ORACLE UNIVERSITY

Hardware and Software
Engineered to Work Together

Oracle Database 12*c*: SQL and PL/SQL New Features

Activity Guide

D90165GC10
Edition 1.0 | February 2015 | D90451

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

ORACLE

Authors

Anupama Mandya
Lauran Serhal
Swarnapriya Shridhar
Dimpi Sarmah

Technical Contributors

Steven Feuerstein
John Haydu
Hermann Baer
Srinivasan Ramakrishnan
Charles Sperry
A.A. Hopeman
Suresh Sridharan
Paul Lane
S. Matt Taylor Jr.

Reviewers

Steven Feuerstein
Iloon Ellen
Gerlinde Frenzen

Editors

Vijayalakshmi Narasimhan
Aju Kumar

Graphic Designer

Rajiv Chandrabhanu

Publishers

Joseph Fernandez
Srividya Rameshkumar
Veena Narasimhan

Table of Contents

Course Practice Environment: Security Credentials

For OS usernames and passwords, see the following:

- If you are attending a classroom-based or a live virtual class (LVC), ask your instructor or LVC producer for OS credential information.
- If you are using a self-study format, refer to the communication that you receive from Oracle University for this course.

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

Product-Specific Credentials		
Product/Application	**Username**	**Password**
Oracle SQL Developer	`hr`	`hr`

Practices for Lesson 1: Getting Started Using SQL Developer

Chapter 1

Practice 1-1: Getting Started Using SQL Developer

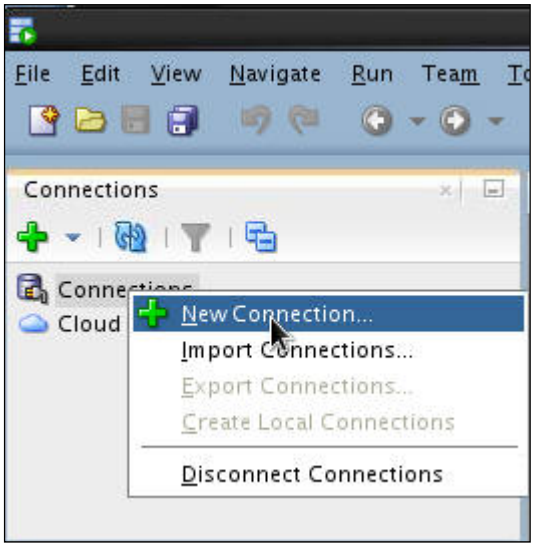
Overview

Oracle SQL Developer is a graphical version of SQL*Plus that gives database developers a convenient way to perform basic tasks. You can browse, create, edit, and delete (drop) database objects; run SQL statements and scripts; edit and debug PL/SQL code; manipulate and export (unload) data; and view and create reports. In this practice, you open Oracle SQL Developer and connect to Oracle Database 12*c* by creating a database connection so that you can view and work with database objects.

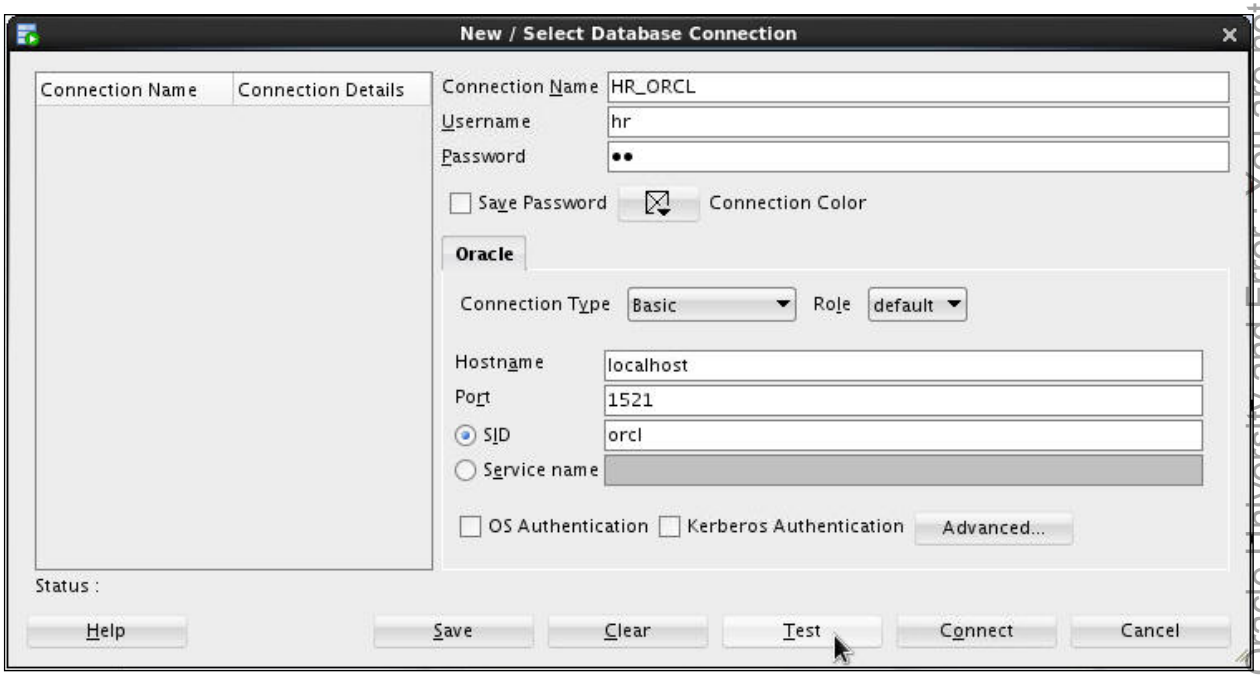
Tasks

1. Start Oracle SQL Developer and create a database connection.
   a. Double-click the SQL Developer icon on the desktop.

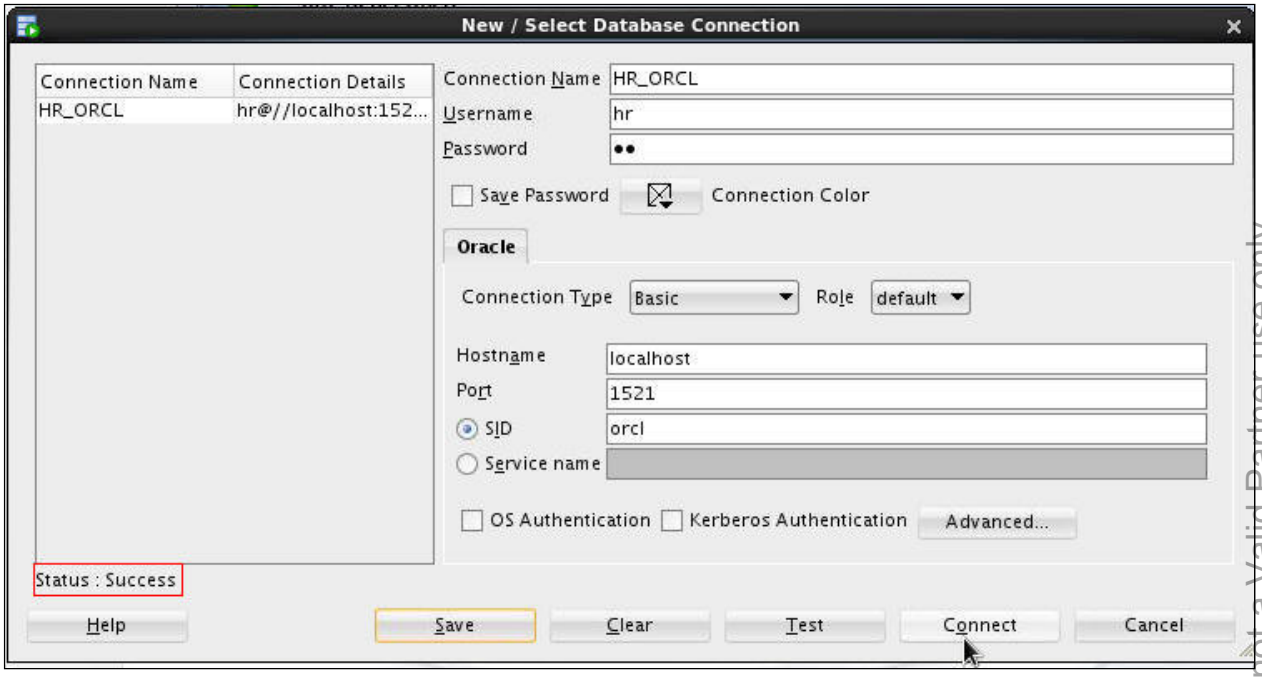
b. In the Connections navigator, right-click **Connections** and select **New Connection**.



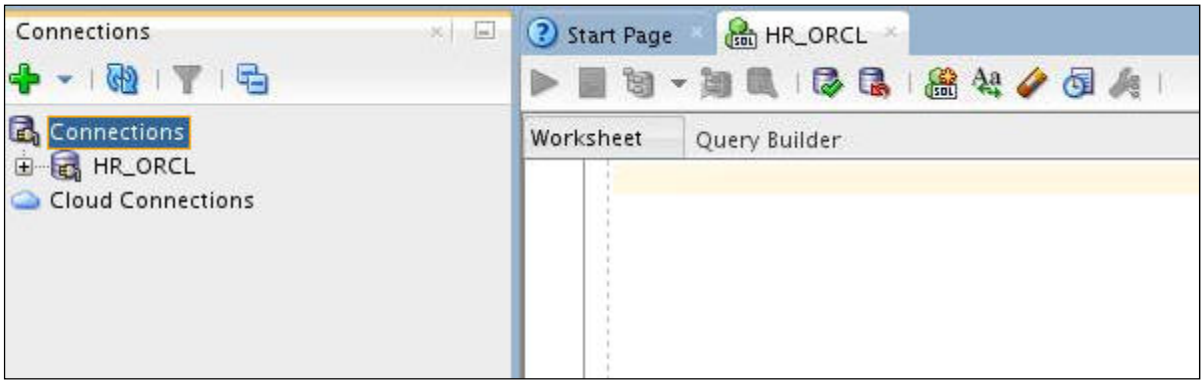
c. Enter **`HR_ORCL`** for the Connection Name (or any other name that identifies your connection). Enter the Username and Password. Specify **`localhost`** for the Hostname and enter **`ORCL`** for the SID. Click **Test**.



d. The status of the connection was tested successfully. The connection was not saved however. Click Save to save the connection, and then click **Connect**.

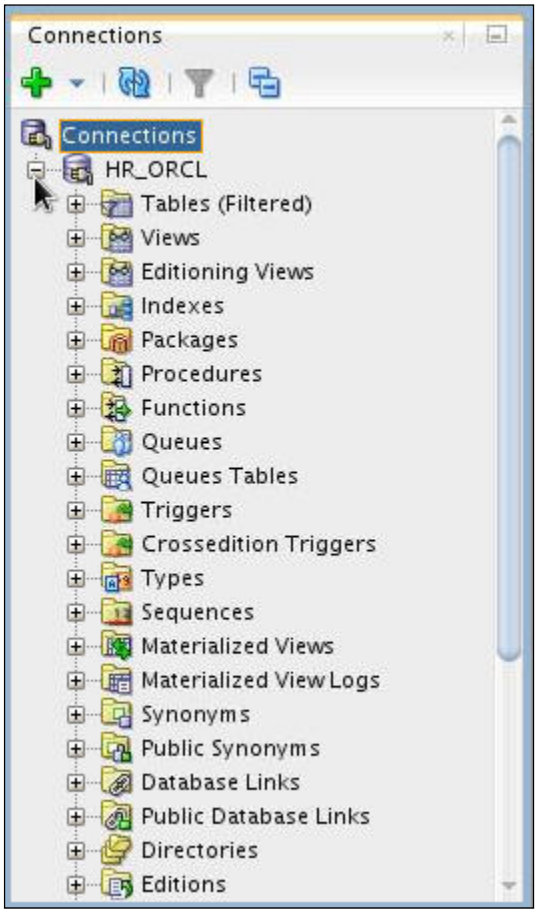


The connection was saved and you see the database in the list.



Note: When a connection is opened, a SQL Worksheet is opened automatically. The SQL Worksheet allows you to execute SQL against the connection that you just created.

e. Expand **HR_ORCL**. The various database objects that are available in that schema are displayed.



Practices for Lesson 2: SQL Language Enhancements

Chapter 2

Practice 2-1: SQL Row Limiting Clause

Overview

In this practice, you execute `SQL SELECT` statements by using the SQL row limiting clauses, such as `FETCH FIRST` and `FETCH NEXT`.

Note: You are encouraged to first code the tasks given as follows and test your own answers before looking at the solution.

Assumptions

You should have performed Practice 1-1.

Tasks

1. Execute a `SQL SELECT` statement by using the `FETCH FIRST` keyword that returns employees with the lowest `employee_id` values from the `employees` table.
2. Execute a `SQL SELECT` statement by using the `FETCH NEXT` and `OFFSET` keywords that return the next five employees with the lowest `employee_id` values from the `employees` table.
3. Execute a `SQL SELECT` statement by using the `FETCH FIRST` keyword that returns 5 percent of the employees with the lowest salaries from the `employees` table.
4. Execute a `SQL SELECT` statement by using the `FETCH FIRST` keyword that returns 5 percent of the employees with the lowest salaries by using the `WITH TIES` keyword from the `employees` table.

Solution 2-1: SQL Row Limiting Clause

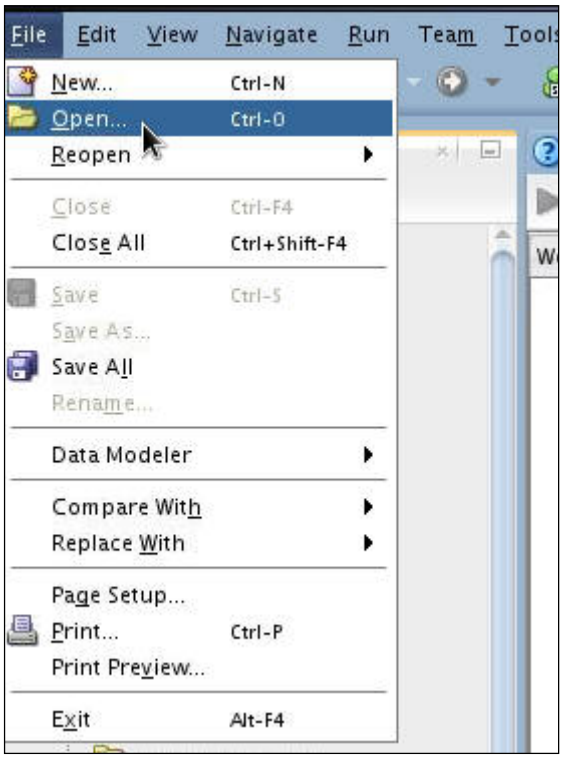
Tasks

1. Execute a `SQL SELECT` statement by using the `FETCH FIRST` keyword that returns employees with the lowest `employee_id` values from the `employees` table.

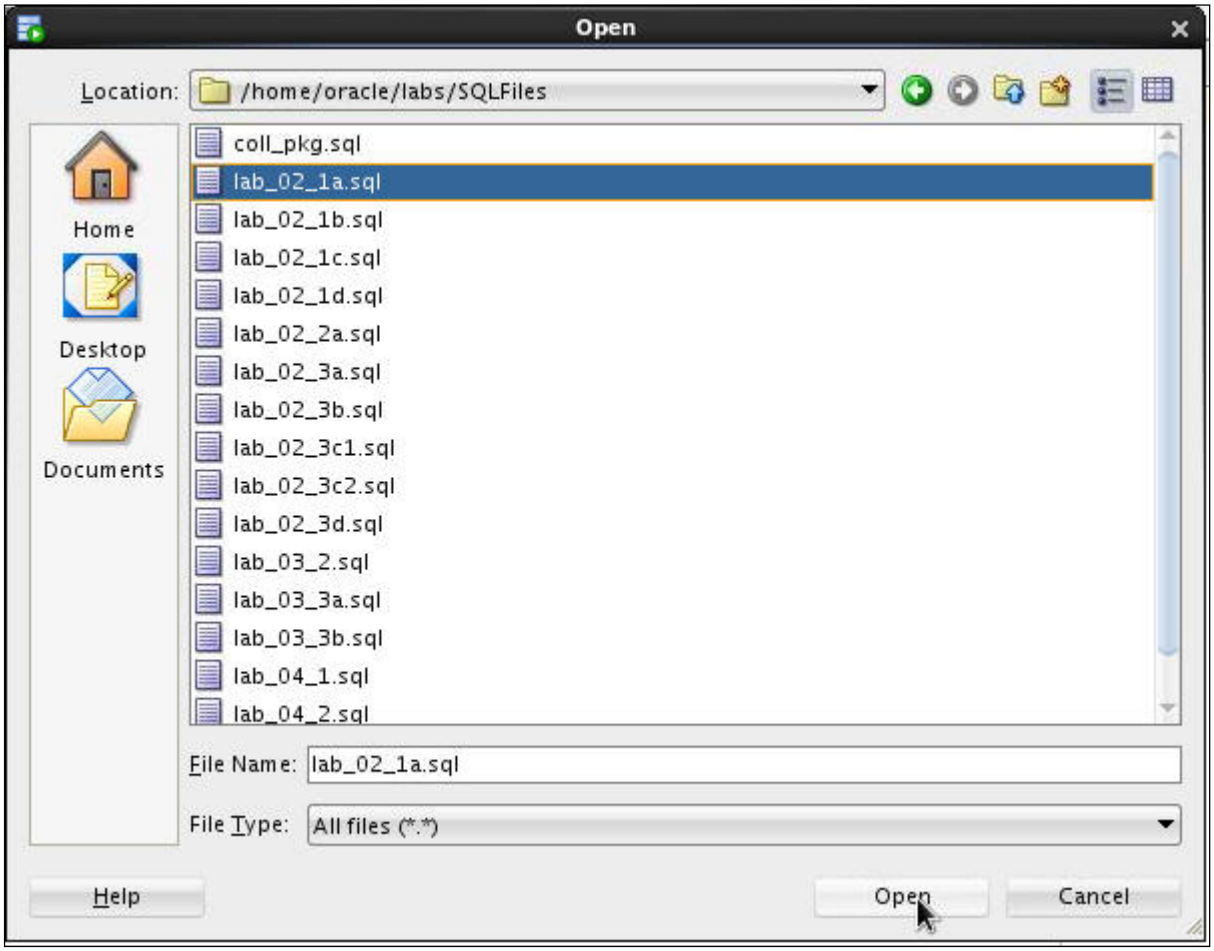
   To perform the preceding task, in SQL Developer, open the `lab_02_1a.sql` file that is available in the `/home/oracle/labs/SQLFiles` folder. This file contains the SQL script that returns the first five `employee_id` and `first_name` values from the `employees` table with the lowest `employee_id` values by using the `FETCH FIRST` keyword.

   Perform the following steps:

   a. In SQL Developer, select **File** > **Open**.

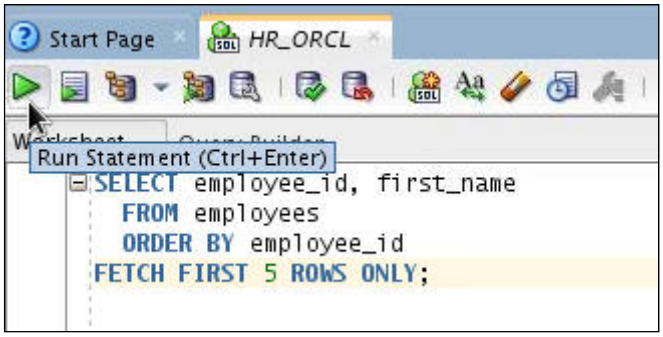


b. Navigate to `/home/oracle/labs/SQLFiles` and select `lab_02_1a.sql`. Click **Open**.

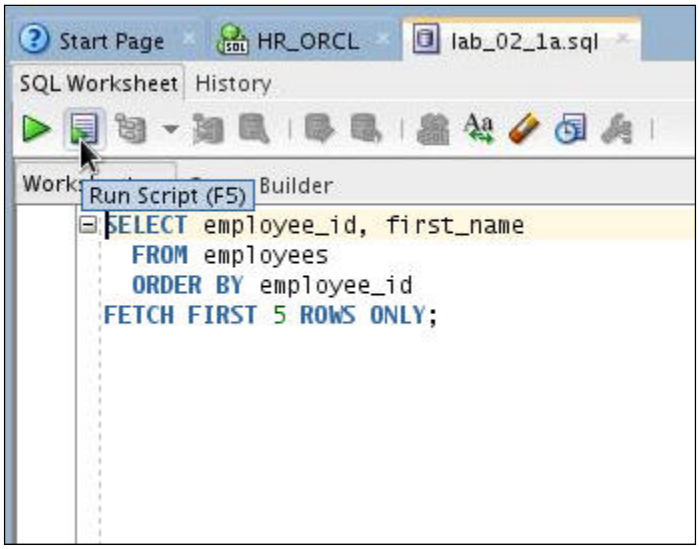


Note: Optionally, you can copy the following SQL statement and paste it in the `HR_ORCL` worksheet. Click the **Run Statement** icon to execute the SQL query.

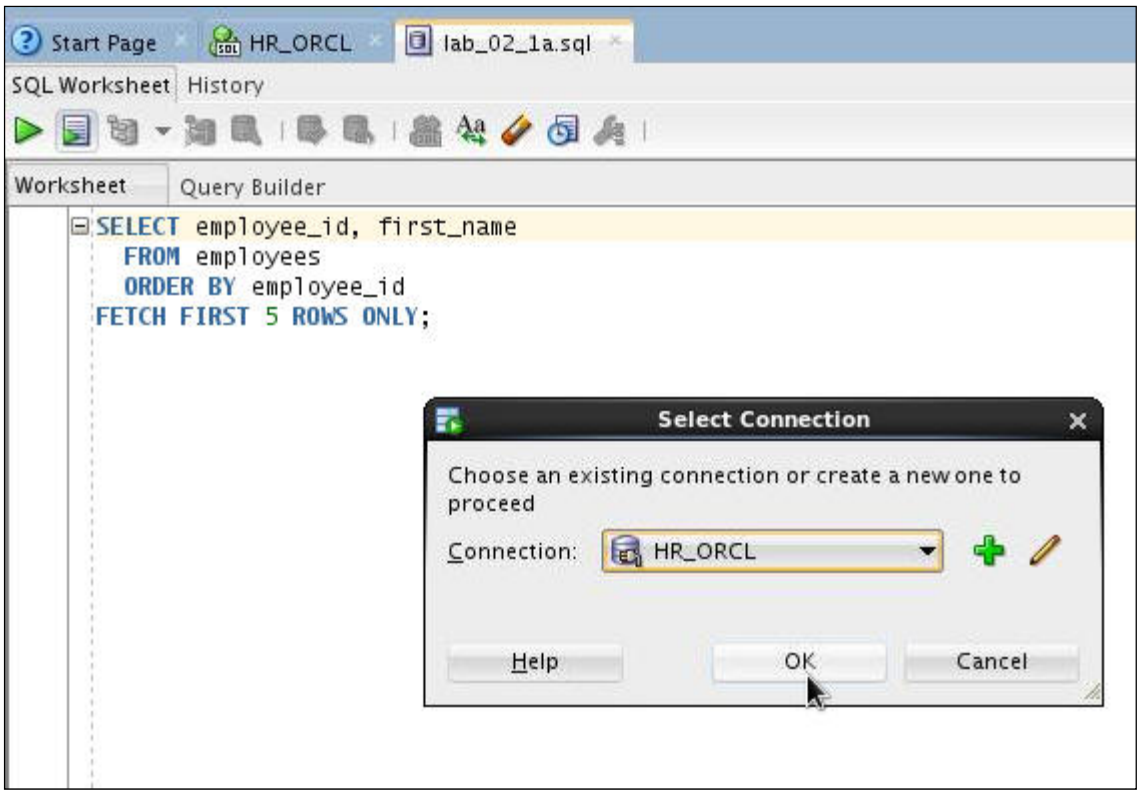
```
SELECT employee_id, first_name
FROM employees
ORDER BY employee_id
FETCH FIRST 5 ROWS ONLY;
```



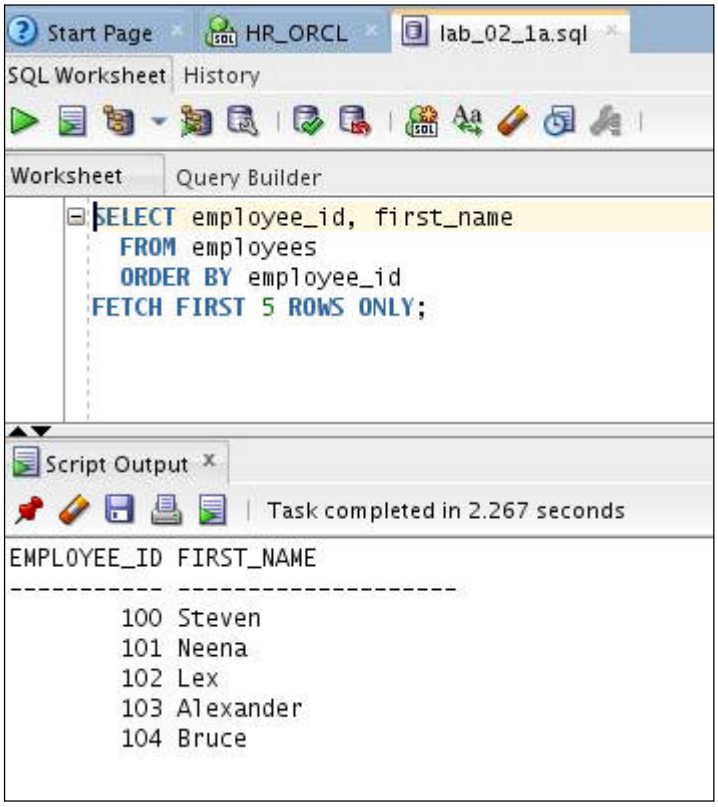
c. The SQL query to display the first five `employee_id` and `first_name` values of employees with the lowest `employee_id` values by using the `FETCH FIRST` keyword is displayed in the SQL Worksheet. Click the **Run Script** icon to run the SQL script.



d. For Select Connection, make sure that the **`HR_ORCL`** connection is selected and click **OK**.



e. Notice that Script Output displays the first five employees with the lowest `employee_id`.



2. Execute a `SQL SELECT` statement by using the `FETCH NEXT` and `OFFSET` keywords that return the next five employees with the lowest `employee_id` values from the `employees` table.

   To perform the preceding task, in SQL Developer, open the `lab_02_1b.sql` file that is available in the `/home/oracle/labs/SQLFiles` folder. This file contains the SQL script that returns the next five `employee_id` and `first_name` (following that fetched in the preceding task from the `employees` table) with the lowest `employee_id` values by using the `FETCH FIRST` and `OFFSET` keywords.

   Perform the following steps:

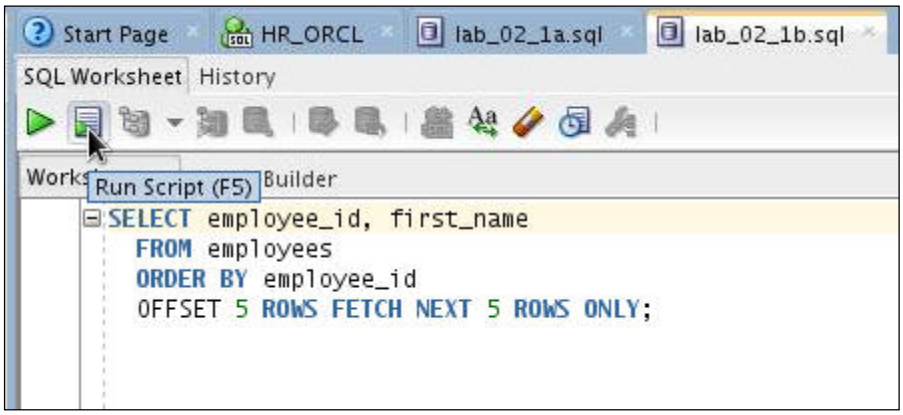
   a. In SQL Developer, select **File** > **Open**.

   b. Navigate to `/home/oracle/labs/SQLFiles` and select `lab_02_1b.sql`. Click **Open**.

      Note: Optionally, you can copy the following SQL statement and paste it in the `HR_ORCL` worksheet. Click the **Run Statement** icon to execute the SQL query.

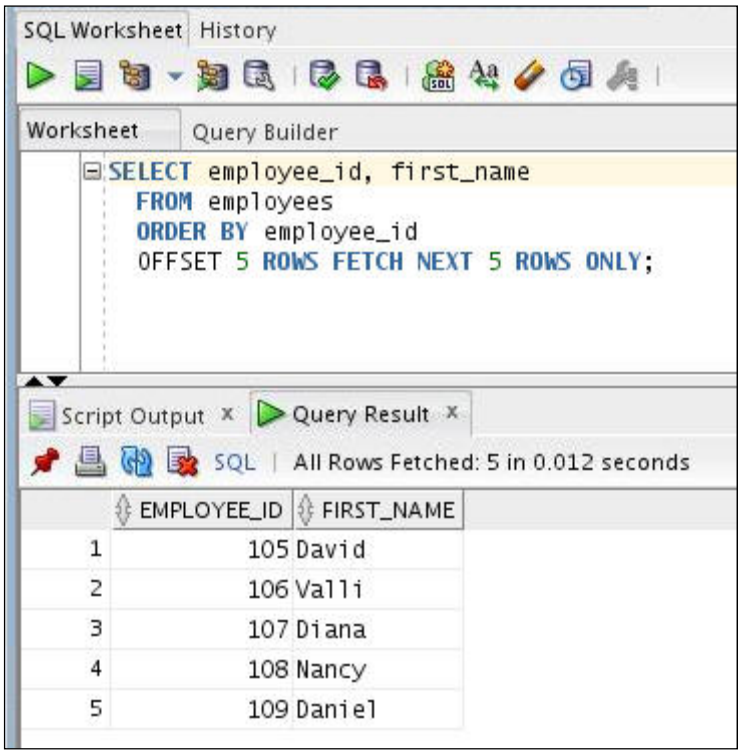
```
SELECT employee_id, first_name
FROM employees
ORDER BY employee_id
OFFSET 5 ROWS FETCH NEXT 5 ROWS ONLY;
```

c. The SQL query to return the next five employees (following that fetched in the preceding task) with the lowest `employee_id` values by using the `FETCH NEXT` and `OFFSET` keywords is displayed in the SQL Worksheet. Click the **Run Script** icon to run the SQL script.



d. Make sure that the **`HR_ORCL`** connection is selected and click **OK**.

e. The next five employees are displayed. The `OFFSET` keyword skips the first five employees and the `FETCH NEXT` keyword fetches the next five rows of employees.



3. Execute a `SQL SELECT` statement by using the `FETCH FIRST` keyword that returns 5 percent of the employees with the lowest salaries from the `employees` table.

   To perform the preceding task, in SQL Developer, open the `lab_02_1c.sql` file that is available in the `/home/oracle/labs/SQLFiles` folder. This file contains the SQL script that returns `employee_id`, `first_name`, and `salary` of 5 percent of the employees with the lowest salaries from the `employees` table by using the `FETCH FIRST` keyword.

   Perform the following steps:

   a. In SQL Developer, select **File** > **Open**.

   b. Navigate to `/home/oracle/labs/SQLFiles` and select `lab_02_1c.sql`. Click **Open**.

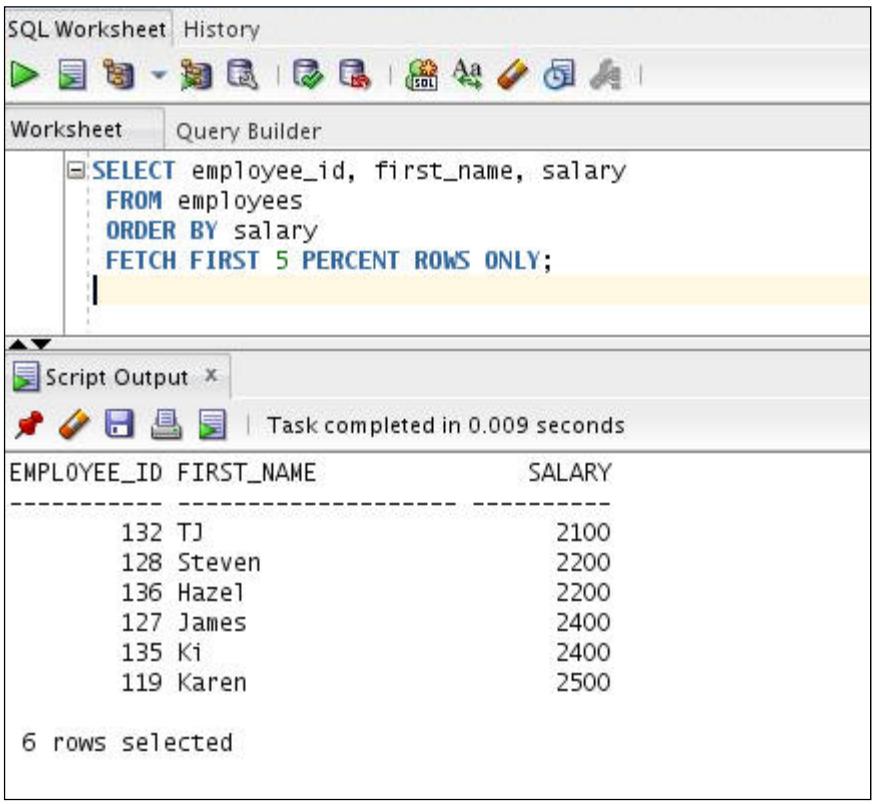
      Note: Optionally, you can copy the following SQL query and paste it in the `HR_ORCL` worksheet. Click the **Run Statement** icon to execute the SQL query.

```
SELECT employee_id, first_name, salary
FROM employees
ORDER BY salary
FETCH FIRST 5 PERCENT ROWS ONLY;
```

   c. The SQL query that returns `employee_id`, `first_name`, and `salary` of 5 percent of the employees with the lowest salaries from the `employees` table by using the `FETCH FIRST` keyword is displayed in the SQL Worksheet. Click the **Run Script** icon to run the SQL script.

   d. Make sure that the **`HR_ORCL`** connection is selected and click **OK**.

   e. The output displays 5 percent of the employees with the lowest salaries.



4. Execute a SQL SELECT statement by using the FETCH FIRST keyword that returns 5 percent of the employees with the lowest salaries by using the WITH TIES keyword from the employees table.

   To perform the preceding task, in SQL Developer, open the lab_02_1d.sql file that is available in the /home/oracle/labs/SQLFiles folder. This file contains the SQL script that returns employee_id, first_name, and salary of 5 percent of the employees with the lowest salaries, plus all additional employees with the same salary as the last row fetched in the previous task from the employees table by using the FETCH FIRST keyword and the WITH TIES keyword.

   Perform the following steps:

   a. In SQL Developer, select **File** > **Open**.

   b. Navigate to /home/oracle/labs/SQLFiles and select lab_02_1d.sql. Click **Open**.

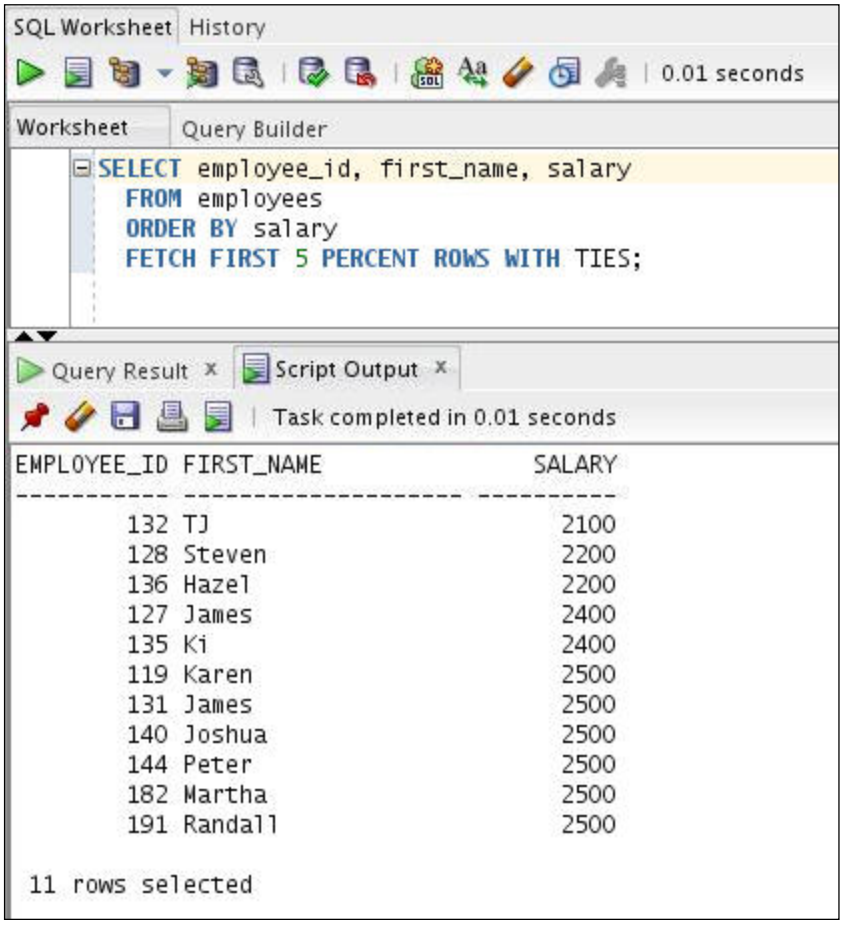
      Note: Optionally, you can copy the following SQL query and paste it in the HR_ORCL worksheet. Click the **Run Statement** icon to execute the SQL query.

```
SELECT employee_id, first_name, salary
  FROM employees
  ORDER BY salary
  FETCH FIRST 5 PERCENT ROWS WITH TIES;
```

   c. The SQL query that returns employee_id, first_name, and salary of 5 percent of the employees with the lowest salaries, plus all additional employees with the same salary as the last row fetched in the previous task from the employees table by using the FETCH FIRST keyword and the WITH TIES keyword is displayed in the SQL Worksheet. Click the **Run Script** icon to run the SQL script.

   d. Make sure that the **HR_ORCL** connection is selected and click **OK**.

e. The 5 percent of employees with the lowest salaries, plus all additional employees with the same salary as the last row fetched in the previous task from the `employees` table is displayed.



Practice 2-2: Invisible and Hidden Columns

Overview

In this practice, you:

- Create a table with an invisible column in SQL Developer
- View information about an invisible column in SQL*Plus by using the `DESCRIBE` command

Note: You are encouraged to first code the task given as follows and test your own answer before looking at the solution.

Assumptions

You should have performed Practice 1-1.

Tasks

1. You want to create a table that stores employee details such as employee ID, employee information, and employee account number. Employee account number is stored by using an invisible column because you do not want to display the account number. To do this, create a table called `test_invisible_cols` with an invisible column. The invisible column would contain the employee account number and be of type number. When you execute the `DESCRIBE` command to describe the table, you notice that the invisible column does not appear in the result. To view the invisible column, you log in to SQL*Plus and set the `COLINVISIBLE` session state to `ON`. You notice that the invisible column is now displayed in the result.

Solution 2-2: Invisible and Hidden Columns

Tasks

1. You want to create a table that stores employee details such as employee ID, employee information, and employee account number. Employee account number is stored by using an invisible column because you do not want to display the account number. To do this, create a table called `test_invisible_cols` with an invisible column. The invisible column would contain the employee account number and be of type number. When you execute the `DESCRIBE` command to describe the table, you notice that the invisible column does not appear. To view the invisible column, you log in to SQL*Plus and set the `COLINVISIBLE` session status to `ON`. You notice that the invisible column is now displayed in the result.

   To perform the preceding task, in SQL Developer, open the `lab_02_2a.sql` file that is available in the `/home/oracle/labs/SQLFiles` folder. This file contains the SQL script to create a table called `test_invisible_cols` with an invisible column. The `DESCRIBE` command lists the table columns. You notice that with the default condition of `SET COLINVISIBLE OFF`, the invisible column does not appear in the result.

   Perform the following steps:

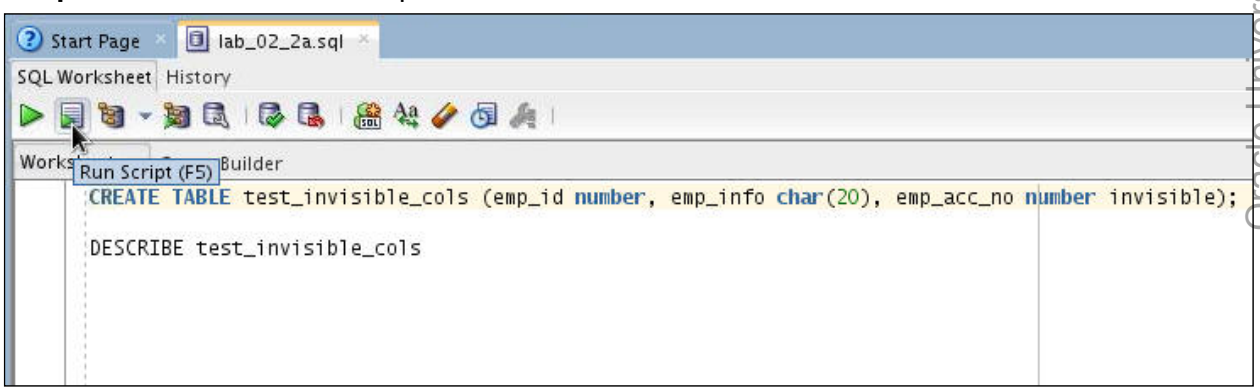
   a. In SQL Developer, select **File** > **Open**.

   b. Navigate to `/home/oracle/labs/SQLFiles` and select `lab_02_2a.sql`. Click **Open**.

      Note: Optionally, you can copy the following SQL statements and paste them in the `HR_ORCL` worksheet. Click the **Run Statement** icon to execute the SQL statements.

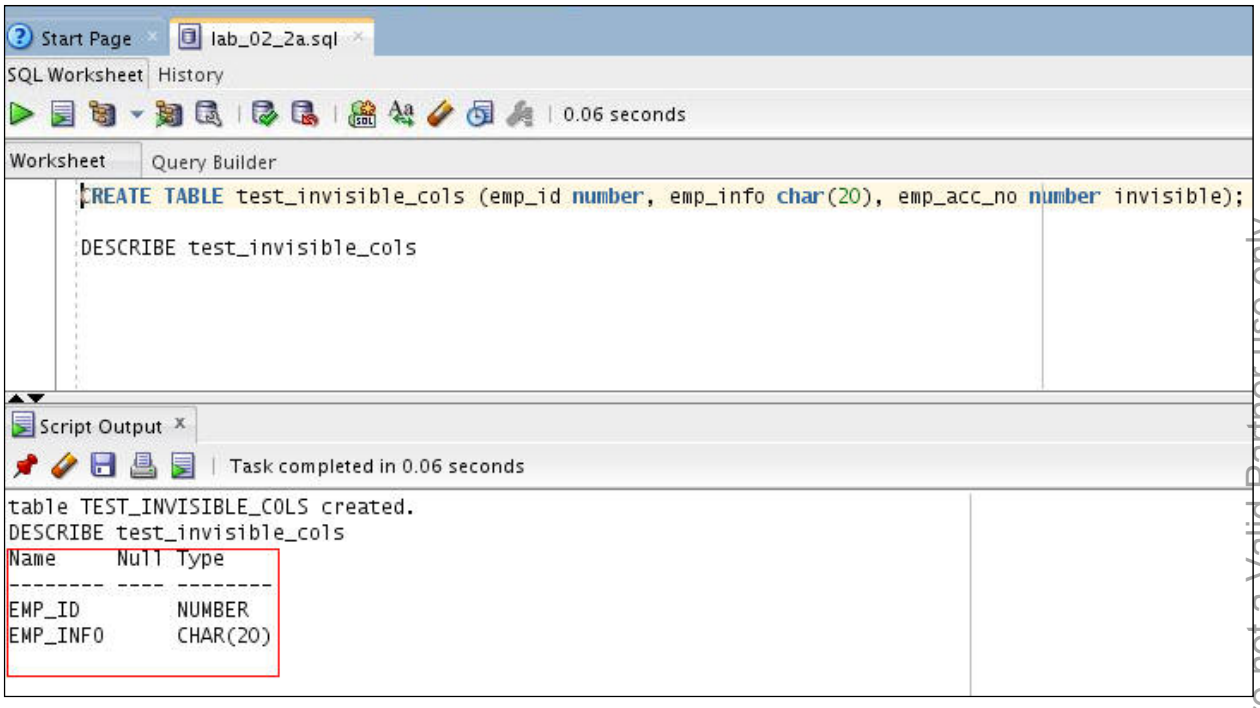
```
CREATE TABLE test_invisible_cols (emp_id number, emp_info
char(20), emp_acc_no number invisible);


DESCRIBE test_invisible_cols
```

   c. The SQL query to create a table called `test_invisible_cols` with an invisible column is displayed. The `DESCRIBE` command lists the table columns. Click the **Run Script** icon to run the SQL script.



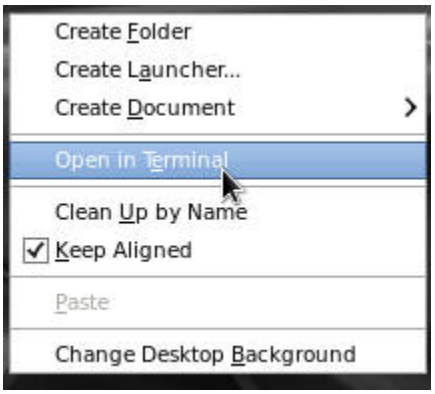
   d. Make sure that the **HR_ORCL** connection is selected and click **OK**.

e. Notice that the invisible column `emp_acc_no` does not appear in the result. This is because of the default value of `SET COLINVISIBLE OFF`.

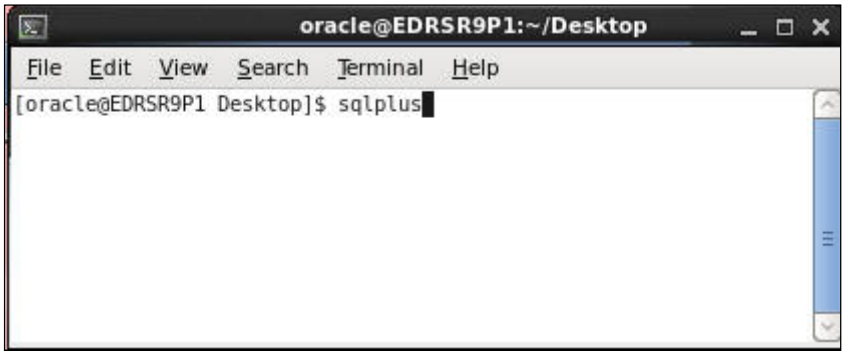


To view the invisible column, log in to SQL*Plus and change the default setting of `SET COLINVISIBLE` to `ON`. You then use the `DESCRIBE` command again to list the table columns. The invisible column will be displayed now.

a. Open a terminal.



b. Type `sqlplus` and press the Enter key.



c. Enter the username and password.

```
oracle@EDRSR9P1:~/Desktop
File  Edit  View  Search  Terminal  Help
[oracle@EDRSR9P1 Desktop]$ sqlplus

SQL*Plus: Release 12.1.0.1.0 Production on Mon Dec 15 08:36:46 2014

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

Enter user-name: hr
Enter password:
```

d. Type `SHOW COLINVISIBLE` and press the Enter key.

```
oracle@EDRSR9P1:~/Desktop
File  Edit  View  Search  Terminal  Help
[oracle@EDRSR9P1 Desktop]$ sqlplus

SQL*Plus: Release 12.1.0.1.0 Production on Mon Dec 15 08:36:46 2014

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

Enter user-name: hr
Enter password:
Last Successful login time: Mon Dec 15 2014 08:33:56 +00:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing opt
ions

SQL> SHOW COLINVISIBLE
```

e. Notice that the `COLINVISIBLE` status is set to `off`.

```
oracle@EDRSR9P1:~/Desktop
File  Edit  View  Search  Terminal  Help
[oracle@EDRSR9P1 Desktop]$ sqlplus

SQL*Plus: Release 12.1.0.1.0 Production on Mon Dec 15 08:36:46 2014

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

Enter user-name: hr
Enter password:
Last Successful login time: Mon Dec 15 2014 08:33:56 +00:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing opt
ions

SQL> SHOW COLINVISIBLE
colinvisible OFF
SQL>
```

f. Type `SET COLINVISIBLE ON` and press the Enter key.

```
oracle@EDRSR9P1:~/Desktop
File  Edit  View  Search  Terminal  Help
[oracle@EDRSR9P1 Desktop]$ sqlplus

SQL*Plus: Release 12.1.0.1.0 Production on Mon Dec 15 08:36:46 2014

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

Enter user-name: hr
Enter password:
Last Successful login time: Mon Dec 15 2014 08:33:56 +00:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing opt
ions

SQL> SHOW COLINVISIBLE
colinvisible OFF
SQL> SET COLINVISIBLE ON
```

g. Type `DESCRIBE TEST_INVISIBLE_COLS` and press the Enter key.

```
oracle@EDRSR9P1:~/Desktop
File  Edit  View  Search  Terminal  Help
[oracle@EDRSR9P1 Desktop]$ sqlplus

SQL*Plus: Release 12.1.0.1.0 Production on Mon Dec 15 08:36:46 2014

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

Enter user-name: hr
Enter password:
Last Successful login time: Mon Dec 15 2014 08:33:56 +00:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing opt
ions

SQL> SHOW COLINVISIBLE
colinvisible OFF
SQL> SET COLINVISIBLE ON
SQL> DESCRIBE TEST_INVISIBLE_COLS
```

Notice that the invisible column is now displayed.

```
oracle@EDRSR9P1:~/Desktop
File  Edit  View  Search  Terminal  Help
SQL*Plus: Release 12.1.0.1.0 Production on Mon Dec 15 08:36:46 2014

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

Enter user-name: hr
Enter password:
Last Successful login time: Mon Dec 15 2014 08:33:56 +00:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing opt
ions

SQL> SHOW COLINVISIBLE
colinvisible OFF
SQL> SET COLINVISIBLE ON
SQL> DESCRIBE TEST_INVISIBLE_COLS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMP_ID                                             NUMBER
 EMP_INFO                                           CHAR(20)
 EMP_ACC_NO (INVISIBLE)                             NUMBER

SQL>
```

Practice 2-3: Enhanced DDL Capabilities By Using the `ONLINE` Keyword

Overview

In this practice, you use the new `ONLINE` keyword to allow the execution of DML statements during the DDL operations.

Note: You are encouraged to first code the tasks given as follows and test your own answers before looking at the solution.

Assumptions

You should have performed Practice 1-1.

Tasks

1. Create an index on a table and execute the `DROP INDEX` statement by using the `ONLINE` keyword to drop the index.
2. Create a table and add a primary key constraint to a column in the table. Execute the `DROP CONSTRAINT` statement by using the `ONLINE` keyword to drop the primary key constraint.
3. Create a table and an index for the table. Alter the index by using the `ALTER INDEX` statement with the `UNUSABLE ONLINE` keyword. Execute the `SELECT` statement to view the status of the index.
4. Create a table and insert values into the table. Alter the table and set a column as unused by using the `SET UNUSED` keyword. Also specify the `ONLINE` keyword in the statement to allow DML operations to be performed on the table while marking the column as `UNUSED`.

Solution 2-3: Enhanced DDL Capabilities by Using the `ONLINE` Keyword

Tasks

1. Create an index on a table and execute the `DROP INDEX` statement by using the `ONLINE` keyword to drop the index.

   To perform the preceding task, in SQL Developer, open the `lab_02_3a.sql` file that is available in the `/home/oracle/labs/SQLFiles` folder. This file contains the SQL script to create a table called `myemp1` and an index called `myemp_ix` on the table. The SQL script also includes the `DROP INDEX` statement that uses the `ONLINE` clause to drop the index, thereby indicating that DML operations on the table are allowed while dropping the index.

   Perform the following steps:

   a. In SQL Developer, select **File** > **Open**.

   b. Navigate to `/home/oracle/labs/SQLFiles` and select `lab_02_3a.sql`. Click **Open**.

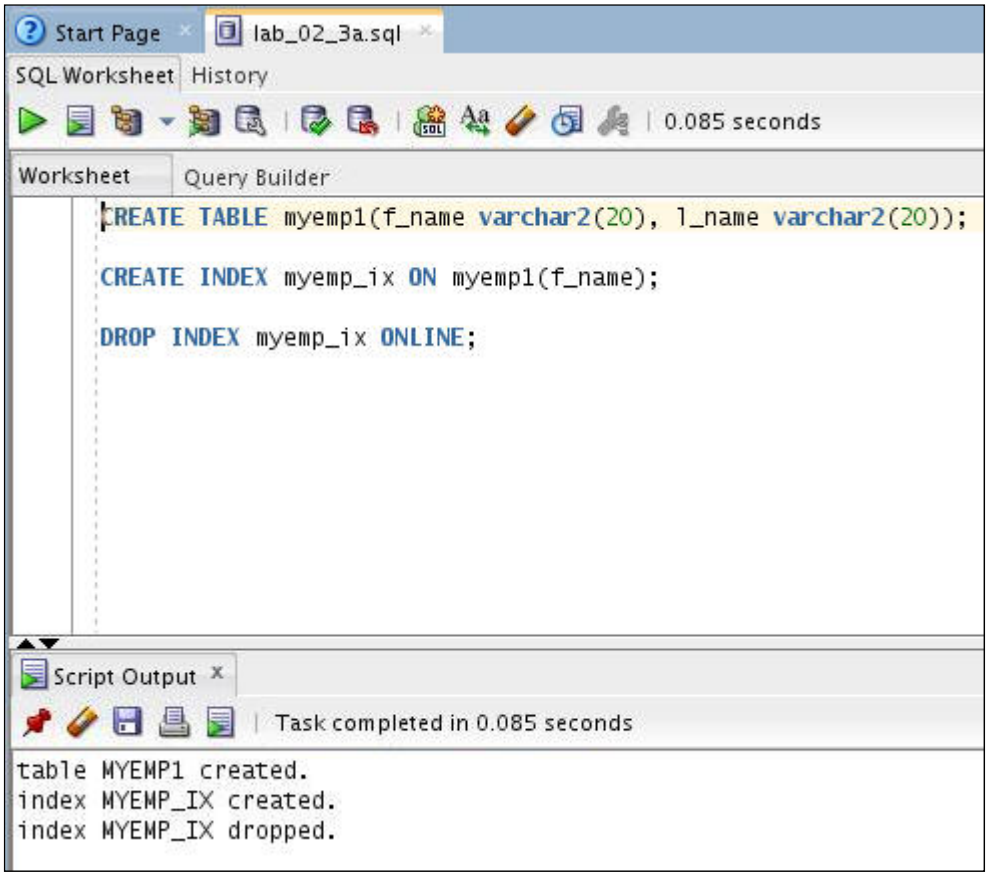
      Note: Optionally, you can copy the following SQL statements and paste them in the `HR_ORCL` worksheet. Click the **Run Statement** icon to execute the SQL statements.

```
CREATE TABLE myemp1(f_name varchar2(20), l_name varchar2(20));
CREATE INDEX myemp_ix ON myemp1(f_name);
DROP INDEX myemp_ix ONLINE;
```

   c. The SQL script contains a query to create a table called `myemp1` and an index called `myemp_ix` on the `myemp1` table. The index `myemp_ix` can be dropped by using the `ONLINE` keyword. Click the **Run Script** icon to run the SQL script.

   d. Make sure that the **`HR_ORCL`** connection is selected and click **OK**.

e. The `ONLINE` clause indicates that DML operations on the table are allowed while dropping the index.



2. Create a table and add a primary key constraint to a column in the table. Execute the `DROP CONSTRAINT` statement by using the `ONLINE` keyword to drop the primary key constraint.

   To perform the preceding task, in SQL Developer, open the `lab_02_3b.sql` file that is available in the `/home/oracle/labs/SQLFiles` folder. This file contains the SQL script to create a table called `emp1` with the columns `emp_id` and `emp_name`. The script also contains a SQL statement to add a primary key constraint to the `emp_id` column in the table. The table is altered to drop the constraint by using the `ONLINE` keyword.

   Perform the following steps:

   a. In SQL Developer, select **File** > **Open**.

   b. Navigate to `/home/oracle/labs/SQLFiles` and select `lab_02_3b.sql`. Click **Open**.

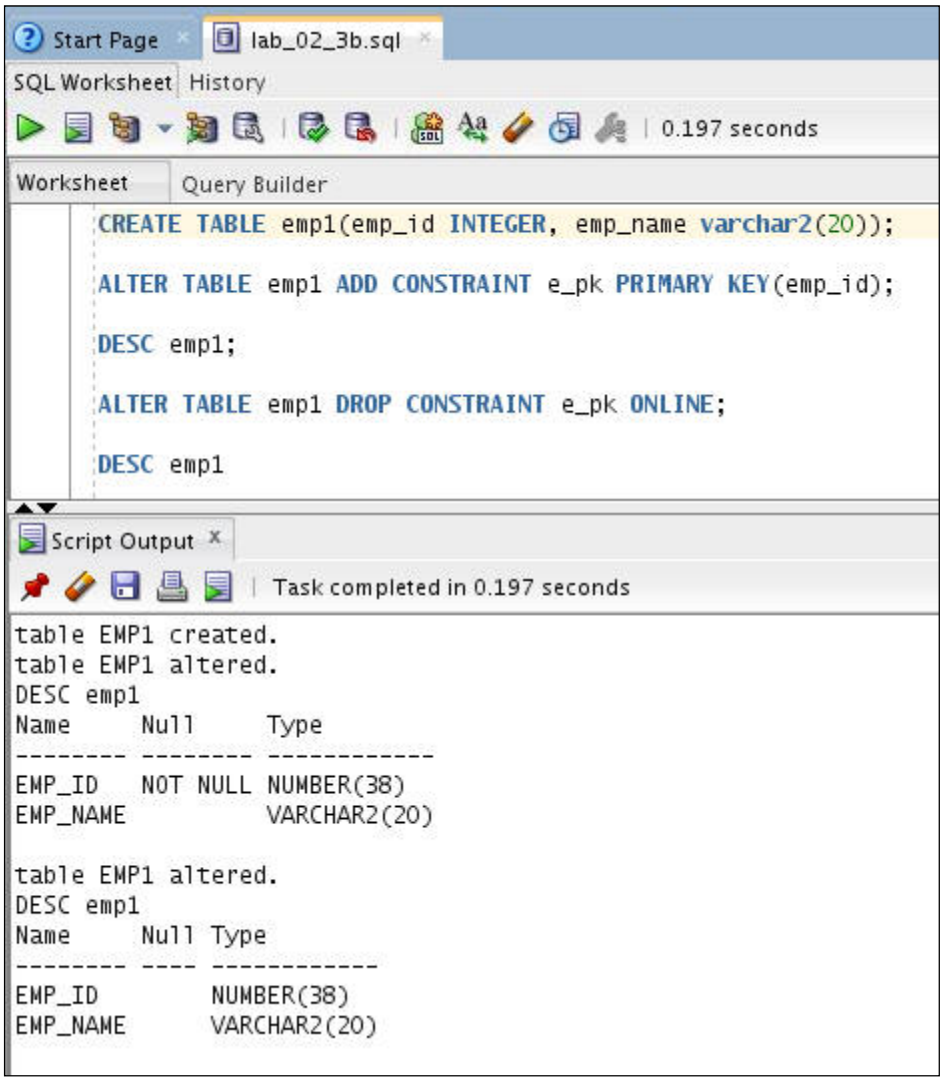
      Note: Optionally, you can copy the following SQL statements and paste them in the `HR_ORCL` worksheet. Click the **Run Statement** icon to execute the SQL statements.

```
CREATE TABLE emp1(emp_id INTEGER, emp_name varchar2(20));

ALTER TABLE emp1 ADD CONSTRAINT e_pk PRIMARY KEY(emp_id);

DESC emp1;

ALTER TABLE emp1 DROP CONSTRAINT e_pk ONLINE;

DESC emp1;
```

c. The SQL script contains a SQL statement to create a table called `emp1`. A primary key constraint is applied on the table column. The SQL query to drop the constraint by using the `ONLINE` keyword is executed. By specifying the `ONLINE` keyword in the drop constraint, the SQL statement indicates that DML operations are allowed on the table while dropping the constraint. This enables simpler application development, especially for application migrations. Click the **Run Script** icon to run the SQL script.

d. Make sure that the **`HR_ORCL`** connection is selected and click **OK**.

e. The `ONLINE` clause indicates that DML operations on the table are allowed while dropping the constraint.



3. Create a table and an index for the table. Alter the index by using the `ALTER INDEX` statement with the `UNUSABLE ONLINE` keyword. Execute the `SELECT` statement to view the status of the index.

   To perform the preceding task, in SQL Developer, open the `lab_02_3c1.sql` file that is available in the `/home/oracle/labs/SQLFiles` folder. This file contains the SQL script to create an `emp2` table and an `ename_ix` index on the table. The script also contains an `ALTER INDEX` statement with the `UNUSABLE ONLINE` keyword and a `SELECT` statement to view the status of the index.

Perform the following steps:

a. In SQL Developer, select **File** > **Open**.

b. Navigate to `/home/oracle/labs/SQLFiles` and select `lab_02_3c1.sql`. Click **Open**.

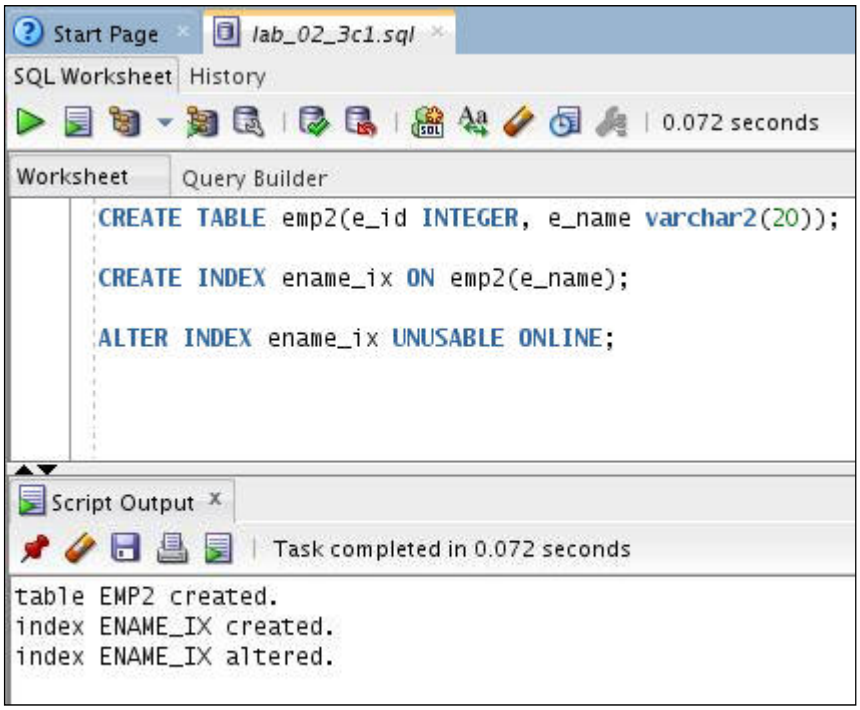
   Note: Optionally, you can copy the following SQL statements and paste them in the `HR_ORCL` SQL Worksheet. Click the **Run Statement** icon to execute the SQL statements.

```
CREATE TABLE emp2(e_id INTEGER, e_name varchar2(20));

CREATE INDEX ename_ix ON emp2(e_name);

ALTER INDEX ename_ix UNUSABLE ONLINE;
```

c. The SQL script contains a SQL statement to create a table called `emp2`. An index `ename_ix` is created on the `emp2` table. The `ename_ix` index is altered by using the `UNUSABLE` and `ONLINE` keywords. The `UNUSABLE` keyword is specified to mark the index, index partitions, or index subpartitions as `UNUSABLE` . The `ONLINE` keyword is used to indicate that DML operations on the table or partition are allowed while marking the index as `UNUSABLE`. Click the **Run Script** icon to run the SQL script.

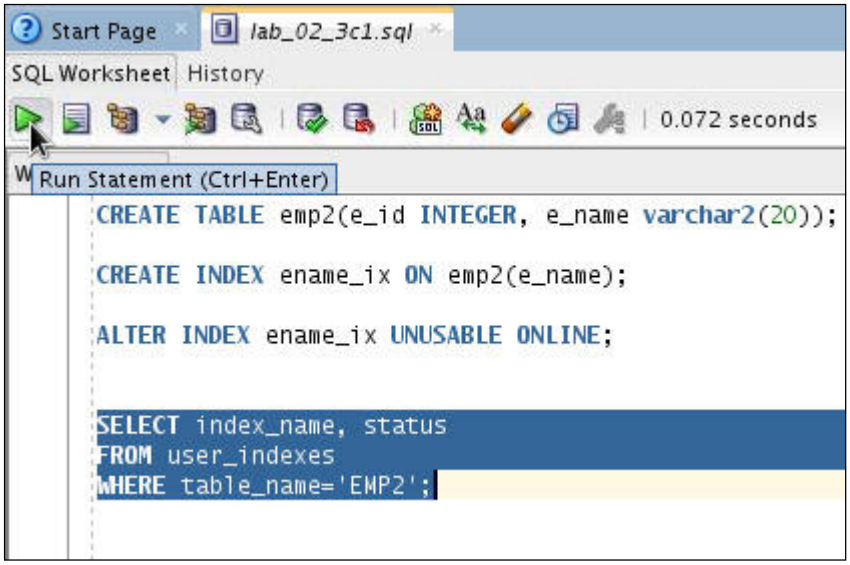
d. Make sure that the **`HR_ORCL`** connection is selected and click **OK**.

e. The `UNUSABLE` keyword is specified to mark the index, index partitions, or index subpartitions as `UNUSABLE`. The `ONLINE` keyword is used to indicate that DML operations on the table or partition are allowed while marking the index as `UNUSABLE`.

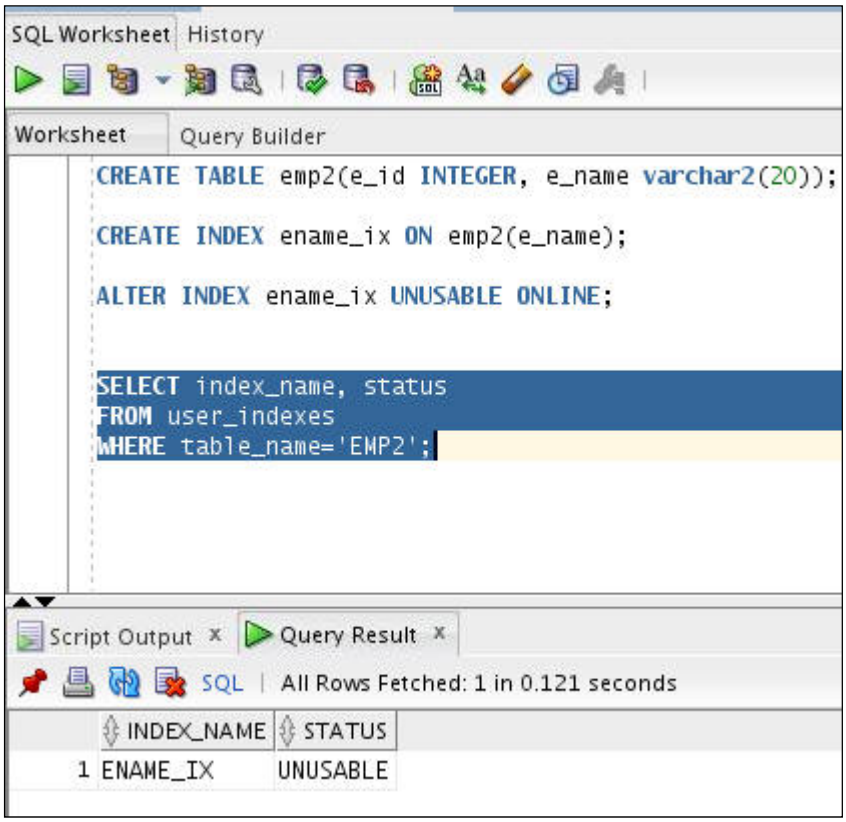


f. To verify the status of the index, enter the following SQL statement in the SQL Worksheet that returns the status of the index. Click the **Run Statement** icon. Alternatively, navigate to `/home/oracle/labs/SQLFiles` and select `lab_02_3c2.sql`. Click **Open**. Click the **Run Script** icon.

```
SELECT index_name, status
FROM user_indexes
WHERE table_name='EMP2';
```



g. Notice that the status of the index is unusable.



4. Create a table and insert values into the table. Alter the table and set a column as unused by using the `SET UNUSED` keyword. Also specify the `ONLINE` keyword in the statement to allow DML operations to be performed on the table while marking the column as `UNUSED`.

   To perform the preceding task, in SQL Developer, open the `lab_02_3d.sql` file that is available in the `/home/oracle/labs/SQLFiles` folder. This file contains the SQL script to create a table called `myemp3` and insert values into the table. The table is altered and the column `emp_name` is set as unused by using the `SET UNUSED` keyword. Also, the `ONLINE` keyword in the statement allows DML operations to be performed on the table while marking the column as `UNUSED`.

   Perform the following steps:

   a. In SQL Developer, select **File** > **Open**.

   b. Navigate to `/home/oracle/labs/SQLFiles` and select `lab_02_3d.sql`. Click **Open**.

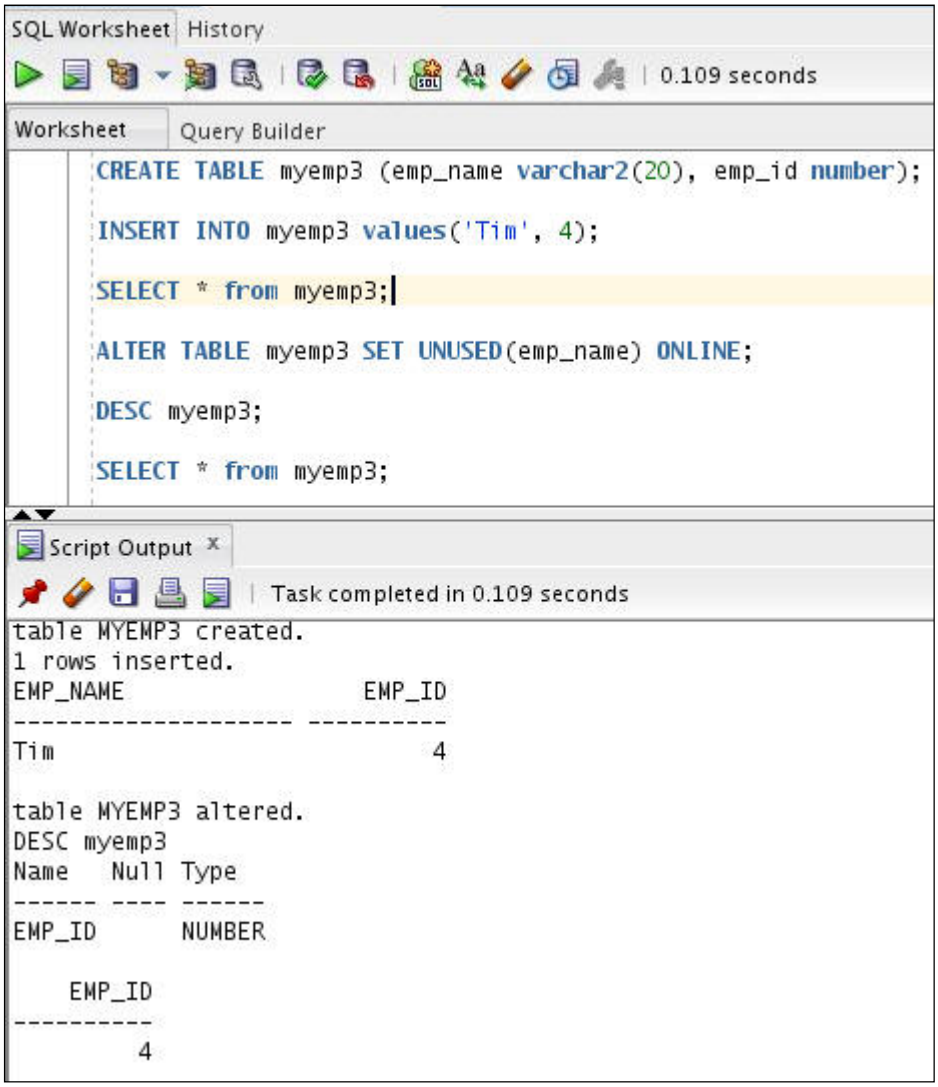
      Note: Optionally, you can copy the following SQL statements and paste them in the `HR_ORCL` SQL Worksheet. Click the **Run Statement** icon to execute the SQL statements.

```
CREATE TABLE myemp3 (emp_name varchar2(20), emp_id number);

INSERT INTO myemp3 values('Tim', 4);

SELECT * from myemp3;

ALTER TABLE myemp3 SET UNUSED(emp_name) ONLINE;

DESC myemp3;

SELECT * from myemp3;
```

   c. Make sure that the **`HR_ORCL`** connection is selected and click **OK**.

d. The SQL script contains the SQL statement to create a table called `myemp3`. Values are inserted into the table, and then the table is altered by using the `SET UNUSED` and `ONLINE` keywords. The `SET UNUSED` keyword is specified to mark one or more columns as unused. The `ONLINE` keyword is specified to indicate that DML operations on the table are allowed while marking the column or columns as `UNUSED`.

Notice that the `emp_name` column is not displayed after altering the table by using the `SET UNUSED` keyword.



Practices for Lesson 3: Data Type Enhancements

Chapter 3

Practice 3-1: Using the SQL `IDENTITY` Column

Overview

In this practice, you use the SQL `IDENTITY` column in a table to automatically generate a sequence of numbers for that column.

Note: You are encouraged to first code the task given as follows and test your own answer before looking at the solution.

Assumptions

You should have completed Practice1-1.

Tasks

1. Create a table with a column as an `IDENTITY` column. While creating the table, the `IDENTITY` column is specified with the `ON_NULL` clause. Using this `IDENTITY` column in the table allows you to generate a sequence of numbers for that column while inserting values.

Solution 3-1: Using the SQL `IDENTITY` Column

Tasks

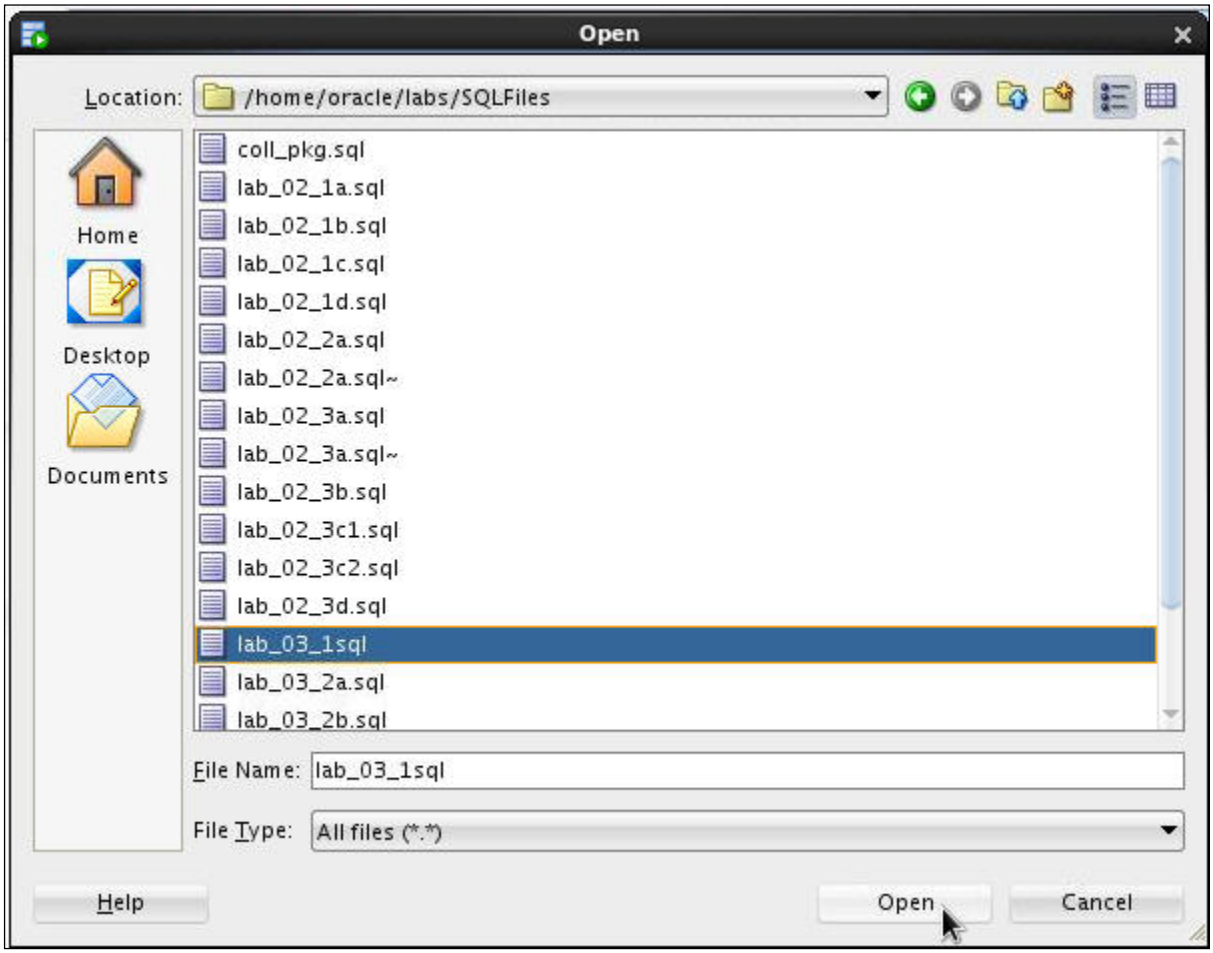
1. Create a table with a column as an `IDENTITY` column. While creating the table, the `IDENTITY` column is specified with the `ON_NULL` clause. Using this `IDENTITY` column in the table allows you to generate a sequence of numbers for that column while inserting values.

   To perform the preceding task, in SQL Developer, open the `lab_03_1.sql` file that is available in the `/home/oracle/labs/SQLFiles` folder. This file contains the SQL script to create a table called `identity_test` with `column1` being an `IDENTITY` column. While creating the table, the `IDENTITY` column is specified with the `ON_NULL` clause where Oracle database uses a sequence generator to assign a value to the column when you insert a value that evaluates to `NULL`.

   Perform the following steps:

   a. In SQL Developer, select **File** > **Open**.

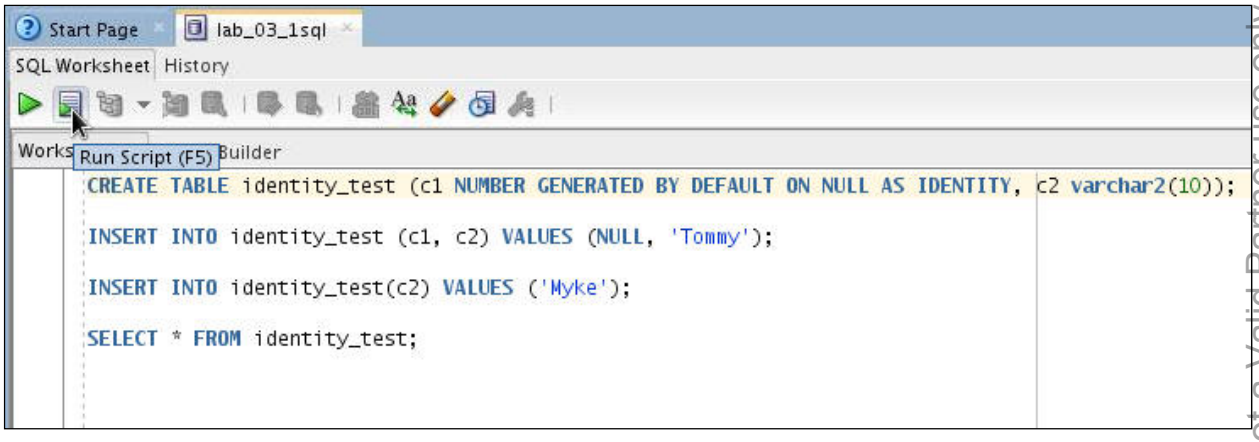
b. Navigate to `/home/oracle/labs/SQLFiles` and select `lab_03_1.sql`. Click **Open**.



Note: Optionally, you can copy the following SQL query and paste it in the `HR_ORCL` worksheet. Click the **Run Statement** icon to execute the SQL query.

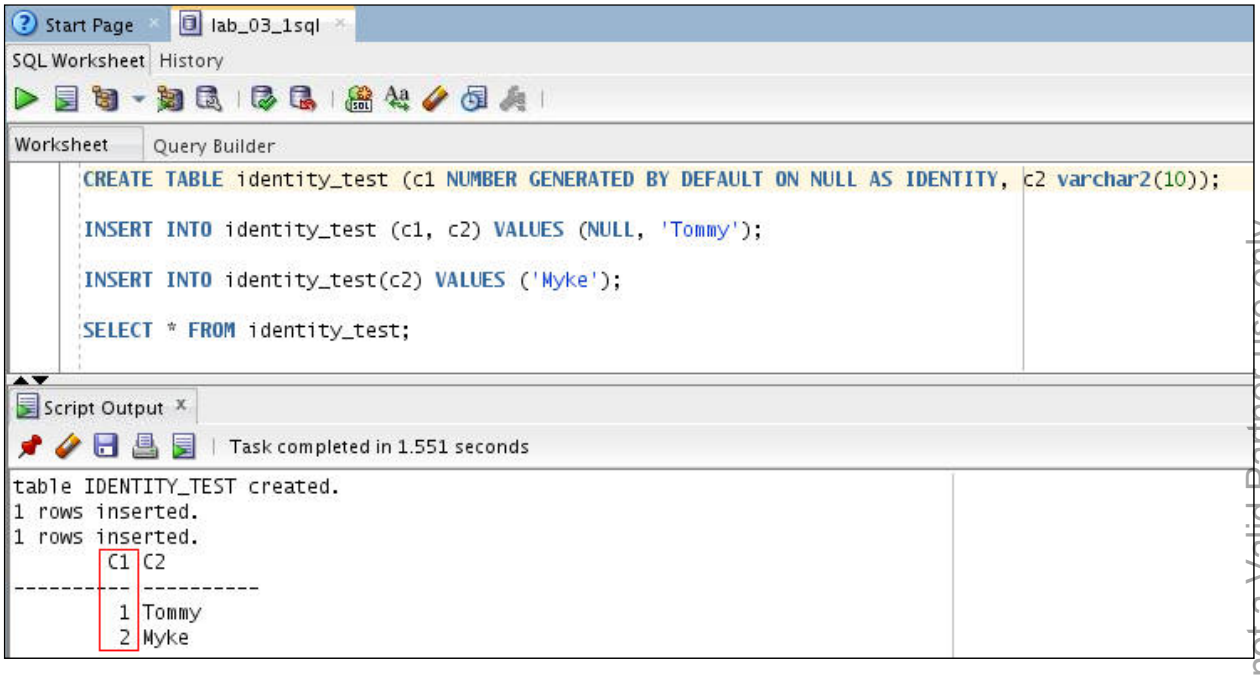
```
CREATE TABLE identity_test (c1 NUMBER GENERATED BY DEFAULT ON
NULL AS IDENTITY, c2 varchar2(10));

INSERT INTO identity_test (c1, c2) VALUES (NULL, 'Tommy');

INSERT INTO identity_test(c2) VALUES ('Myke');

SELECT * FROM identity_test;
```

c. The SQL statement to create a table called `identity_test` is displayed on the SQL Worksheet. The table contains an `IDENTITY` column. The `IDENTITY` column is assigned an increasing or decreasing integer value from a sequence generator for each subsequent `INSERT` statement. The `IDENTITY` column definition can be specified by using the `GENERATED ALWAYS` or `GENERATED BY DEFAULT` clause. The `IDENTITY` column is also specified with the `ON_NULL` clause where Oracle database uses a sequence generator to assign a value to the column when you insert a value that evaluates to `NULL`. Click the **Run Script** icon to run the SQL script.



d. For Select Connection, make sure that the **`HR_ORCL`** connection is selected and click **OK**. If prompted, enter the username and password.

e. Notice that the `IDENTITY` column has the integers generated automatically even when the values are not specified.



Practice 3-2: SQL Column Enhancements

Overview

In this practice, you understand some SQL column enhancements for SQL column defaulting by using explicit `NULL` and SQL column defaulting by using a sequence.

Note: You are encouraged to first code the tasks given as follows and test your own answers before looking at the solution.

Assumptions

You should have completed Practice1-1.

Tasks

1. To use SQL column defaulting by using explicit `NULL`, create a table with a `numeric` column and default the column to a value when a null condition occurs. Observe that when a null value is inserted into the table for this column, it defaults to the value specified.
2. To use SQL column defaulting by using a sequence, create a sequence, which starts with the value `1`. Also, create a table that has a `numeric` column and default the column to `sequence.nextval` as a SQL column defaulting expression when a null value is inserted into the table.

Solution 3-2: SQL Column Enhancements

Tasks

1. To use SQL column defaulting by using explicit NULL, create a table with a numeric column and default the column to a value when a null condition occurs. Observe that when a null value is inserted into the table for this column, it defaults to the value specified.

   To perform the preceding task, in SQL Developer, open the lab_03_2a.sql file that is available in the /home/oracle/labs/SQLFiles folder. This file contains the SQL script to create a table called foo with a numeric column. While creating the table, the numeric column has a NOT NULL condition, which defaults to a value specified if the value is NULL. Thus, if a subsequent insert statement attempts to assign a NULL value, the default value specified is assigned instead.

   Perform the following steps:

   a. In SQL Developer, select **File** > **Open**.

   b. Navigate to /home/oracle/labs/SQLFiles and select lab_03_2a.sql. Click **Open**.

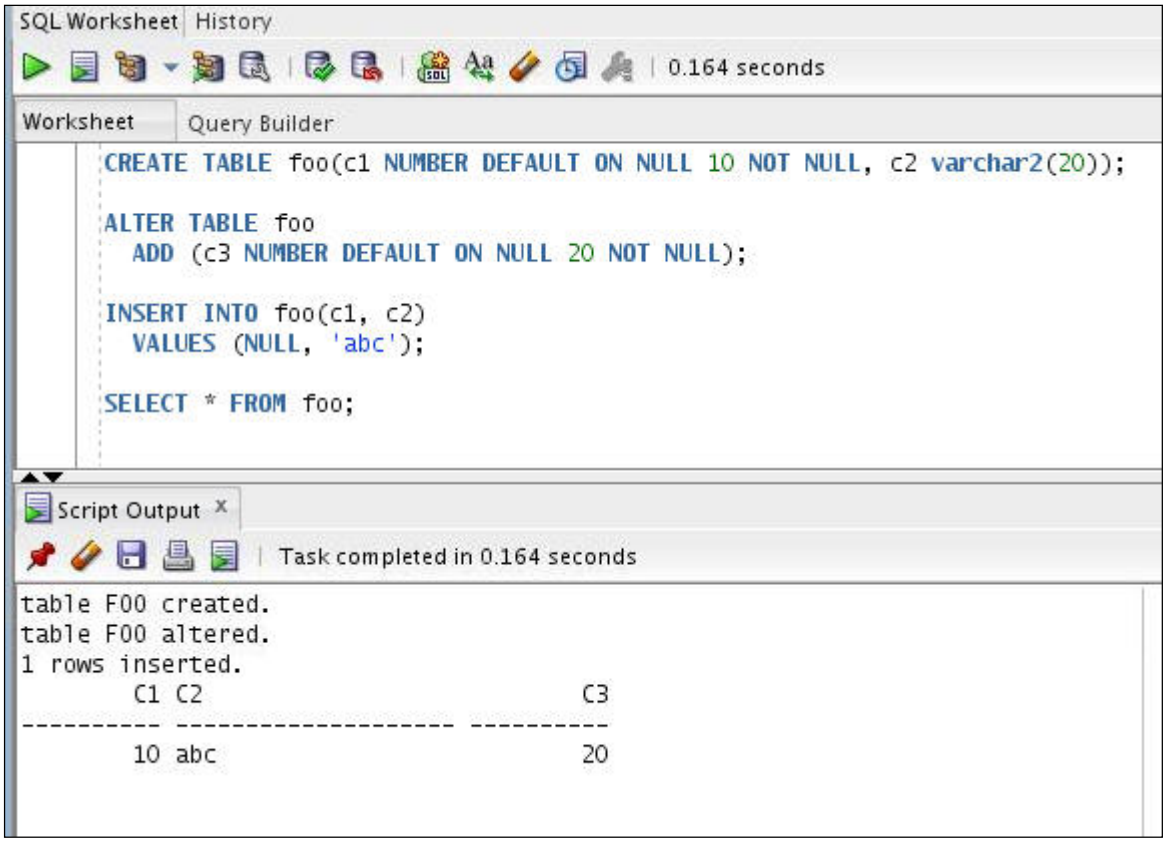
      Note: Optionally, you can copy the following SQL query and paste it in the HR_ORCL worksheet. Click the **Run Statement** icon to execute the SQL query.

```
CREATE TABLE foo(c1 NUMBER DEFAULT ON NULL 10 NOT NULL, c2
varchar2(20));

ALTER TABLE foo
  ADD (c3 NUMBER DEFAULT ON NULL 20 NOT NULL);

INSERT INTO foo(c1, c2)
  VALUES (NULL, 'abc');

SELECT * FROM foo;
```

   c. The script contains a SQL statement to create a table called foo. This table has a number column that has a NOT NULL condition and that defaults to a value specified if the value is NULL. The column defaulting on explicit null is implemented by adding the ON NULL keyword after the DEFAULT keyword in a CREATE or ALTER TABLE statement. Click the **Run Script** icon to run the SQL script.

   d. For Select Connection, make sure that the **HR_ORCL** connection is selected and click **OK**. If prompted, enter the username and password.

e. Notice that the value in column `c1` defaults to 10 when a null value is assigned. Similarly, the column `c3` has a default value of 20 assigned to it when a null value is inserted.



2. To use SQL column defaulting by using a sequence, create a sequence, which starts with the value `1`. Also, create a table that has a `numeric` column and default the column to `sequence.nextval` as a SQL column defaulting expression when a null value is inserted into the table.

   To perform the preceding task, in SQL Developer, open the `lab_03_2b.sql` file that is available in the `/home/oracle/labs/SQLFiles` folder. This file contains the SQL script to create a sequence `s1`, which starts with `1`. A table called `bar` with a numeric column is created. The `numeric` column has a `NOT NULL` condition, which defaults to the sequence `s1` by using the `sequence.nextval` expression when a null value is inserted into the table.

   Perform the following steps:

   a. In SQL Developer, select **File** > **Open**.

   b. Navigate to `/home/oracle/labs/SQLFiles` and select `lab_03_2b.sql`. Click **Open**.

```
CREATE SEQUENCE s1 START WITH 1;


CREATE TABLE bar (c1 number DEFAULT s1. NEXTVAL NOT NULL, c2
varchar2(20));


INSERT INTO bar (c2) VALUES ('Tim');
```

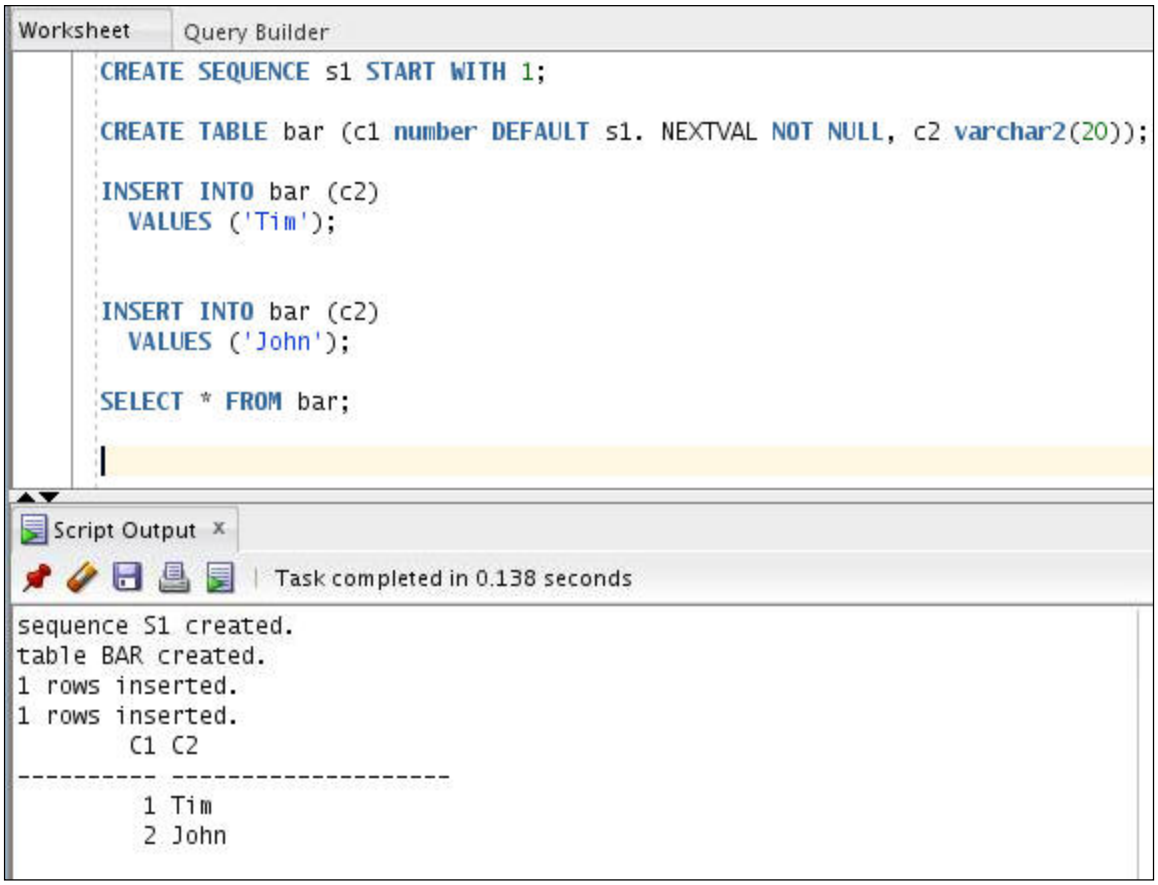
```
INSERT INTO bar (c2) VALUES ('John');

SELECT * FROM bar;
```

Note: Optionally, you can copy the following SQL query and paste it in the `HR_ORCL` worksheet. Click the **Run Statement** icon to execute the SQL query.

c. The script contains a SQL statement to create a sequence `s1`, which starts with `1`. It also contains a SQL statement to create a table called `bar` with a `numeric` column. The `numeric` column has a `NOT NULL` condition and defaults to `s1.nextval` when a null value is inserted into the table. Click **Run Script** icon to run the SQL script.

d. For Select Connection, make sure that the **`HR_ORCL`** connection is selected and click **OK**. If prompted, enter the username and password.

e. Notice that the next value of the sequence is assigned to the column `c1` when the value is null.



Practices for Lesson 4: PL/SQL Enhancements

Chapter 4

Practice 4-1: Using the `ACCESSIBLE BY` Clause in PL/SQL Database Objects

Overview

In this practice, you use the `ACCESSIBLE BY` clause in PL/SQL units to secure the PL/SQL objects.

Note: You are encouraged to first code the task given as follows and test your own answer before looking at the solution.

Tasks

1. Create a procedure by using the `ACCESSIBLE BY` clause and specify the PL/SQL units that can access the procedure. This feature allows you to add an extra layer of security to the PL/SQL objects.

Solution 4-1: Using the `ACCESSIBLE BY` Clause in PL/SQL Database Objects

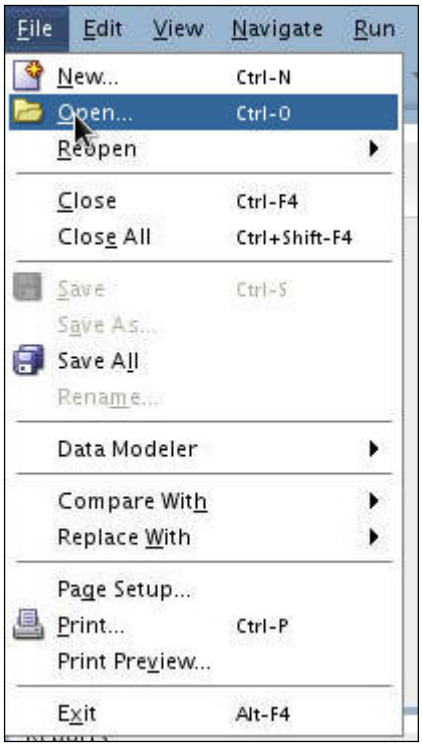
Tasks

1. Create a procedure by using the `ACCESSIBLE BY` clause and specify the PL/SQL units that can access the procedure. This feature allows you to add an extra layer of security to the PL/SQL objects.

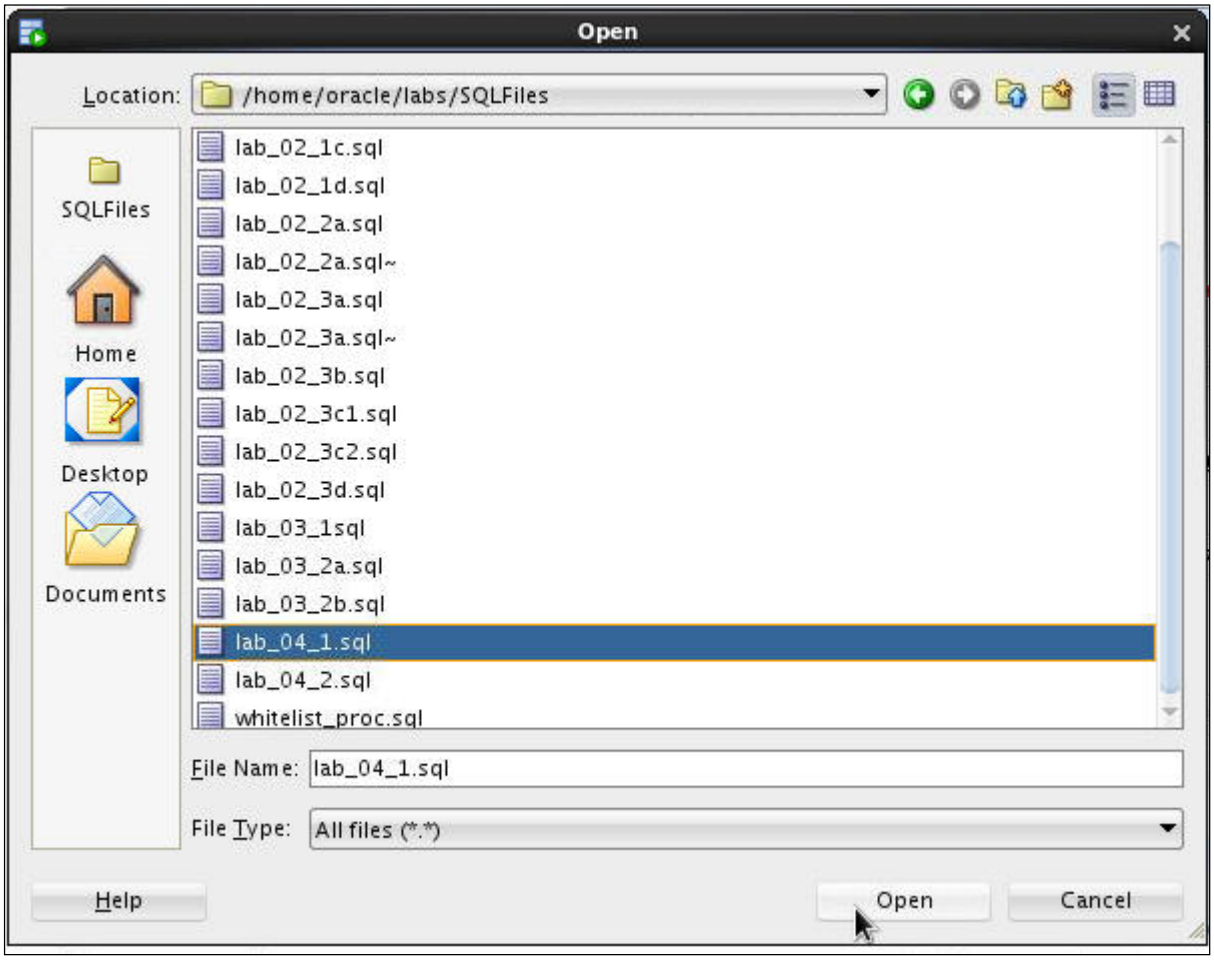
   To perform the preceding task, in SQL Developer, open the `lab_04_1.sql` file that is available in the `/home/oracle/labs/SQLFiles` folder. This file contains the SQL script to create a `test_white_list` procedure. This procedure uses the `ACCESSIBLE BY` clause, thereby allowing only the PL/SQL units, `coll_pkg` package and the `p_white_list_test` procedure, to invoke the `test_white_list` procedure.

   Perform the following steps:

   a. In SQL Developer, select **File** > **Open**.



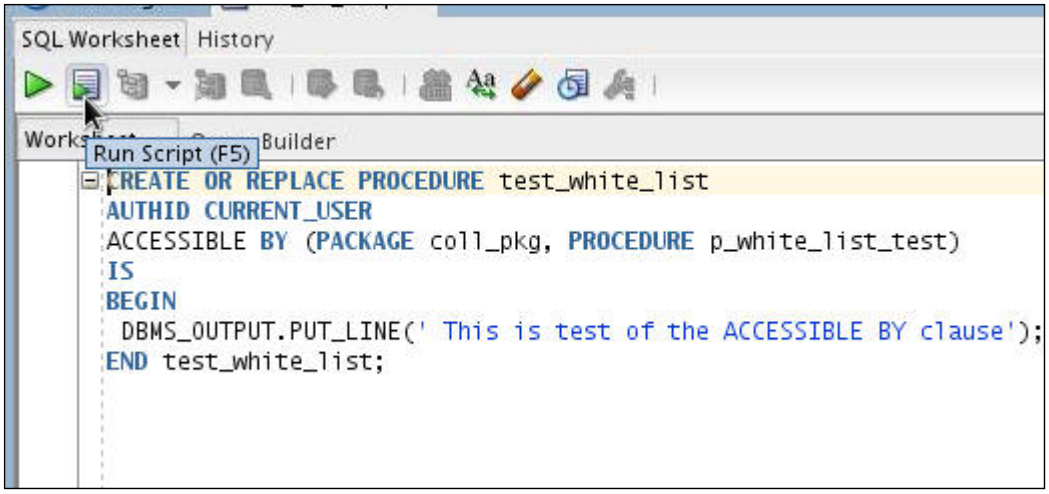
b. Navigate to `/home/oracle/labs/SQLFiles` and select `lab_04_1.sql`. Click **Open**.



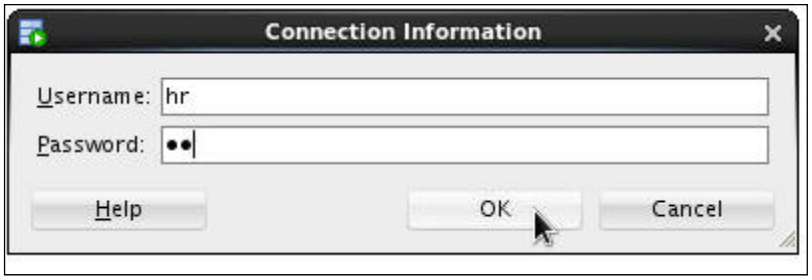
Note: Optionally, you can copy the following PL/SQL block and paste it in the `HR_ORCL` SQL Worksheet. Click the **Run Statement** icon to execute the PL/SQL block.

```
CREATE OR REPLACE PROCEDURE test_white_list
AUTHID CURRENT_USER
ACCESSIBLE BY (PACKAGE coll_pkg, PROCEDURE p_white_list_test)
IS
BEGIN
 DBMS_OUTPUT.PUT_LINE(' This is test of the ACCESSIBLE BY
clause');
END test_white_list;
```

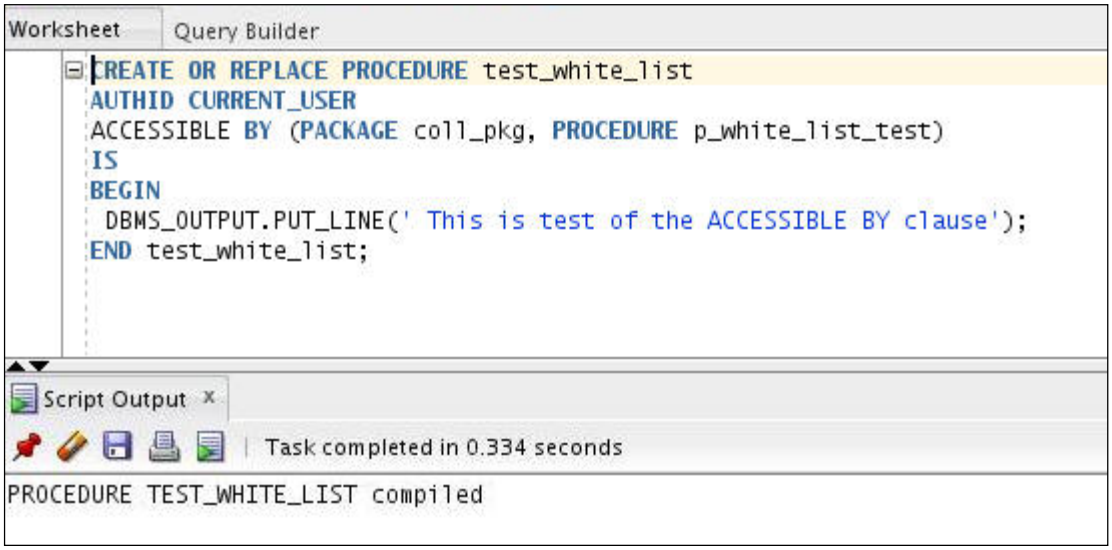
c. The SQL file contains the script to create a procedure that uses the `ACCESSIBLE BY` clause in the syntax. The procedure `test_white_list` specifies the PL/SQL units that can invoke it by using the `ACCESSIBLE BY` clause. In this case, only a package called `coll_pkg` and a procedure `p_white_list_test` can invoke the procedure `test_white_list`. Click the **Run Script** icon to run the SQL script.



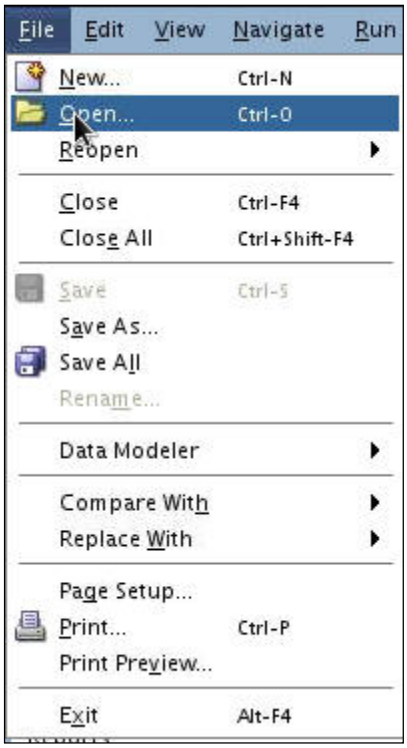
d. For Select Connection, make sure that the **`HR_ORCL`** connection is selected and click **OK**. If prompted, enter the username and password.



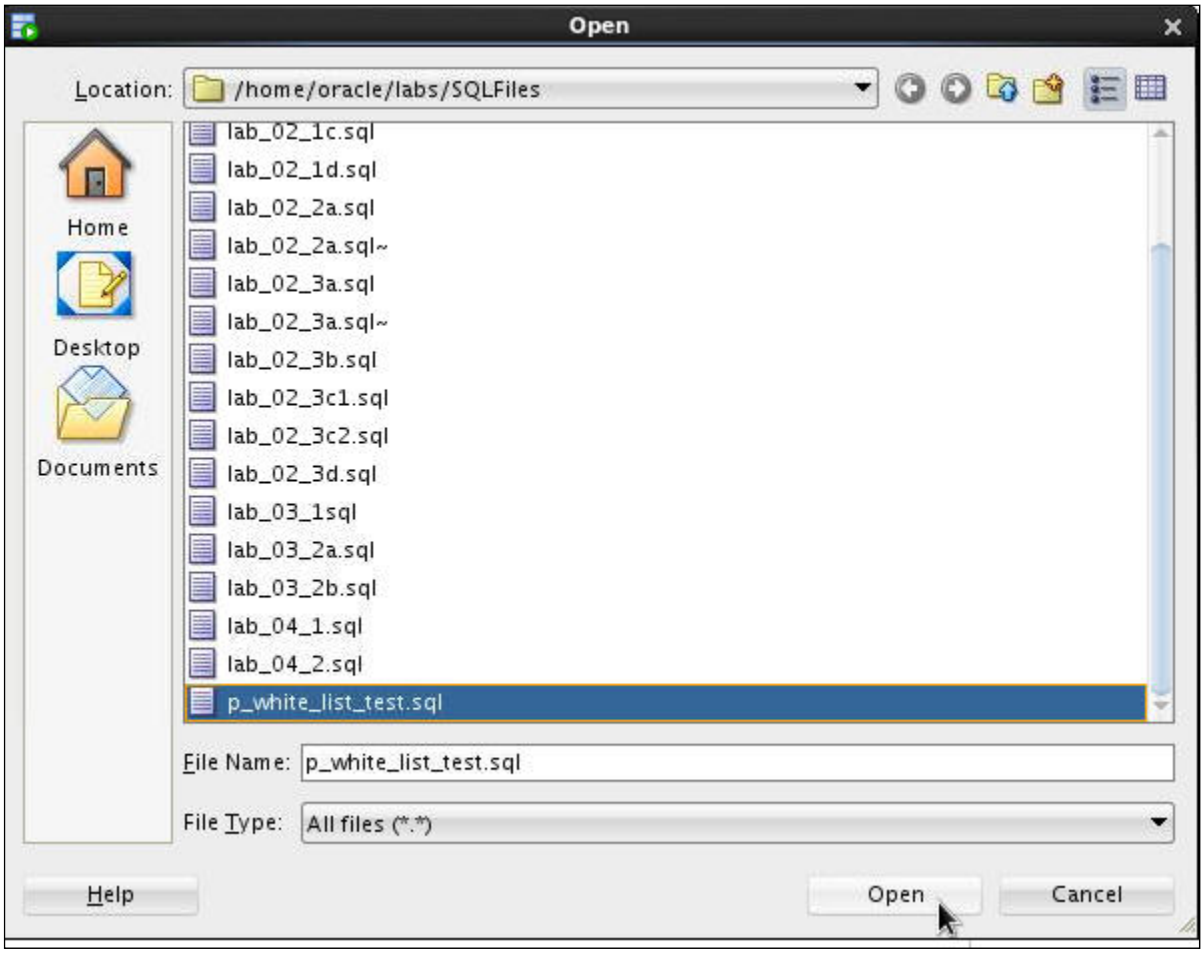
e. The script is compiled successfully.



f. Next, open the file `p_white_list_test.sql` to create the procedure `p_white_list_test`. Select **File** > **Open**.



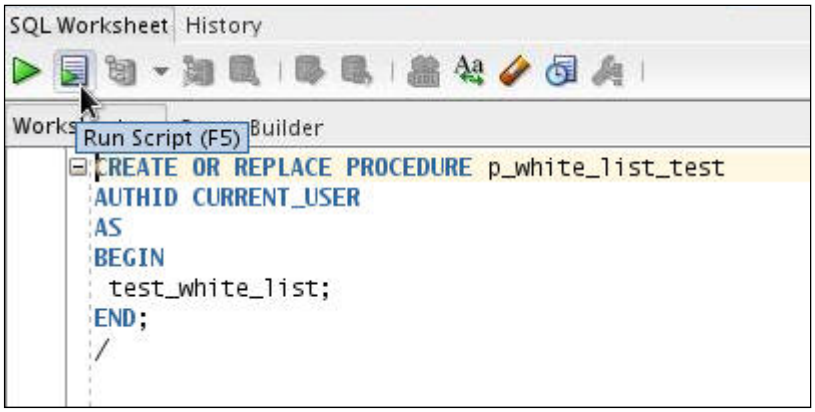
g. Select the `p_white_list_test.sql` file. Click **Open**.



Note: Optionally, you can enter the following lines of code in the SQL Worksheet and click the **Run Script** icon.

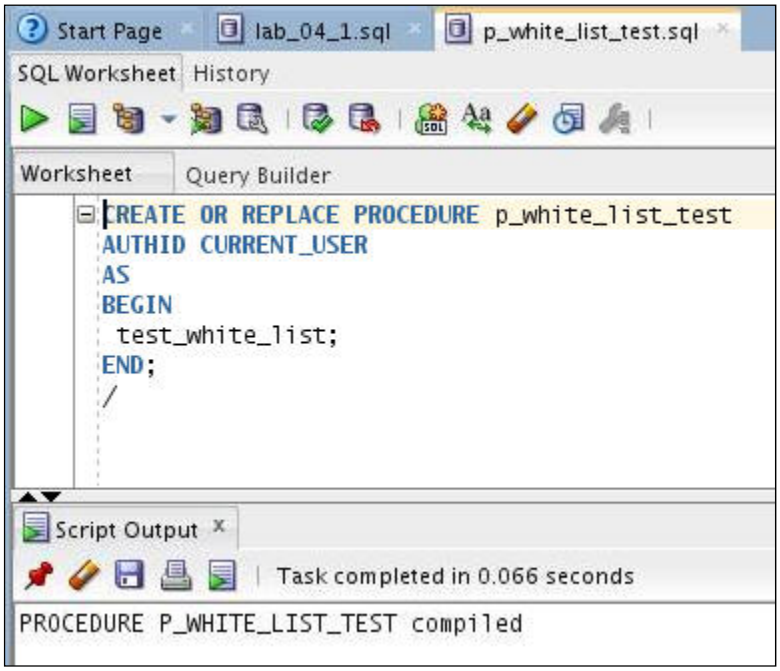
```
CREATE OR REPLACE PROCEDURE p_white_list_test
AUTHID CURRENT_USER
AS
BEGIN
 test_white_list;
END;
/
```

h. The script to create the procedure `p_white_list_test` is displayed. Notice that the procedure `p_white_list_test` invokes the `test_white_list` procedure. Click the **Run Script** icon.

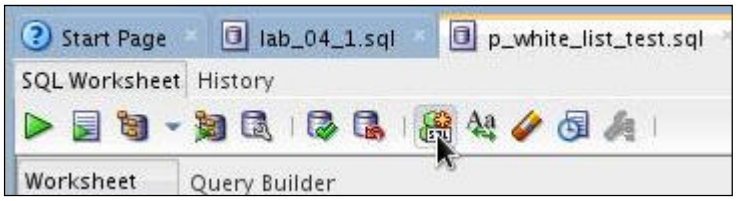


i. For Select Connection, make sure that the **`HR_ORCL`** connection is selected and click **OK**.

j. The procedure is compiled successfully.

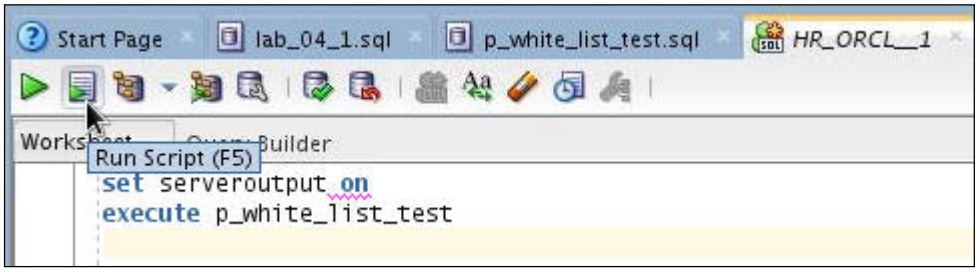


k. To verify that the `test_white_list` procedure can be invoked, execute the `p_white_list_test` procedure. Click the **Unshared SQL Worksheet** icon. A new SQL Worksheet opens.

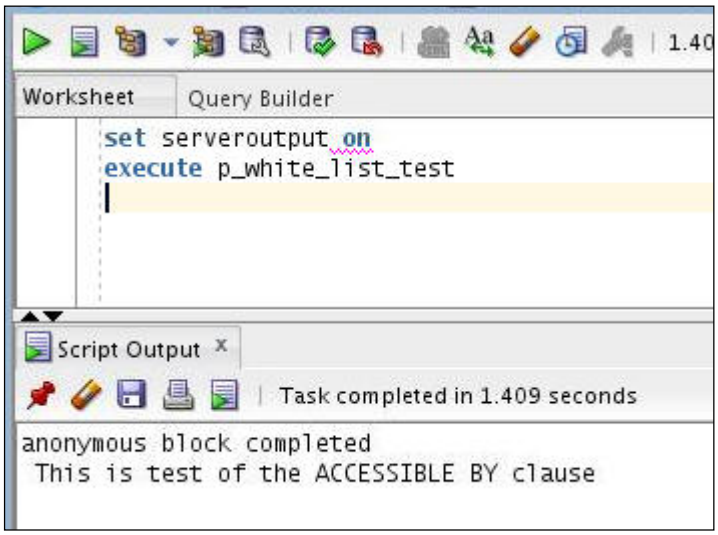


l. Enter the following lines of code in the SQL Worksheet. Click **Run Script**.

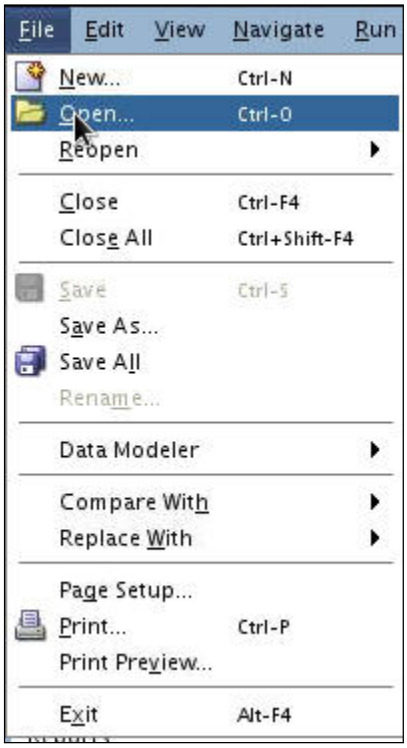
```
set serveroutput on
execute p_white_list_test
```



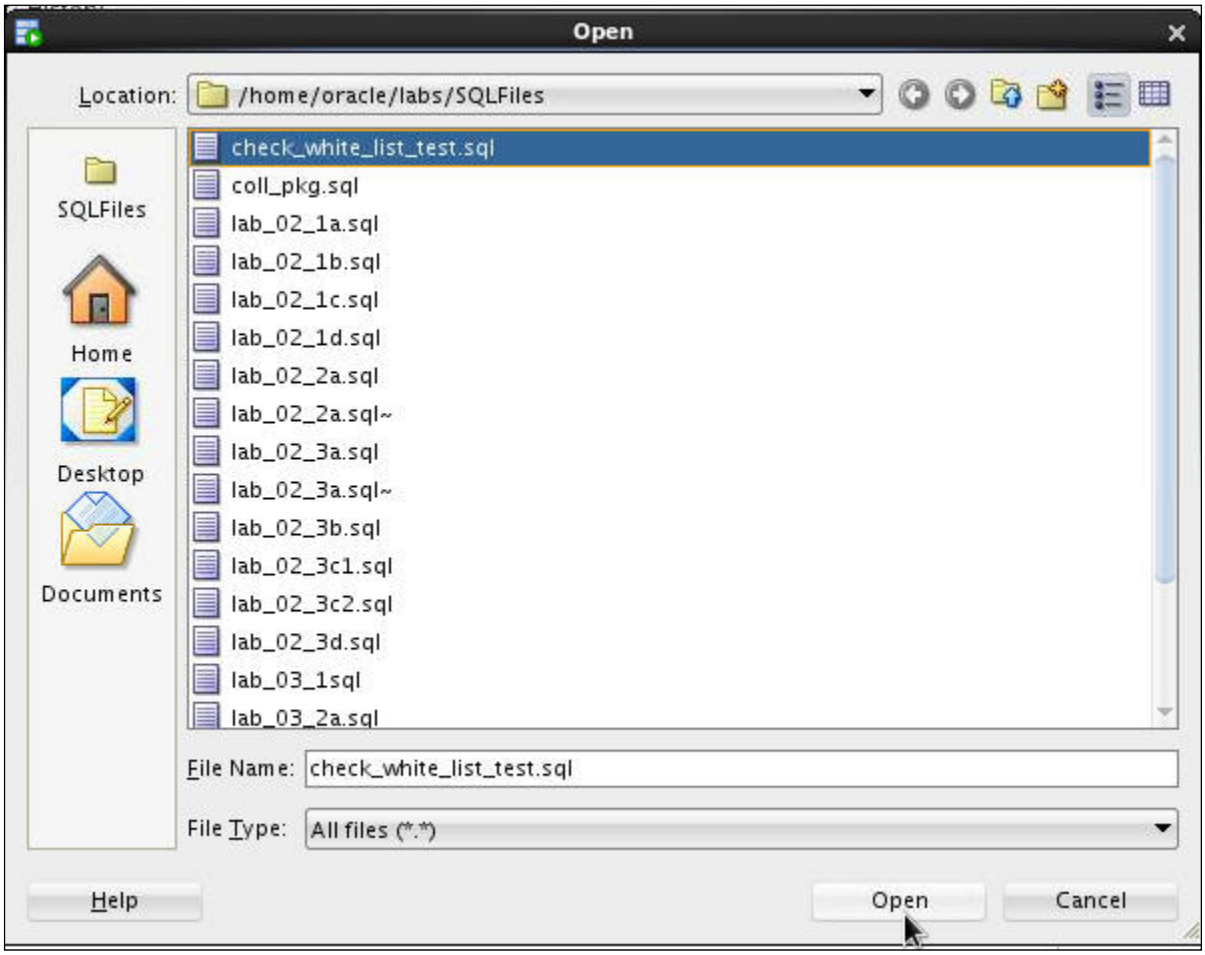
m. Notice that `test_white_list` has been successfully invoked.



n. To check whether the `test_white_list` procedure can be invoked by another procedure that is not part of the white list, select **File** > **Open**.



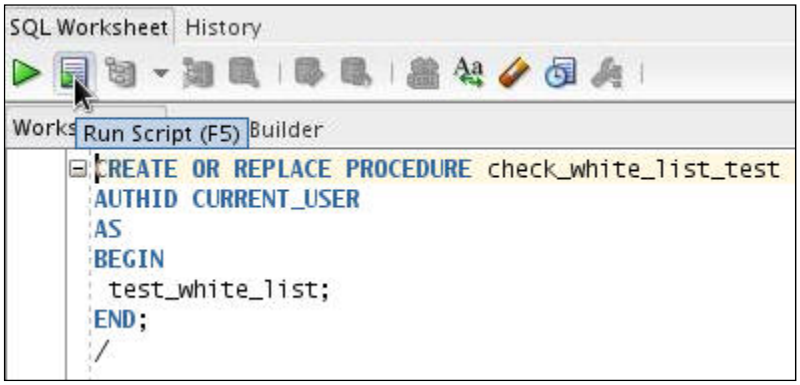
o. Select `check_white_list_test.sql` and click **Open**.



Note: Optionally, you can enter the following lines of code in the SQL Worksheet and click the **Run Script** icon.

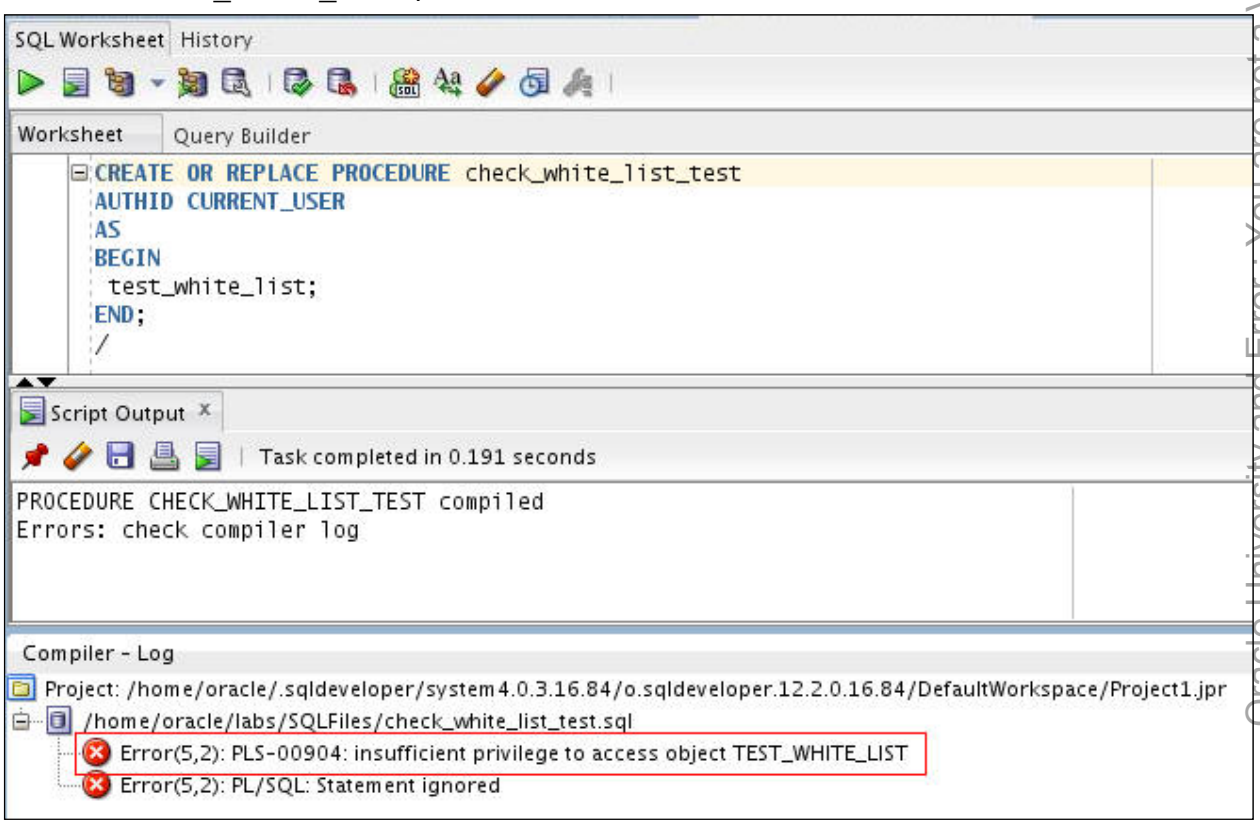
```
CREATE OR REPLACE PROCEDURE check_white_list_test
AUTHID CURRENT_USER
AS
BEGIN
 test_white_list;
END;
/
```

p. Notice that this script contains the SQL that invokes the `test_white_list` procedure. Click **Run Script**.



q. For Select Connection, make sure that the **HR_ORCL** connection is selected and click **OK**. If prompted, enter the username and password and click OK.

r. The script has the procedure compiled with some errors. The Compiler log shows that the procedure was compiled with errors and the error is due to insufficient privileges to invoke the `test_white_list` procedure.



Practice 4-2: Invoker's Right Function That Can Be Result-Cached

Overview

In this practice, you create an Invoker's Rights function that can be result-cached.

Note: You are encouraged to first code the task given as follows and test your own answer before looking at the solution.

Tasks

1. Create a function called `get_hire_date` that can be result cached and has the `AUTHID` property set to invoker's right. The function should take employee ID as the input parameter and return the hire date of the employee ID as the result. To view the result, the function is invoked by passing a value for employee ID.

Solution 4-2: Invoker's Right Function That Can Be Result-Cached

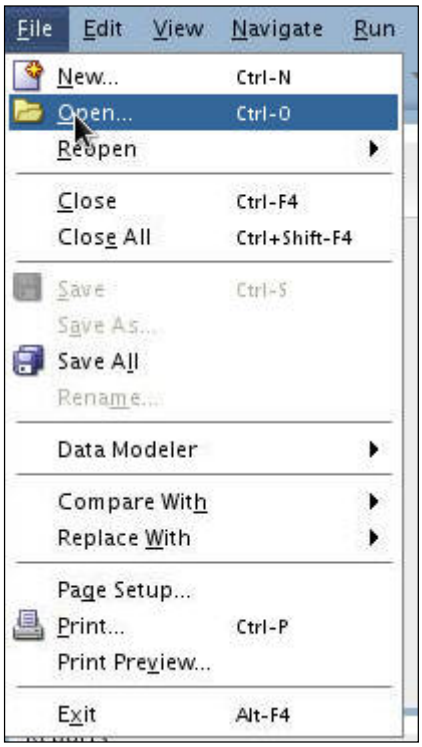
Tasks

1. Create a function called `get_hire_date` that can be result cached and has the `AUTHID` property set to invoker's right. The function should take employee ID as the input parameter and return the hire date of the employee ID as the result. To view the result, the function is invoked by passing a value for employee ID.

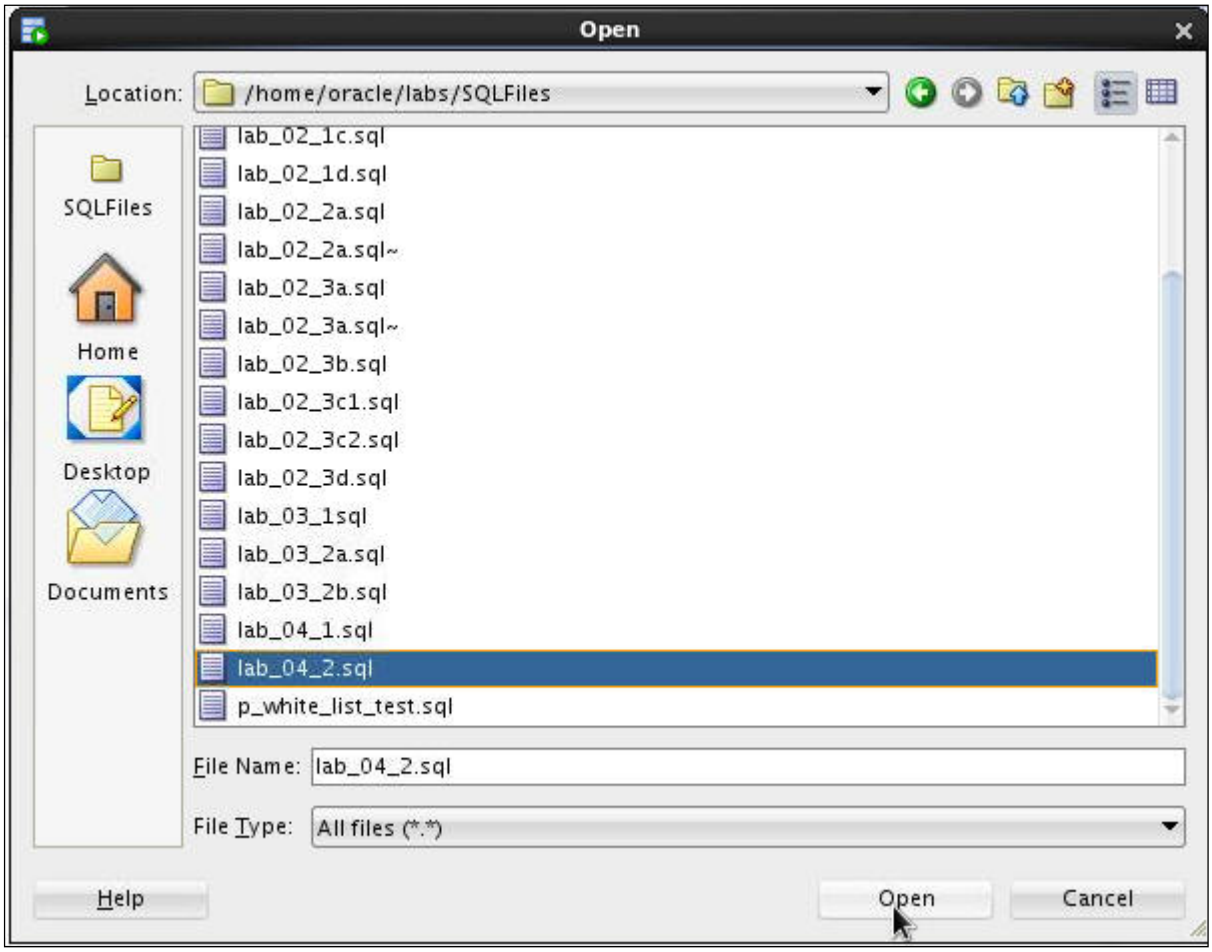
   To perform the preceding task, in SQL Developer, open the `lab_04_2.sql` file that is available in the `/home/oracle/labs/SQLFiles` folder. This file contains the SQL script to create a `get_hire_date` function that can be result cached and has the `AUTHID` property set to invoker's right. The `get_hire_date` function takes `emp_id` as the input parameter and returns `hire_date`.

   Perform the following steps:

   a. In SQL Developer, select **File** > **Open**.



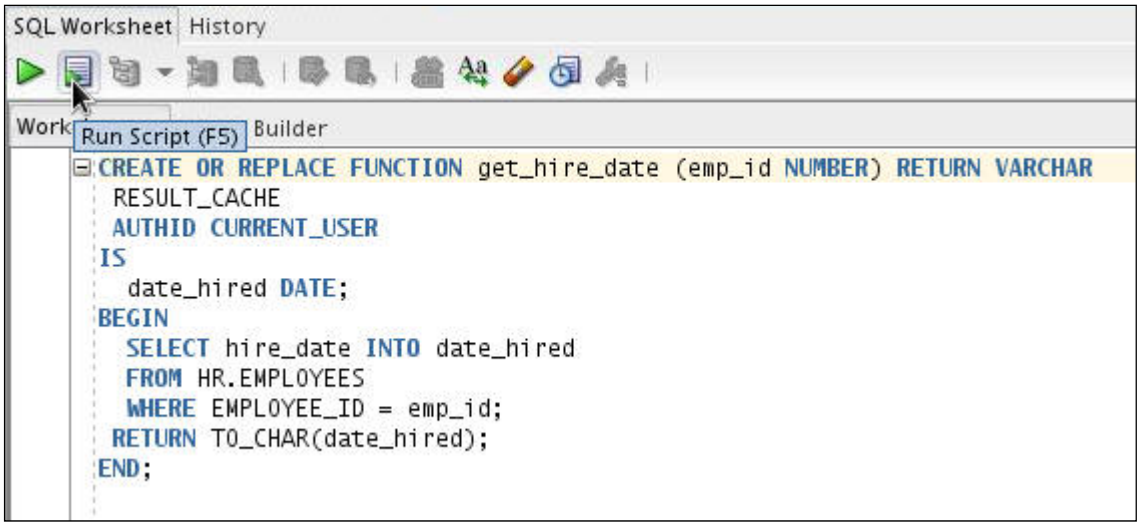
b. Select `lab_04_2.sql` and click **Open**.



Note: Optionally, you can enter the following lines of code in the SQL Worksheet and click the **Run Script** icon.

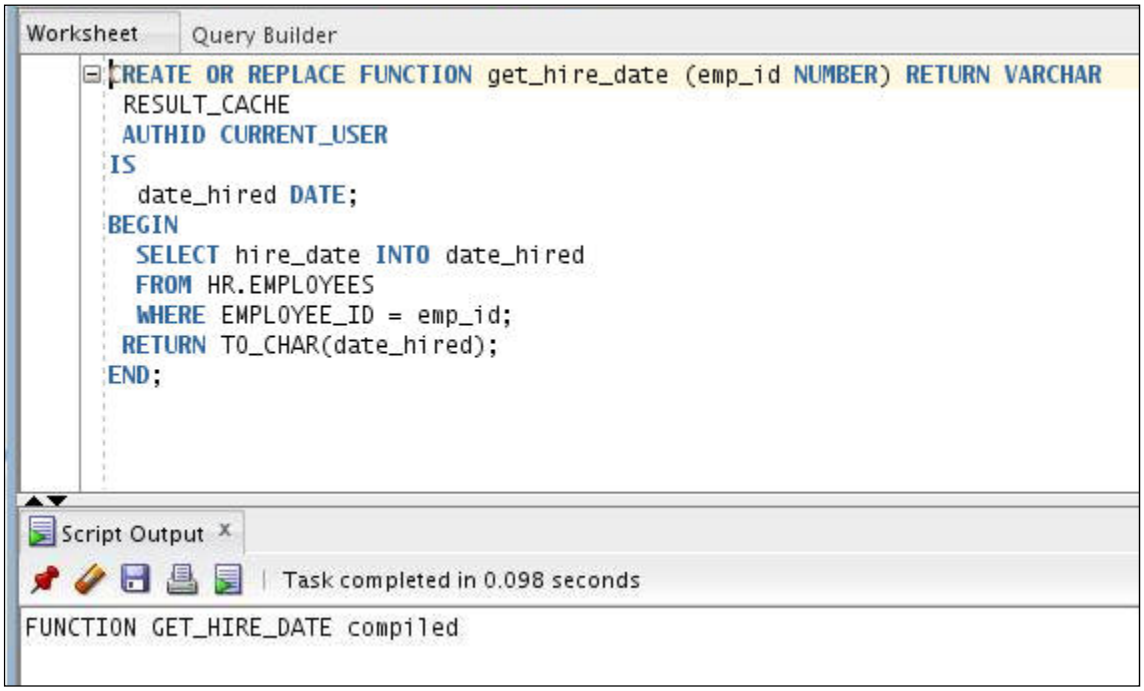
```
CREATE OR REPLACE FUNCTION get_hire_date (emp_id NUMBER) RETURN
VARCHAR
 RESULT_CACHE
 AUTHID CURRENT_USER
IS
  date_hired DATE;
BEGIN
  SELECT hire_date INTO date_hired
  FROM HR.EMPLOYEES
  WHERE EMPLOYEE_ID = emp_id;
 RETURN TO_CHAR(date_hired);
END;
```

c. Notice that this script contains the SQL to create a function called `get_hire_date` that can be result cached and the `AUTHID` property set to `CURRENT_USER`. Click **Run Script.**



d. For Select Connection, make sure that the **HR_ORCL** connection is selected and click **OK**.

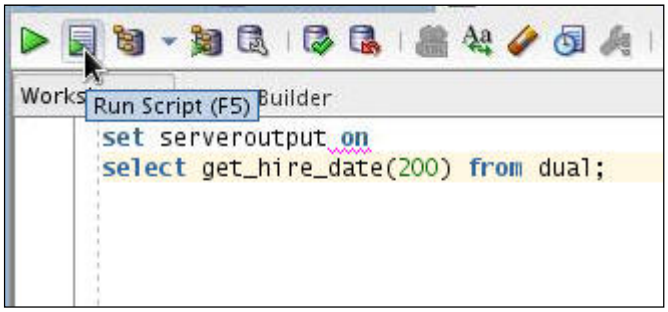
The function gets compiled successfully.



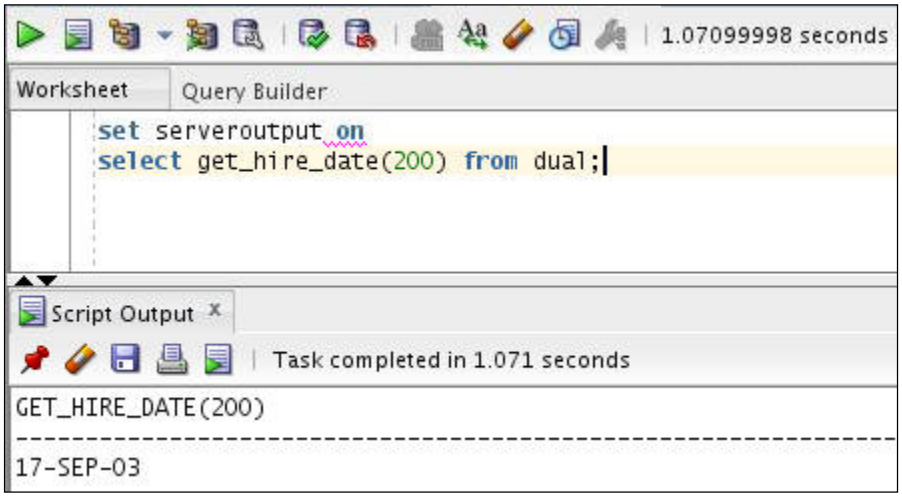
e. Next, open a new SQL Worksheet by clicking the **Unshared SQL Worksheet** icon.

f. Enter the following lines of code to invoke the function. Click the **Run Script** icon.

```
set serveroutput on
select get_hire_date(200) from dual;
```



g. Notice that the hire date of employee ID 200 is displayed.



Practices for Lesson 5: Data Warehousing Enhancements

Chapter 5

Practice 5-1: Multi Partition Maintenance Operations

Overview

In this practice, you create range partitioned tables and add multiple partitions to it. You also merge multiple range partitions into one partition. You understand how a partition can be split into multiple partitions in a table.

Note: You are encouraged to first code the tasks given as follows and test your own answers before looking at the solution.

Tasks

1. Open a terminal and log in to SQL*Plus. Create four tablespaces called `tsa`, `tsb`, `tsc`, and `tsd`. After creating the tablespaces, log in to SQL Developer and create a range partitioned table called `sales` with four partitions, one for each quarter of the year 2006, by using the `time_id` column as the partitioning column and by using the `VALUES LESS THAN` clause that determines the partition bound. Each partition should be contained in a separate tablespace. The `sales` table should then be altered to add multiple partitions for each quarter of the year 2007.
2. Merge four partitions of the range partitioned `sales` table into one partition.
3. Create a table by using list partition and split the partition into multiple partitions.

Solution 5-1: Multi Partition Maintenance Operations

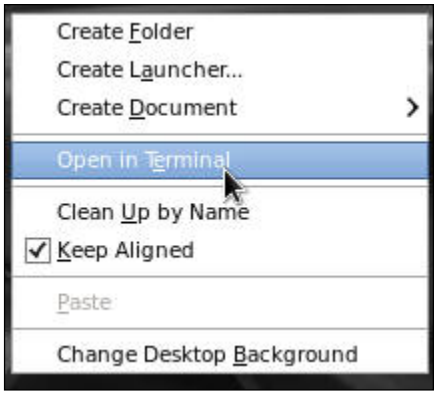
Tasks

1. Open a terminal and log in to SQL*Plus. Create four tablespaces called `tsa`, `tsb`, `tsc`, and `tsd`. After creating the tablespaces, log in to SQL Developer and create a range partitioned table called `sales` with four partitions, one for each quarter of the year 2006, by using the `time_id` column as the partitioning column and by using the `VALUES LESS THAN` clause that determines the partition bound. Each partition should be contained in a separate tablespace. The `sales` table should then be altered to add multiple partitions for each quarter of the year 2007.

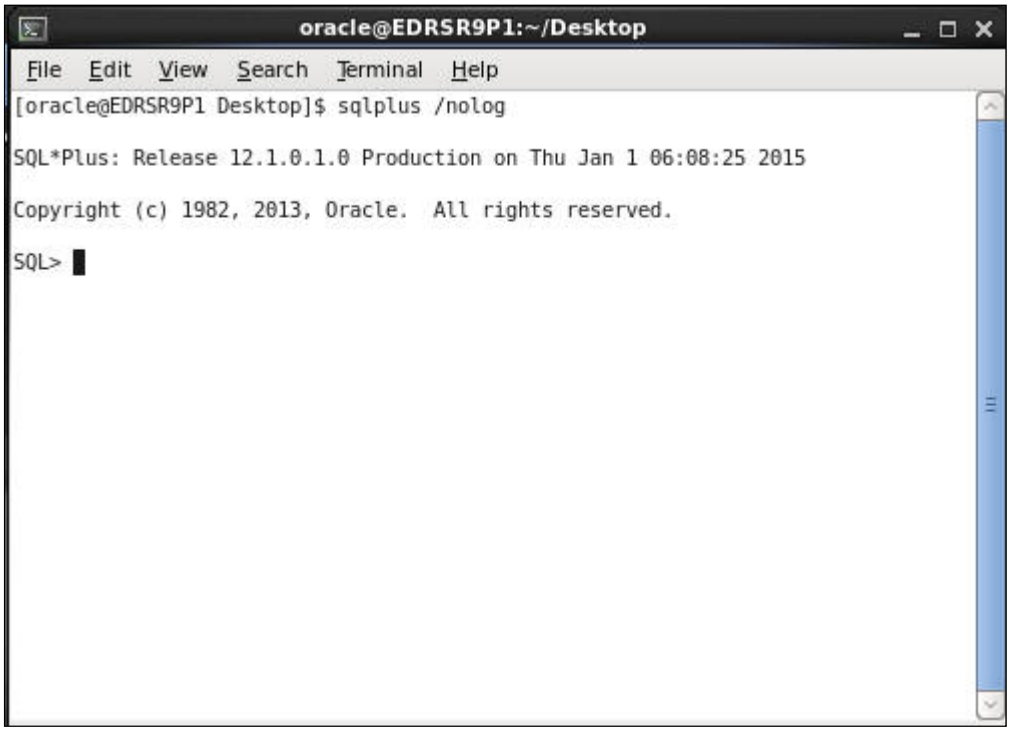
   To create tablespaces, open a terminal and log in to SQL*Plus. Navigate to the `SQLFiles` folder and open the `create_tablespace.sql` file (available in the `/home/oracle/labs/SQLFiles` folder). This file contains the SQL script to create the tablespaces.

   Perform the following steps:

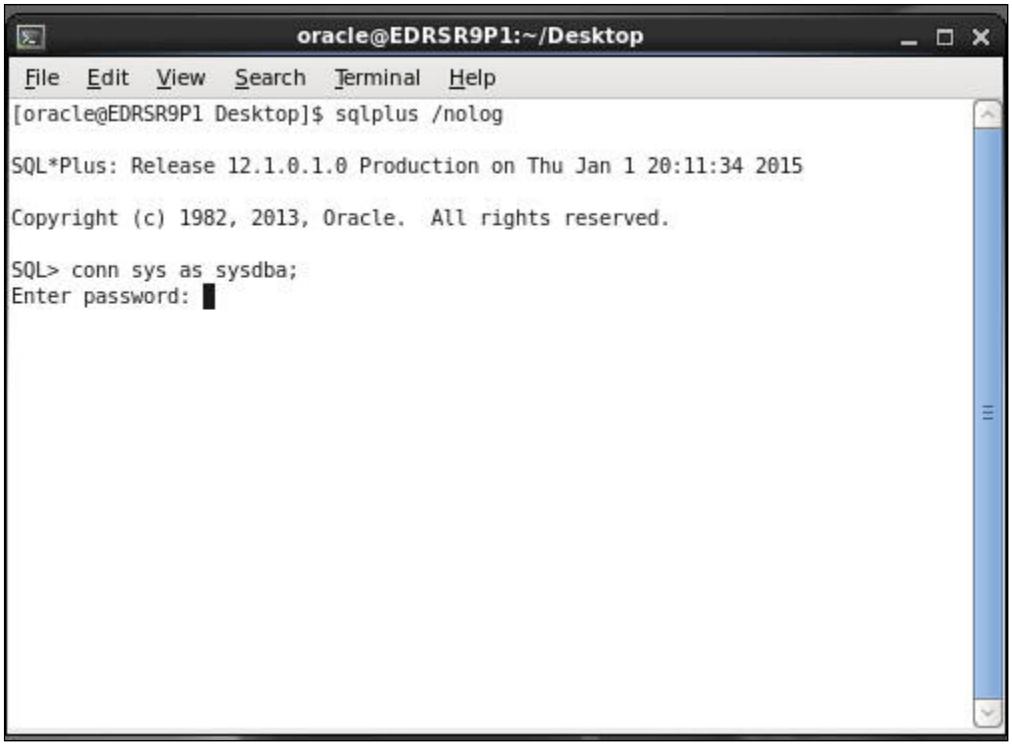
   a. Right-click anywhere in the desktop and select **Open in Terminal**.



b. Enter `sqlplus /nolog` and press the Enter key to log in to SQL*Plus.



c. At the SQL prompt, enter `conn sys as sysdba;` and press the Enter key.



d. Press the Enter key again for `Enter password`:

```
oracle@EDRSR9P1:~/Desktop
File  Edit  View  Search  Terminal  Help
[oracle@EDRSR9P1 Desktop]$ sqlplus /nolog

SQL*Plus: Release 12.1.0.1.0 Production on Thu Jan 1 20:11:34 2015

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

SQL> conn sys as sysdba;
Enter password:
Connected.
SQL>
```

e. Run the `create_tablespace.sql` file to create the tablespaces. Enter the following SQL statement to locate and run the `create_tablespace.sql` file, and press the Enter key.

`@/home/oracle/labs/SQLFiles/create_tablespace.sql`

```
oracle@EDRSR9P1:~/Desktop
File  Edit  View  Search  Terminal  Help
[oracle@EDRSR9P1 Desktop]$ sqlplus /nolog

SQL*Plus: Release 12.1.0.1.0 Production on Thu Jan 1 20:11:34 2015

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

SQL> conn sys as sysdba;
Enter password:
Connected.
SQL> @/home/oracle/labs/SQLFiles/create_tablespace.sql
```

f. In this case, four tablespaces `tsa`, `tsb`, `tsc`, and `tsd` are created.

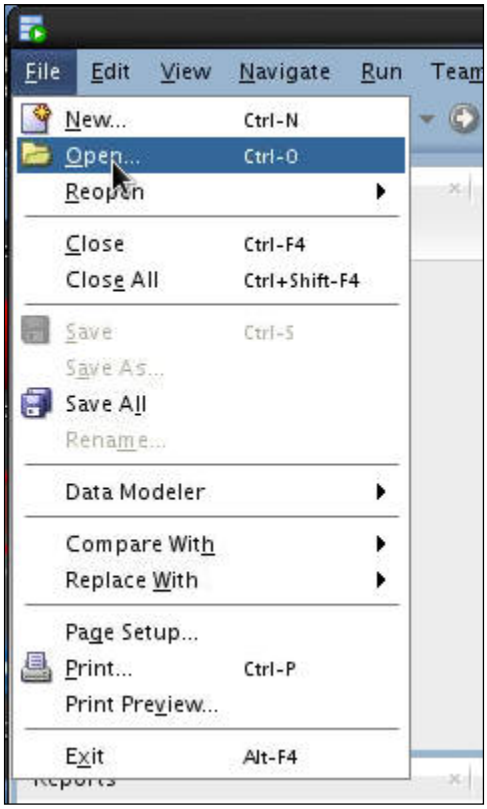
```
oracle@EDRSR9P1:~/Desktop
File  Edit  View  Search  Terminal  Help
[oracle@EDRSR9P1 Desktop]$ sqlplus /nolog

SQL*Plus: Release 12.1.0.1.0 Production on Thu Jan 1 21:50:49 2015

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

SQL> conn sys as sysdba;
Enter password:
Connected.
SQL> @/home/oracle/labs/SQLFiles/create_tablespace.sql

Tablespace created.


Tablespace created.


Tablespace created.


Tablespace created.

SQL>
```

g. Enter `exit` to log out of the terminal.

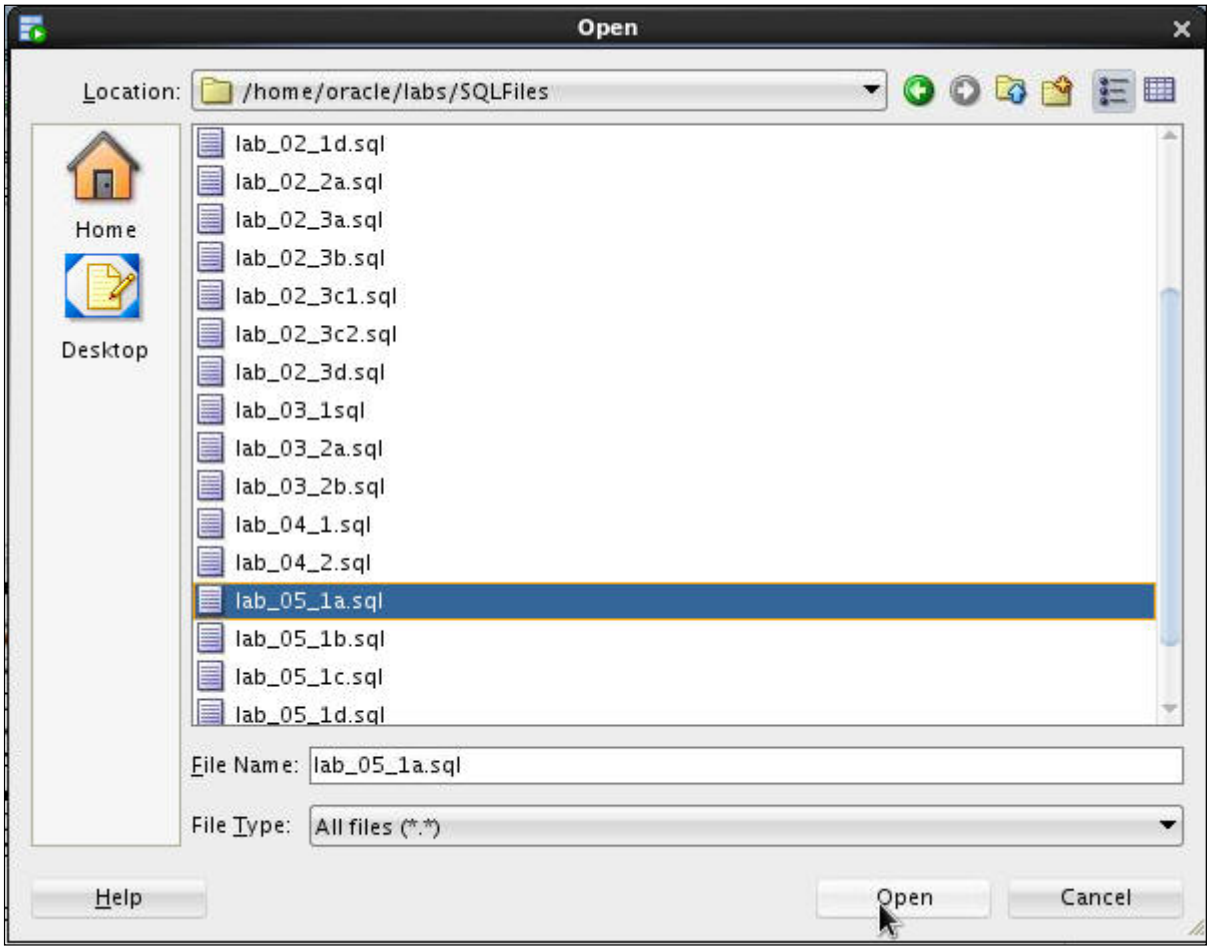
Next, you want to create a range partitioned table called `sales` with four partitions, one for each quarter of the year 2006, by using the `time_id` column as the partitioning column and by using the `VALUES LESS THAN` clause that determines the partition bound. Each partition should be contained in a separate tablespace. The `sales` table should then be altered to add multiple partitions for each quarter of the year 2007.

In SQL Developer, open the `lab_05_1a.sql` file that is available in the `/home/oracle/labs/SQLFiles` folder. This file contains the SQL script to create a range partitioned table called `sales` with four partitions, one for each quarter of the year 2006, by using the `time_id` column as the partitioning column. The `sales` table is then altered to add multiple partitions.

h. Open SQL Developer. Select **File** > **Open**.



i. Navigate to `home/oracle/labs/SQLFiles/lab_05_1a.sql` and click **Open**.

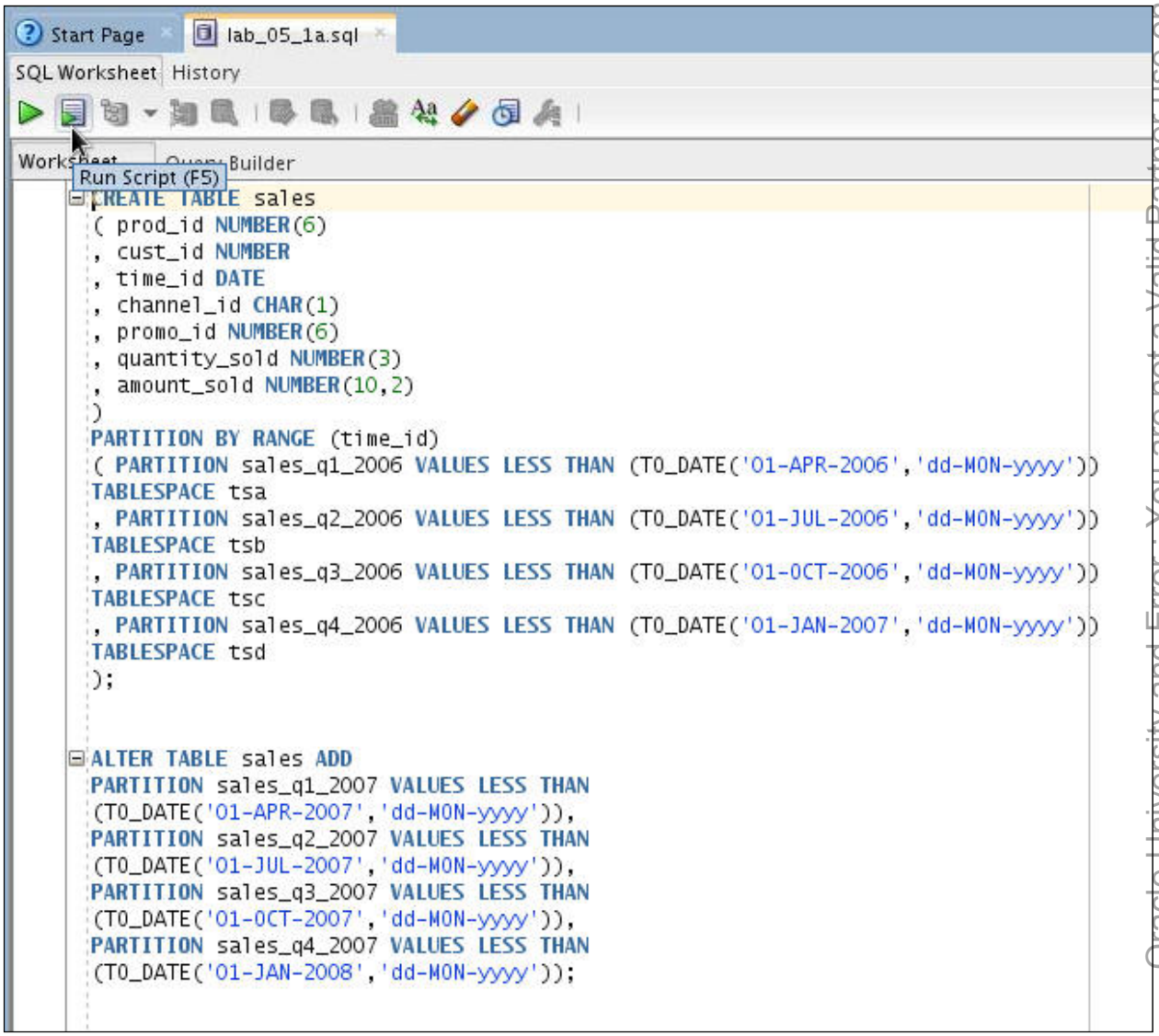


Note: Optionally, you can copy the following lines of code and paste them in the `HR_ORCL` worksheet. Click the **Run Statement** icon to execute the SQL statements.

```
CREATE TABLE sales
( prod_id NUMBER(6)
, cust_id NUMBER
, time_id DATE
, channel_id CHAR(1)
, promo_id NUMBER(6)
, quantity_sold NUMBER(3)
, amount_sold NUMBER(10,2)
)
PARTITION BY RANGE (time_id)
( PARTITION sales_q1_2006 VALUES LESS THAN (TO_DATE('01-APR-
2006','dd-MON-yyyy'))
TABLESPACE tsa
, PARTITION sales_q2_2006 VALUES LESS THAN (TO_DATE('01-JUL-
2006','dd-MON-yyyy'))
TABLESPACE tsb
```

```
, PARTITION sales_q3_2006 VALUES LESS THAN (TO_DATE('01-OCT-
2006','dd-MON-yyyy'))
TABLESPACE tsc
, PARTITION sales_q4_2006 VALUES LESS THAN (TO_DATE('01-JAN-
2007','dd-MON-yyyy'))
TABLESPACE tsd
);


ALTER TABLE sales ADD
PARTITION sales_q1_2007 VALUES LESS THAN
(TO_DATE('01-APR-2007','dd-MON-yyyy')),
PARTITION sales_q2_2007 VALUES LESS THAN
(TO_DATE('01-JUL-2007','dd-MON-yyyy')),
PARTITION sales_q3_2007 VALUES LESS THAN
(TO_DATE('01-OCT-2007','dd-MON-yyyy')),
PARTITION sales_q4_2007 VALUES LESS THAN
(TO_DATE('01-JAN-2008','dd-MON-yyyy'));
```

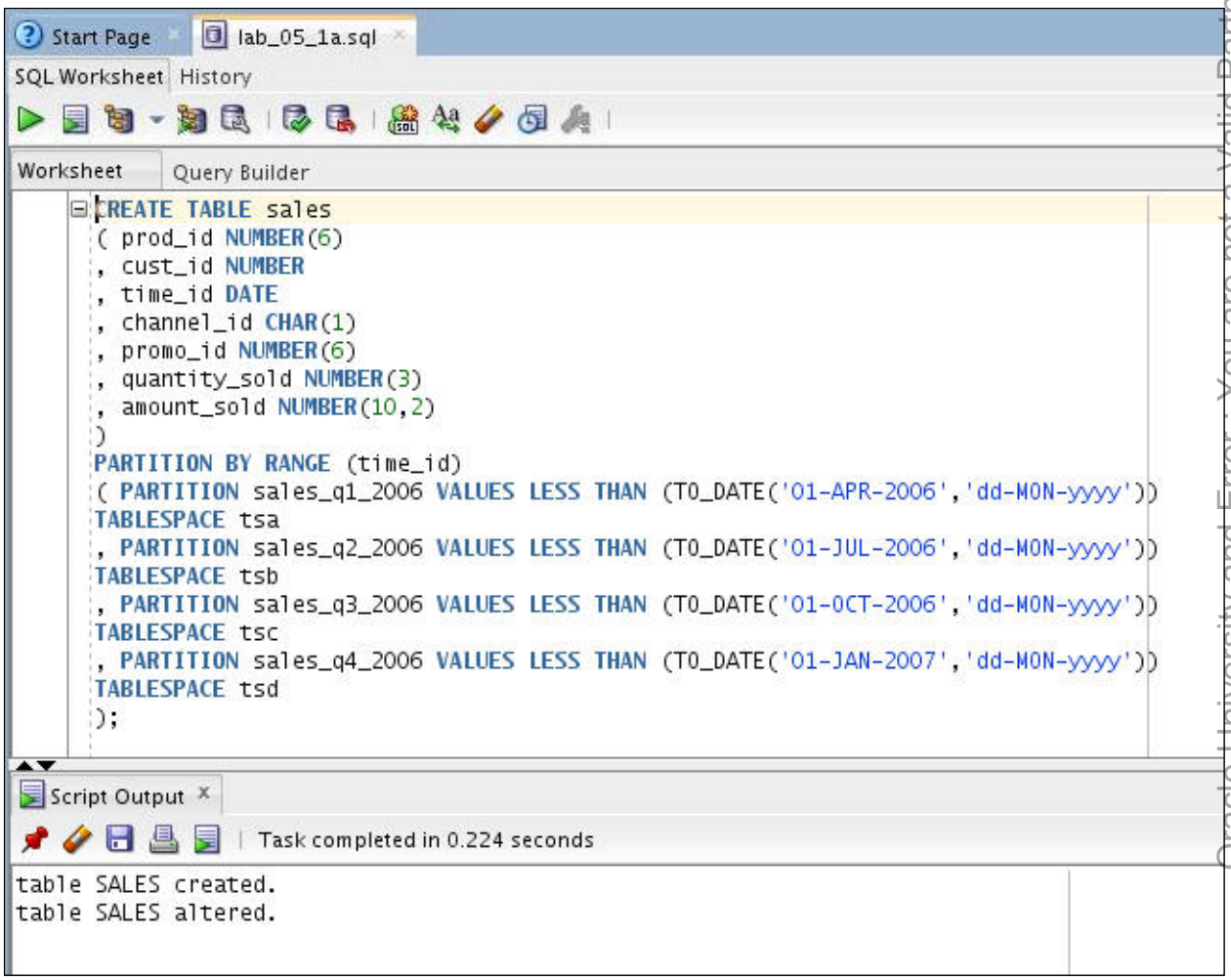
j. The file contains the SQL script to create a sales table with four partitions, one for each quarter of 2006. Each partition is given a name `sales_q1_2006`, `sales_q2_2006`, `sales_q3_2006`, and `sales_q4_2006`. The `time_id` column is the partitioning column, whereas its value represents the partitioning key of a specific row. The `VALUES LESS THAN` clause determines the partition bound: rows with partitioning key values that compare less than the ordered list of values specified by the clause are stored in the partition. Each partition is contained in a separate tablespace: `tsa`, `tsb`, `tsc`, and `tsd`. The file contains the script to add multiple partitions by using a single statement by specifying the individual partitions. Click the **Run Script** icon.



k. For Select Connection, select `HR_ORCL` and click **OK**.

l. If prompted for Connection Information, enter the username and password and click OK.

The table has been created and partitions have been added.

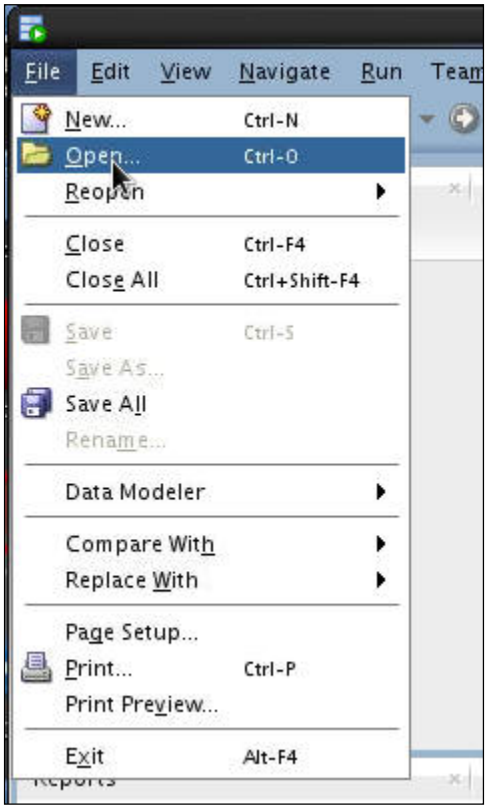


2. Merge four partitions of the range partitioned `sales` table into one partition.

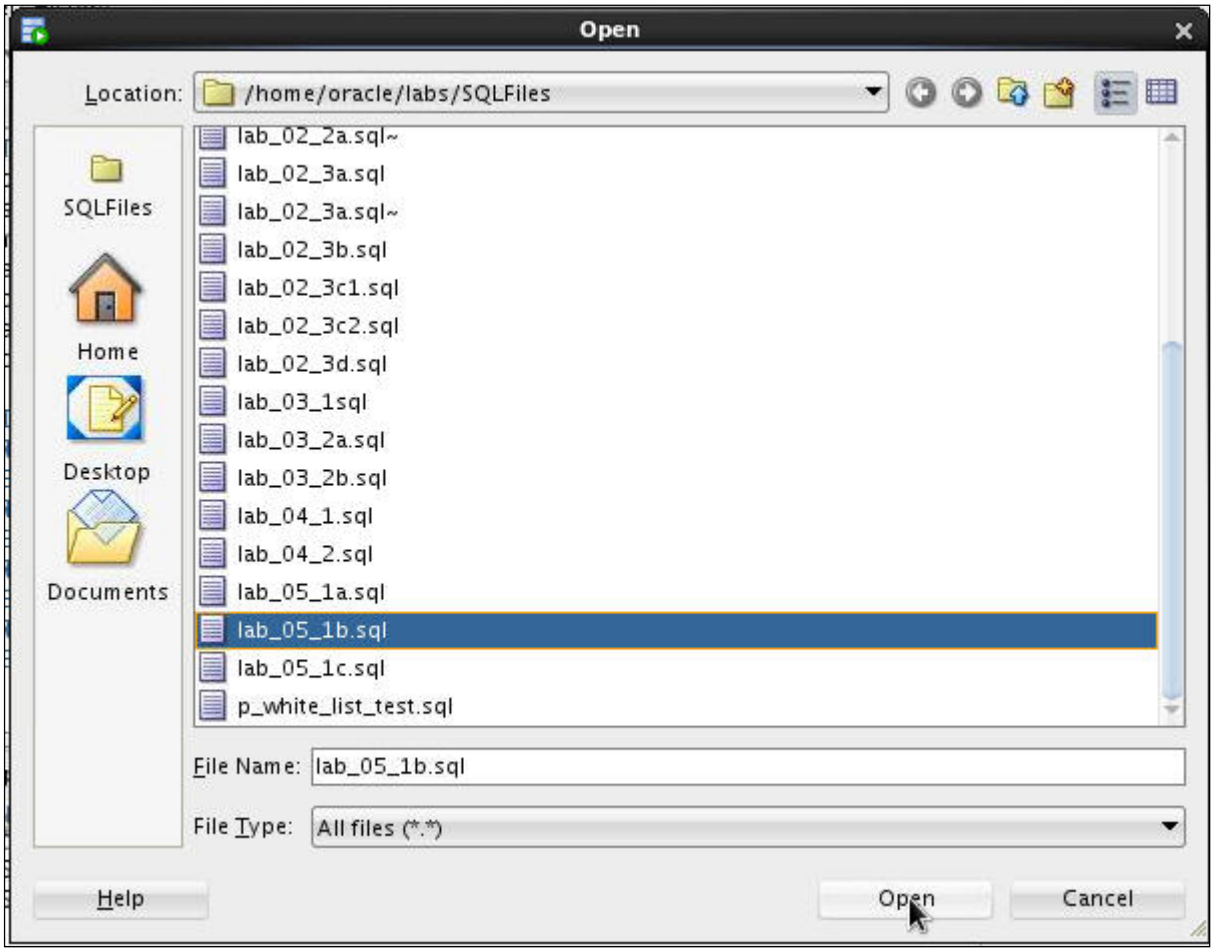
   To perform the preceding task, in SQL Developer, open the `lab_05_1b.sql` file that is available in the `/home/oracle/labs/SQLFiles` folder. This file contains the SQL script to merge four partitions of the range partitioned `sales` table into one partition.

   Perform the following steps:

   a. In SQL Developer, select **File** > **Open**.



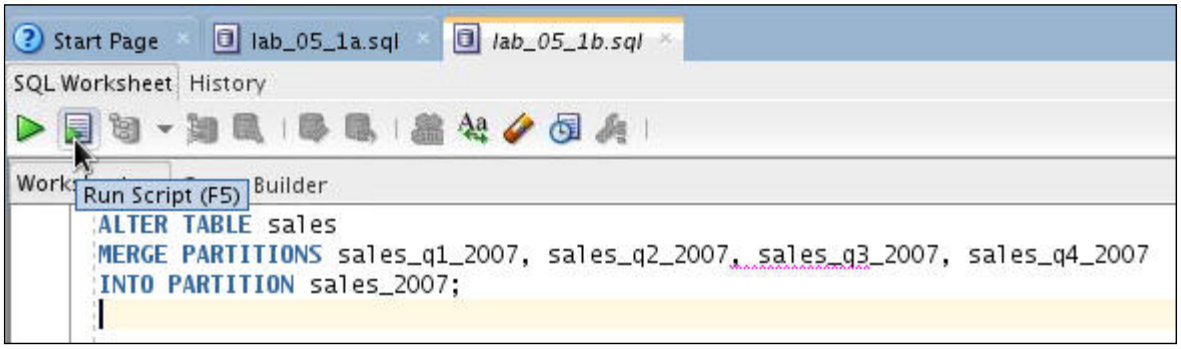
b. Navigate to `home/oracle/labs/SQLFiles/lab_05_1b.sql` and click **Open**.



Note: Optionally, you can copy the following lines of code and paste them in the `HR_ORCL` worksheet. Click the **Run Statement** icon to execute the SQL statements.

```
ALTER TABLE sales
MERGE PARTITIONS sales_q1_2007, sales_q2_2007, sales_q3_2007,
sales_q4_2007
INTO PARTITION sales_2007;
```

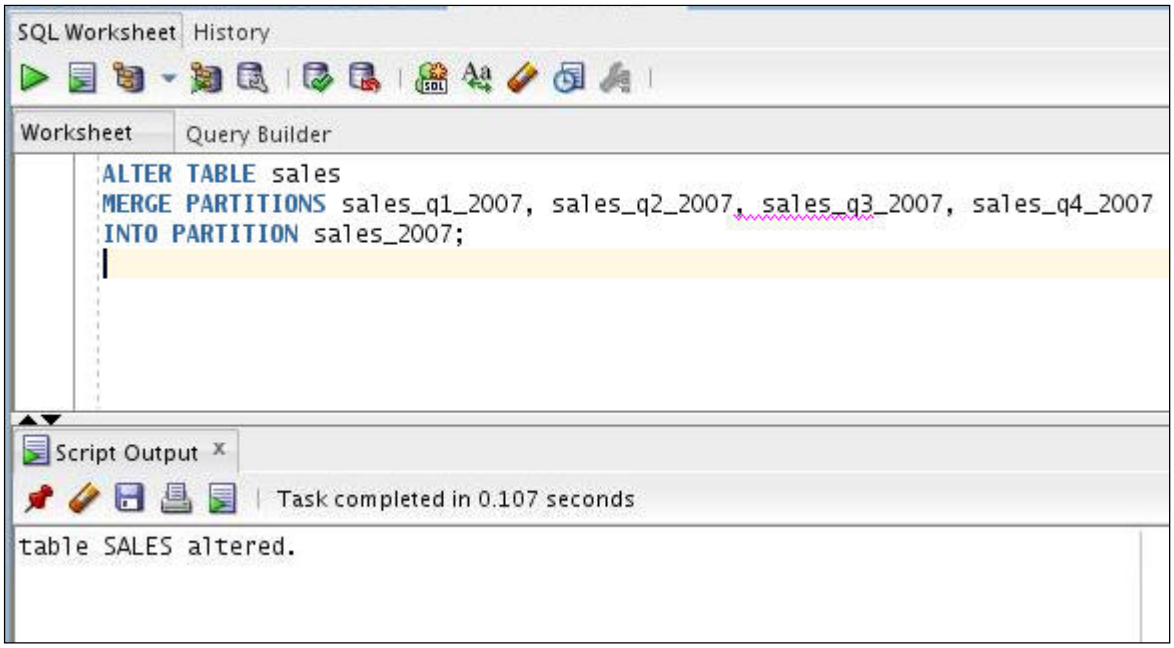
c. The file contains the SQL script to merge the four partitions that were added in the previous task into one partition. Here, the four partitions that correspond to the four quarters of the year 2007 are merged into a single partition that contains the entire sales data for the year. Click **Run Script**.



d. For Select Connection, select `HR_ORCL` and click **OK**.

e. If prompted, for Connection Information, enter the username and password and click OK.

f. The table has been altered.

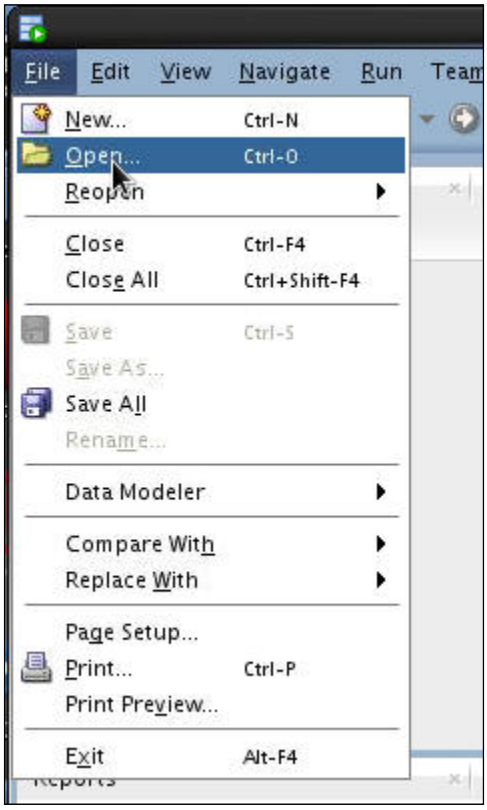


3. Create a table by using list partition and split the partition into multiple partitions.

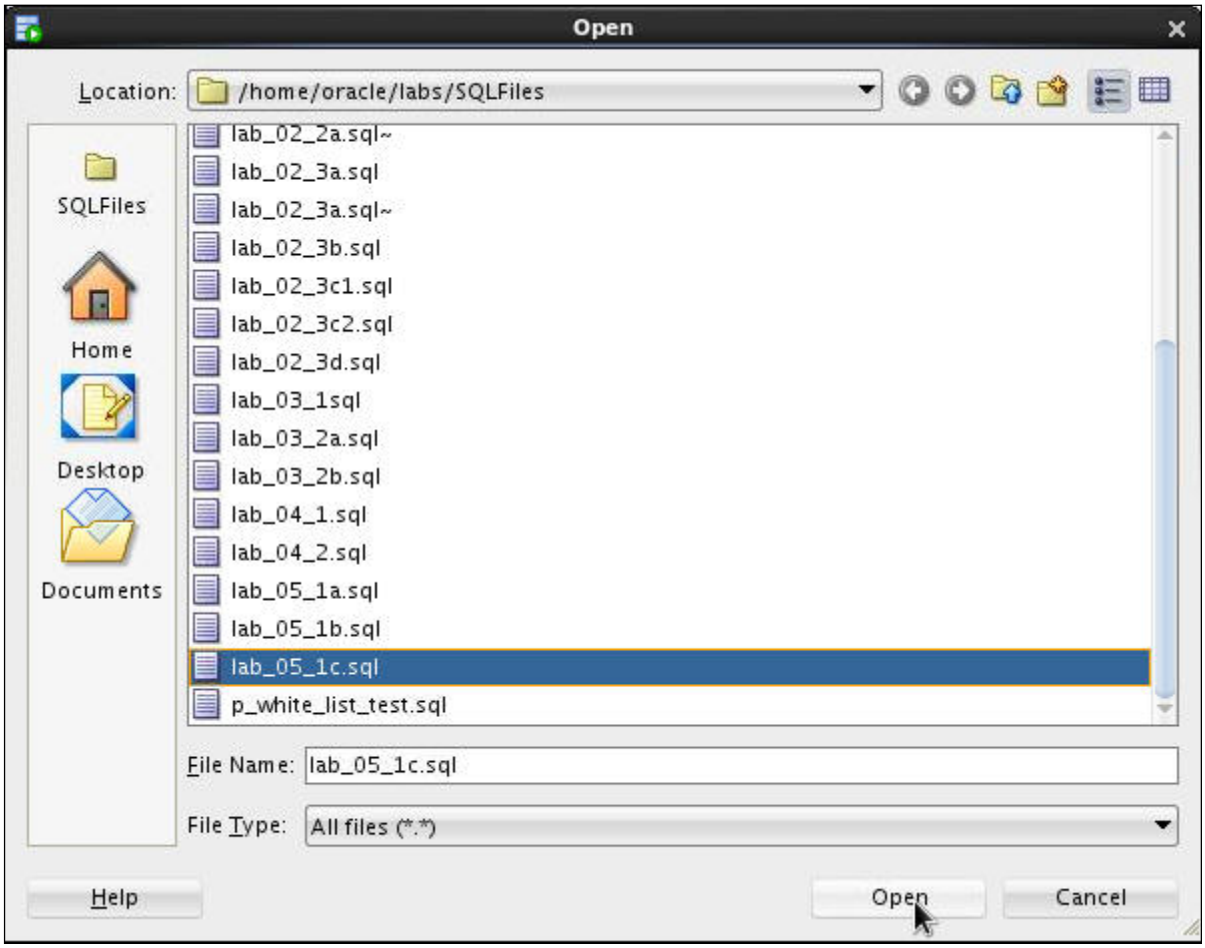
   To perform the preceding task, in SQL Developer, open the `lab_05_1c.sql` file that is available in the `/home/oracle/labs/SQLFiles` folder. This file contains the SQL script to create a table called `list_customers` with list partition. This partition is then split into multiple partitions.

   Perform the following steps:

   a. In SQL Developer, select **File** > **Open**.



b. Navigate to `home/oracle/labs/SQLFiles/lab_05_1c.sql` and click **Open**.



Note: Optionally, you can copy the following lines of code and paste them in the `HR_ORCL` worksheet. Click the **Run Statement** icon to execute the SQL statements.

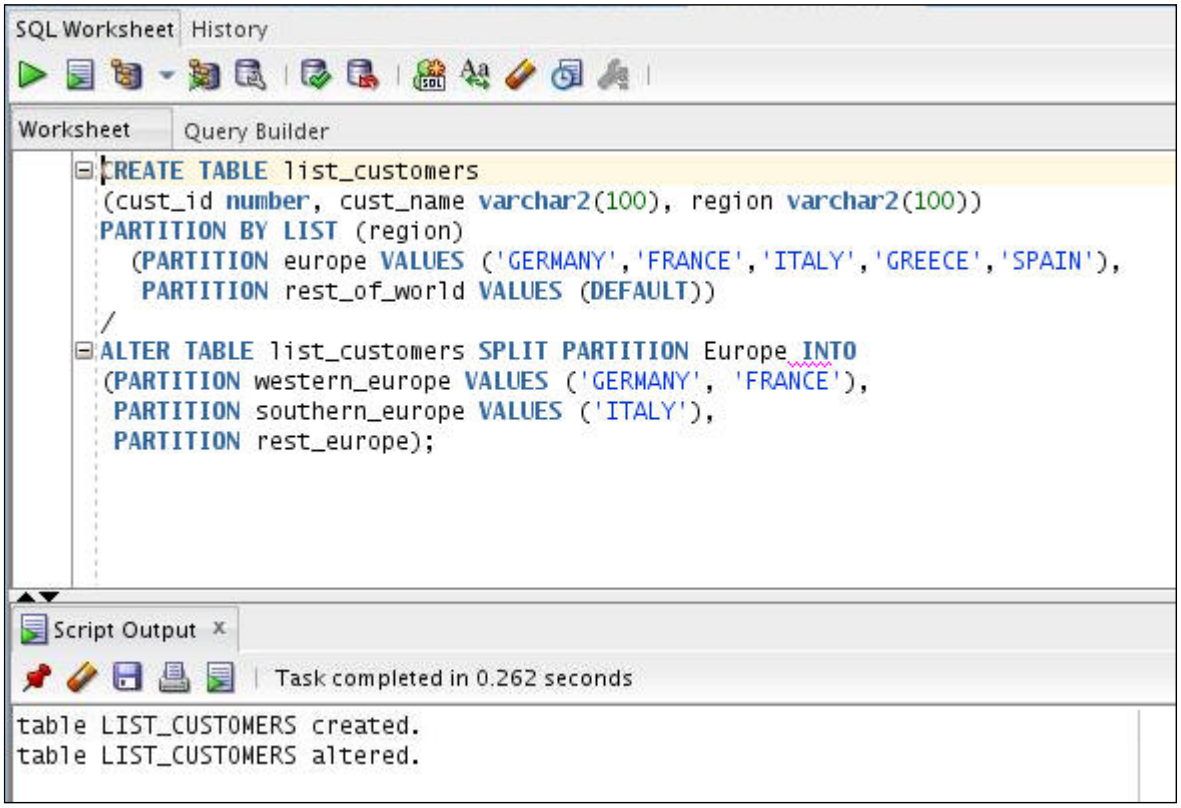
```
CREATE TABLE list_customers
(cust_id number, cust_name varchar2(100), region varchar2(100))
PARTITION BY LIST (region)
  (PARTITION europe VALUES
('GERMANY','FRANCE','ITALY','GREECE','SPAIN'),
   PARTITION rest_of_world VALUES (DEFAULT))
/
ALTER TABLE list_customers SPLIT PARTITION Europe INTO
(PARTITION western_europe VALUES ('GERMANY', 'FRANCE'),
 PARTITION southern_europe VALUES ('ITALY'),
 PARTITION rest_europe);
```

c. This file creates a partitioned table called `list_customers`. The table is then partitioned by List, and splits the partition Europe into three partitions: `western_europe`, `eastern_europe`, and `rest_europe`. Click **Run Script**.

d. For Select Connection, select `HR_ORCL` and click **OK**.

e. If prompted for Connection Information, enter the username and password and click OK.

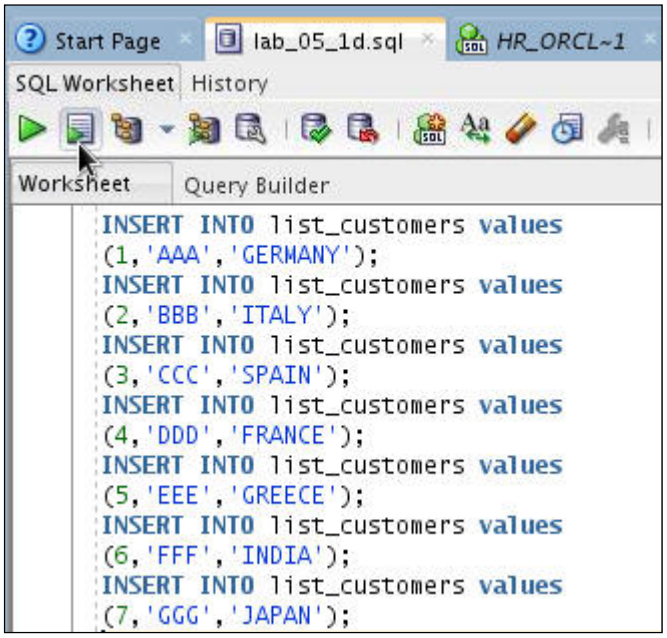
The table has been created and altered.



f. Next, insert some values into the table. Select **File** > **Open**.

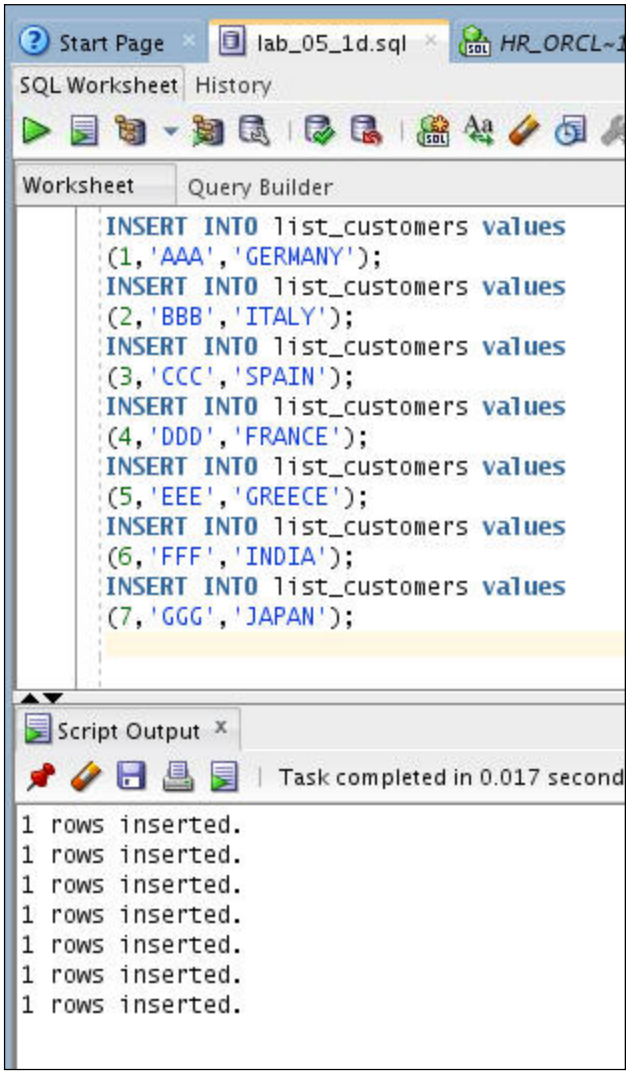
g. Navigate to `home/oracle/labs/SQLFiles/lab_05_1d.sql` and click **Open**.

h. The file contains the SQL script to insert values into the `list_customers` table. Click **Run Script**.

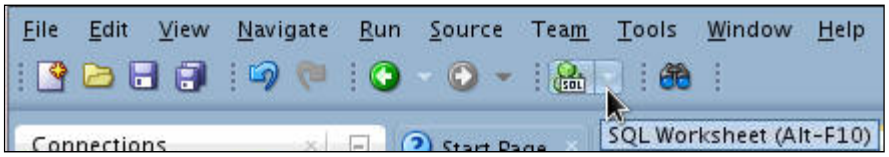


i. If prompted for Select Connection, select `HR_ORCL` and click **OK**.

j. The values are inserted into the table.



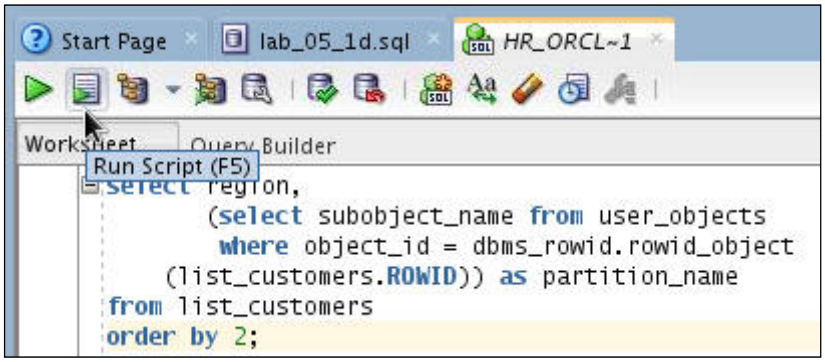
k. To view the table partitions for each region, open SQL Worksheet. Click the **SQL Worksheet** icon.



If prompted, click OK for Select Connection.

I. Enter the following lines of code or alternatively navigate to `home/oracle/labs/SQLFiles/lab_05_1e.sql` and open the SQL file. Click **Run Script**.

```
select region,
       (select subobject_name from user_objects
        where object_id = dbms_rowid.rowid_object

    (list_customers.ROWID)) as partition_name
from list_customers
order by 2;
```



The result displays the region and its corresponding partition name.