_OFFSET ('Egypt') from dual;
```

   b. Alter the session to set the `TIME_ZONE` parameter value to the time zone offset of US/Pacific-New.

```
    ALTER SESSION SET TIME_ZONE = '-7:00';
```

   c. Display `CURRENT_DATE`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP` for this session.
   Note: The output may be different based on the date when the command is executed.

```
          SELECT CURRENT_DATE, CURRENT_TIMESTAMP,
         LOCALTIMESTAMP FROM DUAL;
```

   d. Alter the session to set the `TIME_ZONE` parameter value to the time zone offset of Singapore.

```
 ALTER SESSION SET TIME_ZONE = '+8:00';
```

   e. Display `CURRENT_DATE`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP` for this session.
   Note: The output might be different, based on the date when the command is executed.

```
          SELECT CURRENT_DATE, CURRENT_TIMESTAMP,
    LOCALTIMESTAMP FROM DUAL;
```

Note: Observe in the preceding practice that `CURRENT_DATE`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP` are all sensitive to the session time zone.

3. Write a query to display DBTIMEZONE and SESSIONTIMEZONE.

```
      SELECT DBTIMEZONE,SESSIONTIMEZONE
      FROM DUAL;
```

4. Write a query to extract YEAR from the HIRE_DATE column of the EMPLOYEES table for those employees who work in department 80.

```
      SELECT last_name, EXTRACT (YEAR FROM HIRE_DATE)
      FROM employees
      WHERE department_id = 80;
```

5. Alter the session to set NLS_DATE_FORMAT to DD-MON-YYYY.

```
            ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY';
```

6. Examine and run the lab_10_06.sql script to create the SAMPLE_DATES table and populate it.

   a. Select from the table and view the data.

```
       SELECT * FROM sample_dates;
```

   b. Modify the data type of the DATE_COL column and change it to TIMESTAMP. Select from the table to view the data.

```
ALTER TABLE sample_dates MODIFY date_col TIMESTAMP;
SELECT * FROM sample_dates;
```

   c. Try to modify the data type of the DATE_COL column and change it to TIMESTAMP WITH TIME ZONE. What happens?

```
ALTER TABLE sample_dates MODIFY date_col
TIMESTAMP WITH TIME ZONE;
```

You are unable to change the data type of the DATE_COL column because the Oracle server does not permit you to convert from TIMESTAMP to TIMESTAMP WITH TIMEZONE by using the ALTER statement.

7. Create a query to retrieve last names from the `EMPLOYEES` table and calculate the review status. If the year hired was 2008, display `Needs Review` for the review status; otherwise, display `not this year!` Name the review status column `Review`. Sort the results by the `HIRE_DATE` column.
Hint: Use a `CASE` expression with the `EXTRACT` function to calculate the review status.

```
SELECT e.last_name
   ,     (CASE extract(year from e.hire_date)
           WHEN 2008 THEN 'Needs Review'
           ELSE 'not this year!'
      END )          AS "Review "
FROM   employees e
ORDER BY e.hire_date;
```

8. Create a query to print the last names and the number of years of service for each employee. If the employee has been employed five or more years, print `5 years of service`. If the employee has been employed 10 or more years, print `10 years of service`. If the employee has been employed 15 or more years, print `15 years of service`. If none of these conditions matches, print `maybe next year!` Sort the results by the `HIRE_DATE` column. Use the `EMPLOYEES` table.
Hint: Use `CASE` expressions and `TO_YMINTERVAL`.

```
SELECT e.last_name, hire_date, sysdate,
       (CASE
       WHEN  (sysdate -TO_YMINTERVAL('15-0'))>=
           hire_date THEN'15 years of service'
        WHEN (sysdate -TO_YMINTERVAL('10-0'))>= hire_date
          THEN '10 years of service'
        WHEN (sysdate - TO_YMINTERVAL('5-0'))>= hire_date
           THEN '5 years of service'
            ELSE 'maybe next year!'
          END) AS "Awards"
     FROM   employees e;
```

Practices for Lesson 21: Oracle Cloud Overview

Chapter 21

Practices for Lesson 21: Oracle Cloud overview

Practice Overview

There is no hands-on practice for this lesson. However you have:

- Steps to request Oracle Database Cloud Service Trial Account
- Getting Started with Oracle Public Cloud DBaaS demonstration

Practice 21-1: Requesting an Oracle Cloud Trial Account

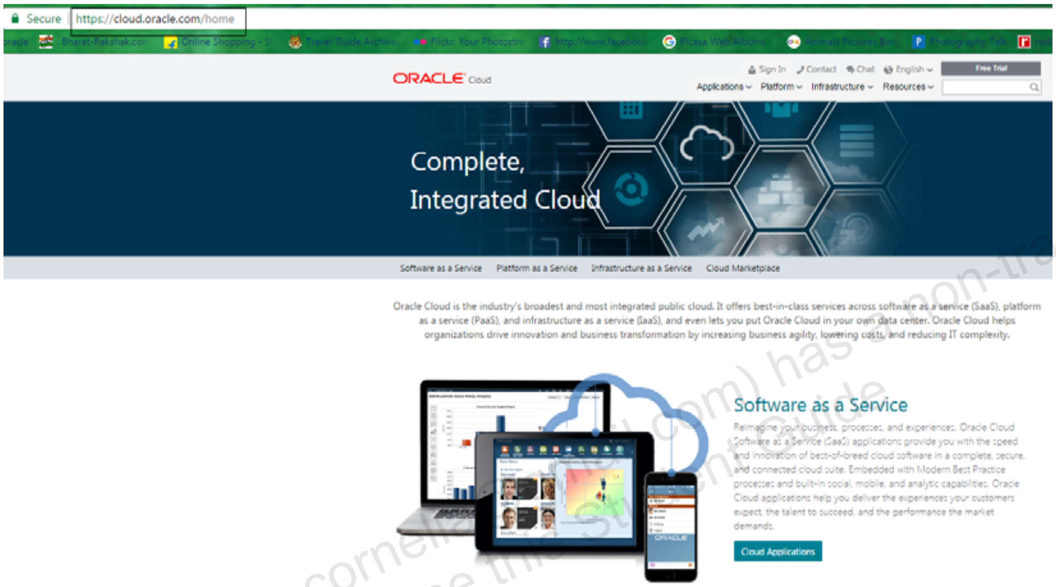
Overview

In this practice, you get an overview on how to request and activate an Oracle Cloud Trial account.

Tasks

A. Request a trial subscription:

1. Open your web browser and go to the Oracle Cloud website: `http://cloud.oracle.com`

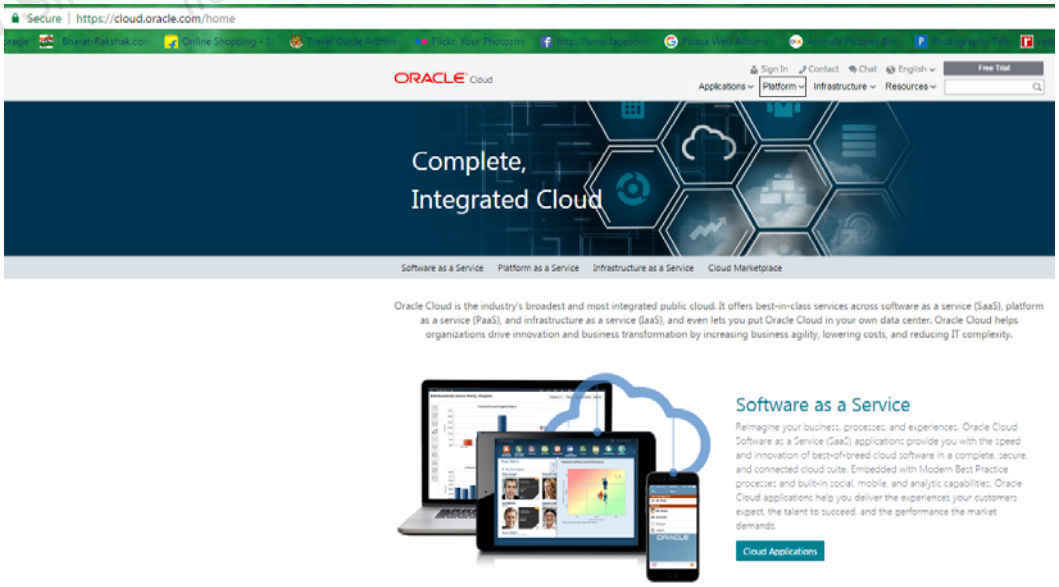


Oracle Cloud is the industry's broadest and most integrated public cloud. It offers best-in-class services across software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS), and even lets you put Oracle Cloud in your own data center. Oracle Cloud helps organizations drive innovation and business transformation by increasing business agility, lowering costs, and reducing IT complexity.

Software as a Service

Reimagine your business, processes, and experiences. Oracle Cloud Software as a Service (SaaS) applications provide you with the speed and innovation of best-of-breed cloud software in a complete, secure, and connected cloud suite. Embedded with Modern Best Practice processes and built-in social, mobile, and analytic capabilities, Oracle Cloud applications help you deliver the experiences your customers expect, the talent to succeed, and the performance the market demands.

2. Select one of the service category tabs. For example, Applications or Platform.

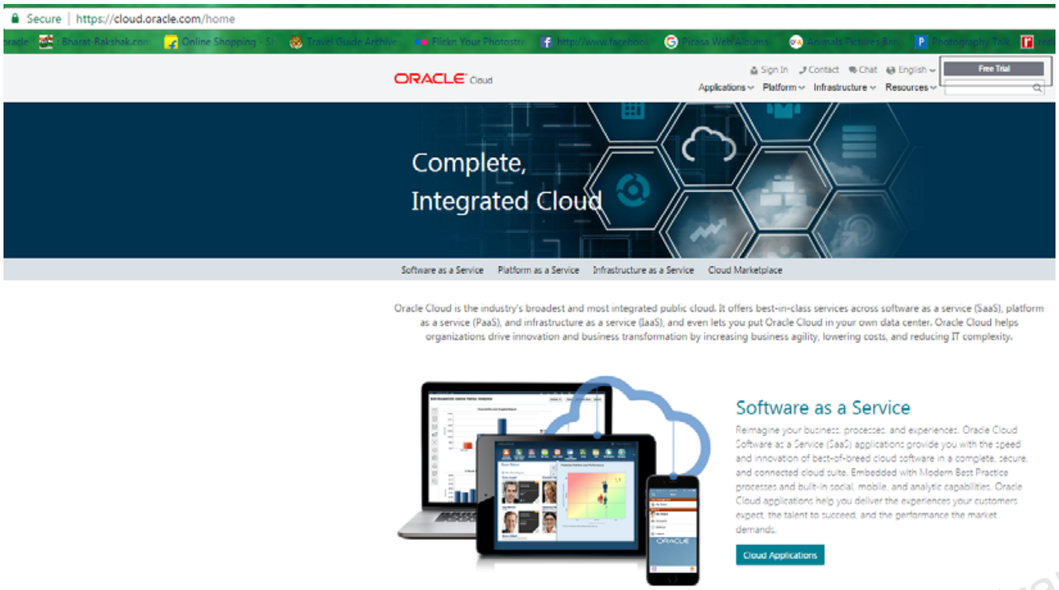


Oracle Cloud is the industry's broadest and most integrated public cloud. It offers best-in-class services across software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS), and even lets you put Oracle Cloud in your own data center. Oracle Cloud helps organizations drive innovation and business transformation by increasing business agility, lowering costs, and reducing IT complexity.

Software as a Service

Reimagine your business, processes, and experiences. Oracle Cloud Software as a Service (SaaS) applications provide you with the speed and innovation of best-of-breed cloud software in a complete, secure, and connected cloud suite. Embedded with Modern Best Practice processes and built-in social, mobile, and analytic capabilities, Oracle Cloud applications help you deliver the experiences your customers expect, the talent to succeed, and the performance the market demands.

3. Click **Free Trial**. The page lists the services that have free subscriptions.

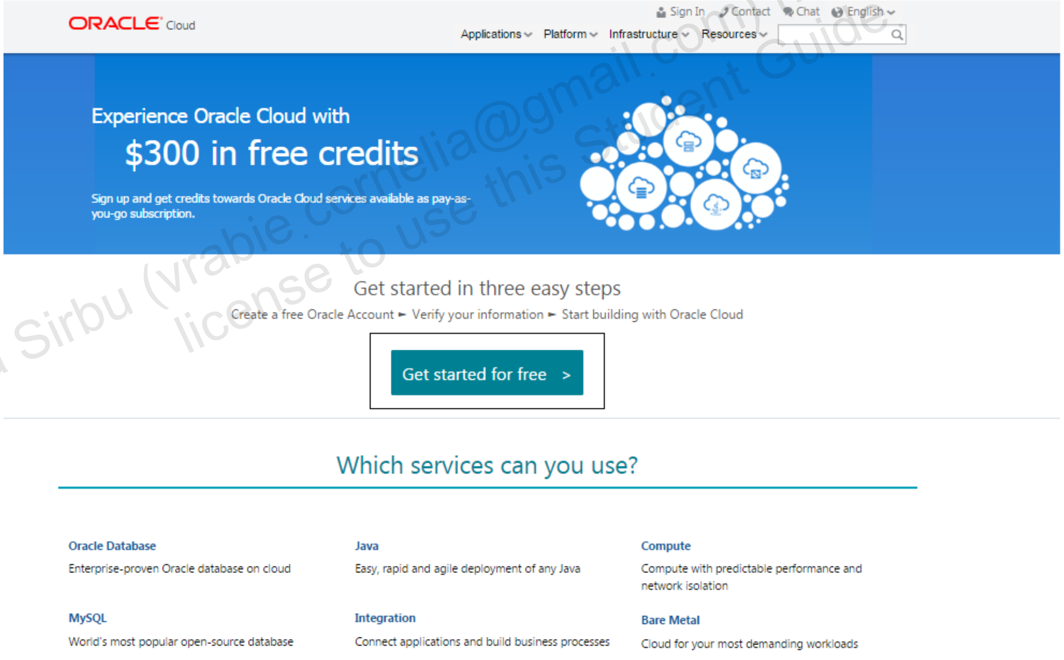


Oracle Cloud is the industry's broadest and most integrated public cloud. It offers best-in-class services across software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS), and even lets you put Oracle Cloud in your own data center. Oracle Cloud helps organizations drive innovation and business transformation by increasing business agility, lowering costs, and reducing IT complexity.

Software as a Service

Reimagine your business, processes, and experiences. Oracle Cloud Software as a Service (SaaS) applications provide you with the speed and innovation of best-of-breed cloud software in a complete, secure, and connected cloud suite. Embedded with Modern Best Practice processes and built-in social, mobile, and analytic capabilities, Oracle Cloud applications help you deliver the experiences your customers expect, the talent to succeed, and the performance the market demands.

4. Click **Get started for free**.



Experience Oracle Cloud with

$300 in free credits

Sign up and get credits towards Oracle Cloud services available as pay-as-you-go subscription.

Get started in three easy steps

Create a free Oracle Account ► Verify your information ► Start building with Oracle Cloud

Which services can you use?

Oracle Database
Enterprise-proven Oracle database on cloud

Java
Easy, rapid and agile deployment of any Java

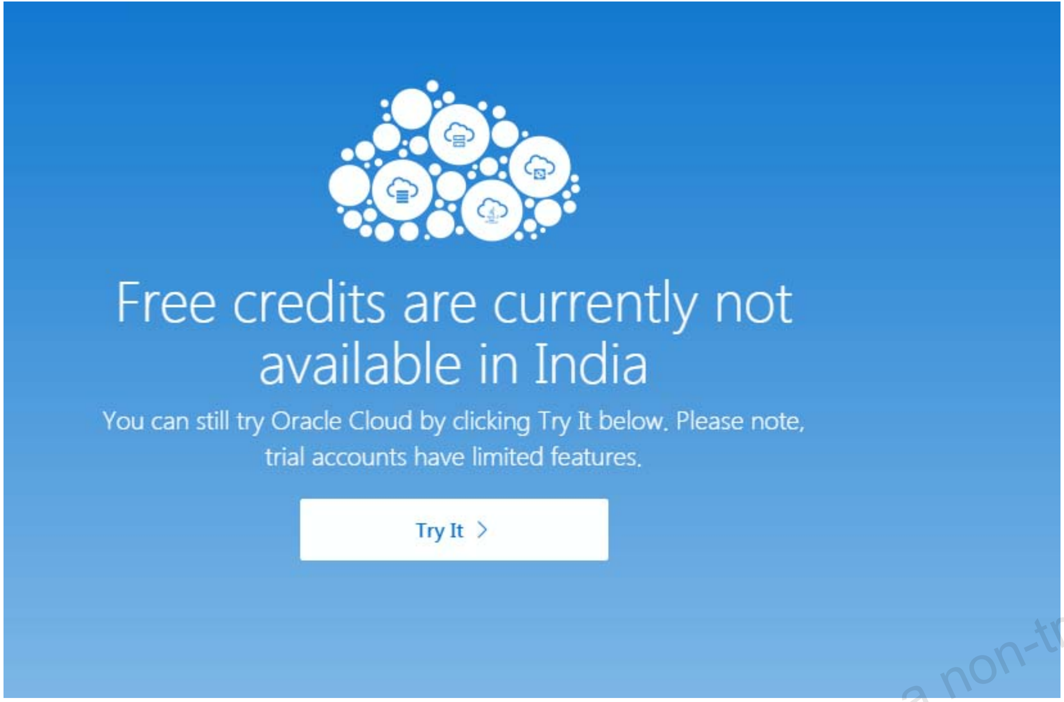
Compute
Compute with predictable performance and network isolation

MySQL
World's most popular open-source database

Integration
Connect applications and build business processes

Bare Metal
Cloud for your most demanding workloads

5. Click **Try It** to setup a free trial account.



Free credits are currently not available in India

You can still try Oracle Cloud by clicking Try It below. Please note, trial accounts have limited features.

6. Select one of the following options to continue:
   – If you already have an Oracle.com account, enter your single sign-on (SSO) user name and password, and click **Sign In**. The Sign Up for a Trial Subscription wizard opens.

     Note that if you are already signed in to your Oracle.com account, the system does not prompt for your credentials again. The Sign Up for a Trial Subscription wizard opens immediately.
   – If you don't have an Oracle.com account, then click **Sign Up** to register for a free account. Follow the on-screen instructions. Your account gets created and you will receive a confirmation email. Follow the instructions in the email to verify the status of your email address. You can then use your Oracle.com account to register for Oracle Cloud services.

7. Enter the information required to set up your Oracle Cloud account as follows:

Field	Description
First Name, Last Name	Enter your name.
Company, Country	Enter a new company name or select an existing one. Parentheses are allowed in the company name. If you are requesting your first trial, enter the company name, and select the country.

Field	Description
	If you have already requested a trial, the Company field shows a company name by default. You can select an existing company name from the list or you can enter a new company name.
Country Calling Code	The country calling code is automatically selected based on the country you choose from this list. If you select **Other** in this field, then you're prompted to enter the country code.
Mobile Number	Enter a valid mobile number.
Request Code/ Verification Code	You must request a verification code, which will be sent to the mobile number you specified, to verify your identity and complete the trial flow. Click **Request Code**. Enter the code (that you received on the mobile phone) in this field. Verification codes are valid for one-time use.
Service Name /Identity Domain/Account Name	Enter the service name that you want to use for which you are requesting a trial subscription. Oracle Cloud determines your options for the **Identity Domain** field based on the company name and country you entered on the Account Information page. For metered services, this is the account name. You can either create a new identity domain when requesting for metered service trials or use an existing domain. Note that you can't activate metered trials in a domain containing applications such as Oracle Human Capital Management Cloud Service (Oracle HCM) or Oracle Customer Relationship Management Cloud Service (Oracle CRM). • If an identity domain for trial subscriptions does not exist for the company and country you entered, then Oracle Cloud automatically generates and displays a unique name for the identity domain. You cannot change the value. • If an identity domain for trial subscriptions already exists for the company and country you entered, then the Identity Domain field displays the name of an existing domain by default. You can select any existing domain from the list or create a new identity domain.

Field	Description
	• If you create a new identity domain, then Oracle Cloud automatically generates a unique name for the identity domain. You can see the assigned name in the **Identity Domain** field. If you change the domain name, then ensure that you enter a unique domain name in the **Identity Domain** field. If you don't, then you'll get an error message when you try to go to the next step of the workflow. When generating names for identity domains, Oracle Cloud uses either of the following formats: • For metered trial subscriptions: *country**company**nnnnn* • For nonmetered trial subscriptions: *country**company**trialnnnnn* where: • *country*: Standard two-letter abbreviation for the country. • ***company***: • For metered trial subscriptions: Up to the first eight characters of the company name that you specified previously. • For nonmetered trial subscriptions: Up to the first 13 characters of the company name that you specified previously. • *trial*: The word “trial” • *nnnnn*: A 5-digit number, randomly generated. Examples: • For metered trial subscriptions*:* *usopenbree94621, caopenbree37518* • For nonmetered trial subscriptions: *usopenbreezetrial94621, caopenbreezetrial37518*
Data Jurisdiction	If you are prompted, then select the jurisdiction where you want us to set up your trial service. Data jurisdictions are filtered based on the requested service and subscription type. A data jurisdiction is automatically selected based on the Company’s country. However, you can select another

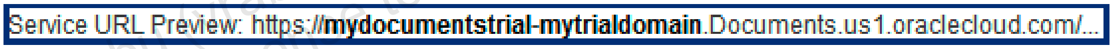
Field	Description
	data jurisdiction to set up your trial, if supported. You can't select a data jurisdiction when you request metered trials. The company's country is mapped to configured data jurisdiction in the system such as: • APAC • EMEA • South America • North America North America is selected by default if the country doesn't belong to the other 3 jurisdictions. If North America isn't available, then the first available data jurisdiction is selected. You can customize the data jurisdiction settings as required.

Note:

The generated service URL preview is displayed at the bottom:

https://<*service_name*>-<*identity_domain_name*>.<*cloud_service*>.<*data_center_name*>.oraclecloud.com

For example:



Service URL Preview: https://**mydocumentstrial-mytrialdomain**.Documents.us1.oraclecloud.com/...

The generated service URL preview changes as and when you change the service name, the identity domain name, or both. Note that the actual format of service URL preview varies based on the service type.

Read and accept the terms and conditions of the trial agreement before continuing.

Click **Sign up**. The system confirms that Oracle has received your request for a trial.

The Review Summary page displays the following details:

- **Service Information:** Displays the type of service you requested, the name of the service, and the identity domain to which the service belongs.

  In addition, the Review Summary page lists the name and types of other services included with your trial subscription request, if any. For example, the trial subscription for Oracle Java Cloud Service includes Oracle Database Cloud Service, which is created in the same identity domain.

  - **Order Information:** Displays the order ID, which is a unique identifier for this order, and the order date. Refer to the order ID whenever you contact us about billing or payment issues.
  - **Trial Information:** Displays the trial duration (usually 30 days).

Once your request for a trial subscription gets processed, you will receive an email with the following subject:

`Welcome to Oracle Cloud. Activate your trial`.

The email includes details about your order and your service. It also includes a link to activate your service. Activating the service makes it available for you to use.

Alternatively, you can sign in to My Account at any time to monitor the status of your services, including when a service is ready for activation.

B. Activating a Trial Subscription

When your trial subscription to a service is ready to be activated, you'll get an email from Oracle Cloud. You then use the My Account application to activate your service.

Only trial requests that were processed by Oracle Cloud team can be activated.

You can activate your service from the link in the email or from My Account.

Notes About Activating a Trial Subscription

These are some points to bear in mind when activating a trial subscription.

When you activate a trial subscription, note that:

- If you activate an Oracle Java Cloud Service trial, both Oracle Java Cloud Service and Oracle Database Cloud Service, which is included with the Java trial, will be activated.
- If you view the record for the trial service just after activation, the service is listed but it may not be fully activated yet by Oracle. When fully activated, the status is set to `Active`.
- When the service activation process is complete, the service and its details will be available in My Services, where you can monitor the status and usage of the service.

If you already have trial or paid subscriptions to Oracle Cloud services, you can go to My Services before you activate your service. However, if this request is your first request for a trial or paid subscription, you will not have access to My Services until after the activation process.

Activating Trial Subscriptions from the Email Link

One way to activate a trial is to use the activation link provided in the email from Oracle Cloud. You will receive the email when your service is ready to be activated.

To activate your trial subscription using the email link:

1. Open the Welcome email you received from Oracle Cloud.
2. Click **Activate My Trial**.
   - If you are not signed in, then the Oracle Sign In page opens. Enter your Oracle.com account user name and password, and click **Sign In**. The My Account application opens and displays the details page for the service.
   - If you are already signed in to your Oracle.com account, then the My Account application opens and displays the details page for the service. You don't need to sign in again.

On the details page for the service, note that the system:

- Displays a message that indicates the service was submitted for activation. You'll get another email when the service is active and ready to use.
- Updates the cloud icon to indicate the current status.
- Updates the **Status** field in the Additional Information section to indicate the current status.

Activating Trial Subscriptions From Oracle Cloud

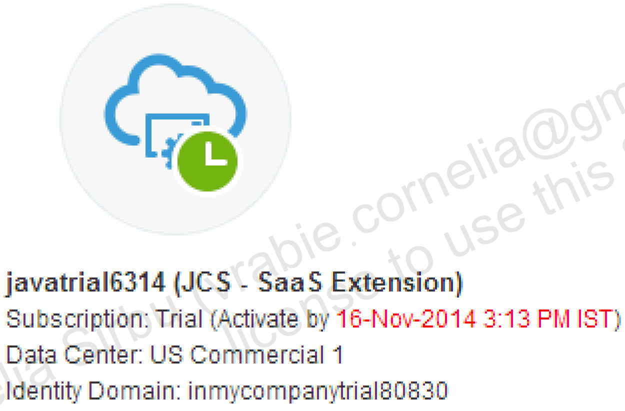
If you activated your trial subscription by using the email link, then you can skip this section.

To open My Account to get your trial subscription activated:

1. Sign in to My Account.
   a. Open your web browser and go to the Oracle Cloud website: `http://cloud.oracle.com`
   b. Click **Sign In** and then click **Sign In to My Account**.
   c. Enter your Oracle.com account user name and password, and click **Sign In**.

The Dashboard page in My Account opens.

2. Navigate to the listing of the trial service that you want to activate.



javatrial6314 (JCS - SaaS Extension)
Subscription: Trial (Activate by 16-Nov-2014 3:13 PM IST)
Data Center: US Commercial 1
Identity Domain: inmycompanytrial80830

Note:

- The cloud icon and its hover text indicate that the service hasn't been activated.
- The **Subscription** field specifies the date by which you must activate the trial subscription for this service. If you don't activate the service by the deadline, then Oracle Cloud cancels the subscription.
- The **Activate** button, which appears only if the service needs to be activated, is now available.

Click **Activate**.

Note that the system:

- Displays a message at the top of the page that indicates the service was submitted for activation. You'll get another email when the service is active and ready to use.
- Places the service listing in alphabetic order on the page.
- Updates the cloud icon and the **Subscription** field to indicate that the activation is in progress.

Practice 21-2: Getting Started with Oracle Public Cloud DBaaS demonstration

Overview

In this demonstration, you get an overview on how to create and access your first Database Cloud Service Instance.

Tasks

Review the Getting Started with Oracle Public Cloud DBaaS (http://oukc.oracle.com/public/redir.html?type=player&offid=1957025749) demonstration. This demonstration covers everything that is needed to create and access your first Database Cloud Service Instance.

Tasks include

- Creating a backup container
- Creating SSH Keys
- Creating the database itself

Additional Practices and Solutions

Chapter 22

Practices for Lesson 1: Overview

Practices Overview

In these practices, you will be working on extra exercises that are based on the following topics:

- Basic SQL `SELECT` statement
- Basic SQL Developer commands
- SQL functions

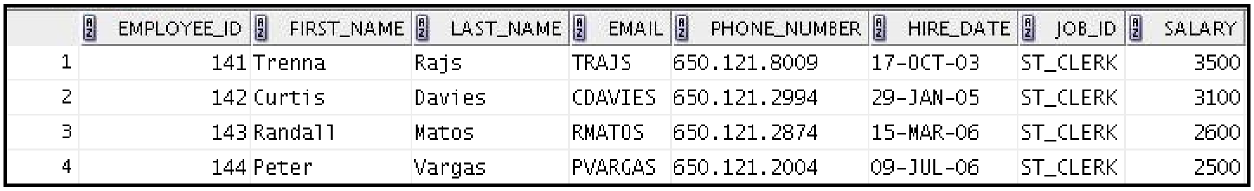
Practice 1-1: Additional Practice

Overview

In this practice, exercises have been designed to be worked on after you have discussed the following topics: basic SQL `SELECT` statement, basic SQL Developer commands, and SQL functions.

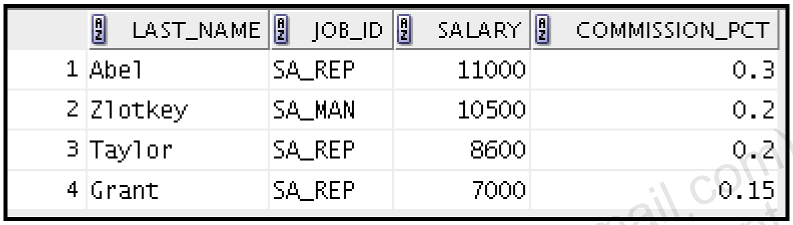
Tasks

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



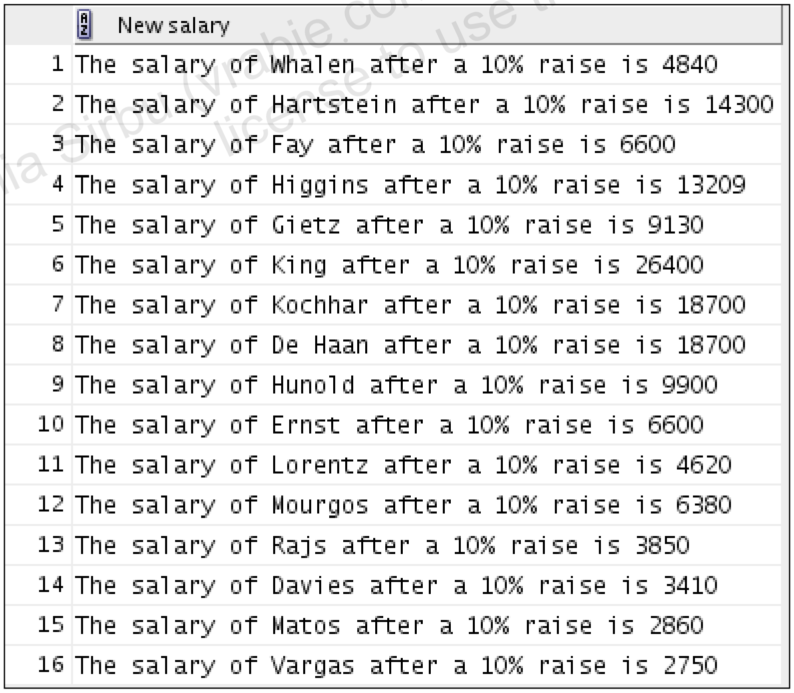
	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
1	141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-03	ST_CLERK	3500
2	142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-05	ST_CLERK	3100
3	143	Randall	Matos	RMATOS	650.121.2874	15-MAR-06	ST_CLERK	2600
4	144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-06	ST_CLERK	2500

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



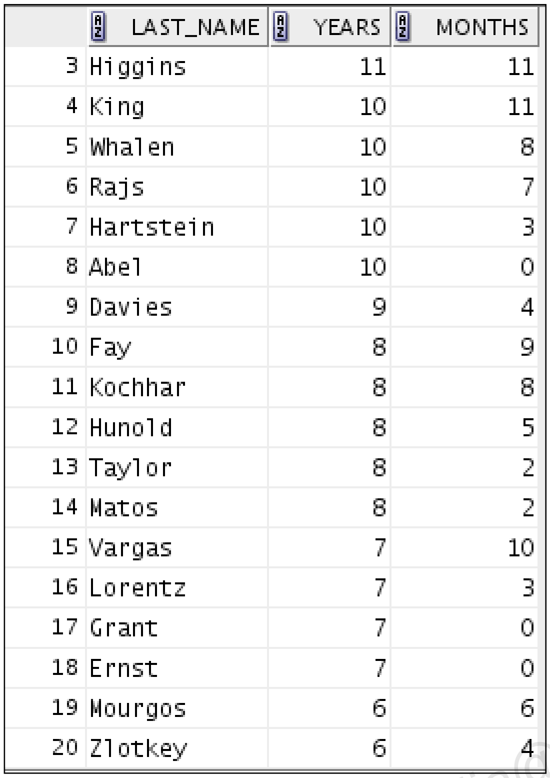
	LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
1	Abel	SA_REP	11000	0.3
2	Zlotkey	SA_MAN	10500	0.2
3	Taylor	SA_REP	8600	0.2
4	Grant	SA_REP	7000	0.15

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).



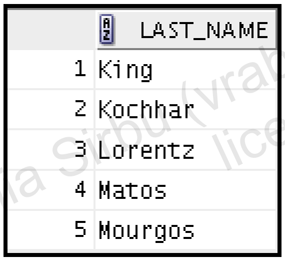
	New salary
1	The salary of Whalen after a 10% raise is 4840
2	The salary of Hartstein after a 10% raise is 14300
3	The salary of Fay after a 10% raise is 6600
4	The salary of Higgins after a 10% raise is 13209
5	The salary of Gietz after a 10% raise is 9130
6	The salary of King after a 10% raise is 26400
7	The salary of Kochhar after a 10% raise is 18700
8	The salary of De Haan after a 10% raise is 18700
9	The salary of Hunold after a 10% raise is 9900
10	The salary of Ernst after a 10% raise is 6600
11	The salary of Lorentz after a 10% raise is 4620
12	The salary of Mourgos after a 10% raise is 6380
13	The salary of Rajs after a 10% raise is 3850
14	The salary of Davies after a 10% raise is 3410
15	The salary of Matos after a 10% raise is 2860
16	The salary of Vargas after a 10% raise is 2750

4. Create a report of employees and their duration 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 duration of their employment. The employee who has been employed the longest should appear at the top of the list.



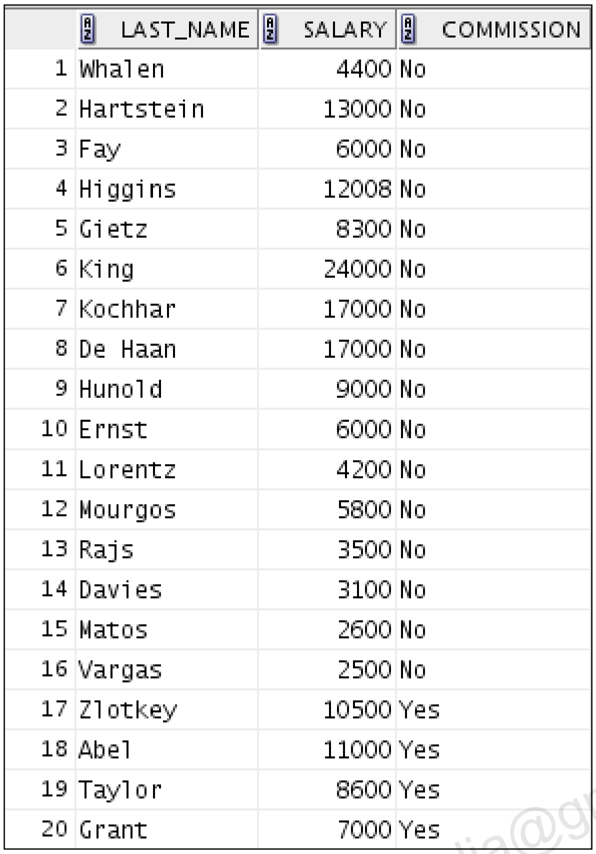
	LAST_NAME	YEARS	MONTHS
3	Higgins	11	11
4	King	10	11
5	Whalen	10	8
6	Rajs	10	7
7	Hartstein	10	3
8	Abel	10	0
9	Davies	9	4
10	Fay	8	9
11	Kochhar	8	8
12	Hunold	8	5
13	Taylor	8	2
14	Matos	8	2
15	Vargas	7	10
16	Lorentz	7	3
17	Grant	7	0
18	Ernst	7	0
19	Mourgos	6	6
20	Zlotkey	6	4

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

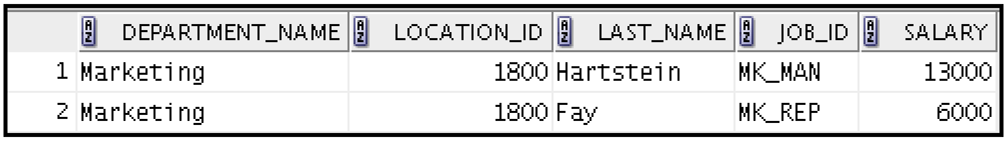
6. Create a report that displays all employees, and indicate with the words *Yes* or *No* whether they receive a commission. Use the DECODE expression in your query.



	LAST_NAME	SALARY	COMMISSION
1	Whalen	4400	No
2	Hartstein	13000	No
3	Fay	6000	No
4	Higgins	12008	No
5	Gietz	8300	No
6	King	24000	No
7	Kochhar	17000	No
8	De Haan	17000	No
9	Hunold	9000	No
10	Ernst	6000	No
11	Lorentz	4200	No
12	Mourgos	5800	No
13	Rajs	3500	No
14	Davies	3100	No
15	Matos	2600	No
16	Vargas	2500	No
17	Zlotkey	10500	Yes
18	Abel	11000	Yes
19	Taylor	8600	Yes
20	Grant	7000	Yes

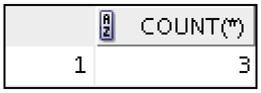
These exercises can be used for extra practice after you have discussed the following topics: basic SQL SELECT statements, basic SQL Developer commands, SQL functions, joins, and group functions.

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 a location. For example, if the user enters 1800, results are as follows:



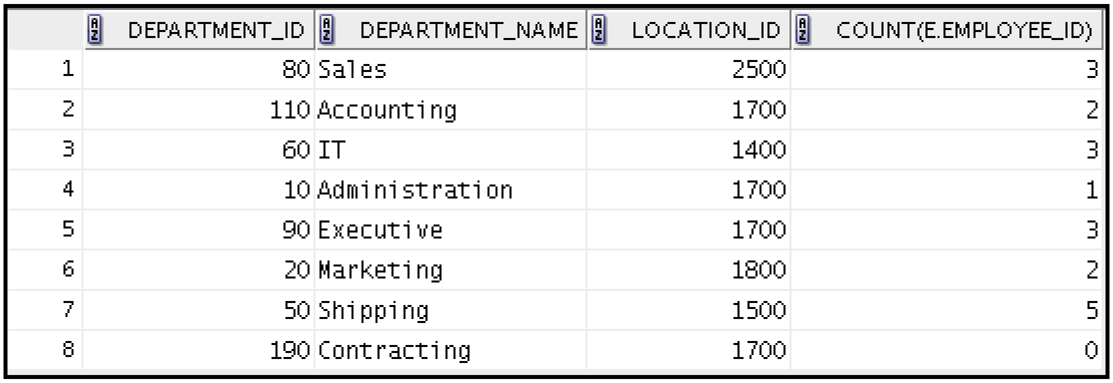
	DEPARTMENT_NAME	LOCATION_ID	LAST_NAME	JOB_ID	SALARY
1	Marketing	1800	Hartstein	MK_MAN	13000
2	Marketing	1800	Fay	MK_REP	6000

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



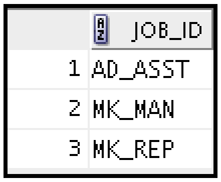
	COUNT(*)
1	3

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



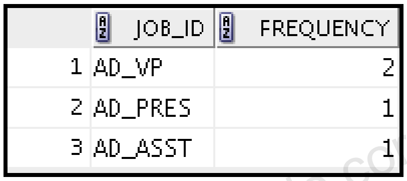
	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	COUNT(E.EMPLOYEE_ID)
1	80	Sales	2500	3
2	110	Accounting	1700	2
3	60	IT	1400	3
4	10	Administration	1700	1
5	90	Executive	1700	3
6	20	Marketing	1800	2
7	50	Shipping	1500	5
8	190	Contracting	1700	0

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
1	AD_ASST
2	MK_MAN
3	MK_REP

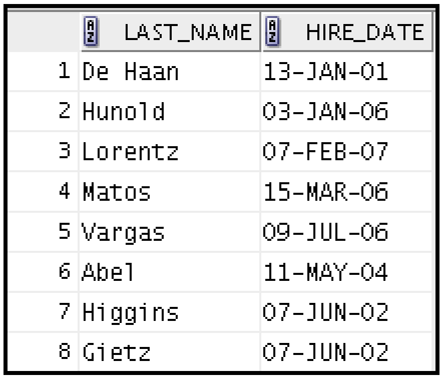
11. Create a report that displays the jobs that are found in the Administration and Executive departments. Also display the number of employees for these jobs. Show the job with the highest number of employees first.



	JOB_ID	FREQUENCY
1	AD_VP	2
2	AD_PRES	1
3	AD_ASST	1

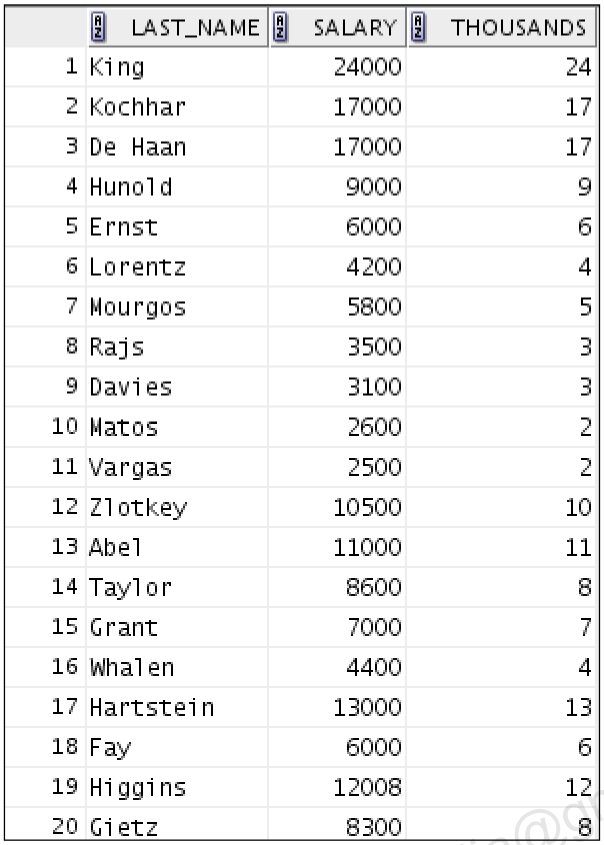
These exercises can be used for extra practice after you have discussed the following topics: basic SQL `SELECT` statements, basic SQL Developer commands, SQL functions, joins, group functions, and subqueries.

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



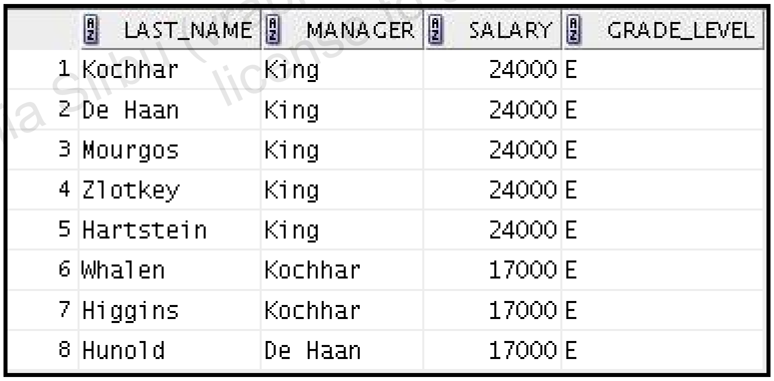
	LAST_NAME	HIRE_DATE
1	De Haan	13-JAN-01
2	Hunold	03-JAN-06
3	Lorentz	07-FEB-07
4	Matos	15-MAR-06
5	Vargas	09-JUL-06
6	Abel	11-MAY-04
7	Higgins	07-JUN-02
8	Gietz	07-JUN-02

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



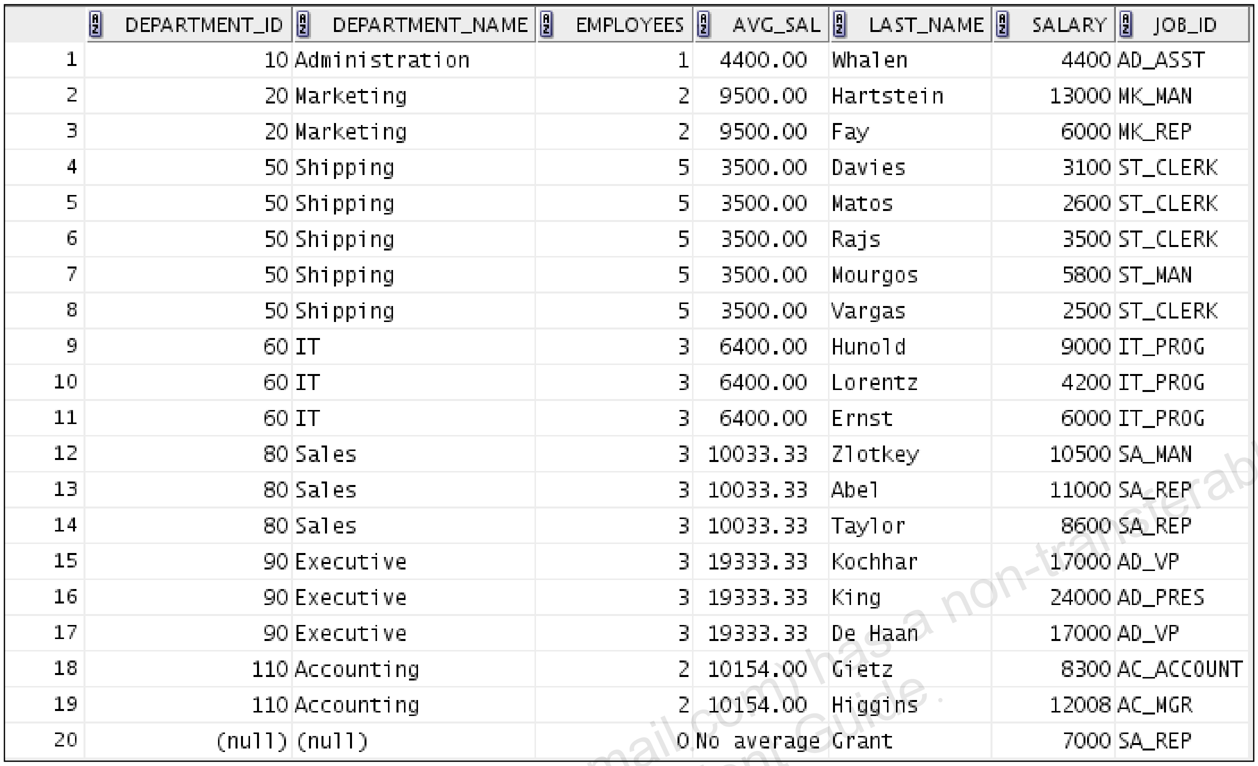
	LAST_NAME	SALARY	THOUSANDS
1	King	24000	24
2	Kochhar	17000	17
3	De Haan	17000	17
4	Hunold	9000	9
5	Ernst	6000	6
6	Lorentz	4200	4
7	Mourgos	5800	5
8	Rajs	3500	3
9	Davies	3100	3
10	Matos	2600	2
11	Vargas	2500	2
12	Zlotkey	10500	10
13	Abel	11000	11
14	Taylor	8600	8
15	Grant	7000	7
16	Whalen	4400	4
17	Hartstein	13000	13
18	Fay	6000	6
19	Higgins	12008	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.



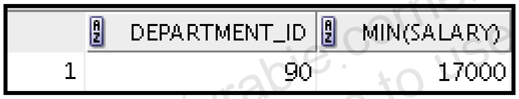
	LAST_NAME	MANAGER	SALARY	GRADE_LEVEL
1	Kochhar	King	24000	E
2	De Haan	King	24000	E
3	Mourgos	King	24000	E
4	Zlotkey	King	24000	E
5	Hartstein	King	24000	E
6	Whalen	Kochhar	17000	E
7	Higgins	Kochhar	17000	E
8	Hunold	De Haan	17000	E

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.



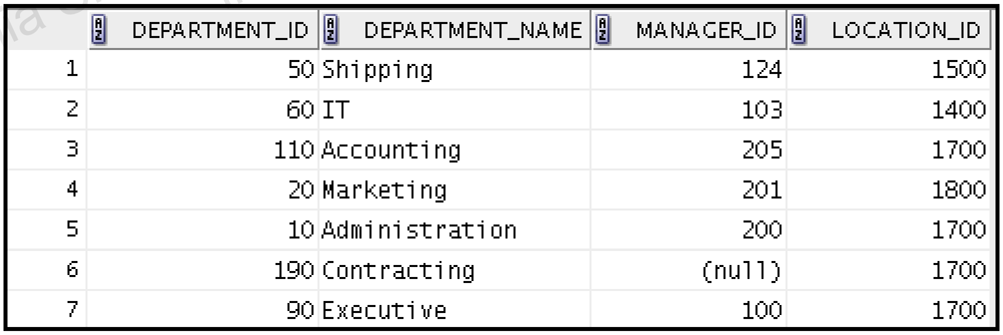
	DEPARTMENT_ID	DEPARTMENT_NAME	EMPLOYEES	AVG_SAL	LAST_NAME	SALARY	JOB_ID
1	10	Administration	1	4400.00	Whalen	4400	AD_ASST
2	20	Marketing	2	9500.00	Hartstein	13000	MK_MAN
3	20	Marketing	2	9500.00	Fay	6000	MK_REP
4	50	Shipping	5	3500.00	Davies	3100	ST_CLERK
5	50	Shipping	5	3500.00	Matos	2600	ST_CLERK
6	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	Abel	11000	SA_REP
14	80	Sales	3	10033.33	Taylor	8600	SA_REP
15	90	Executive	3	19333.33	Kochhar	17000	AD_VP
16	90	Executive	3	19333.33	King	24000	AD_PRES
17	90	Executive	3	19333.33	De Haan	17000	AD_VP
18	110	Accounting	2	10154.00	Gietz	8300	AC_ACCOUNT
19	110	Accounting	2	10154.00	Higgins	12008	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

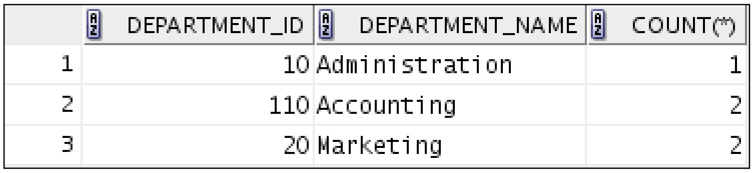
17. Create a report that displays departments where no sales representatives work. Include the department number, department name, manager ID, and location in the output.



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

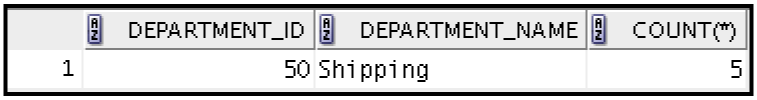
18. Create the following statistical reports for the HR department. Include the department number, department name, and the number of employees working in each department that:

    a. Employs fewer than three employees:



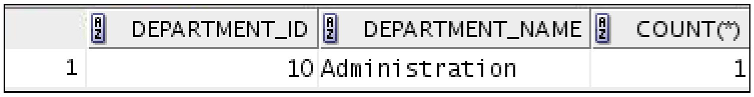
	DEPARTMENT_ID	DEPARTMENT_NAME	COUNT(*)
1	10	Administration	1
2	110	Accounting	2
3	20	Marketing	2

    b. Has the highest number of employees:



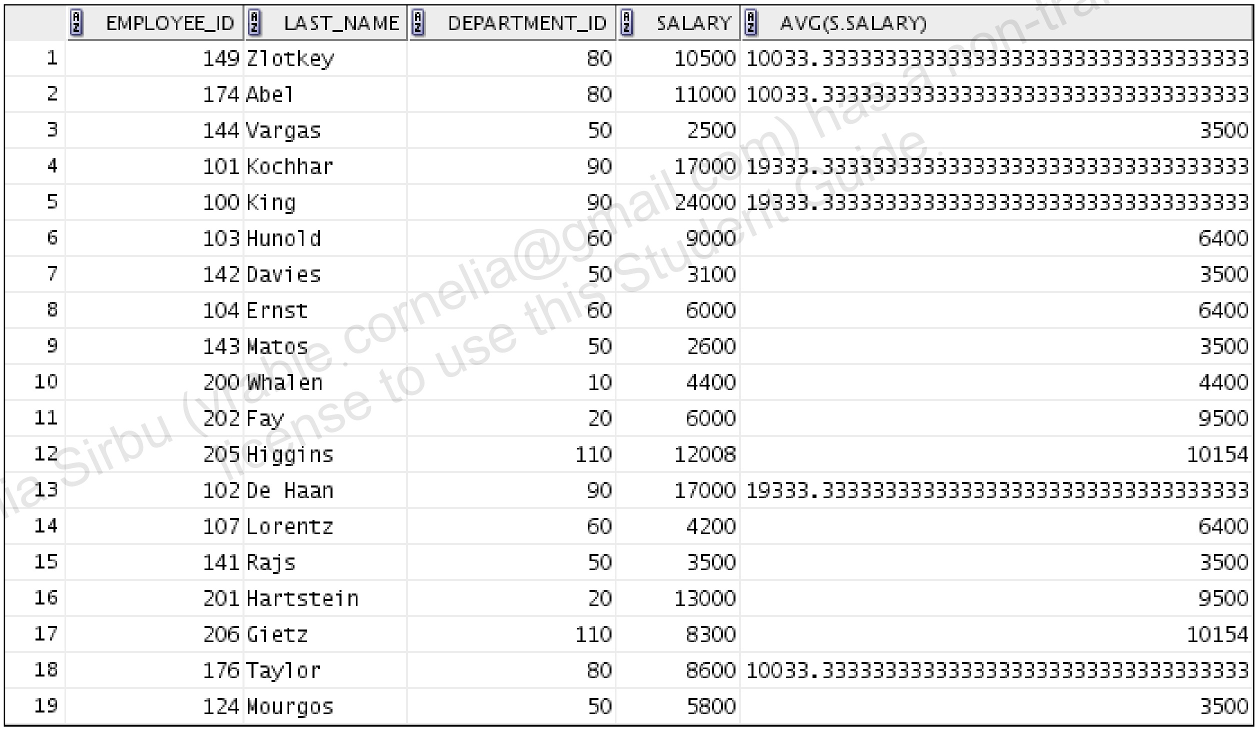
	DEPARTMENT_ID	DEPARTMENT_NAME	COUNT(*)
1	50	Shipping	5

    c. Has the lowest number of employees:



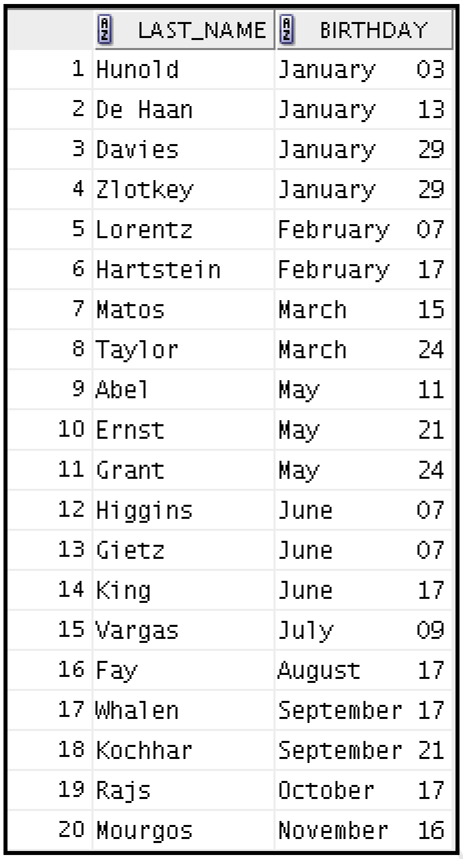
	DEPARTMENT_ID	DEPARTMENT_NAME	COUNT(*)
1	10	Administration	1

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	10033.333333333333333333333333333333333
2	174	Abel	80	11000	10033.333333333333333333333333333333333
3	144	Vargas	50	2500	3500
4	101	Kochhar	90	17000	19333.333333333333333333333333333333333
5	100	King	90	24000	19333.333333333333333333333333333333333
6	103	Hunold	60	9000	6400
7	142	Davies	50	3100	3500
8	104	Ernst	60	6000	6400
9	143	Matos	50	2600	3500
10	200	Whalen	10	4400	4400
11	202	Fay	20	6000	9500
12	205	Higgins	110	12008	10154
13	102	De Haan	90	17000	19333.333333333333333333333333333333333
14	107	Lorentz	60	4200	6400
15	141	Rajs	50	3500	3500
16	201	Hartstein	20	13000	9500
17	206	Gietz	110	8300	10154
18	176	Taylor	80	8600	10033.333333333333333333333333333333333
19	124	Mourgos	50	5800	3500

20. 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: Additional Practice

Overview

Solutions to Additional Practice 1-1 are given as follows.

Tasks

1. The HR department needs to find data for all the clerks who were hired after 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 a commission. Show the last name, job, salary, and commission of these 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 the duration of their 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 that starts 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, last name, job title, and salary of those employees who work in a specific location. Prompt the user for a location. 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 JOIN departments d
ON   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 `department_ID`s 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 these 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, irrespective of the year).

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

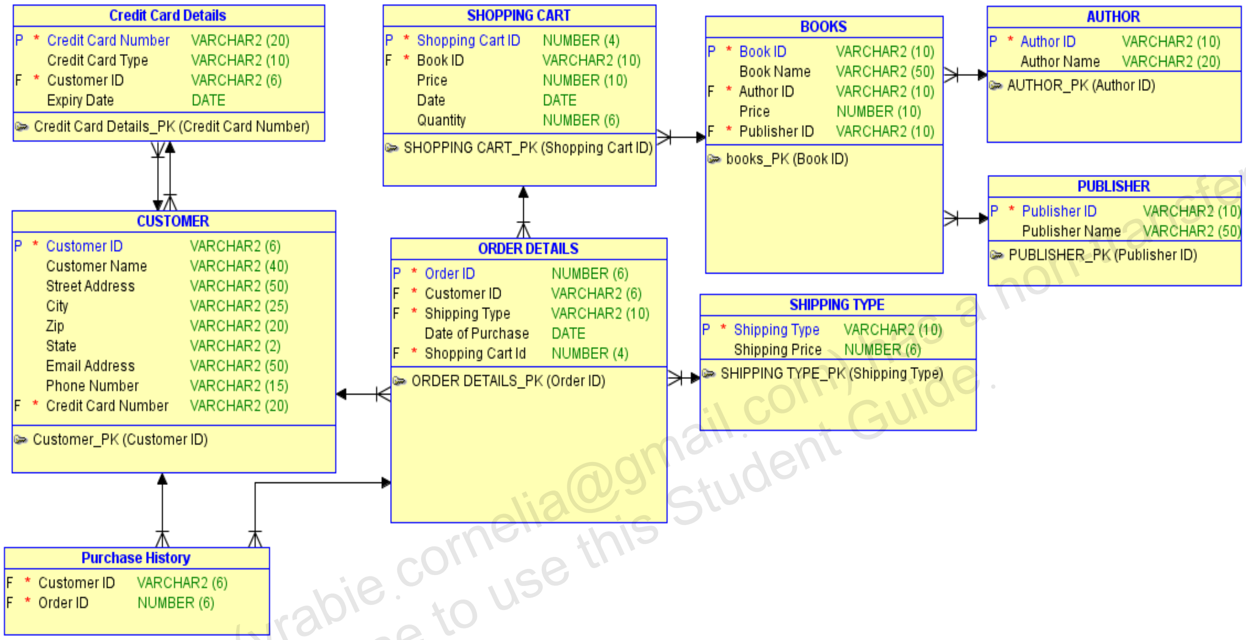
```
SELECT last_name, TO_CHAR(hire_date, 'Month DD') BIRTHDAY
FROM   employees
ORDER BY TO_CHAR(hire_date, 'DDD');
```

Case Study: Online Book Store

Overview

In this case study, you build a set of database tables for an online book store (E-Commerce Shopping Cart). After you create the tables, you insert, update, and delete records in the book store database and generate a report. The database contains only the essential tables.

The following is a diagram of the table and columns for the online book store application:



Credit Card Details

P	*	Credit Card Number	VARCHAR2 (20)
		Credit Card Type	VARCHAR2 (10)
F	*	Customer ID	VARCHAR2 (6)
		Expiry Date	DATE

Credit Card Details_PK (Credit Card Number)

SHOPPING CART

P	*	Shopping Cart ID	NUMBER (4)
F	*	Book ID	VARCHAR2 (10)
		Price	NUMBER (10)
		Date	DATE
		Quantity	NUMBER (6)

SHOPPING CART_PK (Shopping Cart ID)

BOOKS

P	*	Book ID	VARCHAR2 (10)
		Book Name	VARCHAR2 (50)
F	*	Author ID	VARCHAR2 (10)
		Price	NUMBER (10)
F	*	Publisher ID	VARCHAR2 (10)

books_PK (Book ID)

AUTHOR

P	*	Author ID	VARCHAR2 (10)
		Author Name	VARCHAR2 (20)

AUTHOR_PK (Author ID)

PUBLISHER

P	*	Publisher ID	VARCHAR2 (10)
		Publisher Name	VARCHAR2 (50)

PUBLISHER_PK (Publisher ID)

CUSTOMER

P	*	Customer ID	VARCHAR2 (6)
		Customer Name	VARCHAR2 (40)
		Street Address	VARCHAR2 (50)
		City	VARCHAR2 (25)
		Zip	VARCHAR2 (20)
		State	VARCHAR2 (2)
		Email Address	VARCHAR2 (50)
		Phone Number	VARCHAR2 (15)
F	*	Credit Card Number	VARCHAR2 (20)

Customer_PK (Customer ID)

ORDER DETAILS

P	*	Order ID	NUMBER (6)
F	*	Customer ID	VARCHAR2 (6)
F	*	Shipping Type	VARCHAR2 (10)
		Date of Purchase	DATE
F	*	Shopping Cart Id	NUMBER (4)

ORDER DETAILS_PK (Order ID)

SHIPPING TYPE

P	*	Shipping Type	VARCHAR2 (10)
		Shipping Price	NUMBER (6)

SHIPPING TYPE_PK (Shipping Type)

Purchase History

F	*	Customer ID	VARCHAR2 (6)
F	*	Order ID	NUMBER (6)

Note: If you want to build the tables, you can execute the commands in the `Online_Book_Store_Create_Table.sql` script in SQL Developer. If you want to drop the tables, you can execute the commands in the `Online_Book_Store_Drop_Tables.sql` script in SQL Developer. Then you can execute the commands in the `<<Online_Book_Store_Populate.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 `Online_Book_Store_Create_Table.sql` script to build the tables, start with step 2.
- If you use the `Online_Book_Store_Drop_Tables.sql` script to remove the tables, start with step 1.
- If you use the `Online_Book_Store_Populate.sql` script to build and populate the tables, start with step 6.

Practice 1-2

Overview

In this practice, you create the tables based on the following table instance charts. Select the appropriate data types and be sure to add integrity constraints.

Tasks

1. **Table Details**

   a. Table Name: AUTHOR

Column	Data type	Key	Table Dependent Type
Author_ID	VARCHAR2	PK	
Author_Name	VARCHAR2		

   b. Table Name: BOOKS

Column	Datatype	Key	Table Dependent On
Book_ID	VARCHAR2	PK	
Book_Name	VARCHAR2		
Author_ID	VARCHAR2	FK	AUTHORS
Price	NUMBER		
Publisher_ID	VARCHAR2	FK	PUBLISHER

   c. Table Name: CUSTOMER

Column Name	Data type	Key	Table Dependent On
Customer_ID	VARCHAR2	PK	
Customer_Name	VARCHAR2		
Street_Address	VARCHAR2		
City	VARCHAR2		
Phone_Number	VARCHAR2		
Credit_Card_Number	VARCHAR2	FK	Credit_Card_Details

   d. CREDIT_CARD_DETAILS

Column Name	Data type	Key	Table Dependent On
Credit_Card_Number	VARCHAR2	PK	
Credit_Card_Type	VARCHAR2		
Expiry_Date	DATE		

   e. Table Name: ORDER_DETAILS

Column	Data type	Key	Table Dependent On
Order_ID	NUMBER	PK	
Customer_ID	VARCHAR2	FK	CUSTOMER
Shipping_Type	VARCHAR2	FK	SHIPPING_TYPE
Date_of_Purchase	DATE		
Shopping_Cart_ID	NUMBER	FK	SHOPPING_CART

   f.

Table Name: PUBLISHER

Column	Data type	Key	Table Dependent Type
Publisher_ID	VARCHAR2	PK	
Publisher_Name	VARCHAR2		

g. Table Name: PURCHASE_HISTORY

Column	Data type	Key	Table Dependent Type
Customer_ID	VARCHAR2	FK	CUSTOMER
Order_ID	NUMBER	FK	ORDER_DETAILS

h. Table Name: SHIPPING_TYPE

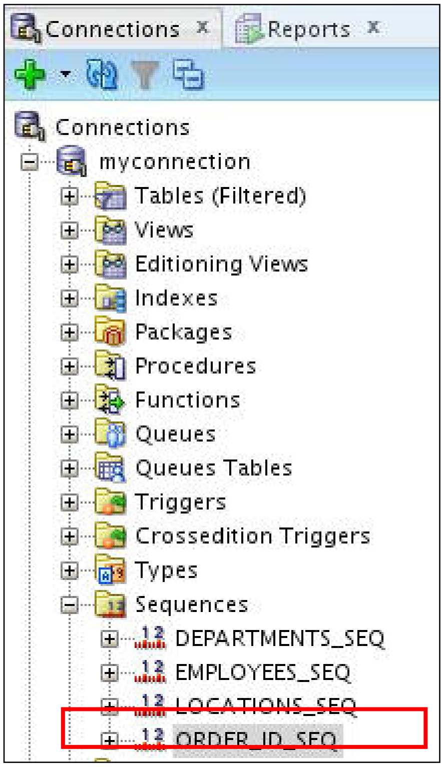
Column	Data type	Key	Table Dependent Type
Shipping_Type	VARCHAR2	PK	
Shipping_Price	NUMBER		

i. Table Name: SHOPPING_CART

Column	Data type	Key	Table Dependent On
Shopping_Cart_ID	NUMBER	PK	
Book_ID	VARCHAR2	FK	BOOKS
Price	NUMBER		
Date	DATE		
Quantity	NUMBER		

2. Add additional Referential Integrity constraints to the tables created.
3. Verify that the tables were created properly by checking in the Connections Navigator in SQL Developer.
4. Create a sequence to uniquely identify each row in the ORDER_DETAILS table.
   a. Start with 100; do not allow caching of the values. Name the sequence ORDER_ID_SEQ.

b. Verify the existence of the sequences in the Connections Navigator in SQL Developer.



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

Add data to the following tables:

a. AUTHOR

b. PUBLISHER

c. SHIPPING_TYPE

d. CUSTOMER

e. CREDIT_CARD_DETAILS

f. BOOKS

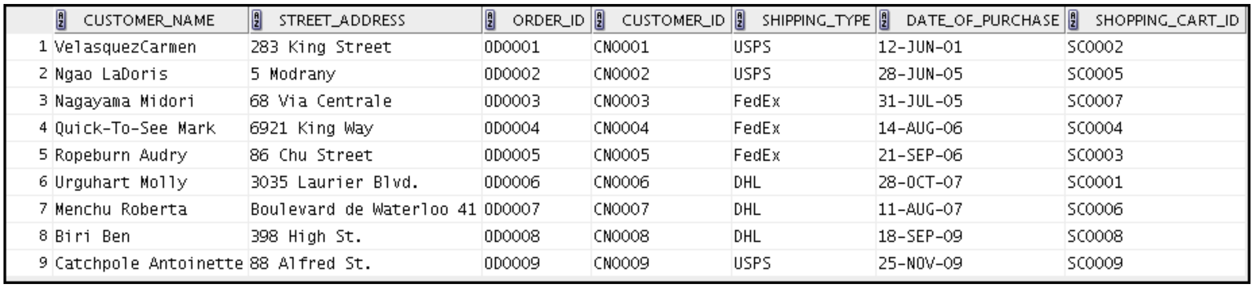
g. SHOPPING_CART

h. ORDER_DETAILS

i. PURCHASE_HISTORY

> **Note:** Save the scripts using the task number. For example, to save the script created for the BOOKS table, you can save it as labs_apcs_5a_1.sql. Ensure that you save the scripts in the /home/oracle/labs folder.

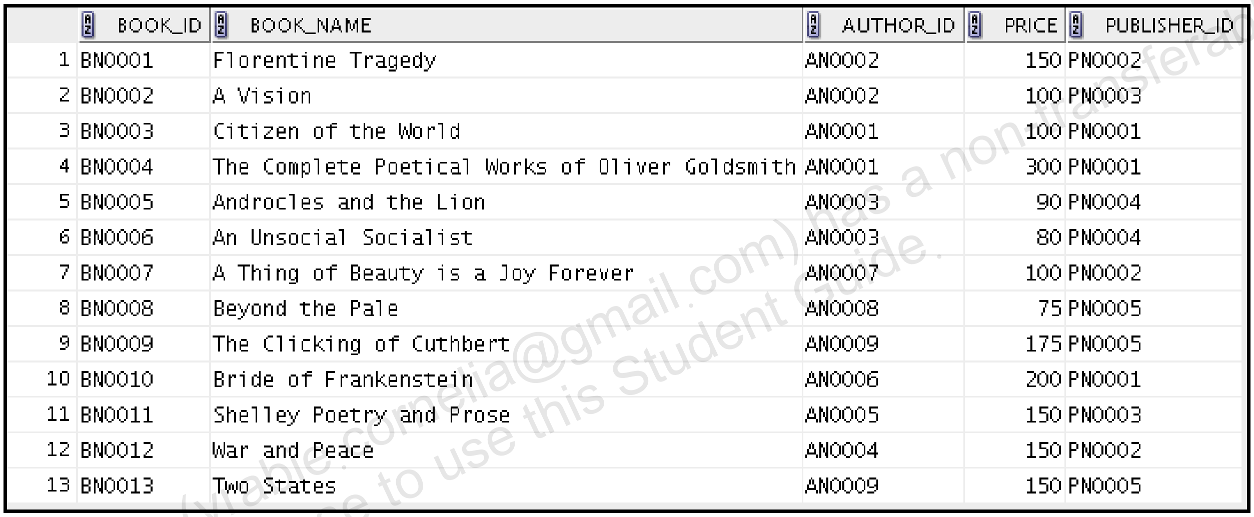
6. Create a view named `CUSTOMER_DETAILS` to show the Customer Name, Customer Address, and the details of the order placed by the customer. Order the results by Customer ID.



	CUSTOMER_NAME	STREET_ADDRESS	ORDER_ID	CUSTOMER_ID	SHIPPING_TYPE	DATE_OF_PURCHASE	SHOPPING_CART_ID
1	VelasquezCarmen	283 King Street	OD0001	CN0001	USPS	12-JUN-01	SC0002
2	Ngao LaDoris	5 Modrany	OD0002	CN0002	USPS	28-JUN-05	SC0005
3	Nagayama Midori	68 Via Centrale	OD0003	CN0003	FedEx	31-JUL-05	SC0007
4	Quick-To-See Mark	6921 King Way	OD0004	CN0004	FedEx	14-AUG-06	SC0004
5	Ropeburn Audry	86 Chu Street	OD0005	CN0005	FedEx	21-SEP-06	SC0003
6	Urguhart Molly	3035 Laurier Blvd.	OD0006	CN0006	DHL	28-OCT-07	SC0001
7	Menchu Roberta	Boulevard de Waterloo 41	OD0007	CN0007	DHL	11-AUG-07	SC0006
8	Biri Ben	398 High St.	OD0008	CN0008	DHL	18-SEP-09	SC0008
9	Catchpole Antoinette	88 Alfred St.	OD0009	CN0009	USPS	25-NOV-09	SC0009

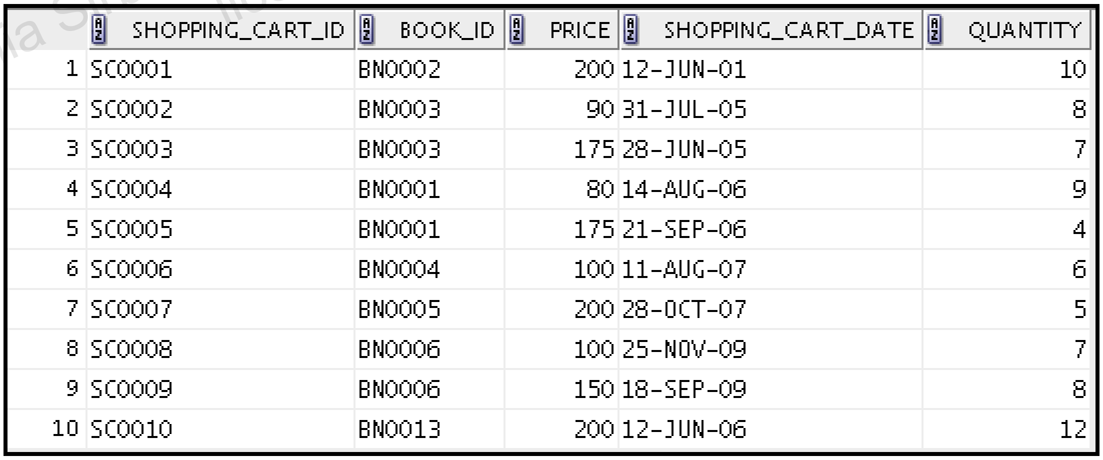
7. Make changes to the data in the tables.

   a. Add a new book detail. Verify if the author detail for the book is available in the `AUTHOR` table. If not, make an entry in the `AUTHOR` table.



	BOOK_ID	BOOK_NAME	AUTHOR_ID	PRICE	PUBLISHER_ID
1	BN0001	Florentine Tragedy	AN0002	150	PN0002
2	BN0002	A Vision	AN0002	100	PN0003
3	BN0003	Citizen of the World	AN0001	100	PN0001
4	BN0004	The Complete Poetical Works of Oliver Goldsmith	AN0001	300	PN0001
5	BN0005	Androcles and the Lion	AN0003	90	PN0004
6	BN0006	An Unsocial Socialist	AN0003	80	PN0004
7	BN0007	A Thing of Beauty is a Joy Forever	AN0007	100	PN0002
8	BN0008	Beyond the Pale	AN0008	75	PN0005
9	BN0009	The Clicking of Cuthbert	AN0009	175	PN0005
10	BN0010	Bride of Frankenstein	AN0006	200	PN0001
11	BN0011	Shelley Poetry and Prose	AN0005	150	PN0003
12	BN0012	War and Peace	AN0004	150	PN0002
13	BN0013	Two States	AN0009	150	PN0005

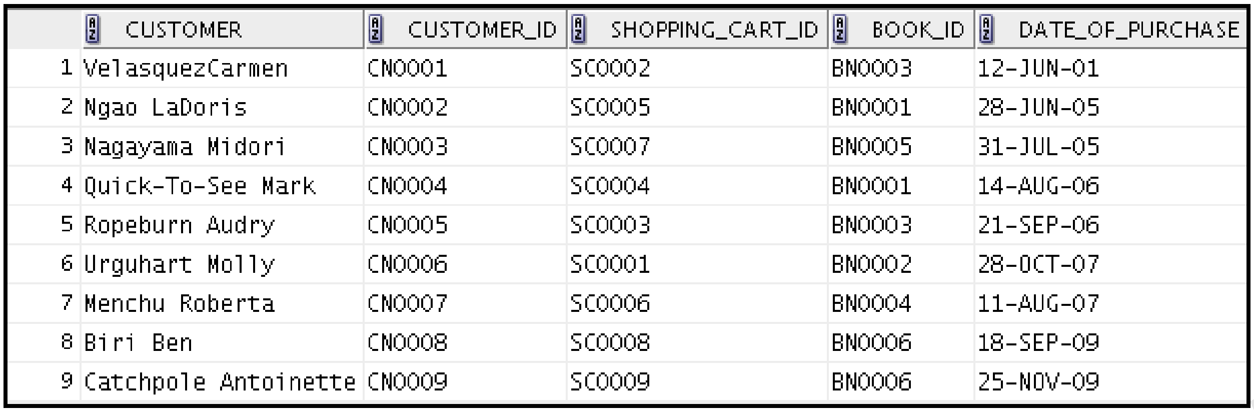
   b. Enter a shopping cart detail for the book details that you just entered in 7(a).



	SHOPPING_CART_ID	BOOK_ID	PRICE	SHOPPING_CART_DATE	QUANTITY
1	SC0001	BN0002	200	12-JUN-01	10
2	SC0002	BN0003	90	31-JUL-05	8
3	SC0003	BN0003	175	28-JUN-05	7
4	SC0004	BN0001	80	14-AUG-06	9
5	SC0005	BN0001	175	21-SEP-06	4
6	SC0006	BN0004	100	11-AUG-07	6
7	SC0007	BN0005	200	28-OCT-07	5
8	SC0008	BN0006	100	25-NOV-09	7
9	SC0009	BN0006	150	18-SEP-09	8
10	SC0010	BN0013	200	12-JUN-06	12

8. Create a report that contains each customer's history of purchasing books. Be sure to include the customer name, customer ID, book ID, date of purchase, and shopping cart ID. Save the commands that generate the report in a script file named `lab_apcs_8.sql`.

   Note: Your results may be different.



	CUSTOMER	CUSTOMER_ID	SHOPPING_CART_ID	BOOK_ID	DATE_OF_PURCHASE
1	VelasquezCarmen	CN0001	SC0002	BN0003	12-JUN-01
2	Ngao LaDoris	CN0002	SC0005	BN0001	28-JUN-05
3	Nagayama Midori	CN0003	SC0007	BN0005	31-JUL-05
4	Quick-To-See Mark	CN0004	SC0004	BN0001	14-AUG-06
5	Ropeburn Audry	CN0005	SC0003	BN0003	21-SEP-06
6	Urguhart Molly	CN0006	SC0001	BN0002	28-OCT-07
7	Menchu Roberta	CN0007	SC0006	BN0004	11-AUG-07
8	Biri Ben	CN0008	SC0008	BN0006	18-SEP-09
9	Catchpole Antoinette	CN0009	SC0009	BN0006	25-NOV-09

Solution 1-2

Overview

The solution to Practice 1-2 is given as follows.

Tasks

1. **Table Details**

   a. AUTHOR

```
 CREATE TABLE AUTHOR
    (
      Author_ID VARCHAR2 (10)  NOT NULL ,
      Author_Name VARCHAR2 (20)
    )
;




COMMENT ON TABLE AUTHOR IS 'Author'
;



ALTER TABLE AUTHOR
    ADD CONSTRAINT AUTHOR_PK PRIMARY KEY (Author_ID);
```

   b. BOOKS

```
  CREATE TABLE BOOKS
    (
     Book_ID VARCHAR2 (10)  NOT NULL ,
     Book_Name VARCHAR2 (50) ,
     Author_ID VARCHAR2 (10)  NOT NULL ,
     Price NUMBER (10) ,
     Publisher_ID VARCHAR2 (10)  NOT NULL
    )
;

COMMENT ON TABLE BOOKS IS 'Books'
;



ALTER TABLE BOOKS
    ADD CONSTRAINT books_PK PRIMARY KEY ( Book_ID );
```

c. CUSTOMER

```
    CREATE TABLE CUSTOMER
    (
      Customer_ID VARCHAR2 (6)  NOT NULL ,
      Customer_Name VARCHAR2 (40) ,
      Street_Address VARCHAR2 (50) ,
      City VARCHAR2 (25) ,
      Phone_Number VARCHAR2 (15) ,
      Credit_Card_Number VARCHAR2 (20)  NOT NULL
    )
;

COMMENT ON TABLE CUSTOMER IS 'Customer'
;


ALTER TABLE CUSTOMER
    ADD CONSTRAINT Customer_PK PRIMARY KEY ( Customer_ID ) ;
```

d. CREDIT_CARD_DETAILS

```
CREATE TABLE CREDIT_CARD_DETAILS
    (
     Credit_Card_Number VARCHAR2 (20)  NOT NULL ,
     Credit_Card_Type VARCHAR2 (10) ,
     Expiry_Date DATE
    )
;

COMMENT ON TABLE CREDIT_CARD_DETAILS IS 'Credit Card Details'
;


ALTER TABLE CREDIT_CARD_DETAILS
    ADD CONSTRAINT Credit_Card_Details_PK PRIMARY KEY (
Credit_Card_Number) ;
```

e. ORDER_DETAILS

```
CREATE TABLE ORDER_DETAILS
   (
 Order_ID VARCHAR2 (6)  NOT NULL ,
 Customer_ID VARCHAR2 (6)  NOT NULL ,
    Shipping_Type VARCHAR2 (10)  NOT NULL ,
 Date_of_Purchase DATE ,
    Shopping_Cart_ID varchar2(6)  NOT NULL
    )
;

    COMMENT ON TABLE ORDER_DETAILS IS 'Order Details'
;
ALTER TABLE ORDER_DETAILS
   ADD CONSTRAINT ORDER_DETAILS_PK PRIMARY KEY (Order_ID ) ;
```

f. PUBLISHER

```
CREATE TABLE PUBLISHER
    (
      Publisher_ID VARCHAR2 (10)  NOT NULL ,
      Publisher_Name VARCHAR2 (50)
    )
;




COMMENT ON TABLE PUBLISHER IS 'Publisher'
;



ALTER TABLE PUBLISHER
    ADD CONSTRAINT PUBLISHER_PK PRIMARY KEY ( Publisher_ID) ;
```

g. PURCHASE_HISTORY

```
     CREATE TABLE PURCHASE_HISTORY
     (
      Customer_ID VARCHAR2 (6)  NOT NULL ,
      Order_ID VARCHAR2 (6)  NOT NULL
     )
     ;

     COMMENT ON TABLE PURCHASE_HISTORY IS 'Purchase History'
     ;
```

h. SHIPPING_TYPE

```
CREATE TABLE SHIPPING_TYPE
    (
     Shipping_Type VARCHAR2 (10)  NOT NULL ,
     Shipping_Price NUMBER (6)
    )
;

COMMENT ON TABLE SHIPPING_TYPE IS 'Shipping Type'
;

ALTER TABLE SHIPPING_TYPE
    ADD CONSTRAINT SHIPPING_TYPE_PK PRIMARY KEY ( Shipping_Type
) ;
```

i. SHOPPING _CART

```
    CREATE TABLE SHOPPING_CART
    (
     Shopping_Cart_ID VARCHAR2 (6)  NOT NULL ,
     Book_ID VARCHAR2 (10)  NOT NULL ,
     Price NUMBER (10) ,
     Shopping_cart_Date DATE ,
     Quantity NUMBER (6)
    )
;

COMMENT ON TABLE SHOPPING_CART IS 'Shopping Cart'
;

ALTER TABLE SHOPPING_CART
ADD CONSTRAINT SHOPPING_CART_PK PRIMARY KEY (SHOPPING_CART_ID)
;
```

2. **Adding Additional Referential Integrity Constraints to the Table Created**

a. Include a Foreign Key constraint in the BOOKS table.

```
ALTER TABLE BOOKS
    ADD CONSTRAINT BOOKS_AUTHOR_FK FOREIGN KEY
    (
     Author_ID
    )
    REFERENCES AUTHOR
    (
      Author_ID
    )
;

ALTER TABLE BOOKS
    ADD CONSTRAINT BOOKS_PUBLISHER_FK FOREIGN KEY
    (
     Publisher_ID
    )
    REFERENCES PUBLISHER
    (
     Publisher_ID
);
```

b. Include a Foreign Key constraint in the `ORDER_DETAILS` table.

```
ALTER TABLE ORDER_DETAILS
    ADD CONSTRAINT Order_ID_FK FOREIGN KEY
    (
     Customer_ID
    )
    REFERENCES CUSTOMER
    (
     Customer_ID
    )
;


ALTER TABLE ORDER_DETAILS
    ADD CONSTRAINT FK_Order_details FOREIGN KEY
    (
     Shipping_Type
    )
    REFERENCES SHIPPING_TYPE
    (
     Shipping_Type
    )
;


ALTER TABLE ORDER_DETAILS
    ADD CONSTRAINT Order_Details_fk FOREIGN KEY
    (
      Shopping_Cart_ID
    )
    REFERENCES SHOPPING_CART
    (
     Shopping_Cart_ID
    )
;
```

c. Include a Foreign Key constraint in the PURCHASE_HISTORY table.

```
ALTER TABLE PURCHASE_HISTORY
           ADD CONSTRAINT Pur_Hist_ORDER_DETAILS_FK FOREIGN KEY
          (
           Order_ID
          )
        REFERENCES ORDER_DETAILS
        (
         Order_ID
    )
;
ALTER TABLE PURCHASE_ HISTORY
    ADD CONSTRAINT Purchase_History_CUSTOMER_FK FOREIGN KEY
    (
     Customer_ID
    )
    REFERENCES CUSTOMER
    (
     Customer_ID
    ) ;
```

d. Include a Foreign Key constraint in the SHOPPING_CART table.

```
ALTER TABLE SHOPPING_CART
    ADD CONSTRAINT SHOPPING_CART_BOOKS_FK FOREIGN KEY
    (
     Book_ID
    )
    REFERENCES BOOKS
    (
      Book_ID
    )
;
```

3. Verify that the tables were created properly by checking in the Connections Navigator in SQL Developer. In the Connections Navigator, expand Connections > myconnection > Tables.

4. Create a sequence to uniquely identify each row in the `ORDER DETAILS` table.

   a. Start with 100; do not allow caching of the values. Name the sequence `ORDER_ID_SEQ`.

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

   b. Verify the existence of the sequences in the Connections Navigator in SQL Developer.

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

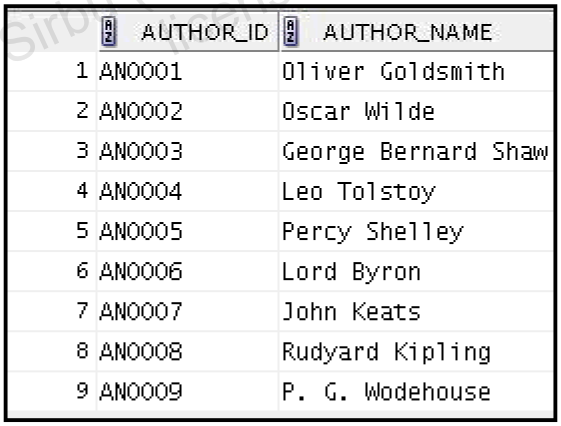
   Alternatively, you can also query the `user_sequences` data dictionary view:

```
SELECT * FROM user_sequences;
```

5. Add data to the tables.

   a. `AUTHOR` Table

Author_ID	Author_Name
AN0001	Oliver Goldsmith
AN0002	Oscar Wilde
AN0003	George Bernard Shaw
AN0004	Leo Tolstoy
AN0005	Percy Shelley
AN0006	Lord Byron
AN0007	John Keats
AN0008	Rudyard Kipling
AN0009	P. G. Wodehouse

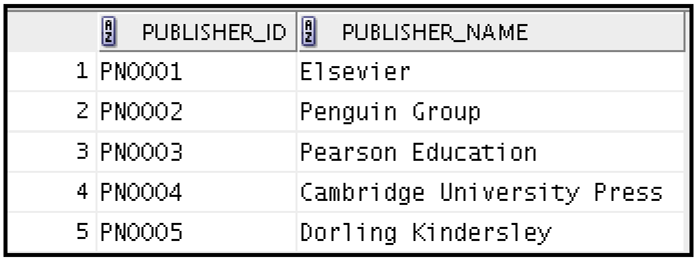


	AUTHOR_ID	AUTHOR_NAME
1	AN0001	Oliver Goldsmith
2	AN0002	Oscar Wilde
3	AN0003	George Bernard Shaw
4	AN0004	Leo Tolstoy
5	AN0005	Percy Shelley
6	AN0006	Lord Byron
7	AN0007	John Keats
8	AN0008	Rudyard Kipling
9	AN0009	P. G. Wodehouse

b.

PUBLISHER Table

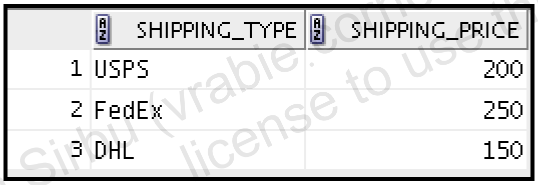
Publisher_ID	Publisher_Name
PN0001	Elsevier
PN0002	Penguin Group
PN0003	Pearson Education
PN0004	Cambridge University Press
PN0005	Dorling Kindersley



	PUBLISHER_ID	PUBLISHER_NAME
1	PN0001	Elsevier
2	PN0002	Penguin Group
3	PN0003	Pearson Education
4	PN0004	Cambridge University Press
5	PN0005	Dorling Kindersley

c. SHIPPING _TYPE

Shipping_Type	Shipping_Price
USPS	200
FedEx	250
DHL	150

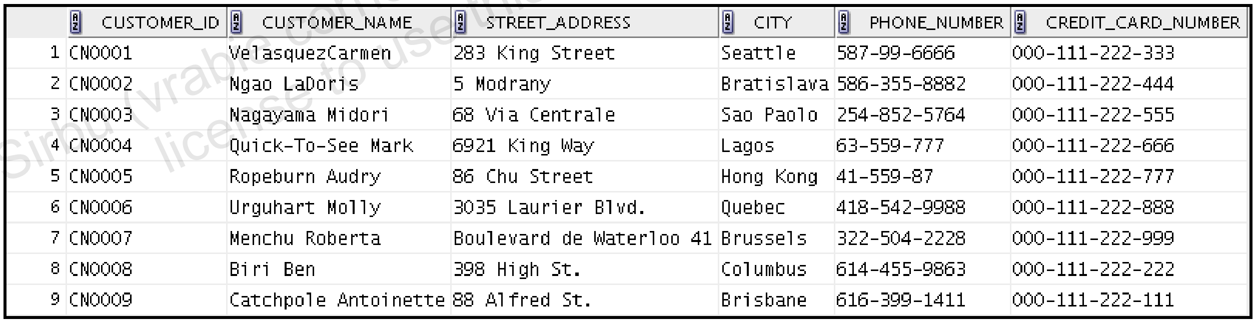


	SHIPPING_TYPE	SHIPPING_PRICE
1	USPS	200
2	FedEx	250
3	DHL	150

d.

CUSTOMER

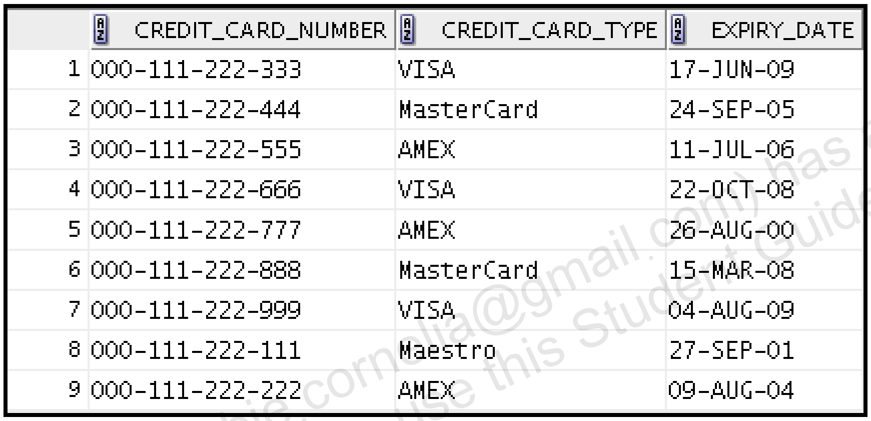
Customer _ ID	Customer _Name	Street _Address	City	Phone _number	Credit _Card _Number
CN0001	VelasquezCarmen	283 King Street	Seattle	587-99-6666	000-111-222-333
CN0002	Ngao LaDoris	5 Modrany	Bratislava	586-355-8882	000-111-222-444
CN0003	Nagayama Midori	68 Via Centrale	Sao Paolo	254-852-5764	000-111-222-555
CN0004	Quick-To-See Mark	6921 King Way	Lagos	63-559-777	000-111-222-666
CN0005	Ropeburn Audry	86 Chu Street	Hong Kong	41-559-87	000-111-222-777
CN0006	Urguhart Molly	3035 Laurier Blvd.	Quebec	418-542-9988	000-111-222-888
CN0007	Menchu Roberta	Boulevard de Waterloo 41	Brussels	322-504-2228	000-111-222-999
CN0008	Biri Ben	398 High St.	Columbus	614-455-9863	000-111-222-222
CN0009	Catchpole Antoinette	88 Alfred St.	Brisbane	616-399-1411	000-111-222-111



	CUSTOMER_ID	CUSTOMER_NAME	STREET_ADDRESS	CITY	PHONE_NUMBER	CREDIT_CARD_NUMBER
1	CN0001	VelasquezCarmen	283 King Street	Seattle	587-99-6666	000-111-222-333
2	CN0002	Ngao LaDoris	5 Modrany	Bratislava	586-355-8882	000-111-222-444
3	CN0003	Nagayama Midori	68 Via Centrale	Sao Paolo	254-852-5764	000-111-222-555
4	CN0004	Quick-To-See Mark	6921 King Way	Lagos	63-559-777	000-111-222-666
5	CN0005	Ropeburn Audry	86 Chu Street	Hong Kong	41-559-87	000-111-222-777
6	CN0006	Urguhart Molly	3035 Laurier Blvd.	Quebec	418-542-9988	000-111-222-888
7	CN0007	Menchu Roberta	Boulevard de Waterloo 41	Brussels	322-504-2228	000-111-222-999
8	CN0008	Biri Ben	398 High St.	Columbus	614-455-9863	000-111-222-222
9	CN0009	Catchpole Antoinette	88 Alfred St.	Brisbane	616-399-1411	000-111-222-111

e.
CREDIT_CARD_DETAILS

Credit _Card_ Number	Credit _Card _Type	Expiry _Date
000-111-222-333	VISA	17-JUN-2009
000-111-222-444	MasterCard	24-SEP-2005
000-111-222-555	AMEX	11-JUL-2006
000-111-222-666	VISA	22-OCT-2008
000-111-222-777	AMEX	26-AUG-2000
000-111-222-888	MasterCard	15-MAR-2008
000-111-222-999	VISA	4-AUG-2009
000-111-222-111	Maestro	27-SEP-2001
000-111-222-222	AMEX	9-AUG-2004

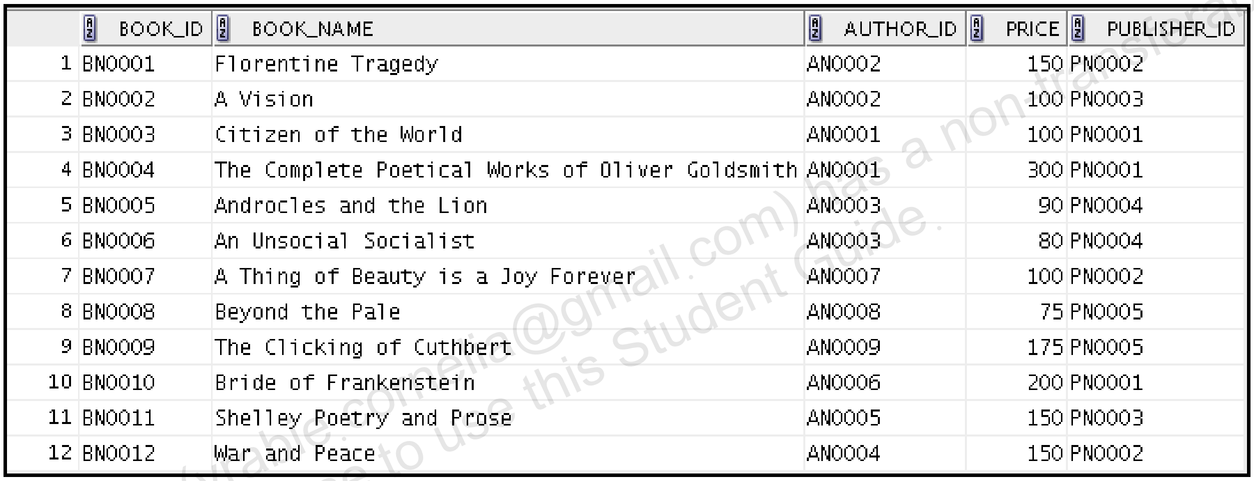


	CREDIT_CARD_NUMBER	CREDIT_CARD_TYPE	EXPIRY_DATE
1	000-111-222-333	VISA	17-JUN-09
2	000-111-222-444	MasterCard	24-SEP-05
3	000-111-222-555	AMEX	11-JUL-06
4	000-111-222-666	VISA	22-OCT-08
5	000-111-222-777	AMEX	26-AUG-00
6	000-111-222-888	MasterCard	15-MAR-08
7	000-111-222-999	VISA	04-AUG-09
8	000-111-222-111	Maestro	27-SEP-01
9	000-111-222-222	AMEX	09-AUG-04

f. BOOKS

Book _ID	Book _Name	Author _ID	Price	Publisher _ID
BN0001	Florentine Tragedy	AN0002	150	PN0002
BN0002	A Vision	AN0002	100	PN0003
BN0003	Citizen of the World	AN0001	100	PN0001
BN0004	The Complete Poetical Works of Oliver Goldsmith	AN0001	300	PN0001
BN0005	Androcles and the Lion	AN0003	90	PN0004
BN0006	An Unsocial Socialist	AN0003	80	PN0004
BN0007	A Thing of Beauty is a Joy Forever	AN0007	100	PN0002

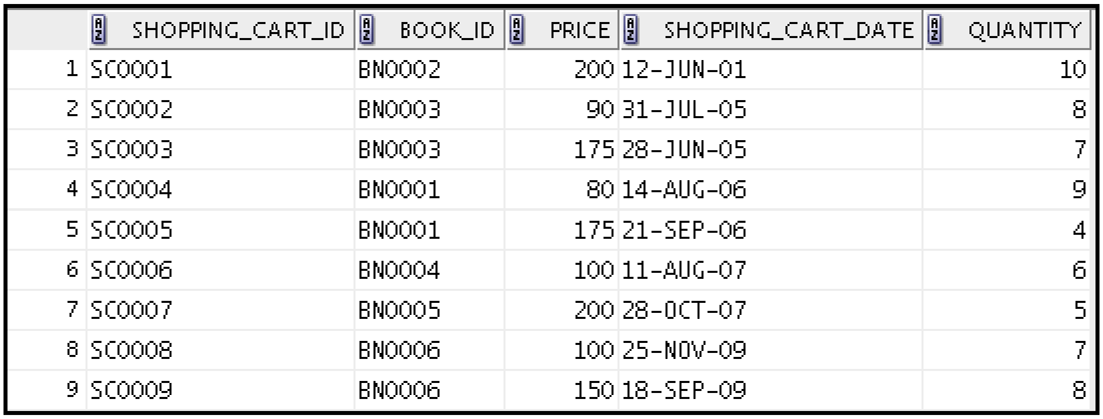
BN0008	Beyond the Pale	AN0008	75	PN0005
BN0009	The Clicking of Cuthbert	AN0009	175	PN0005
BN00010	Bride of Frankenstein	AN0006	200	PN0001
BN00011	Shelley's Poetry and Prose	AN0005	150	PN0003
BN00012	War and Peace	AN0004	150	PN0002



	BOOK_ID	BOOK_NAME	AUTHOR_ID	PRICE	PUBLISHER_ID
1	BN0001	Florentine Tragedy	AN0002	150	PN0002
2	BN0002	A Vision	AN0002	100	PN0003
3	BN0003	Citizen of the World	AN0001	100	PN0001
4	BN0004	The Complete Poetical Works of Oliver Goldsmith	AN0001	300	PN0001
5	BN0005	Androcles and the Lion	AN0003	90	PN0004
6	BN0006	An Unsocial Socialist	AN0003	80	PN0004
7	BN0007	A Thing of Beauty is a Joy Forever	AN0007	100	PN0002
8	BN0008	Beyond the Pale	AN0008	75	PN0005
9	BN0009	The Clicking of Cuthbert	AN0009	175	PN0005
10	BN0010	Bride of Frankenstein	AN0006	200	PN0001
11	BN0011	Shelley Poetry and Prose	AN0005	150	PN0003
12	BN0012	War and Peace	AN0004	150	PN0002

g. SHOPPING_CART

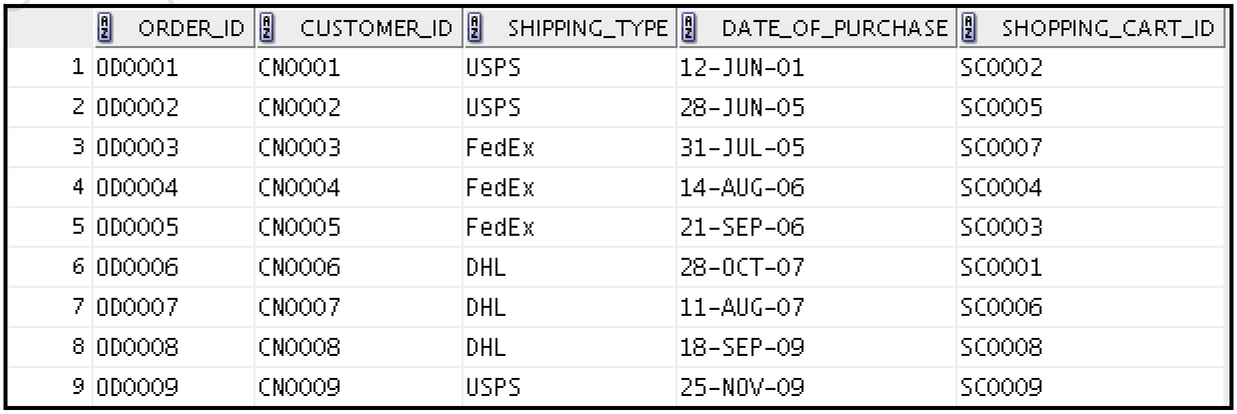
Shopping _Cart _ID	Book _ID	Price	Shopping _Cart _Date	Quantity
SC0001	BN0002	200	12-JUN-2001	10
SC0002	BN0003	90	31-JUL-2004	8
SC0003	BN0003	175	28-JUN-2005	7
SC0004	BN0001	80	14-AUG-2006	9
SC0005	BN0001	175	21-SEP-2006	4
SC0006	BN0004	100	11-AUG-2007	6
SC0007	BN0005	200	28-OCT-2007	5
SC0008	BN0006	100	25-NOV-2009	7
SC0009	BN0006	150	18-SPET-2009	8



	SHOPPING_CART_ID	BOOK_ID	PRICE	SHOPPING_CART_DATE	QUANTITY
1	SC0001	BN0002	200	12-JUN-01	10
2	SC0002	BN0003	90	31-JUL-05	8
3	SC0003	BN0003	175	28-JUN-05	7
4	SC0004	BN0001	80	14-AUG-06	9
5	SC0005	BN0001	175	21-SEP-06	4
6	SC0006	BN0004	100	11-AUG-07	6
7	SC0007	BN0005	200	28-OCT-07	5
8	SC0008	BN0006	100	25-NOV-09	7
9	SC0009	BN0006	150	18-SEP-09	8

h. ORDER _DETAILS

Order _ID	Customer _ID	Shipping_ Type	Date _of _Purchase	Shopping _Cart _ID
OD0001	CN0001	USPS	12-JUN-2001	SC0002
OD0002	CN0002	USPS	28-JUN-2005	SC0005
OD0003	CN0003	FedEx	31-JUL-2004	SC0007
OD0004	CN0004	FedEx	14-AUG-2006	SC0004
OD0005	CN0005	FedEx	21-SEP-2006	SC0003
OD0006	CN0006	DHL	28-OCT-2007	SC0001
OD0007	CN0007	DHL	11-AUG-2007	SC0006
OD0008	CN0008	DHL	18-SEP-2009	SC0008
OD0009	CN0009	USPS	25-NOV-2009	SC0009

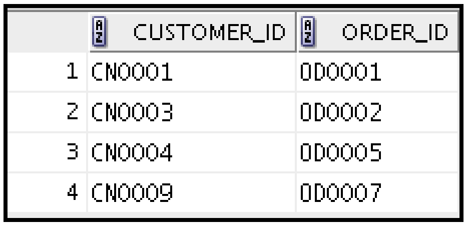


	ORDER_ID	CUSTOMER_ID	SHIPPING_TYPE	DATE_OF_PURCHASE	SHOPPING_CART_ID
1	OD0001	CN0001	USPS	12-JUN-01	SC0002
2	OD0002	CN0002	USPS	28-JUN-05	SC0005
3	OD0003	CN0003	FedEx	31-JUL-05	SC0007
4	OD0004	CN0004	FedEx	14-AUG-06	SC0004
5	OD0005	CN0005	FedEx	21-SEP-06	SC0003
6	OD0006	CN0006	DHL	28-OCT-07	SC0001
7	OD0007	CN0007	DHL	11-AUG-07	SC0006
8	OD0008	CN0008	DHL	18-SEP-09	SC0008
9	OD0009	CN0009	USPS	25-NOV-09	SC0009

i. PURCHASE_HISTORY

Customer _ID	Order _ID
CN0001	OD0001

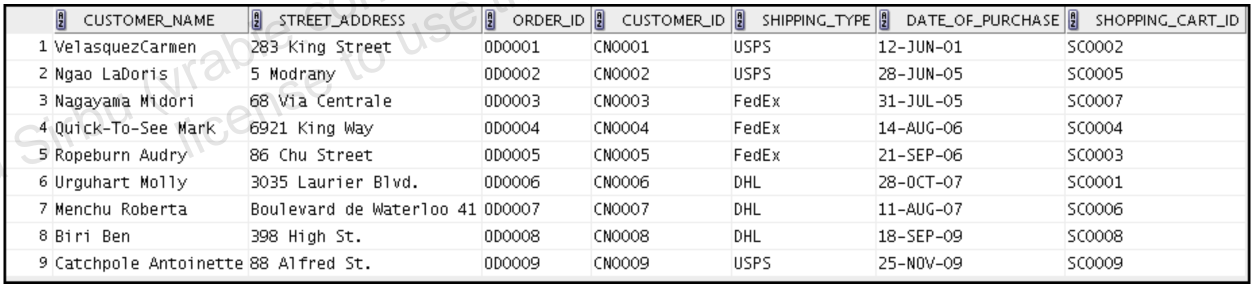
CN0003	OD0002
CN0004	OD0005
CN0009	OD0007



	CUSTOMER_ID	ORDER_ID
1	CN0001	OD0001
2	CN0003	OD0002
3	CN0004	OD0005
4	CN0009	OD0007

6. Create a view named CUSTOMER_DETAILS to show the Customer Name, Customer Address, and the details of the order placed by the customer. Order the results by Customer ID.

```
CREATE VIEW customer_details AS
   SELECT   c.customer_name, c.street_address, o.order_id,
o.customer_id, o.shipping_type, o.date_of_purchase,
o.shopping_cart_id
   FROM     customer c JOIN order_details o
   ON       c.customer_id = o.customer_id;

SELECT   *
FROM     customer_details
ORDER BY customer_id;
```

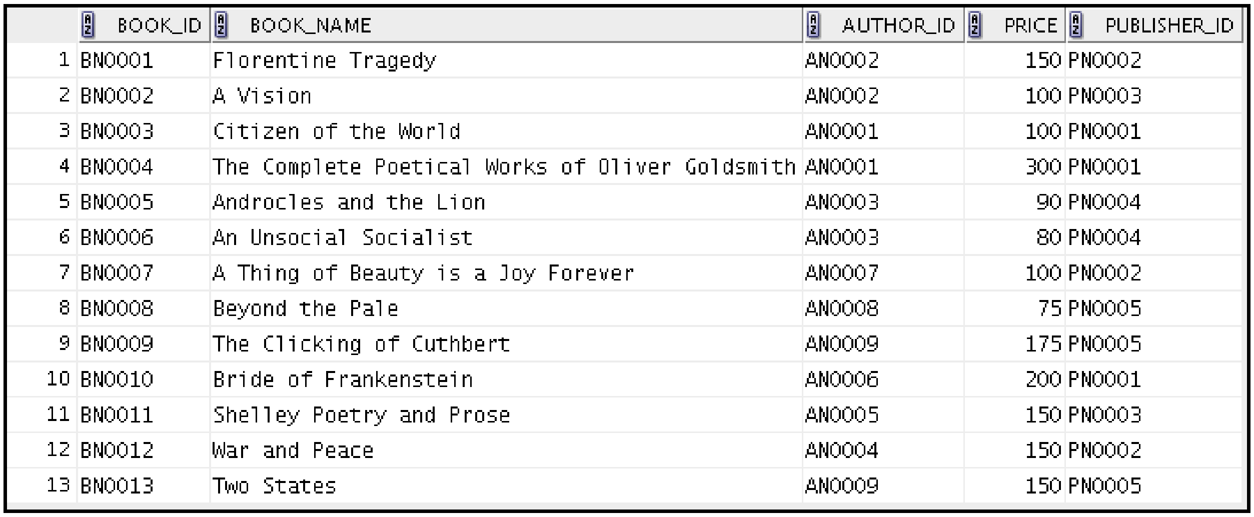


	CUSTOMER_NAME	STREET_ADDRESS	ORDER_ID	CUSTOMER_ID	SHIPPING_TYPE	DATE_OF_PURCHASE	SHOPPING_CART_ID
1	VelasquezCarmen	283 King Street	OD0001	CN0001	USPS	12-JUN-01	SC0002
2	Ngao LaDoris	5 Modrany	OD0002	CN0002	USPS	28-JUN-05	SC0005
3	Nagayama Midori	68 Via Centrale	OD0003	CN0003	FedEx	31-JUL-05	SC0007
4	Quick-To-See Mark	6921 King Way	OD0004	CN0004	FedEx	14-AUG-06	SC0004
5	Ropeburn Audry	86 Chu Street	OD0005	CN0005	FedEx	21-SEP-06	SC0003
6	Urguhart Molly	3035 Laurier Blvd.	OD0006	CN0006	DHL	28-OCT-07	SC0001
7	Menchu Roberta	Boulevard de Waterloo 41	OD0007	CN0007	DHL	11-AUG-07	SC0006
8	Biri Ben	398 High St.	OD0008	CN0008	DHL	18-SEP-09	SC0008
9	Catchpole Antoinette	88 Alfred St.	OD0009	CN0009	USPS	25-NOV-09	SC0009

7. Make changes to the data in the tables.

   a. Add a new book detail. Verify if the author detail for the book is available in the AUTHOR table. If not, make an entry in the AUTHOR table.

```
INSERT INTO books(book_id, book_name, author_id, price,
publisher_id)
VALUES ('BN0013','Two States','AN0009','150','PN0005');
```



	BOOK_ID	BOOK_NAME	AUTHOR_ID	PRICE	PUBLISHER_ID
1	BN0001	Florentine Tragedy	AN0002	150	PN0002
2	BN0002	A Vision	AN0002	100	PN0003
3	BN0003	Citizen of the World	AN0001	100	PN0001
4	BN0004	The Complete Poetical Works of Oliver Goldsmith	AN0001	300	PN0001
5	BN0005	Androcles and the Lion	AN0003	90	PN0004
6	BN0006	An Unsocial Socialist	AN0003	80	PN0004
7	BN0007	A Thing of Beauty is a Joy Forever	AN0007	100	PN0002
8	BN0008	Beyond the Pale	AN0008	75	PN0005
9	BN0009	The Clicking of Cuthbert	AN0009	175	PN0005
10	BN0010	Bride of Frankenstein	AN0006	200	PN0001
11	BN0011	Shelley Poetry and Prose	AN0005	150	PN0003
12	BN0012	War and Peace	AN0004	150	PN0002
13	BN0013	Two States	AN0009	150	PN0005

b. Enter a shopping cart detail for the book details that you just entered in 7(a).

```
INSERT INTO shopping_cart(shopping_cart_id, book_id, price,
Shopping_cart_date,quantity)
VALUES ('SC0010','BN0013','200',TO_DATE('12-JUN-2006','DD-MON-
YYYY'),'12');
```

8. Create a report that contains each customer's history of purchasing books. Be sure to include the customer name, customer ID, book ID, date of purchase, and shopping cart ID. Save the commands that generate the report in a script file named `lab_apcs_8.sql`.

Note: Your results may be different.

```
SELECT  c.customer_name CUSTOMER, c.customer_id,
s.shopping_cart_id, s.book_id,o.date_of_purchase
FROM  customer c
JOIN order_details o
ON   o.customer_id=c.customer_id
JOIN shopping_cart s
ON o.shopping_cart_id=s.shopping_cart_id;
```