Appendix A
Practices and Solutions

Table of Contents

Practices and Solutions for Lesson I

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

Practice I-1: Accessing SQL Developer Resources

In this practice, you do the following:

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

Practice I-2: Using SQL Developer

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

c) Place the file under your /home/oracle/labs/sql2/labs folder.

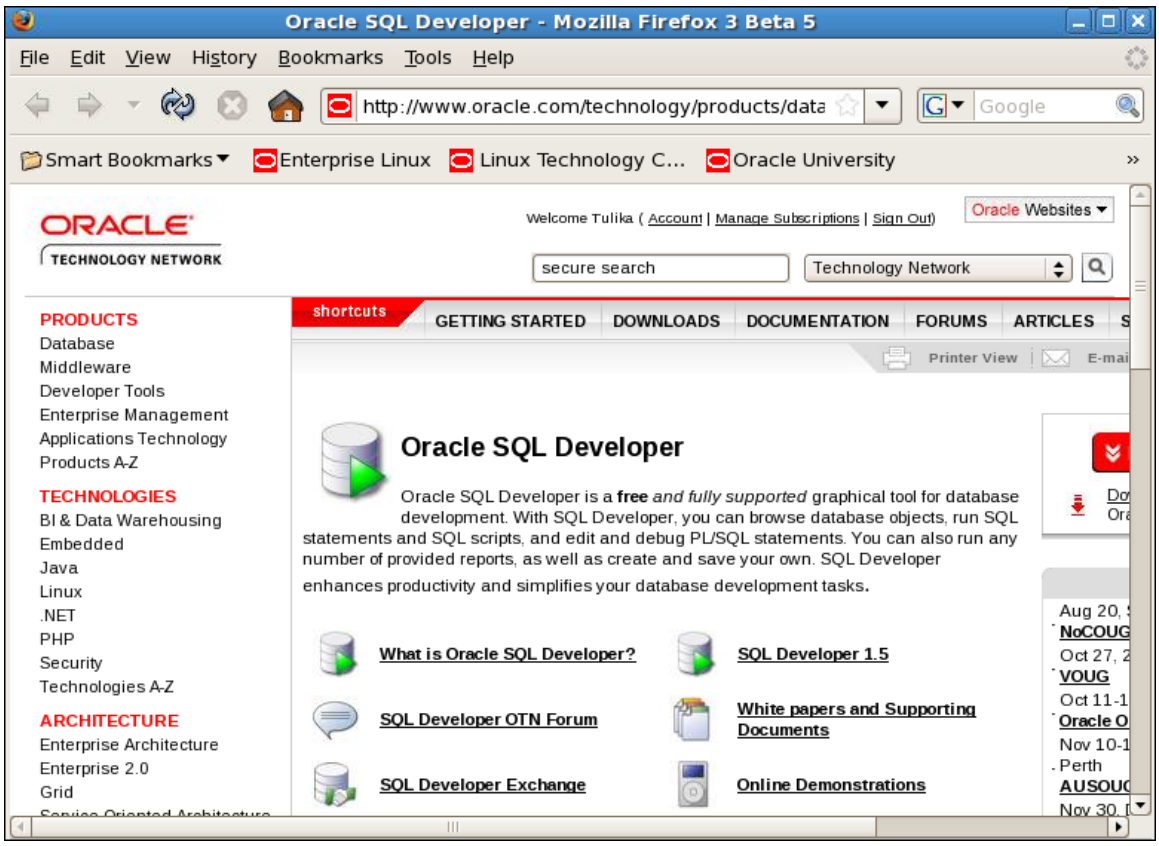
9) Open and run `confidence.sql` from your /home/oracle/labs/sql2/labs folder, and observe the output.

Practice Solutions I-1: Accessing SQL Developer Resources

1) Access the SQL Developer home page.

a) Access the online SQL Developer home page available online at:
http://www.oracle.com/technology/products/database/sql_developer/index.html

The SQL Developer home page is displayed as follows:



b) Bookmark the page for easier future access.

2) Access the SQL Developer tutorial available online at:
http://st-curriculum.oracle.com/tutorial/SQLDeveloper/index.htm

Then, review the following sections and associated demos:

a) What to Do First

b) Working with Database Objects

c) Accessing Data

Practice Solutions I-2: Using SQL Developer

1) Start SQL Developer by using the desktop icon.

2) Create a database connection using the following information:

a. Connection Name: `myconnection`

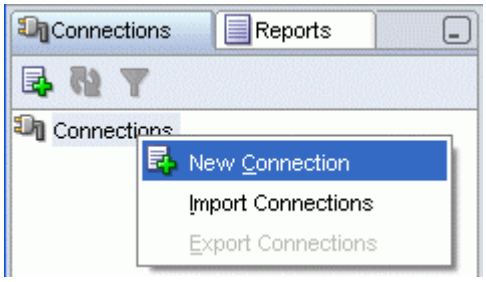
b. Username: `oraxx` (Ask your instructor to assign you one `ora` account out of the `ora21–ora40` range of accounts.)

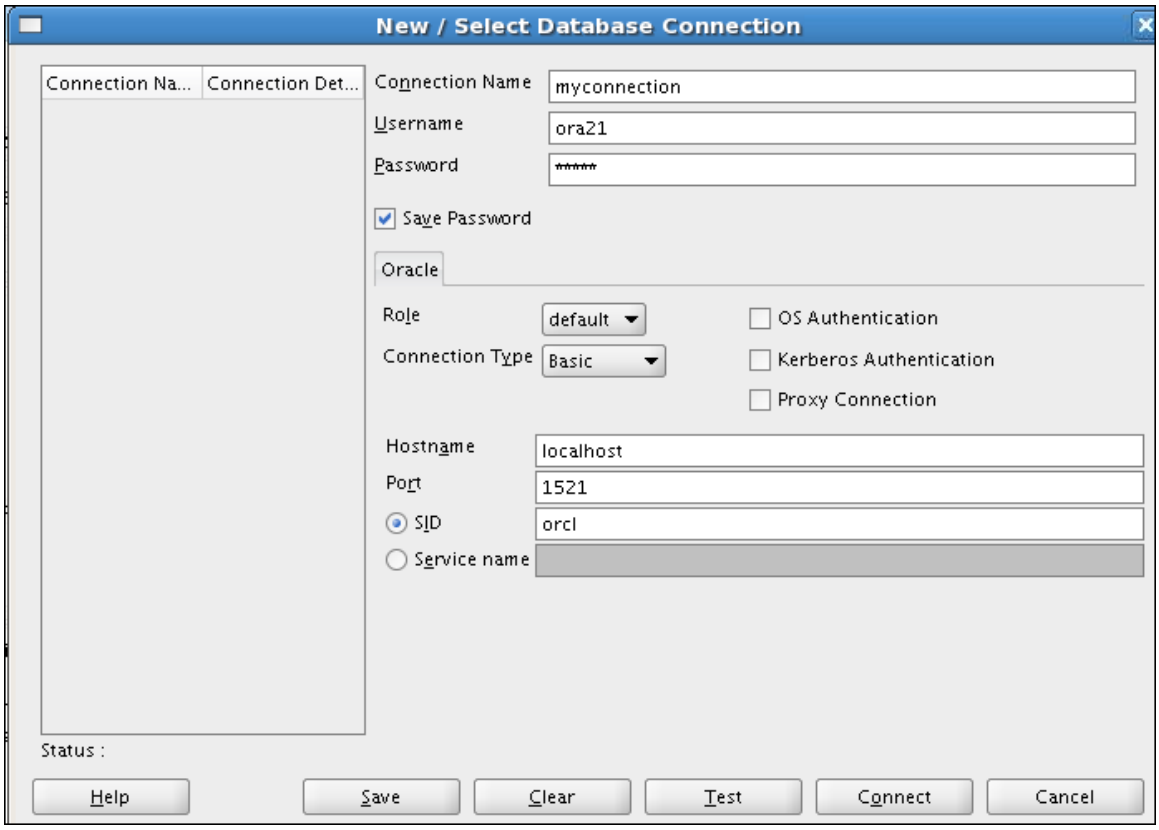
c. Password: `oraxx`

d. Hostname: `localhost`

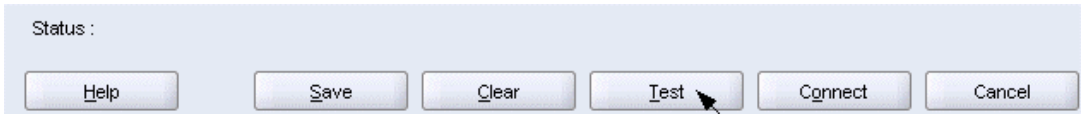
e. Port: `1521`

f. SID: `orcl` (or the value provided to you by the instructor)

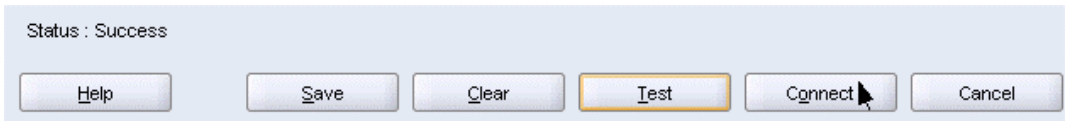




3) Test the new connection. If the status is Success, connect to the database by using this new connection.
   a) Click the Test button in the New/Select Database Connection window.

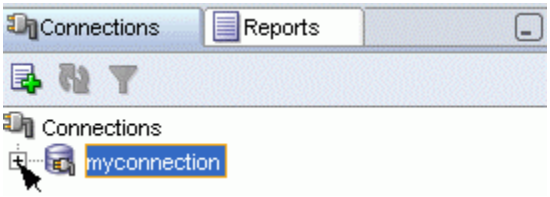


   b) If the status is Success, click the Connect button.

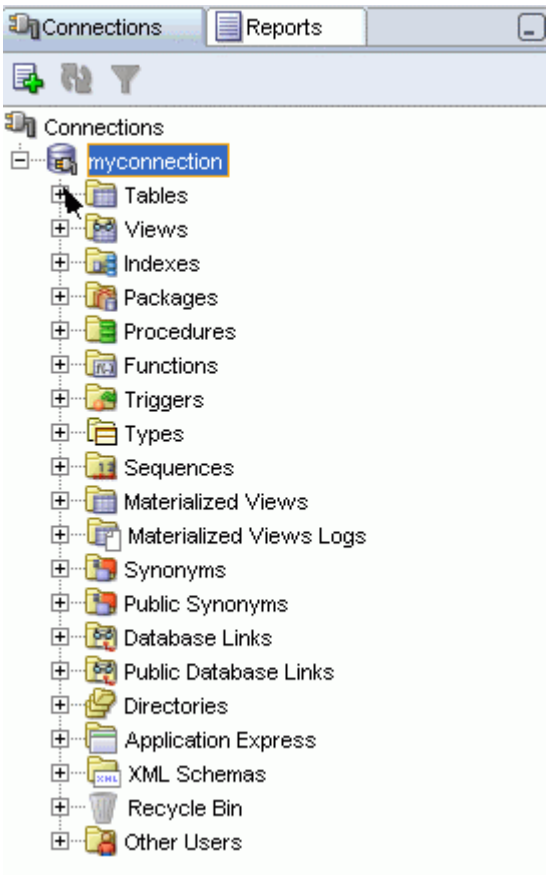


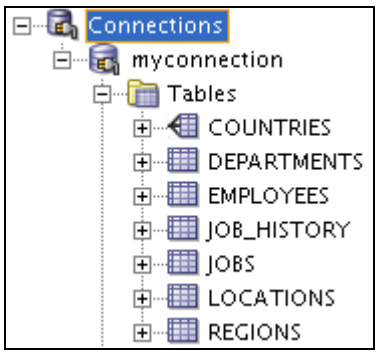
Browsing the Tables

4) Browse the structure of the `EMPLOYEES` table and display its data.
   a) Expand the myconnection connection by clicking the plus sign next to it.



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





c) Display the structure of the `EMPLOYEES` table.

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

myconnection | EMPLOYEES

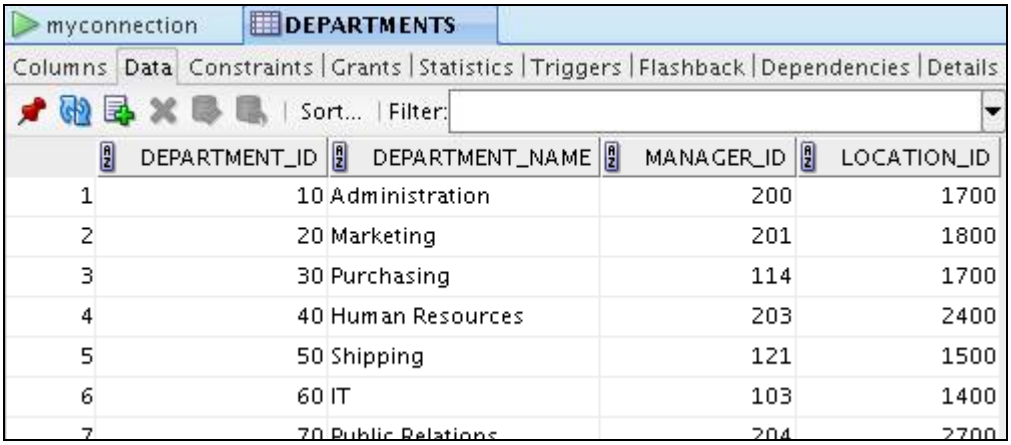
Columns | Data | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Indexes | SQL

Actions...

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

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

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


myconnection DEPARTMENTS

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

Sort... | Filter:

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

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

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

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

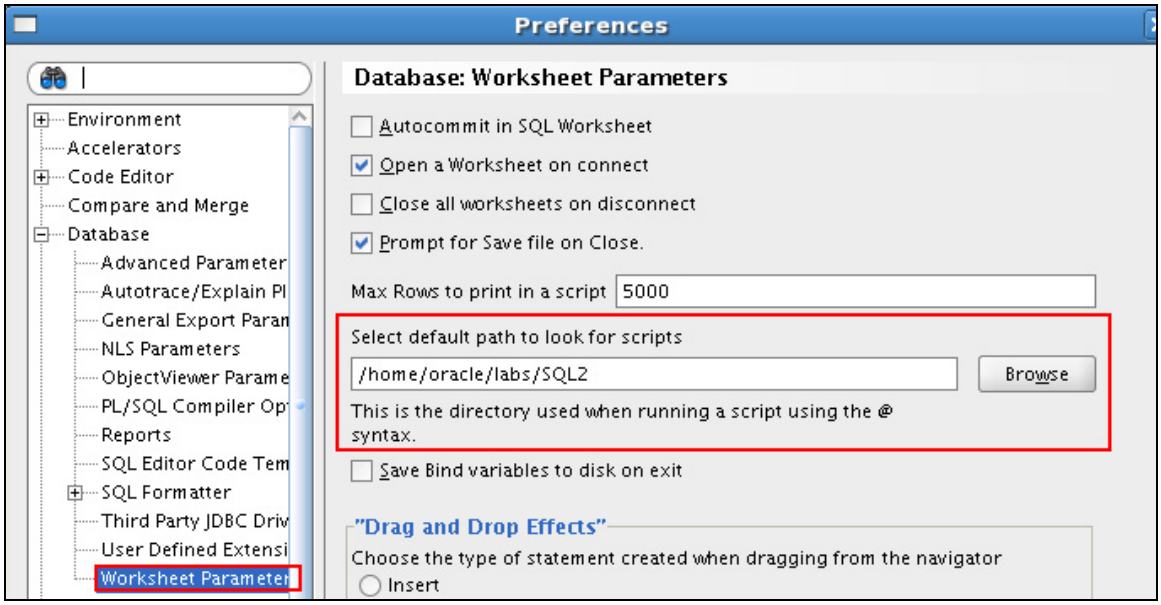
   b) Write a query to display last name, job ID, and commission for all employees who are not entitled to receive a commission.

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

6) Set your script pathing preference to /home/oracle/labs/sql2.

   a) Select Tools > Preferences > Database > Worksheet Parameters.

   b) Enter the value in the **Select default path to look for scripts** field. Then, click OK.

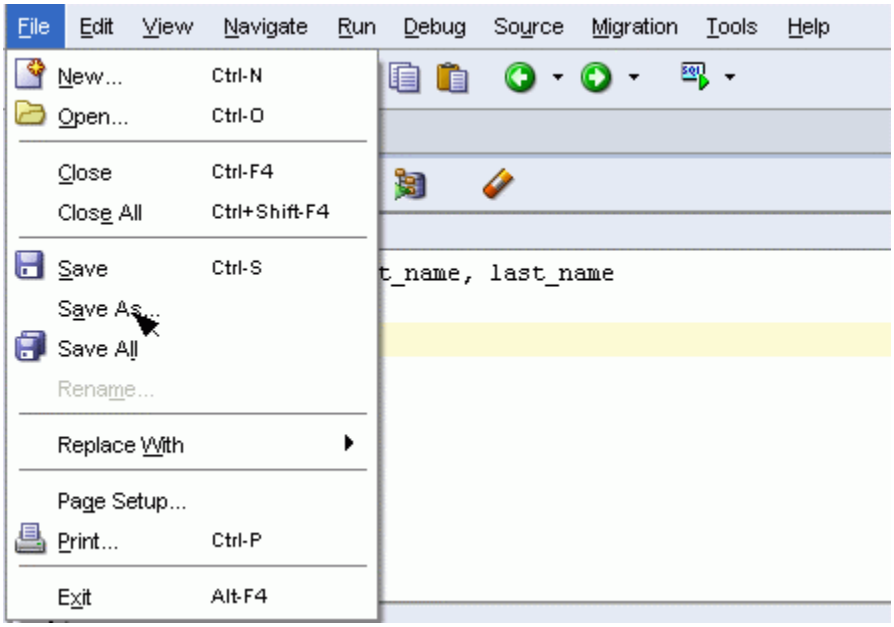


7) Enter the following SQL statement:

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

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

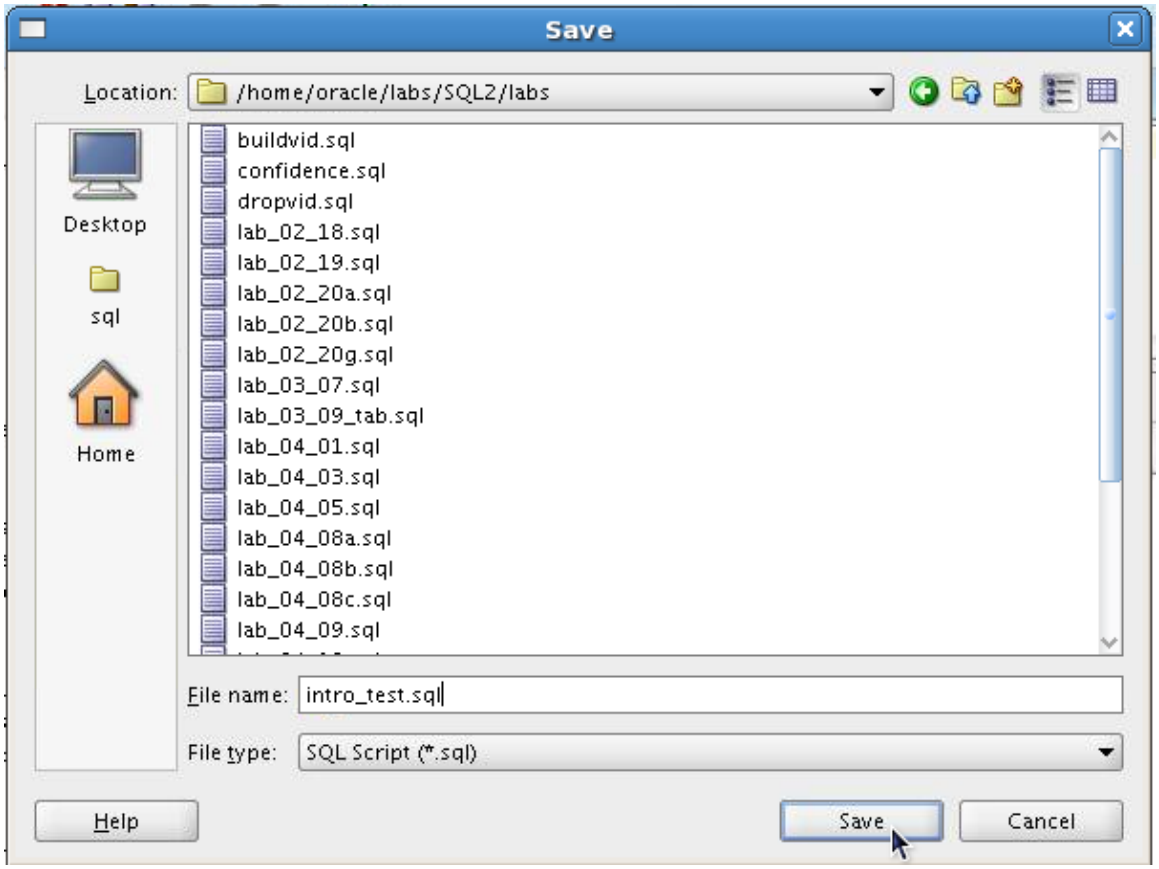
a) Select File > Save As.



b) Name the file **`intro_test.sql`**.

Enter `intro_test.sql` in the File_name text box.

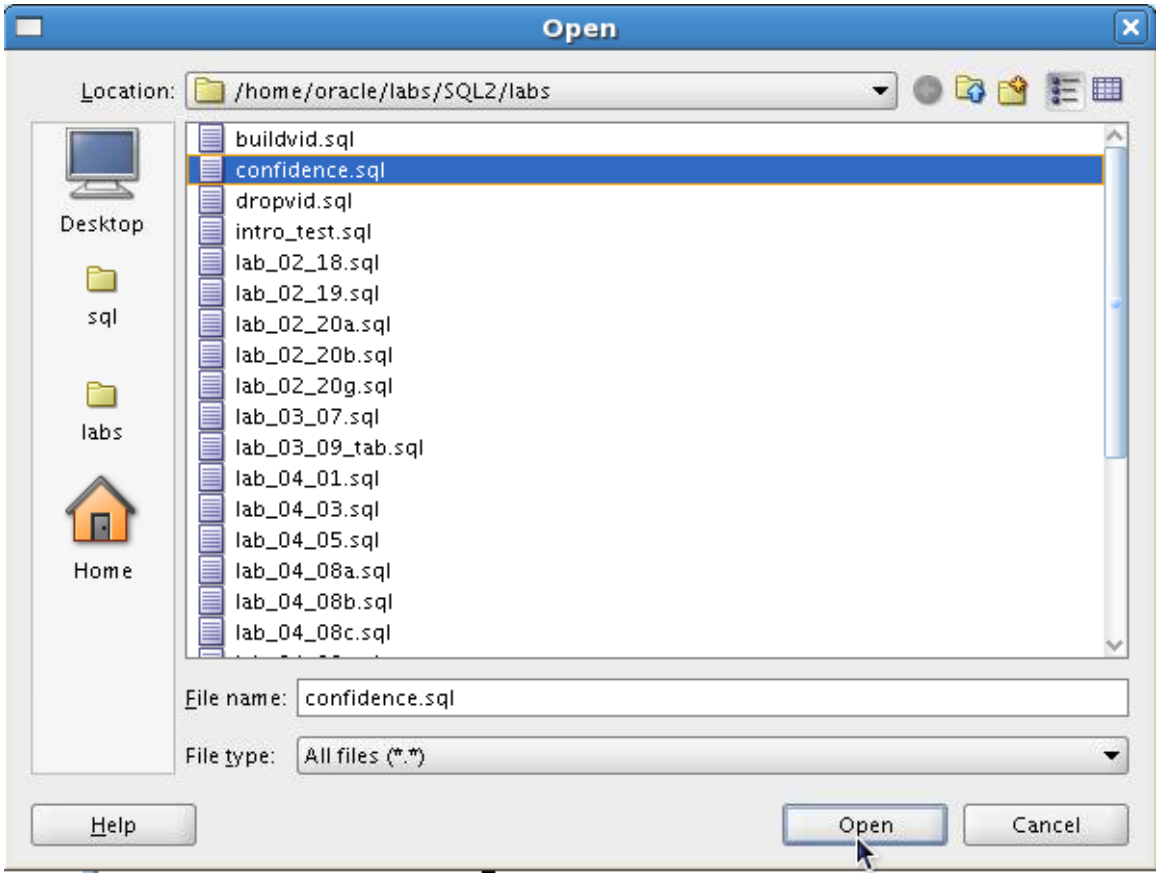
c) Place the file under the **/home/oracle/labs/SQL2/labs** folder.



Then, click Save.

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

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



Then, press F5 to execute the script.

The following is the expected result:

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

1 rows selected

COUNT(*)
----------------------
107

1 rows selected

COUNT(*)
----------------------
25

1 rows selected
```

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

1 rows selected

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

1 rows selected

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

1 rows selected

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

1 rows selected

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

1 rows selected
```

Practices and Solutions for Lesson 1

Practice 1-1: Controlling User Access

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

   _____

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

   _____

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

   _____

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

   _____

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

   _____

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

   _____

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

   _____

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

1. **Click the SQL Developer desktop icon.**
2. **In the Connections Navigator, use the *`oraxx`* account and the corresponding password provided by your instructor to log on to the database.**

8. Grant another user query privilege on your table. Then, verify whether that user can use the privilege.
   Note: For this exercise, team up with another group. For example, if you are user `ora21`, team up with another user `ora22`.
   a. Grant another user privilege to view records in your `REGIONS` table. Include an option for this user to further grant this privilege to other users.
   b. Have the user query your `REGIONS` table.
   c. Have the user pass on the query privilege to a third user (for example, `ora23`).

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

Note: Each team can run exercises 9 and 10 independently.

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

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

11. Grant another user access to your `DEPARTMENTS` table. Have the user grant you query access to his or her `DEPARTMENTS` table.
12. Query all the rows in your `DEPARTMENTS` table.

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

. . .

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

Team 1 `SELECT` statement results:

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

Team 2 SELECT statement results:

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

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

Practice Solutions 1-1: Controlling User Access

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

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

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

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

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

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

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

   Only `User21`

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

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

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

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

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

   Team 1 executes this statement:

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

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

   Team 2 executes this statement:

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

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

   Team 2 executes this statement.

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

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

   Team 1 executes this statement.

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

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

Team 1 executes this statement.

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

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

Team 1 executes this statement.

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

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

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

Team 2 executes the GRANT statement.

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

Team 1 executes the GRANT statement.

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

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

12. Query all the rows in your DEPARTMENTS table.

```
SELECT  *
FROM    departments;
```

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

Team 1 executes this INSERT statement.

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

Team 2 executes this INSERT statement.

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

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

Team 1 creates a synonym named team2.

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

Team 2 creates a synonym named team1.

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

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

Team 1 executes this SELECT statement.

```
SELECT  *
   FROM    team2;
```

Team 2 executes this SELECT statement.

```
SELECT  *
    FROM    team1;
```

16. Revoke the SELECT privilege from the other team.

Team 1 revokes the privilege.

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

Team 2 revokes the privilege.

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

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

Team 1 executes this DELETE statement.

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

Team 2 executes this DELETE statement.

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

Practices and Solutions for Lesson 2

Practice 2-1: Managing Schema Objects

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

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

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

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

2 rows selected
```

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

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

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

4 rows selected
```

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

```
Name                                      Null     Type
----------------------------------------- -------- ----------------
ID                                                 NUMBER(7)
LAST_NAME                                          VARCHAR2(50)
FIRST_NAME                                         VARCHAR2(25)
DEPT_ID                                            NUMBER(7)

4 rows selected
```

5. Create the `EMPLOYEES2` table based on the structure of the `EMPLOYEES` table. Include only the `EMPLOYEE_ID`, `FIRST_NAME`, `LAST_NAME`, `SALARY`, and `DEPARTMENT_ID` columns. Name the columns in your new table `ID`, `FIRST_NAME`, `LAST_NAME`, `SALARY`, and `DEPT_ID`, respectively.
6. Drop the `EMP2` table.
7. Query the recycle bin to see whether the table is present.

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

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

```
Name                                      Null     Type
----------------------------------------- -------- ----------------
ID                                                 NUMBER(7)
LAST_NAME                                          VARCHAR2(50)
FIRST_NAME                                         VARCHAR2(25)
DEPT_ID                                            NUMBER(7)

4 rows selected
```

9. Drop the `FIRST_NAME` column from the `EMPLOYEES2` table. Confirm your modification by checking the description of the table.

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

4 rows selected
```

10. In the `EMPLOYEES2` table, mark the `DEPT_ID` column as `UNUSED`. Confirm your modification by checking the description of the table.

```
Name                          Null     Type
----------------------------- -------- ----------------
ID                                     NUMBER(6)
LAST_NAME                     NOT NULL VARCHAR2(25)
SALARY                                 NUMBER(8,2)

3 rows selected
```

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

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

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

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

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

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

b. Query the `library_items_ext` table.

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

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

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

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

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

    c. Query the `dept_add_ext` table.

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

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

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

```
create table succeeded.
```

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

```
1 rows inserted

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

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

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

    d. Drop the `emp_books_pk` constraint.

```
alter table emp_books succeeded.
```

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

```
alter table emp_books succeeded.
```

f. Set the `emp_books_pk` constraint as deferred.

```
set constraint succeeded.
```

g. Run the `lab_02_20_g.sql` file to populate data into the `emp_books` table. What do you observe?

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

h. Commit the transaction. What do you observe?

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

Practice Solutions 2-1: Managing Schema Objects

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

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

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

DESCRIBE dept2
```

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

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

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

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

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

DESCRIBE emp2
```

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

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

DESCRIBE emp2
```

5. Create the EMPLOYEES2 table based on the structure of the EMPLOYEES table. Include only the EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY, and DEPARTMENT_ID columns. Name the columns in your new table ID, FIRST_NAME, LAST_NAME, SALARY, and DEPT_ID, respectively.

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

6. Drop the EMP2 table.

```
DROP TABLE emp2;
```

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

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

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

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

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

9. Drop the FIRST_NAME column from the EMPLOYEES2 table. Confirm your modification by checking the description of the table.

```
       ALTER TABLE employees2
      DROP COLUMN first_name;

   DESCRIBE employees2
```

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

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

   DESCRIBE employees2
```

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

```
   ALTER TABLE employees2
      DROP UNUSED COLUMNS;

      DESCRIBE employees2
```

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

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

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

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

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

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

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

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

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

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

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

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

17. Create the DEPT_NAMED_INDEX table based on the following table instance chart. Name the index for the PRIMARY KEY column as DEPT_PK_IDX.

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

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

18. Create an external table `library_items_ext`. Use the `ORACLE_LOADER` access driver.
    Note: The `emp_dir` directory and `library_items.dat` are already created for this exercise.

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

    2354, 2264, 13.21, 150,

    2355, 2289, 46.23, 200,

    2355, 2264, 50.00, 100,

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

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

b) Query the `library_items_ext` table.

```
SELECT * FROM library_items_ext;
```

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

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

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

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

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

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

b) Query the dept_add_ext table.

```
SELECT * FROM dept_add_ext;
```

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

a) Run the lab_02_20a.sql script to create the emp_books table. Observe that the emp_books_pk primary key is not created as deferrable.

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

b) Run the lab_02_20b.sql script to populate data into the emp_books table.

What do you observe?

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

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

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

```
SET CONSTRAINT emp_books_pk DEFERRED;
```

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

d) Drop the emp_books_pk constraint.

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

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

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

f) Set the emp_books_pk constraint as deferred.

```
SET CONSTRAINT emp_books_pk DEFERRED;
```

g) Run the `lab_02_20g.sql` script to populate data into the `emp_books` table.

What do you observe?

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

You see that all the rows are inserted.

h) Commit the transaction. What do you observe?

```
COMMIT;
```

You see that the transaction is rolled back.

Practices and Solutions for Lesson 3

Practice 3-1: Managing Objects with Data Dictionary Views

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

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

	TABLE_NAME
1	REGIONS
2	LOCATIONS
3	DEPARTMENTS
4	JOBS
5	EMPLOYEES
6	JOB_HISTORY
7	EMP_NEW_SAL
8	EMPLOYEES2
9	DEPT_NAMED_INDEX

…

2. Query the `ALL_TABLES` data dictionary view to see information about all the tables that you can access. Exclude the tables that you own.
   Note: Your list may not exactly match the following list:

	TABLE_NAME	OWNER
1	DUAL	SYS
2	SYSTEM_PRIVILEGE_MAP	SYS
3	TABLE_PRIVILEGE_MAP	SYS

…

98	PLAN_TABLE$	SYS
99	WRI$_ADV_ASA_RECO_DATA	SYS
100	PSTUBTBL	SYS

3. For a specified table, create a script that reports the column names, data types, and data types' lengths, as well as whether nulls are allowed. Prompt the user to enter the table name. Give appropriate aliases to the `DATA_PRECISION` and `DATA_SCALE` columns. Save this script in a file named `lab_03_01.sql`.
   For example, if the user enters `DEPARTMENTS`, the following output results:

Practice 3-1: Managing Objects with Data Dictionary Views (continued)

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

4. Create a script that reports the column name, constraint name, constraint type, search condition, and status for a specified table. You must join the `USER_CONSTRAINTS` and `USER_CONS_COLUMNS` tables to obtain all this information. Prompt the user to enter the table name. Save the script in a file named `lab_03_04.sql`.
   For example, if the user enters `DEPARTMENTS`, the following output results:

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

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

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

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

	SYNONYM_NAME	TABLE_OWNER	TABLE_NAME	DB_LINK
1	TEAM2	ORA22	DEPARTMENTS	(null)
2	EMP	ORA21	EMPLOYEES	(null)

7. Run `lab_03_07.sql` to create the `dept50` view for this exercise.
   You need to determine the names and definitions of all the views in your schema. Create a report that retrieves view information: the view name and text from the `USER_VIEWS` data dictionary view.
   Note: The `EMP_DETAILS_VIEW` was created as part of your schema.
   Note: You can see the complete definition of the view if you use Run Script (or press F5) in SQL Developer. If you use Execute Statement (or press F9) in SQL Developer, scroll horizontally in the result pane. If you use SQL*Plus, to see more contents of a `LONG` column, use the `SET LONG n` command, where *n* is the value of the number of characters of the `LONG` column that you want to see.

Practice 3-1: Managing Objects with Data Dictionary Views (continued)

	VIEW_NAME	TEXT
1	DEPT50	SELECT employee_id empno, last_name employee, department_id deptno
2	EMP_DETAILS_VIEW	SELECT e.employee_id, e.job_id, e.manager_id, e.department_id, d.location_id,

8. Find the names of your sequences. Write a query in a script to display the following information about your sequences: sequence name, maximum value, increment size, and last number. Name the script `lab_03_08.sql`. Run the statement in your script.

	SEQUENCE_NAME	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
1	DEPARTMENTS_SEQ	9990	10	280
2	EMPLOYEES_SEQ	999999999999999999999999999	1	207
3	LOCATIONS_SEQ	9900	100	3300

Run the `lab_03_09_tab.sql` script as a prerequisite for exercises 9 through 11. Alternatively, open the script file to copy the code and paste it into your SQL Worksheet. Then execute the script. This script:

- Drops if there are existing tables `DEPT2` and `EMP2`
- Creates the `DEPT2` and `EMP2` tables

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

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

	TABLE_NAME
1	DEPT2
2	EMP2

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

	CONSTRAINT_NAME	CONSTRAINT_TYPE
1	MY_DEPT_ID_PK	P
2	MY_EMP_ID_PK	P
3	MY_EMP_DEPT_ID_FK	R

11. Display the object names and types from the `USER_OBJECTS` data dictionary view for the `EMP2` and `DEPT2` tables.
12. Create the `SALES_DEPT` table based on the following table instance chart. Name the index for the `PRIMARY KEY` column `SALES_PK_IDX`. Then query the data dictionary view to find the index name, table name, and whether the index is unique.

Practice 3-1: Managing Objects with Data Dictionary Views (continued)

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

	INDEX_NAME	TABLE_NAME	UNIQUENESS
1	SALES_PK_IDX	SALES_DEPT	NONUNIQUE

Practice Solutions 3-1: Managing Objects with Data Dictionary Views

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

```
SELECT table_name
   FROM   user_tables;
```

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

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

3. For a specified table, create a script that reports the column names, data types, and data types' lengths, as well as whether nulls are allowed. Prompt the user to enter the table name. Give appropriate aliases to the DATA_PRECISION and DATA_SCALE columns. Save this script in a file named lab_03_01.sql.

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

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

4. Create a script that reports the column name, constraint name, constraint type, search condition, and status for a specified table. You must join the USER_CONSTRAINTS and USER_CONS_COLUMNS tables to obtain all this information. Prompt the user to enter the table name. Save the script in a file named lab_03_04.sql.

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

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

Practice Solutions 3-1: Managing Objects with Data Dictionary Views (continued)

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

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

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

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

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

7. Run lab_03_07.sql to create the dept50 view for this exercise. You need to determine the names and definitions of all the views in your schema. Create a report that retrieves view information: the view name and text from the USER_VIEWS data dictionary view.

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

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

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

Practice Solutions 3-1: Managing Objects with Data Dictionary Views (continued)

8. Find the names of your sequences. Write a query in a script to display the following information about your sequences: sequence name, maximum value, increment size, and last number. Name the script `lab_03_08.sql`. Run the statement in your script.

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

   Run the `lab_03_09_tab.sql` script as a prerequisite for exercises 9 through 11. Alternatively, open the script file to copy the code and paste it into your SQL Worksheet. Then execute the script. This script:

   - Drops the `DEPT2` and `EMP2` tables
   - Creates the `DEPT2` and `EMP2` tables

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

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

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

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

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

11. Query the data dictionary to display the object names and types for both the tables.

```
SELECT   object_name, object_type
FROM     user_objects
WHERE    object_name LIKE 'EMP%'
OR       object_name LIKE 'DEPT%';
```

Practice Solutions 3-1: Managing Objects with Data Dictionary Views (continued)

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

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

```
CREATE TABLE SALES_DEPT
       (team_id NUMBER(3)
       PRIMARY KEY USING INDEX
       (CREATE INDEX sales_pk_idx ON
        SALES_DEPT(team_id)),
        location VARCHAR2(30));

SELECT INDEX_NAME, TABLE_NAME, UNIQUENESS
FROM USER_INDEXES
WHERE TABLE_NAME = 'SALES_DEPT';
```

Practices and Solutions for Lesson 4

Practice 4-1: Manipulating Large Data Sets

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

1. Run the `lab_04_01.sql` script in the lab folder to create the `SAL_HISTORY` table.
2. Display the structure of the `SAL_HISTORY` table.

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

3 rows selected
```

3. Run the `lab_04_03.sql` script in the lab folder to create the `MGR_HISTORY` table.
4. Display the structure of the `MGR_HISTORY` table.

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

3 rows selected
```

5. Run the `lab_04_05.sql` script in the lab folder to create the `SPECIAL_SAL` table.
6. Display the structure of the `SPECIAL_SAL` table.

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

2 rows selected
```

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

- Insert details such as the employee ID, hire date, and salary into the `SAL_HISTORY` table.
- Insert details such as the employee ID, manager ID, and salary into the `MGR_HISTORY` table.

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

	EMPLOYEE_ID	SALARY
1	100	24000

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

	EMPLOYEE_ID	HIRE_DATE	SALARY
1	101	21-SEP-89	17000
2	102	13-JAN-93	17000
3	103	03-JAN-90	9000
4	104	21-MAY-91	6000
5	105	25-JUN-97	4800
6	106	05-FEB-98	4800
7	107	07-FEB-99	4200

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

	EMPLOYEE_ID	MANAGER_ID	SALARY
1	101	100	17000
2	102	100	17000
3	103	102	9000
4	104	103	6000
5	105	103	4800
6	106	103	4800
7	107	103	4200

8.

a. Run the `lab_04_08a.sql` script in the lab folder to create the `SALES_WEEK_DATA` table.

b. Run the `lab_04_08b.sql` script in the lab folder to insert records into the `SALES_WEEK_DATA` table.

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

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

7 rows selected
```

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

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

e. Run the `lab_04_08_e.sql` script in the lab folder to create the `EMP_SALES_INFO` table.

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

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

3 rows selected
```

g. Write a query to do the following:

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

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

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

9. You have the data of past employees stored in a flat file called `emp.data`. You want to store the names and email IDs of all employees, past and present, in a table. To do

this, first create an external table called `EMP_DATA` using the `emp.dat` source file in the `emp_dir` directory. Use the `lab_04_09.sql` script to do this.

10. Next, run `the lab_04_10.sql` script to create the `EMP_HIST` table.
    a. Increase the size of the email column to 45.
    b. Merge the data in the `EMP_DATA` table created in the last lab into the data in the `EMP_HIST` table. Assume that the data in the external `EMP_DATA` table is the most up-to-date. If a row in the `EMP_DATA` table matches the `EMP_HIST` table, update the email column of the `EMP_HIST` table to match the `EMP_DATA` table row. If a row in the `EMP_DATA` table does not match, insert it into the `EMP_HIST` table. Rows are considered matching when the employee's first and last names are identical.
    c. Retrieve the rows from `EMP_HIST` after the merge.

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

11. Create the EMP3 table by using the `lab_04_11.sql` script. In the `EMP3` table, change the department for `Kochhar` to `60` and commit your change. Next, change the department for `Kochhar` to `50` and commit your change. Track the changes to `Kochhar` by using the Row Versions feature.

	START_DATE	END_DATE	DEPARTMENT_ID
1	18-JUN-09 06.04.26.000000000 PM	(null)	50
2	18-JUN-09 06.04.26.000000000 PM	18-JUN-09 06.04.26.000000000 PM	60
3	(null)	18-JUN-09 06.04.26.000000000 PM	90

Practice Solutions 4-1: Manipulating Large Data Sets

1. Run the `lab_04_01.sql` script in the lab folder to create the `SAL_HISTORY` table.

2. Display the structure of the `SAL_HISTORY` table.

```
DESC sal_history
```

3. Run the `lab_04_03.sql` script in the lab folder to create the `MGR_HISTORY` table.

4. Display the structure of the `MGR_HISTORY` table.

```
DESC mgr_history
```

5. Run the `lab_04_05.sql` script in the lab folder to create the `SPECIAL_SAL` table.

6. Display the structure of the `SPECIAL_SAL` table.

```
DESC special_sal
```

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

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

b) Display the records from the SPECIAL_SAL table.

```
SELECT * FROM  special_sal;
```

c) Display the records from the SAL_HISTORY table.

```
SELECT * FROM  sal_history;
```

d) Display the records from the MGR_HISTORY table.

```
SELECT * FROM mgr_history;
```

8. a) Run the lab_04_08a.sql script in the lab folder to create the SALES_WEEK_DATA table.

b) Run the lab_04_08b.sql script in the lab folder to insert records into the SALES_WEEK_DATA table.

c) Display the structure of the SALES_WEEK_DATA table.

```
DESC sales_week_data
```

d) Display the records from the SALES_WEEK_DATA table.

```
SELECT * FROM SALES_WEEK_DATA;
```

e) Run the `lab_04_08_e.sql` script in the lab folder to create the `EMP_SALES_INFO` table.

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

```
DESC emp_sales_info
```

g) Write a query to do the following:

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

  Hint: Use a pivoting `INSERT` statement.

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

h) Display the records from the `SALES_INFO` table.

```
SELECT * FROM emp_sales_info;
```

9. You have the data of past employees stored in a flat file called `emp.data`. You want to store the names and email IDs of all employees past and present in a table. To do this, first create an external table called `EMP_DATA` using the `emp.dat` source file in the `emp_dir` directory. You can use the script in `lab_04_09.sql` to do this.

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

10. Next, run the `lab_04_10.sql` script to create the `EMP_HIST` table.
    a) Increase the size of the email column to 45.

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

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

```
MERGE INTO EMP_HIST f USING EMP_DATA h
 ON (f.first_name = h.first_name
 AND f.last_name = h.last_name)
```

```
WHEN MATCHED THEN
 UPDATE SET f.email = h.email
WHEN NOT MATCHED THEN
 INSERT (f.first_name
    , f.last_name
    , f.email)
 VALUES (h.first_name
    , h.last_name
    , h.email);
```

c) Retrieve the rows from EMP_HIST after the merge.

```
SELECT * FROM emp_hist;
```

11. Create the EMP3 table using the lab_04_11.sql script. In the EMP3 table, change the department for Kochhar to 60 and commit your change. Next, change the department for Kochhar to 50 and commit your change. Track the changes to Kochhar using the Row Versions feature.

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

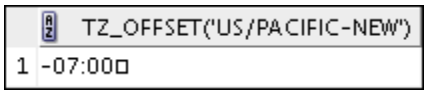
```
SELECT VERSIONS_STARTTIME "START_DATE",
   VERSIONS_ENDTIME "END_DATE",  DEPARTMENT_ID
FROM EMP3
   VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
WHERE LAST_NAME ='Kochhar';
```

Practices and Solutions for Lesson 5

Practice 5-1: Managing Data in Different Time Zones

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

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

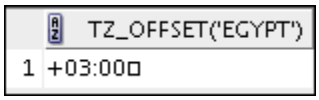


	TZ_OFFSET('US/PACIFIC-NEW')
1	-07:00

   - *Singapore*

	TZ_OFFSET('SINGAPORE')
1	+08:00

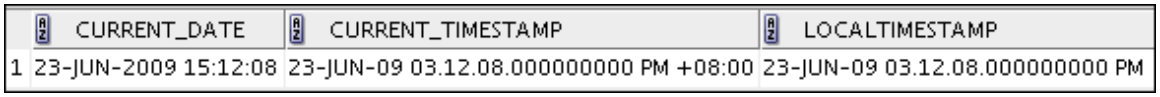
   - *Egypt*



	TZ_OFFSET('EGYPT')
1	+03:00

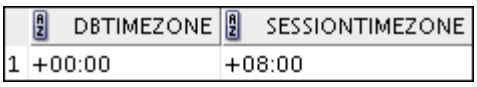
   b. Alter the session to set the `TIME_ZONE` parameter value to the time zone offset of US/Pacific-New.
   c. Display `CURRENT_DATE`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP` for this session.
   d. Alter the session to set the `TIME_ZONE` parameter value to the time zone offset of Singapore.
   e. Display `CURRENT_DATE`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP` for this session.

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



	CURRENT_DATE	CURRENT_TIMESTAMP	LOCALTIMESTAMP
1	23-JUN-2009 15:12:08	23-JUN-09 03.12.08.000000000 PM +08:00	23-JUN-09 03.12.08.000000000 PM

   Note: Observe in the preceding practice that `CURRENT_DATE`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP` are sensitive to the session time zone.
3. Write a query to display `DBTIMEZONE` and `SESSIONTIMEZONE`.



	DBTIMEZONE	SESSIONTIMEZONE
1	+00:00	+08:00

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

	LAST_NAME	EXTRACT(YEARFROMHIRE_DATE)
1	Russell	1996
2	Partners	1997
3	Errazuriz	1997
4	Cambrault	1999
5	Zlotkey	2000
6	Tucker	1997
7	Bernstein	1997

5. Alter the session to set `NLS_DATE_FORMAT` to `DD-MON-YYYY`.
6. Examine and run the `lab_05_06.sql` script to create the `SAMPLE_DATES` table and populate it.
   a. Select from the table and view the data.

	DATE_COL
1	23-JUN-2009

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

	DATE_COL
1	23-JUN-09 02.14.52.000000000 PM

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

7. Create a query to retrieve last names from the `EMPLOYEES` table and calculate the review status. If the year hired was 1998, display `Needs Review` for the review status; otherwise, display `not this year!` Name the review status column `Review`. Sort the results by the `HIRE_DATE` column.
   Hint: Use a `CASE` expression with the `EXTRACT` function to calculate the review status.

	LAST_NAME	Review
1	King	not this year!
2	Whalen	not this year!
3	Kochhar	not this year!
4	Hunold	not this year!
5	Ernst	not this year!
6	De Haan	not this year!
7	Mavris	not this year!

…

Practice 5-1: Managing Data in Different Time Zones (continued)

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

	LAST_NAME	HIRE_DATE	SYSDATE	Awards
1	OConnell	21-JUN-1999	23-JUN-2009	10 years of service
2	Grant	13-JAN-2000	23-JUN-2009	5 years of service
3	Whalen	17-SEP-1987	23-JUN-2009	15 years of service
4	Hartstein	17-FEB-1996	23-JUN-2009	10 years of service
5	Fay	17-AUG-1997	23-JUN-2009	10 years of service
6	Mavris	07-JUN-1994	23-JUN-2009	15 years of service

...

Practice Solutions 5-1: Managing Data in Different Time Zones

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

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

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

US/Pacific-New

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

Singapore

```
SELECT TZ_OFFSET ('Singapore') from dual;
```

Egypt

```
SELECT TZ_OFFSET ('Egypt') from dual;
```

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

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

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

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

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

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

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

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

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

3. Write a query to display DBTIMEZONE and SESSIONTIMEZONE.

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

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

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

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

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

6. Examine and run the lab_05_06.sql script to create the SAMPLE_DATES table and populate it.

   a. Select from the table and view the data.

```
SELECT * FROM sample_dates;
```

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

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

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

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

Practice Solutions 5-1: Managing Data in Different Time Zones (continued)

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

7. Create a query to retrieve last names from the EMPLOYEES table and calculate the review status. If the year hired was 1998, display Needs Review for the review status; otherwise, display not this year! Name the review status column Review. Sort the results by the HIRE_DATE column.
   Hint: Use a CASE expression with the EXTRACT function to calculate the review status.

```
SELECT e.last_name
   ,     (CASE extract(year from e.hire_date)
           WHEN 1998 THEN 'Needs Review'
           ELSE 'not this year!'
     END )           AS "Review "
FROM   employees e
ORDER BY e.hire_date;
```

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

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

Practices and Solutions for Lesson 6

Practice 6-1: Retrieving Data by Using Subqueries

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

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

	LAST_NAME	DEPARTMENT_ID	SALARY
1	Russell	80	14000
2	Partners	80	13500
3	Errazuriz	80	12000

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

	LAST_NAME	DEPARTMENT_NAME	SALARY
1	Whalen	Administration	4400
2	Higgins	Accounting	12000
3	Greenberg	Finance	12000
4	Gietz	Accounting	8300

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

   Note: Do not display Kochhar in the result set.

	LAST_NAME	HIRE_DATE	SALARY
1	De Haan	13-JAN-1993	17000

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

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

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

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	202	Fay	20
2	201	Hartstein	20

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

	ENAME	SALARY	DEPTNO	DEPT_AVG
1	Fripp	8200	50	3475.555555555555555555555555555555555556
2	Kaufling	7900	50	3475.555555555555555555555555555555555556
3	Chung	3800	50	3475.555555555555555555555555555555555556
4	Mourgos	5800	50	3475.555555555555555555555555555555555556
5	Bell	4000	50	3475.555555555555555555555555555555555556
6	Rajs	3500	50	3475.555555555555555555555555555555555556
7	Bull	4100	50	3475.555555555555555555555555555555555556
8	Everett	3900	50	3475.555555555555555555555555555555555556

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

	LAST_NAME
1	Abel
2	Ande
3	Atkinson
4	Austin
5	Baer
6	Baida

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

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

	LAST_NAME
1	Chen
2	Sciarra
3	Urman
4	Popp
5	Khoo
6	Baida

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

	LAST_NAME
1	Vargas
2	Patel
3	Olson
4	Marlow
5	Landry
6	Perkins

10. Write a query to display the employee ID, last names, and department names of all the employees.
Note: Use a scalar subquery to retrieve the department name in the `SELECT` statement.

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT
1	205	Higgins	Accounting
2	206	Gietz	Accounting
3	200	Whalen	Administration
4	100	King	Executive
5	101	Kochhar	Executive

• • •

105	196	Walsh	Shipping
106	197	Feeney	Shipping
107	178	Grant	(null)

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

	DEPARTMENT_NAME	DEPT_TOTAL
1	Sales	304500
2	Shipping	156400

Practice Solutions 6-1: Retrieving Data by Using Subqueries

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

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

2. Display the last name, department name, and salary of any employee whose salary and commission match the salary and commission of any employee located in location ID1700.

```
SELECT e.last_name, d.department_name, e.salary
FROM   employees e, departments d
WHERE  e.department_id = d.department_id
AND   (salary, NVL(commission_pct,0)) IN
            (SELECT salary, NVL(commission_pct,0)
           FROM     employees e, departments  d
           WHERE    e.department_id = d.department_id
           AND d.location_id = 1700);
```

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

   Note: Do not display Kochhar in the result set.

```
SELECT last_name, hire_date, salary
FROM   employees
WHERE  (salary, NVL(commission_pct,0)) IN
             (SELECT salary, NVL(commission_pct,0)
              FROM   employees
              WHERE  last_name = 'Kochhar')
AND last_name != 'Kochhar';
```

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

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

Practice Solutions 6-1: Retrieving Data by Using Subqueries (continued)

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

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

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

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

Practice Solutions 6-1: Retrieving Data by Using Subqueries (continued)

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

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

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

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

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

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

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

Practice Solutions 6-1: Retrieving Data by Using Subqueries (continued)

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

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

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

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

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

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

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

Practices and Solutions for Lesson 7

Practice 7-1: Regular Expression Support

In this practice, you use regular expressions functions to search for, replace, and manipulate data. You also create a new CONTACTS table and add a CHECK constraint to the p_number column to ensure that phone numbers are entered into the database in a specific standard format.

1. Write a query to search the EMPLOYEES table for all the employees whose first names start with "Ki" or "Ko."

	FIRST_NAME	LAST_NAME
1	Janette	King
2	Steven	King
3	Neena	Kochhar

2. Create a query that removes the spaces in the STREET_ADDRESS column of the LOCATIONS table in the display. Use "Street Address" as the column heading.

	Street Address
1	1297ViaColadiRie
2	93091CalledellaTesta
3	2017Shinjuku-ku
4	9450Kamiya-cho
5	2014JabberwockyRd
6	2011InteriorsBlvd
7	2007ZagoraSt

3. Create a query that displays "St" replaced by "Street" in the STREET_ADDRESS column of the LOCATIONS table. Be careful that you do not affect any rows that already have "Street" in them. Display only those rows that are affected.

	REGEXP_REPLACE(STREET_ADDRESS,'ST$','STREET')
1	2007 Zagora Street
2	6092 Boxwood Street
3	12-98 Victoria Street
4	8204 Arthur Street

4. Create a contacts table and add a check constraint to the p_number column to enforce the following format mask to ensure that phone numbers are entered into the database in the following standard format: (XXX) XXX-XXXX. The table should have the following columns:
   - l_name varchar2(30)
   - p_number varchar2 (30)

Practice 7-1: Regular Expression Support (continued)

5. Run the SQL script `lab_07_05.sql` to insert the following seven phone numbers into the `contacts` table. Which numbers are added?

l_name Column Value	**p_number Column Value**
NULL	'(650) 555-5555'
NULL	'(215) 555-3427'
NULL	'650 555-5555'
NULL	'650 555 5555'
NULL	'650-555-5555'
NULL	'(650)555-5555'
NULL	' (650) 555-5555'

6. Write a query to find the number of occurrences of the DNA pattern `ctc` in the string `gtctcgtctcgttctgtctgtcgttctg`. Ignore case-sensitivity.

	COUNT_DNA
1	2

Practice Solutions 7-1: Regular Expression Support

1. Write a query to search the EMPLOYEES table for all employees whose first names start with "Ki" or "Ko."

```
SELECT first_name, last_name
FROM employees
WHERE REGEXP_LIKE (last_name, '^K(i|o).');
```

2. Create a query that removes the spaces in the STREET_ADDRESS column of the LOCATIONS table in the display. Use "Street Address" as the column heading.

```
SELECT regexp_replace (street_address, ' ', '') AS "Street
Address"
FROM locations;
```

3. Create a query that displays "St" replaced by "Street" in the STREET_ADDRESS column of the LOCATIONS table. Be careful that you do not affect any rows that already have "Street" in them. Display only those rows, which are affected.

```
         SELECT regexp_replace (street_address, 'St$',
'Street')
 FROM locations
    WHERE regexp_like (street_address, 'St');
```

4. Create a contacts table and add a check constraint to the p_number column to enforce the following format mask to ensure that phone numbers are entered into the database in the following standard format: **(XXX) XXX-XXXX**. The table should have the following columns:
   - l_name varchar2(30)
   - p_number varchar2 (30)

```
CREATE TABLE contacts
(
  l_name    VARCHAR2(30),
  p_number  VARCHAR2(30)
    CONSTRAINT p_number_format
      CHECK ( REGEXP_LIKE ( p_number, '^\(\d{3}\) \d{3}-
\d{4}$' ) )
);
```

5. Run the `lab_07_05.sql` SQL script to insert the following seven phone numbers into the contacts table. Which numbers are added?
   Only the first two `INSERT` statements use a format that conforms to the `c_contacts_pnf` constraint; the remaining statements generate `CHECK` constraint errors.
6. Write a query to find the number of occurrences of the DNA pattern `ctc` in the string `gtctcgtctcgttctgtctgtcgttctg`. Use the alias `Count_DNA`. Ignore case-sensitivity.

```
SELECT REGEXP_COUNT('gtctcgtctcgttctgtctgtcgttctg','ctc')
AS Count_DNA
FROM dual;
```