Introduction to SQL Queries

Table of Contents

Course Practice Environment: SecurityCredentials

Chapter I

Course Practice Environment: Security Credentials

For connection-specific credentials used in this course, see the following table:

Connection-Specific Credentials		
Connection_Name	**Username**	**Password**
system	system	fenago
myconnection	hr	hr

Practices for Lesson 1:Introduction

Chapter 1

Practices for Lesson 1: Overview

Practice Overview

In this practice, you start Oracle 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 the practices for each lesson. 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 `hr` account.

- Use Oracle SQL Developer to examine the data objects in the `hr` account. The `hr` account contains the `HR` schema tables.

Note the following location for the practice files:

```
Desktop/oracle19c-labs/sql-intro/labs/sql1/labs
```

If you are asked to save any practice files, save them in the above 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: `hr`

   Password: `Enter the password from the Course Practice Environment: Security Credentials document`

   Hostname: `localhost`

   Port: `1521`

   Service Name: `FENAGO`

   Ensure that you select the Save Password check box.
3. Test the Oracle SQL Developer Database Connection and Connect to the Database
   a. Test the new connection.
   b. If the status is Success, connect to the database by using this new connection.
4. Browse 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. Start Oracle SQL Developer by Using the SQL Developer Desktop Icon
   Double-click the SQL Developer desktop icon.

   The SQL Developer interface appears.
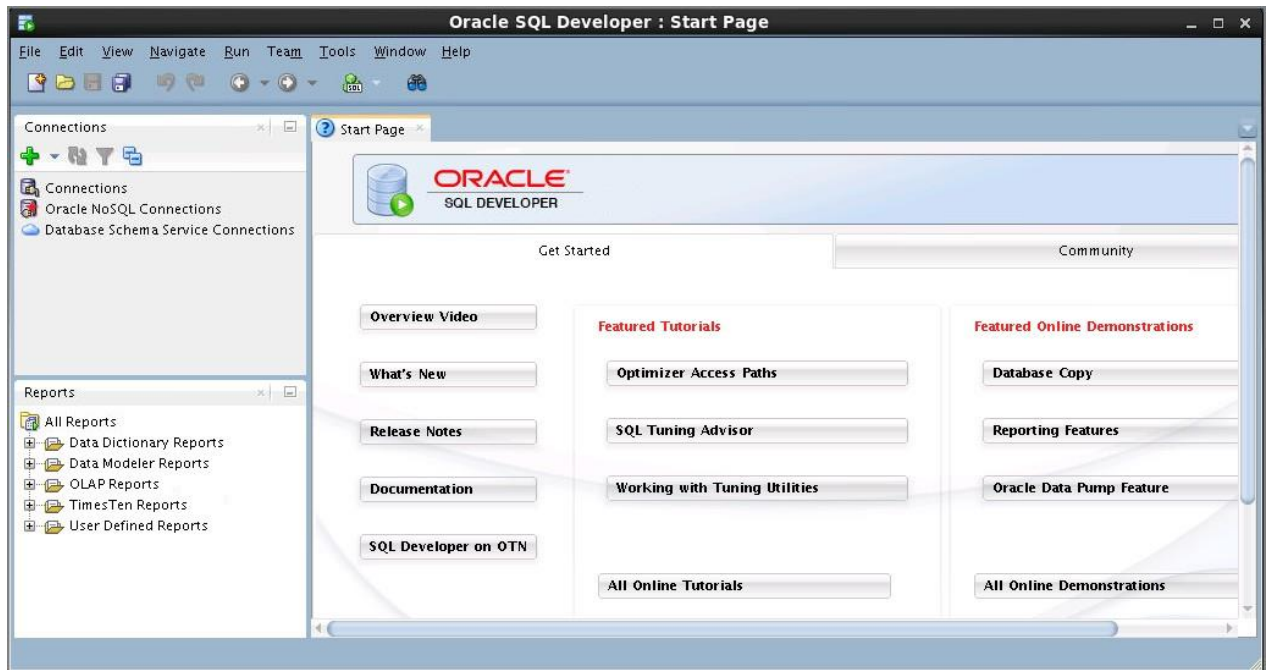
2. Create a New Oracle SQL Developer Database Connection
   a. To create a new database connection, in the Connections Navigator, right-click
      Connections and select New Connection from the context menu.
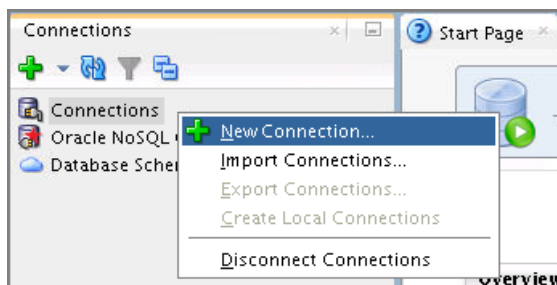
The New / Select Database Connection dialog box appears.

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

   i. Connection Name: `myconnection`

   ii. Username: `hr`

   iii.   Password: `Enter the password from the Course Practice Environment: Security Credentials document`

   iv.   Hostname: `localhost`

   v. Port: `1521`

   vi.   Service Name: `FENAGO`

   Ensure that you select the Save Password check box.

T

3. Test the Oracle SQL Developer Database Connection and Connect to the Database

   a. Test the new connection.

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

When you create a connection, a SQL Worksheet for that connection opens automatically.
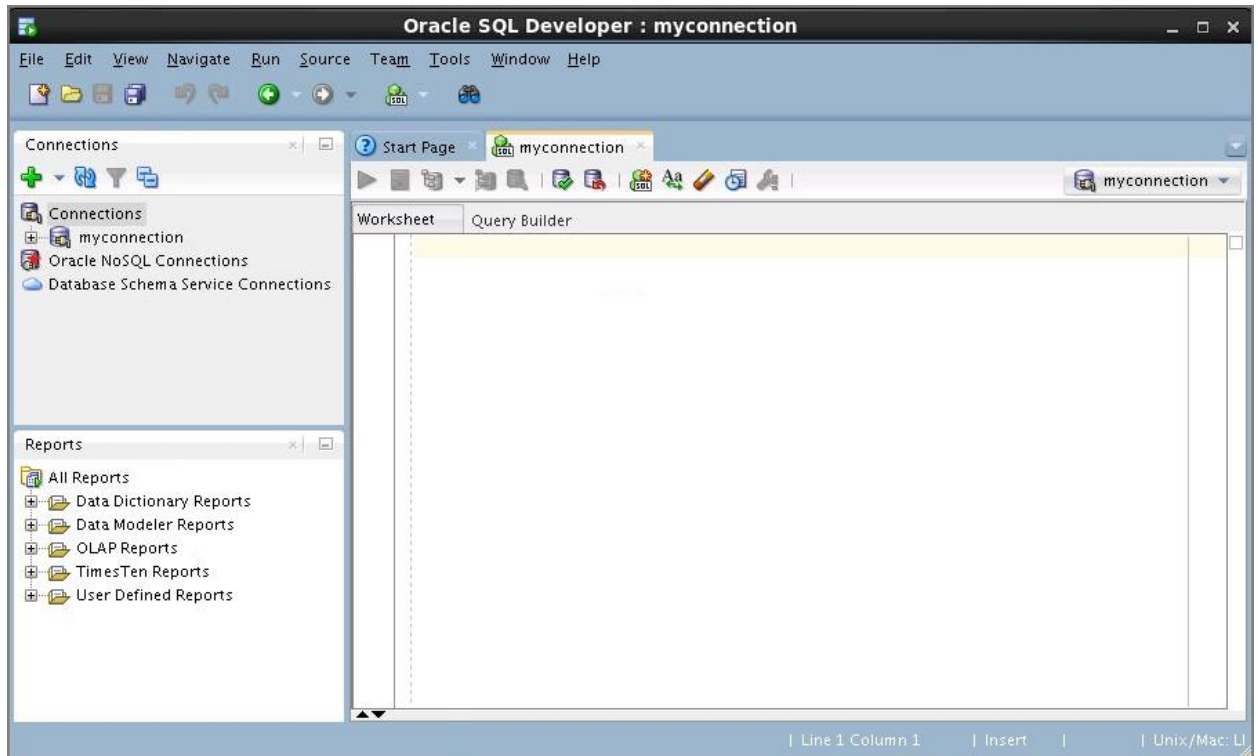
4. Browse the Tables in the Connections Navigator

   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.

Columns | Data | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
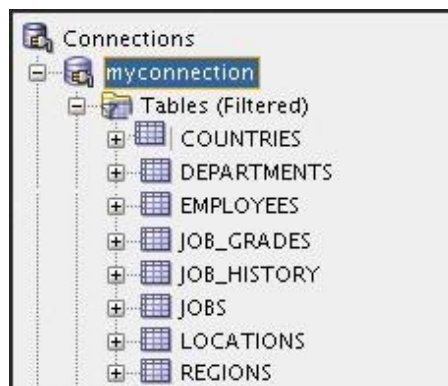
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	EMPLOYEE_ID	NUMBER(6,0)	No	(null)	1	Primary key of employees table.
2	FIRST_NAME	VARCHAR2(20 BYTE)	Yes	(null)	2	First name of the employee. A not null column.
3	LAST_NAME	VARCHAR2(25 BYTE)	No	(null)	3	Last name of the employee. A not null column.
4	EMAIL	VARCHAR2(25 BYTE)	No	(null)	4	Email id of the employee
5	PHONE_NUMBER	VARCHAR2(20 BYTE)	Yes	(null)	5	Phone number of the employee; includes country code and are
6	HIRE_DATE	DATE	No	(null)	6	Date when the employee started on this job. A not null col
7	JOB_ID	VARCHAR2(10 BYTE)	No	(null)	7	Current job of the employee; foreign key to job_id column
8	SALARY	NUMBER(8,2)	Yes	(null)	8	Monthly salary of the employee. Must be greaterthan zero (
9	COMMISSION_PCT	NUMBER(2,2)	Yes	(null)	9	Commission percentage of the employee; Only employees in s
10	MANAGER_ID	NUMBER(6,0)	Yes	(null)	10	Manager id of the employee; has same domain as manager_id
11	DEPARTMENT_ID	NUMBER(4,0)	Yes	(null)	11	Department id where employee works; foreign key to departm

c. View the data of the DEPARTMENTS table.

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

	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 theSQL 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

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:

  `Desktop/oracle19c-labs/sql-intro/labs/sql1/labs`

- Enter your SQL statements in a SQL Worksheet. To open a new worksheet, click File menu, select New. A New Gallery dialog window appears. Click Database Files under Database Tier on the left pane. Select SQL File on the right pane and click OK.

- 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 Run Statement icon in the SQL Worksheet. Alternatively, you can press F9. For DML and DDL statements, use the Run Script icon or press F5.

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

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

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

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

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

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-11,24000,,90
 2 101,Neena,Kochhar,NKOCHHAR,515.123.4568,AD_VP,100,21-SEP-09,17000,,90
 3 102,Lex,De Haan,LDEHAAN,515.123.4569,AD_VP,100,13-JAN-09,17000,,90
 4 103,Alexander,Hunold,AHUNOLD,590.423.4567,IT_PROG,102,03-JAN-14,9000,,60
 5 104,Bruce,Ernst,BERNST,590.423.4568,IT_PROG,103,21-MAY-15,6000,,60
 6 107,Diana,Lorentz,DLORENTZ,590.423.5567,IT_PROG,103,07-FEB-15,4200,,60
```

...

```
18 202,Pat,Fay,PFAY,603.123.6666,MK_REP,201,17-AUG-13,6000,,20
19 205,Shelley,Higgins,SHIGGINS,515.123.8080,AC_MGR,101,07-JUN-10,12008,,110
20 206,William,Gietz,WGIETZ,515.123.8181,AC_ACCOUNT,205,07-JUN-10,8300,,80
```

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 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 next 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

Practice 3-1: Restricting and Sorting Data

Overview

In this practice, you build reports by using statements with 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.

	LAST_NAME	DEPARTMENT_ID
1	Taylor	80

3. The HR department needs to find high-salaried and low-salaried 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-14
2	Taylor	SA_REP	24-MAR-14

5. Display the last name and department ID of all employees in department 20 or department 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 department 50. Label the columns `Employee` and `Monthly Salary`, respectively. Save `lab_03_03.sql` as `lab_03_06.sql`. 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 2010.

	LAST_NAME	HIRE_DATE
1	Higgins	07-JUN-10
2	Gietz	07-JUN-10

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`. 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-salaried and low-salaried 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 department 20 or department 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 department 50. Label the columns `Employee` and `Monthly Salary`, respectively. Save `lab_03_03.sql` as `lab_03_06.sql`. 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 2010.

```
SELECT     last_name, hire_date
FROM       employees
WHERE      hire_date >= '01-JAN-10' AND hire_date < '01-JAN-11';
```

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.

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

Enter 12000 when prompted for a value in a dialog box. Click OK.
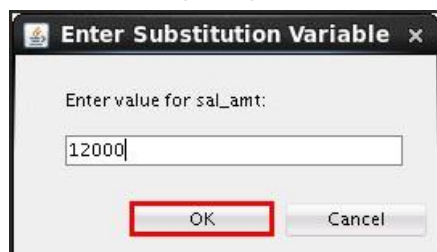
11. The HR department wants to run reports based on a manager. Create a query that prompts the user for a manager ID, and generates the employee ID, last name, salary, and department for that manager's employees. The HR department wants the ability to sort the report on a selected column. You can test 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`. 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:
UsingSingle-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 system date
- Creating queries that require the use of numeric, character, and date functions
- Performing calculations of years and months of service for an employee

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

Overview

In this practice, you use 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	14-JUL-16

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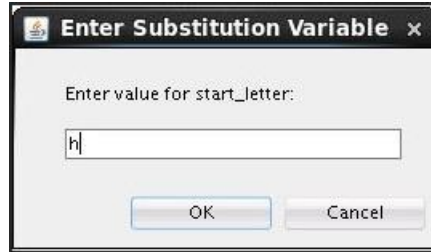
5. Perform the following tasks:
   a. Write a query that displays the last name (with the first letter in uppercase and all the other letters in lowercase) and the length of the last name for all employees whose name starts with the letters "A," or "M." Give each column an appropriate label. Sort the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

	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	6
2	Mourgos	8
3	Grant	14
4	Ernst	14
5	Lorentz	17
6	Vargas	24
7	Matos	28
8	Taylor	28
9	Hunold	30
10	Fay	35
11	Davies	42
12	Abel	50
13	Hartstein	53
14	Rajs	57
15	Whalen	58
16	King	61
17	Higgins	73
18	Gietz	73
19	Kochhar	82
20	De Haan	90

T

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 SALARIES_IN_ASTERISK.

	LAST_NAME	SALARIES_IN_ASTERISK
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	391
2	Kochhar	355
3	King	264

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 `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 "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 '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 SALARIES_IN_ASTERISK.

```
SELECT last_name,
       rpad(' ', (salary/1000)+1, '*')
              SALARIES_IN_ASTERISK
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: UsingConversion 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`

Practice 5-1: Using Conversion Functions and Conditional Expressions

Overview

In this practice, you use 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 $9,000.00 monthly but wants $27,000.00.
    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-11	Monday, the Nineteenth of December, 2011
2	Kochhar	21-SEP-09	Monday, the Twenty-Second of March, 2010
3	De Haan	13-JAN-09	Monday, the Twentieth of July, 2009
4	Hunold	03-JAN-14	Monday, the Seventh of July, 2014
5	Ernst	21-MAY-15	Monday, the Twenty-Third of November, 2015
6	Lorentz	07-FEB-15	Monday, the Tenth of August, 2015
7	Mourgos	16-NOV-15	Monday, the Twenty-Third of May, 2016
8	Rajs	17-OCT-11	Monday, the Twenty-Third of April, 2012
9	Davies	29-JAN-13	Monday, the Fifth of August, 2013
10	Matos	15-MAR-14	Monday, the Twenty-Second of September, 2014
11	Vargas	09-JUL-14	Monday, the Twelfth of January, 2015
12	Zlotkey	29-JAN-16	Monday, the First of August, 2016
13	Abel	11-MAY-12	Monday, the Twelfth of November, 2012
14	Taylor	24-MAR-14	Monday, the Twenty-Ninth of September, 2014
15	Grant	24-MAY-15	Monday, the Thirtieth of November, 2015
16	Whalen	17-SEP-11	Monday, the Nineteenth of March, 2012
17	Hartstein	17-FEB-12	Monday, the Twentieth of August, 2012
18	Fay	17-AUG-13	Monday, the Twenty-Fourth of February, 2014
19	Higgins	07-JUN-10	Monday, the Thirteenth of December, 2010
20	Gietz	07-JUN-10	Monday, the Thirteenth of December, 2010

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	0
2	AC_MGR	0
3	AD_ASST	0
4	AD_PRES	A
5	AD_VP	0
6	AD_VP	0
7	IT_PROG	C
8	IT_PROG	C
9	IT_PROG	C
10	MK_MAN	0
11	MK_REP	0
12	SA_MAN	0
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 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 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;
```

T

Practices for Lesson 6: Reporting Aggregated DataUsing 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

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

Overview

In this practice, you use group functions and select 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`. 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 a 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 2009, 2010, 2011, and 2012. Create appropriate column headings.

	TOTAL	2009	2010	2011	2012
1	20	2	2	3	2

11. Create a matrix query to display the job, the salary for that job based on the department numbers 20, 50, 80, and 90, and the total salary for that job. Ensure to give 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`. 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 a 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 2009, 2010, 2011, and 2012. Create appropriate column headings.

```
SELECT    COUNT(*) total,
          SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),2009,1,0))"2009",
          SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),2010,1,0))"2010",
          SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),2011,1,0))"2011",
          SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),2012,1,0))"2012"
FROM      employees;
```

11. Create a matrix query to display the job, the salary for that job based on the department
    numbers 20, 50, 80, and 90, and the total salary for that job. Ensure to give 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;
```

T

Practices for Lesson 7: Displaying Data from MultipleTables 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

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

Overview

In this practice, you extract data from multiple tables using 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	2012 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	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Davies	50	Shipping
5	Vargas	50	Shipping
6	Rajs	50	Shipping
7	Mourgos	50	Shipping
8	Matos	50	Shipping
9	Hunold	60	IT
10	Ernst	60	IT
11	Lorentz	60	IT
12	Taylor	80	Sales
13	Zlotkey	80	Sales
14	Abel	80	Sales
15	De Haan	90	Executive
16	King	90	Executive
17	Kochhar	90	Executive
18	Higgins	110	Accounting
19	Gietz	110	Accounting

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

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

4. Create a report to display employees' last 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	Ernst	104	Hunold	103
5	Lorentz	107	Hunold	103
6	Kochhar	101	King	100
7	De Haan	102	King	100
8	Mourgos	124	King	100
9	Zlotkey	149	King	100
10	Hartstein	201	King	100
11	Whalen	200	Kochhar	101
12	Higgins	205	Kochhar	101
13	Rajs	141	Mourgos	124
14	Davies	142	Mourgos	124
15	Matos	143	Mourgos	124
16	Vargas	144	Mourgos	124
17	Abel	174	Zlotkey	149
18	Taylor	176	Zlotkey	149
19	Grant	178	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
7	Mourgos	124	King	100
8	Rajs	141	Mourgos	124
9	Davies	142	Mourgos	124
10	Matos	143	Mourgos	124
11	Vargas	144	Mourgos	124
12	Zlotkey	149	King	100
13	Abel	174	Zlotkey	149
14	Taylor	176	Zlotkey	149
15	Grant	178	Zlotkey	149
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

…

	DEPARTMENT	EMPLOYEE	COLLEAGUE
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	Hunold	03-JAN-14
2	Ernst	21-MAY-15
3	Lorentz	07-FEB-15
4	Mourgos	16-NOV-15
5	Matos	15-MAR-14
6	Vargas	09-JUL-14
7	Zlotkey	29-JAN-16
8	Taylor	24-MAR-14
9	Grant	24-MAY-15
10	Fay	17-AUG-13

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	MANAGER	Manager_hire_date
1	Kochhar	21-SEP-09	King	17-JUN-11
2	De Haan	13-JAN-09	King	17-JUN-11
3	Rajs	17-OCT-11	Mourgos	16-NOV-15
4	Davies	29-JAN-13	Mourgos	16-NOV-15
5	Matos	15-MAR-14	Mourgos	16-NOV-15
6	Vargas	09-JUL-14	Mourgos	16-NOV-15
7	Abel	11-MAY-12	Zlotkey	29-JAN-16
8	Taylor	24-MAR-14	Zlotkey	29-JAN-16
9	Grant	24-MAY-15	Zlotkey	29-JAN-16

T

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 these 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 MANAGER,
m.hire_date "Manager_hire_date"
FROM   employees w JOIN employees m
ON     (w.manager_id = m.employee_id)
WHERE   w.hire_date <  m.hire_date;
```

T

Practices for Lesson 8: UsingSubqueries to Solve Queries

Chapter 8

T

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

Practice 8-1: Using Subqueries to Solve Queries
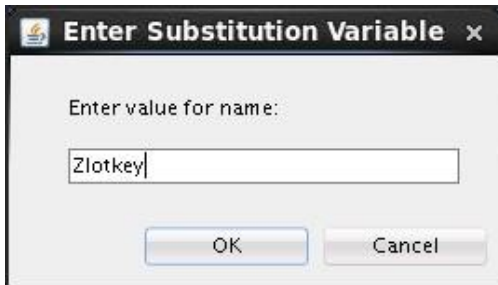
Overview

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).

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.

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 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 = '&&name')
AND     last_name <> '&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 =
&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: Usingthe Set Operators
Chapter 9

Practices for Lesson 9: Overview

Practice Overview

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

- UNION
- INTERSECT
- MINUS

Practice 9-1: Using Set Operators

Overview

In this practice, you write queries by 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 details 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

T

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 Using DMLStatements

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 the
`Desktop/oracle19c-labs/sql-intro/labs/sql1/code_ex`
`/cleanup_scripts/cleanup_10.sql` script.

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 DML statements, use the Run Script icon (or press F5) to execute the query. You can 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 the `cleanup_10.sql` script from `Desktop/oracle19c-labs/sql-intro/labs/sql1/code_ex /cleanup_scripts/` before performing the following tasks.

Tasks

1. Run the `lab_10_01.sql` script from `Desktop/oracle19c-labs/sql-intro/labs/sql1/labs/` to create the `MY_EMPLOYEE` table. 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)
```

2. 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

3. 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.
4. 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

5. Write an INSERT statement in a dynamic reusable script file to load the next two rows of sample data 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.
6. 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.
7. 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

8. Make the data additions permanent.

Update and delete data in the MY_EMPLOYEE table.
9. Change the last name of employee 3 to Drexler.
10. Change the salary to $1,000 for all employees who have a salary less than $900.

11. 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

12. Delete Betty Dancs from the MY_EMPLOYEE table.

13. 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

14. Commit all pending changes.

Control the data transaction to the MY_EMPLOYEE table.

15. 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.

16. 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

17. Mark an intermediate point in the processing of the transaction.

18. Delete all the rows from the MY_EMPLOYEE table.

19. Confirm that the table is empty.

20. Discard the most recent DELETE operation without discarding the earlier INSERT operation.

21. 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

22. Make the data addition permanent.

    If you have time, complete the following exercise:

23. 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.

24. Run the `lab_10_24.sql` script to insert the following record:

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
6	Anthony	Mark	manthony	1230

25. Confirm that the new row was added with the correct USERID.

Solution 10-1: Managing Tables by Using DML Statements

Insert data into the MY_EMPLOYEE table.

1. Run the lab_10_01.sql script from Desktop/oracle19c-labs/sql-intro/labs/sql1/labs/ to create the MY_EMPLOYEE table. Describe the structure of the MY_EMPLOYEE table to identify the column names.

```
DESCRIBE my_employee
```

2. 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

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

3. 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);
```

4. Confirm your additions to the table.

```
SELECT   *
FROM     my_employee;
```

5. Write an INSERT statement in a dynamic reusable script file to load the next two rows of sample data 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 (&id, '&last_name', '&first_name',
        '&userid', &salary);
```

6. 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 (&id, '&last_name', '&first_name',
        '&userid', &salary);
```

7. Confirm your additions to the table.

```
SELECT   *
FROM my_employee;
```

8. Make the data additions permanent.

```
COMMIT;
```

Update and delete data in the MY_EMPLOYEE table.

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

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

10. Change the salary to $1,000 for all employees with a salary less than $900.

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

11. Verify your changes to the table.

```
SELECT   *
FROM     my_employee;
```

12. Delete Betty Dancs from the MY_EMPLOYEE table.

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

13. Confirm your changes to the table.

```
SELECT   *
FROM     my_employee;
```

14. Commit all pending changes.

```
COMMIT;
```

Control the data transaction to the MY_EMPLOYEE table.

15. 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 (&id, '&last_name', '&first_name',
    '&userid', &salary);
```

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

16. Confirm your addition to the table.

```
SELECT   *
FROM     my_employee;
```

17. Mark an intermediate point in the processing of the transaction.

```
SAVEPOINT step_17;
```

18. Delete all the rows from the MY_EMPLOYEE table.

```
DELETE
FROM  my_employee;
```

19. Confirm that the table is empty.

```
SELECT *
FROM     my_employee;
```

20. Discard the most recent DELETE operation without discarding the earlier INSERT operation.

```
ROLLBACK TO step_17;
```

21. Confirm that the new row is still intact.

```
SELECT *
FROM     my_employee;
```

22. Make the data addition permanent.

```
COMMIT;
```

If you have time, complete the following exercise:

23. 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.

```
INSERT INTO my_employee
VALUES (&id, '&&last_name', '&&first_name',
    lower(substr('&first_name', 1, 1) ||
    substr('&last_name', 1, 7)), &salary);


UNDEFINE first_name
UNDEFINE last_name
```

24. Run the lab_10_24.sql script to insert the following record:

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
6	Anthony	Mark	manthony	1230

25. Confirm that the new row was added with the correct USERID.

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

T

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 `Desktop/oracle19c-labs/sql-intro/labs/sql1/code_ex/cleanup_scripts/cleanup_11.sql` script.

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.

Notes

- For all 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
  Desktop/oracle19c-labs/sql-intro/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

```
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.


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.


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.


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.


Name        Null Type
---------- ---- ------------
ID               NUMBER(7)
LAST_NAME        VARCHAR2(50)
COMMISSION       NUMBER(2,2)
```

7. Drop all of 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. Confirm that the table is created.

```
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 delete all rows in the `EMPLOYEES2` table.

   Note: You will get the "Update operation not allowed on table" error message. You will not be allowed to delete data from the table because it is assigned a `READ ONLY` status.

```
Error starting at line : 7 in command -
truncate table employees2
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 the READ WRITE status. Now try to truncate the rows again.

Now, because the table is assigned a READ WRITE status, you will be allowed to truncate rows in the table.

You should get the following messages:

```
Table EMPLOYEES2 altered.

Table EMPLOYEES2 truncated.
```


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.

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

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

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

```
DESCRIBE emp
```

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 of 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 delete all rows in the `EMPLOYEES2` table.

Note: You will get the "Update operation not allowed on table" error message. You will not be allowed to delete data from the table because it is assigned a `READ ONLY` status.

```
TRUNCATE TABLE employees2;
```

11. Revert the `EMPLOYEES2` table to the `READ WRITE` status. Now try to truncate the rows again.

Now, because the table is assigned a `READ WRITE` status, you will be allowed to truncate rows in the table.

```
ALTER TABLE employees2 READ WRITE;


TRUNCATE TABLE employees2;
SELECT * from employees2;
```

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

Note: You can even drop a table that is in the `READ ONLY` status. 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;
```

T

Additional Practices andSolutions
Chapter 12

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 2010.

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
1	141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-11	ST_CLERK	3500	(null)	124	50
2	142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-13	ST_CLERK	3100	(null)	124	50
3	143	Randall	Matos	RMATOS	650.121.2874	15-MAR-14	ST_CLERK	2600	(null)	124	50
4	144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-14	ST_CLERK	2500	(null)	124	50

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

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

	LAST_NAME
1	King
2	Kochhar
3	Lorentz
4	Matos
5	Mourgos

Additional Practices and Solutions

6. Create a report that displays all employees, and indicate whether they receive a commission by using the words *Yes* or *No*. 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-09
2	Hunold	03-JAN-14
3	Lorentz	07-FEB-15
4	Matos	15-MAR-14
5	Vargas	09-JUL-14
6	Abel	11-MAY-12
7	Higgins	07-JUN-10
8	Gietz	07-JUN-10

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, department name, number of employees, and average salary of all the departments, together with the employee 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	10033.3333333333333333333333333333333333
2	174	Abel	80	11000	10033.3333333333333333333333333333333333
3	144	Vargas	50	2500	3500
4	101	Kochhar	90	17000	19333.3333333333333333333333333333333333
5	100	King	90	24000	19333.3333333333333333333333333333333333
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	19333.3333333333333333333333333333333333
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	10033.3333333333333333333333333333333333
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

T

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 2010.

```
SELECT *
FROM    employees
WHERE   job_id  =  'ST_CLERK'
AND hire_date > '31-DEC-2010';
```

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 "K," "L," or "M."

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

6. Create a report that displays all employees, and indicate whether they receive a commission by using the words *Yes* or *No*. 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_IDs 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, department name, number of employees, and average salary of all departments, together with the employee 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:
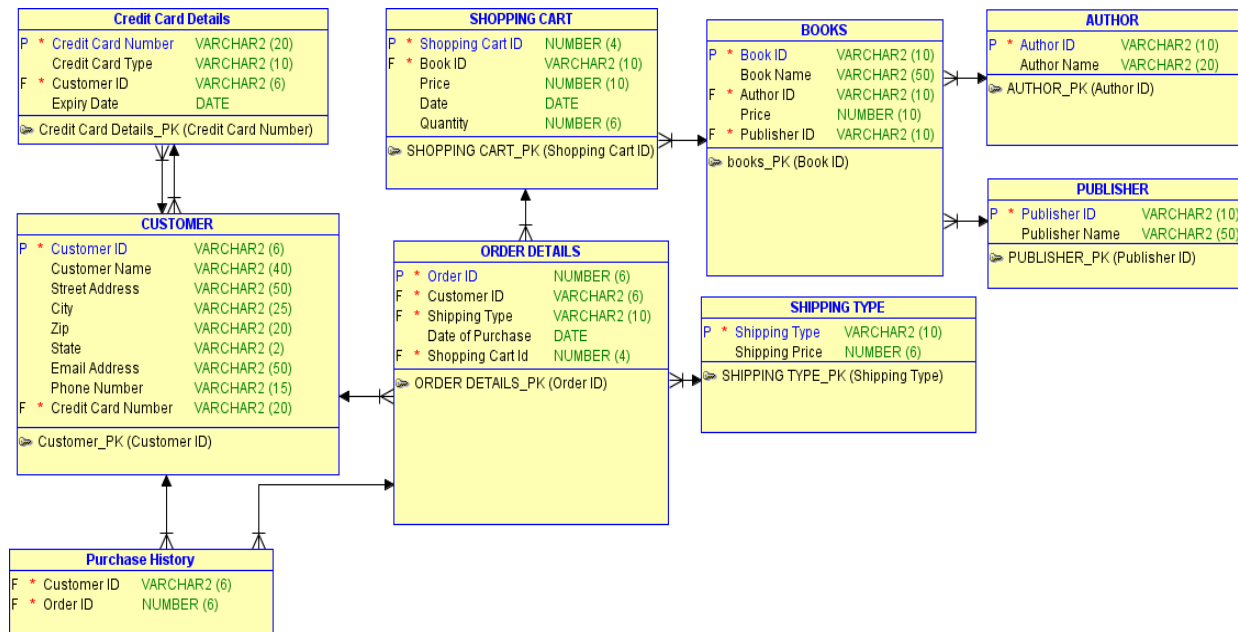
Note: If you want to build the tables, you can execute the commands in the `Online_Book_Store_Create_Table.sql` script in SQL Developer. If you want to drop the tables, you can execute the commands in the `Online_Book_Store_Drop_Tables.sql` script in SQL Developer. Then you can execute the commands in 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 `Desktop/oracle19c-labs/sql-intro/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		
Shopping_Cart_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.
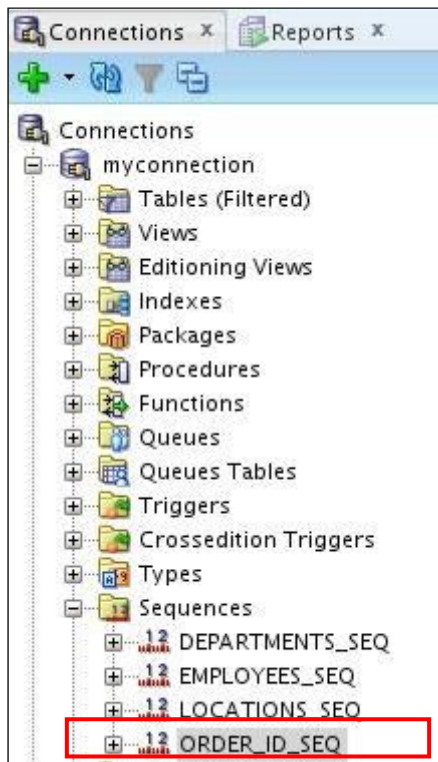
5. Add data to the tables. Create a script for each set of data to be added.
   Add data to the following tables:
   a. `AUTHOR`
   b. `PUBLISHER`
   c. `SHIPPING_TYPE`
   d. `CUSTOMER`
   e. `CREDIT_CARD_DETAILS`
   f. `BOOKS`
   g. `SHOPPING_CART`
   h. `ORDER_DETAILS`
   i. `PURCHASE_HISTORY`

Note: Save the scripts by using the task number. For example, to save the script created for the `AUTHOR` table, you can save it as `labs_apcs_5a_1.sql`. Ensure that you save the scripts in `/home/oracle/labs/sql1/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-11	SC0002
2	Ngao LaDoris	5 Modrany	OD0002	CN0002	USPS	28-JUN-15	SC0005
3	Nagayama Midori	68 Via Centrale	OD0003	CN0003	FedEx	31-JUL-14	SC0007
4	Quick-To-See Mark	6921 King Way	OD0004	CN0004	FedEx	14-AUG-16	SC0004
5	Ropeburn Audry	86 Chu Street	OD0005	CN0005	FedEx	21-SEP-16	SC0003
6	Urguhart Molly	3035 Laurier Blvd.	OD0006	CN0006	DHL	28-OCT-12	SC0001
7	Menchu Roberta	Boulevard de Waterloo 41	OD0007	CN0007	DHL	11-AUG-12	SC0006
8	Biri Ben	398 High St.	OD0008	CN0008	DHL	18-SEP-13	SC0008
9	Catchpole Antoinette	88 Alfred St.	OD0009	CN0009	USPS	25-NOV-13	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-11	10
2	SC0002	BN0003	90	31-JUL-14	8
3	SC0003	BN0003	175	28-JUN-15	7
4	SC0004	BN0001	80	14-AUG-16	9
5	SC0005	BN0001	175	21-SEP-16	4
6	SC0006	BN0004	100	11-AUG-12	6
7	SC0007	BN0005	200	28-OCT-12	5
8	SC0008	BN0006	100	25-NOV-13	7
9	SC0009	BN0006	150	18-SEP-13	8
10	SC0010	BN0013	200	12-JUN-16	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-11
2	Ngao LaDoris	CN0002	SC0005	BN0001	28-JUN-15
3	Nagayama Midori	CN0003	SC0007	BN0005	31-JUL-14
4	Quick-To-See Mark	CN0004	SC0004	BN0001	14-AUG-16
5	Ropeburn Audry	CN0005	SC0003	BN0003	21-SEP-16
6	Urguhart Molly	CN0006	SC0001	BN0002	28-OCT-12
7	Menchu Roberta	CN0007	SC0006	BN0004	11-AUG-12
8	Biri Ben	CN0008	SC0008	BN0006	18-SEP-13
9	Catchpole Antoinette	CN0009	SC0009	BN0006	25-NOV-13

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  (15)  ,
     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
    )
    ;
```

Additional Practices and Solutions

```
      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 CUSTOMER table.

```
ALTER TABLE CUSTOMER
    ADD CONSTRAINT CREDIT_CARD_FK FOREIGN KEY
    (
     Credit_Card_Number
    )
    REFERENCES CREDIT_CARD_DETAILS
    (
      Credit_Card_Number
    )
;
```

    b.   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
);
```

c. 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
    )
;
```

d. 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
```

Additional Practices and Solutions

```
            )
        REFERENCES ORDER_DETAILS
        (
          Order_ID
        )
;
ALTER TABLE PURCHASE_HISTORY
    ADD CONSTRAINT Purchase_History_CUSTOMER_FK FOREIGN KEY
    (
      Customer_ID
    )
    REFERENCES CUSTOMER
    (
      Customer_ID
    ) ;
```

e. 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	Platinum	17-JUN-2019
000-111-222-444	SpecialCard	24-SEP-2025
000-111-222-555	PayM	11-JUL-2017
000-111-222-666	Platinum	22-OCT-2018
000-111-222-777	PayM	26-AUG-2020
000-111-222-888	SpecialCard	15-MAR-2018
000-111-222-999	Platinum	4-AUG-2019
000-111-222-111	Maestro	27-SEP-2021
000-111-222-222	PayM	9-AUG-2024

	CREDIT_CARD_NUMBER	CREDIT_CARD_TYPE	EXPIRY_DATE
1	000-111-222-333	PLATINUM	17-JUN-19
2	000-111-222-444	SpecialCard	24-SEP-25
3	000-111-222-555	PayM	11-JUL-17
4	000-111-222-666	PLATINUM	22-OCT-18
5	000-111-222-777	PayM	26-AUG-20
6	000-111-222-888	SpecialCard	15-MAR-18
7	000-111-222-999	PLATINUM	04-AUG-19
8	000-111-222-111	Maestro	27-SEP-21
9	000-111-222-222	PayM	09-AUG-14

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-2011	10

SC0002	BN0003	90	31-JUL-2014	8
SC0003	BN0003	175	28-JUN-2015	7
SC0004	BN0001	80	14-AUG-2016	9
SC0005	BN0001	175	21-SEP-2016	4
SC0006	BN0004	100	11-AUG-2012	6
SC0007	BN0005	200	28-OCT-2012	5
SC0008	BN0006	100	25-NOV-2013	7
SC0009	BN0006	150	18-SPET-2013	8

	SHOPPING_CART_ID	BOOK_ID	PRICE	SHOPPING_CART_DATE	QUANTITY
1	SC0001	BN0002	200	12-JUN-11	10
2	SC0002	BN0003	90	31-JUL-14	8
3	SC0003	BN0003	175	28-JUN-15	7
4	SC0004	BN0001	80	14-AUG-16	9
5	SC0005	BN0001	175	21-SEP-16	4
6	SC0006	BN0004	100	11-AUG-12	6
7	SC0007	BN0005	200	28-OCT-12	5
8	SC0008	BN0006	100	25-NOV-13	7
9	SC0009	BN0006	150	18-SEP-13	8

h. `ORDER _DETAILS`

Order _ID	Customer _ID	Shipping_ Type	Date _of _Purchase	Shopping _Cart _ID
OD0001	CN0001	USPS	12-JUN-2011	SC0002
OD0002	CN0002	USPS	28-JUN-2015	SC0005
OD0003	CN0003	FedEx	31-JUL-2014	SC0007
OD0004	CN0004	FedEx	14-AUG-2016	SC0004
OD0005	CN0005	FedEx	21-SEP-2016	SC0003
OD0006	CN0006	DHL	28-OCT-2012	SC0001
OD0007	CN0007	DHL	11-AUG-2012	SC0006
OD0008	CN0008	DHL	18-SEP-2013	SC0008
OD0009	CN0009	USPS	25-NOV-2013	SC0009

	ORDER_ID	CUSTOMER_ID	SHIPPING_TYPE	DATE_OF_PURCHASE	SHOPPING_CART_ID
1	OD0001	CN0001	USPS	12-JUN-11	SC0002
2	OD0002	CN0002	USPS	28-JUN-15	SC0005
3	OD0003	CN0003	FedEx	31-JUL-14	SC0007
4	OD0004	CN0004	FedEx	14-AUG-16	SC0004
5	OD0005	CN0005	FedEx	21-SEP-16	SC0003
6	OD0006	CN0006	DHL	28-OCT-12	SC0001
7	OD0007	CN0007	DHL	11-AUG-12	SC0006
8	OD0008	CN0008	DHL	18-SEP-13	SC0008
9	OD0009	CN0009	USPS	25-NOV-13	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 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-11	SC0002
2	Ngao LaDoris	5 Modrany	OD0002	CN0002	USPS	28-JUN-15	SC0005
3	Nagayama Midori	68 Via Centrale	OD0003	CN0003	FedEx	31-JUL-14	SC0007
4	Quick-To-See Mark	6921 King Way	OD0004	CN0004	FedEx	14-AUG-16	SC0004
5	Ropeburn Audry	86 Chu Street	OD0005	CN0005	FedEx	21-SEP-16	SC0003
6	Urguhart Molly	3035 Laurier Blvd.	OD0006	CN0006	DHL	28-OCT-12	SC0001
7	Menchu Roberta	Boulevard de Waterloo 41	OD0007	CN0007	DHL	11-AUG-12	SC0006
8	Biri Ben	398 High St.	OD0008	CN0008	DHL	18-SEP-13	SC0008
9	Catchpole Antoinette	88 Alfred St.	OD0009	CN0009	USPS	25-NOV-13	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');

SELECT * FROM books;
```

	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-2016','DD-MON-
YYYY'),'12');

SELECT * FROM shopping_cart;
```

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;
```