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

1) Run the statement in the lab\_09\_01.sql script to build the MY\_EMPLOYEE table used in this practice.

**CREATE TABLE my\_employee**

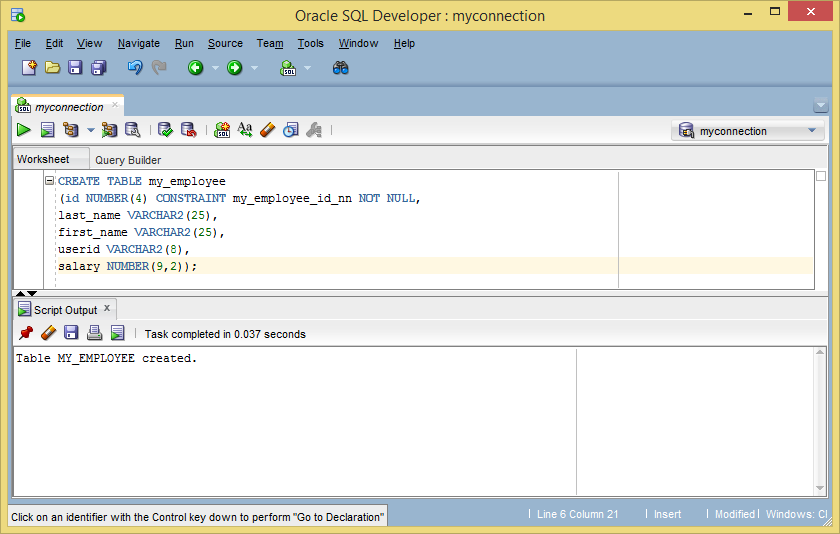
**(id NUMBER(4) CONSTRAINT my\_employee\_id\_nn NOT NULL,**

**last\_name VARCHAR2(25),**

**first\_name VARCHAR2(25),**

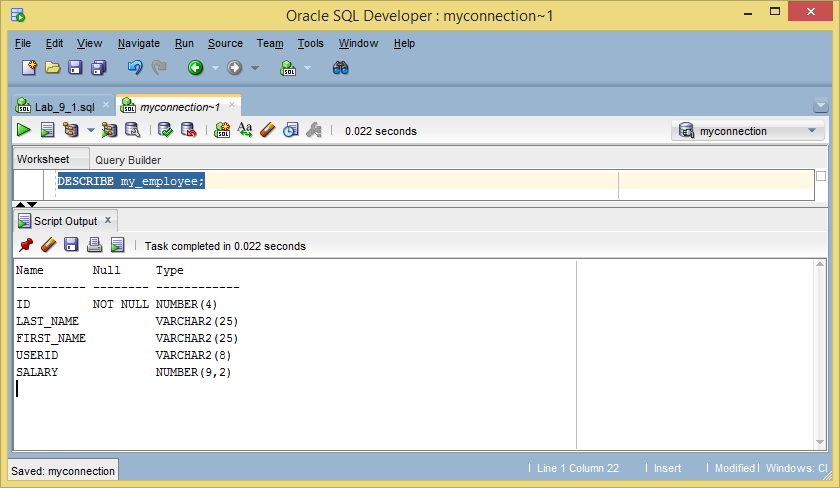
**userid VARCHAR2(8),**

**salary NUMBER(9,2));**



2) Describe the structure of the MY\_EMPLOYEE table to identify the column names.

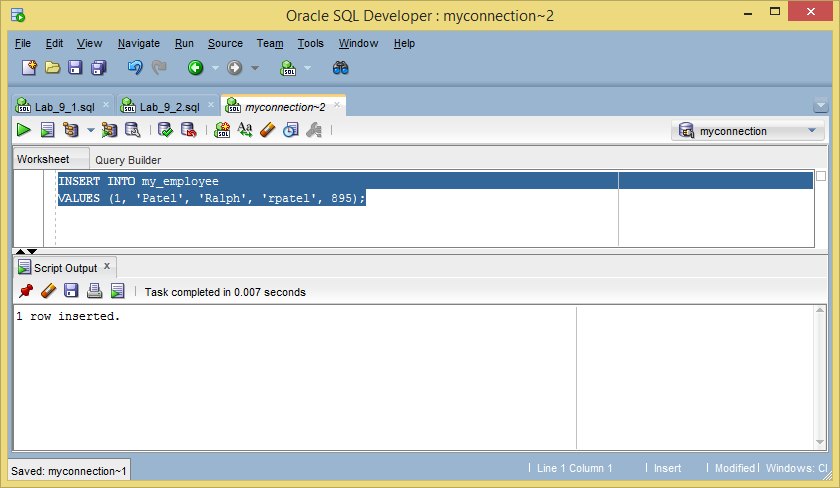
**DESCRIBE my\_employee;**



3) Create an INSERT statement to add the first row of data to the MY\_EMPLOYEE table from the following sample data. Do not list the columns in the INSERT clause. Do not enter all rows yet.

**INSERT INTO my\_employee**

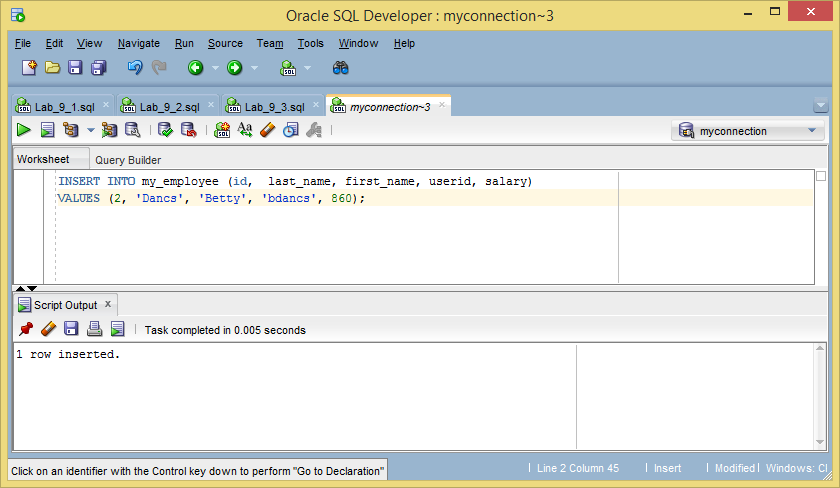
**VALUES (1, 'Patel', 'Ralph', 'rpatel', 895);**



4) Populate the MY\_EMPLOYEE table with the second row of the sample data from the preceding list. This time, list the columns explicitly in the INSERT clause.

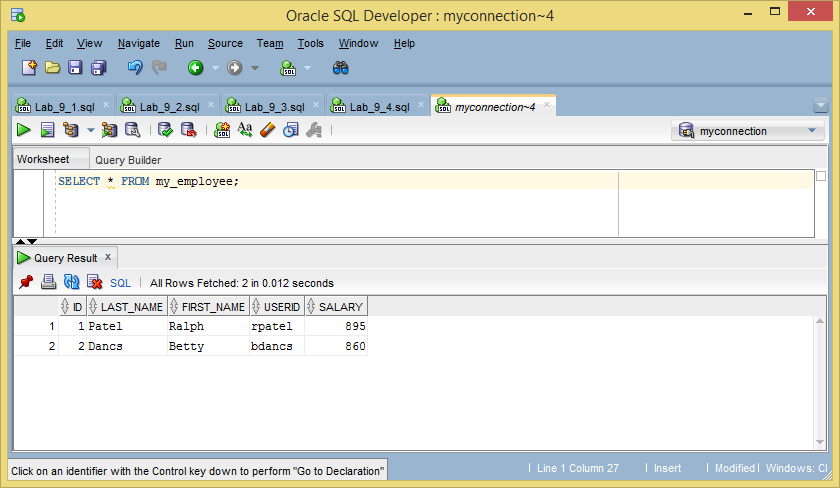
**INSERT INTO my\_employee (id, last\_name, first\_name, userid, salary)**

**VALUES (2, 'Dancs', 'Betty', 'bdancs', 860);**



5) Confirm your addition to the table.

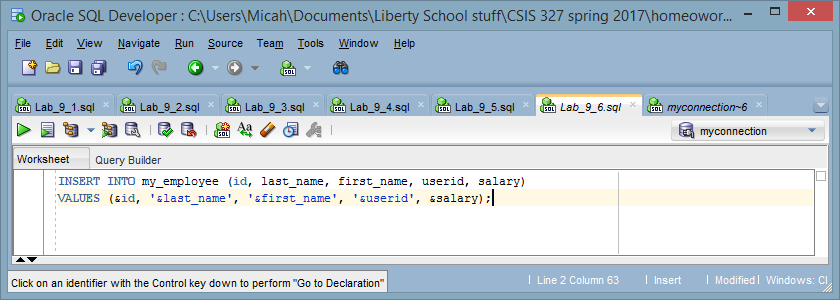
**SELECT \* FROM my\_employee;**



6) Write an INSERT statement in a dynamic reusable script file to load the remaining rows into the MY\_EMPLOYEE table. The script should prompt for all the columns (ID, LAST\_NAME, FIRST\_NAME, USERID, and SALARY). Save this script to a lab\_09\_06.sql file.

**INSERT INTO my\_employee (id, last\_name, first\_name, userid, salary)**

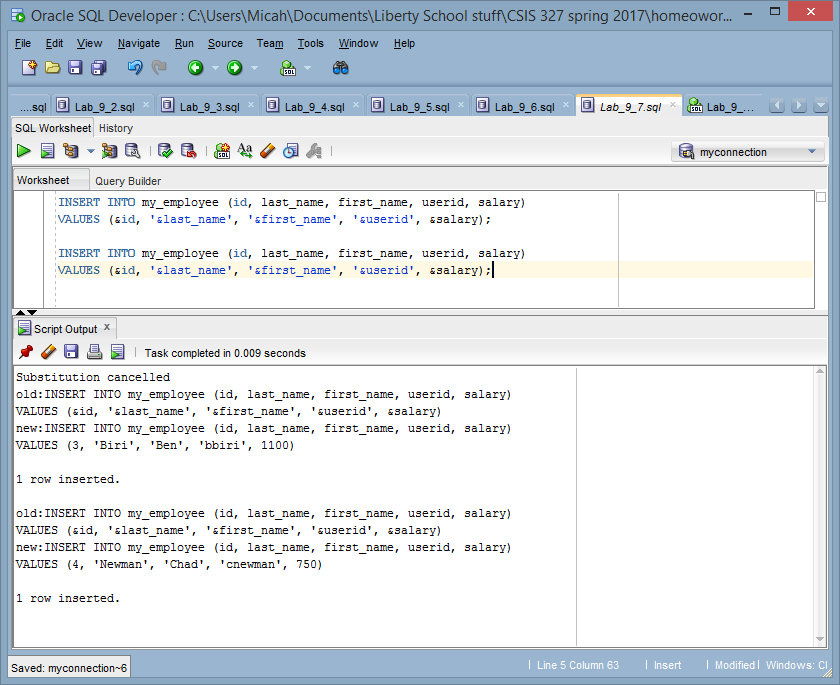
**VALUES (&id, '&last\_name', '&first\_name', '&userid', &salary);**



7) Populate the table with the next two rows of the sample data listed in step 3 by running the INSERT statement in the script that you created.

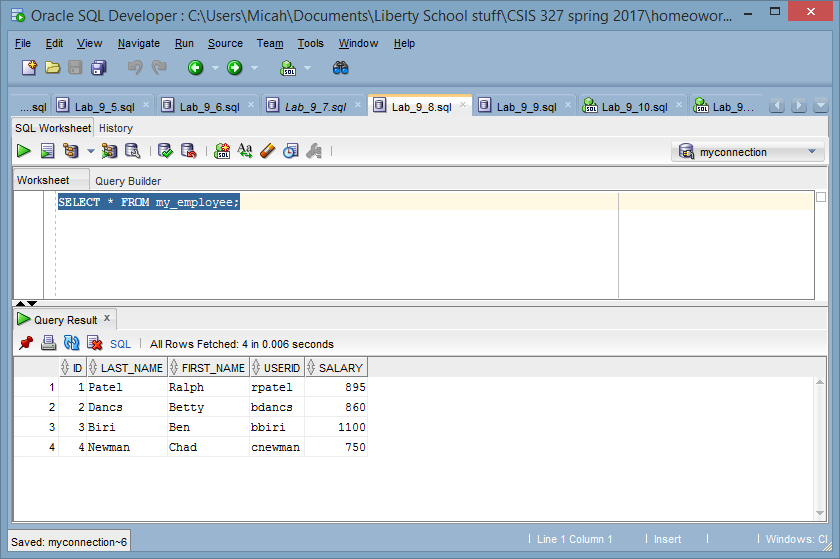
**INSERT INTO my\_employee (id, last\_name, first\_name, userid, salary)**

**VALUES (&id, '&last\_name', '&first\_name', '&userid', &salary);**



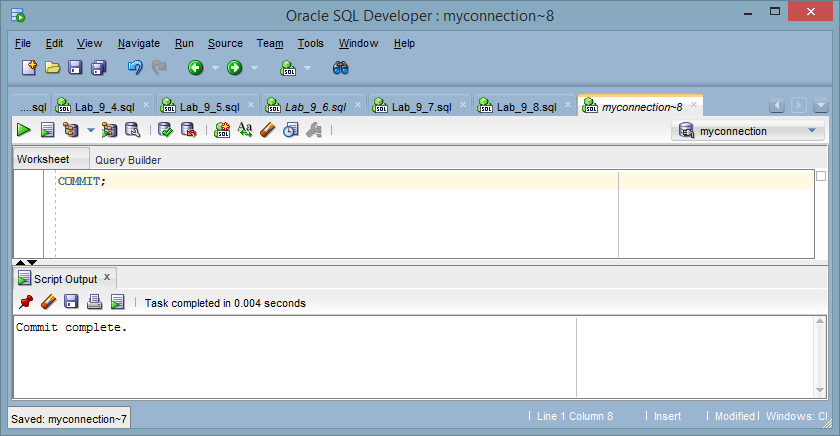
8) Confirm your additions to the table.

**SELECT \* FROM my\_employee;**



9) Make the data additions permanent. Update and delete data in the MY\_EMPLOYEE table.

**COMMIT;**

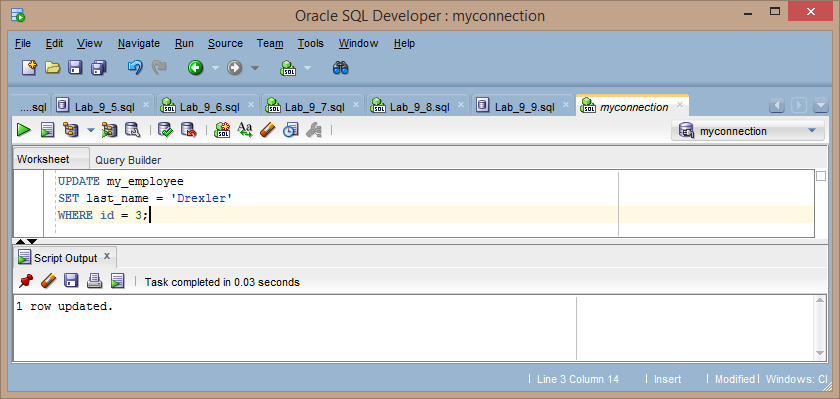


10) Change the last name of employee 3 to Drexler.

**UPDATE my\_employee**

**SET last\_name = 'Drexler'**

**WHERE id = 3;**

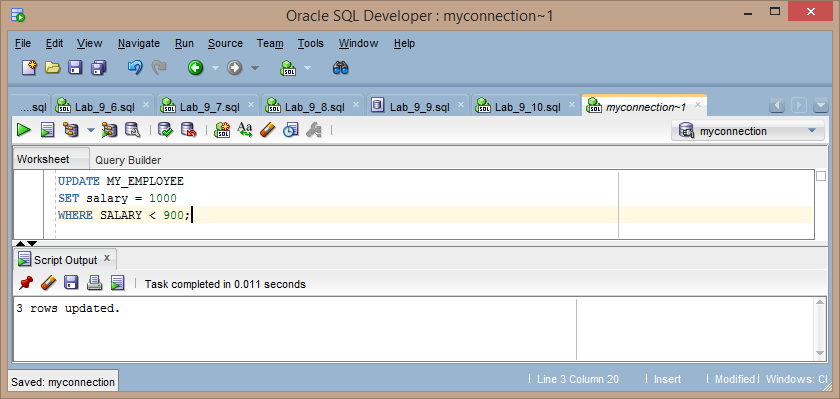


11) Change the salary to $1,000 for all employees who have a salary less than $900.

**UPDATE MY\_EMPLOYEE**

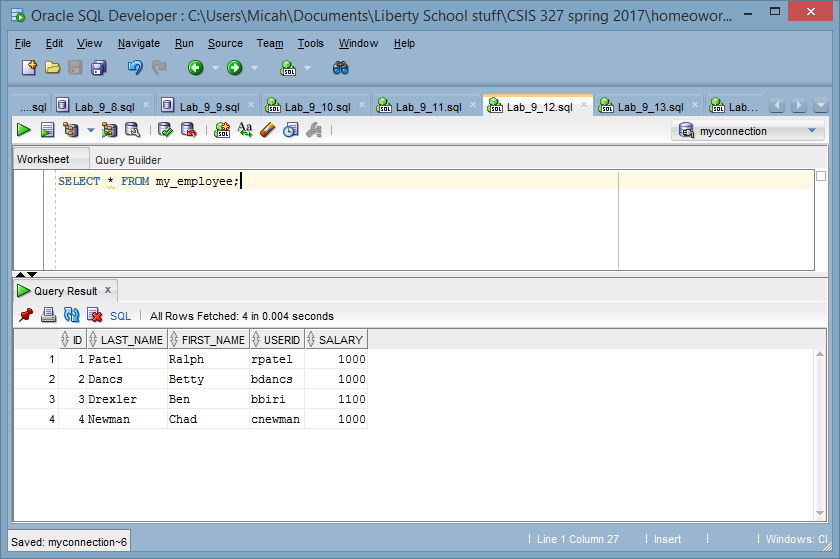
**SET salary = 1000**

**WHERE SALARY < 900;**



12) Verify your changes to the table.

**SELECT \* FROM my\_employee;**

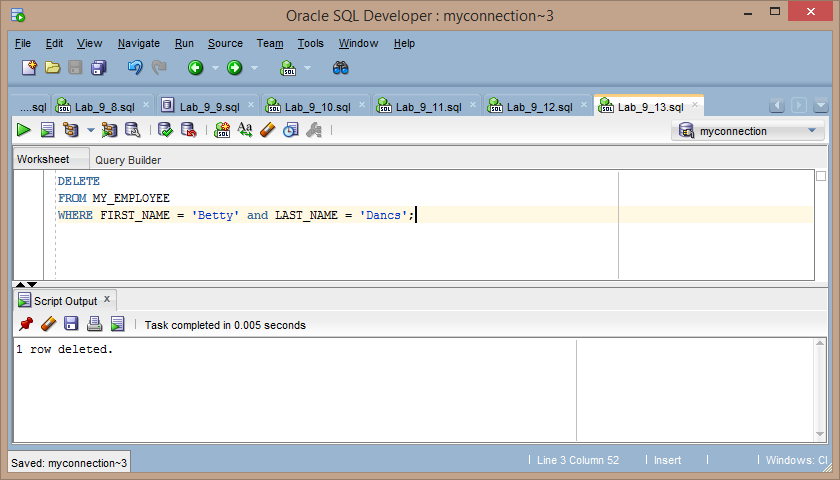


13) Delete Betty Dancs from the MY\_EMPLOYEE table.

**DELETE**

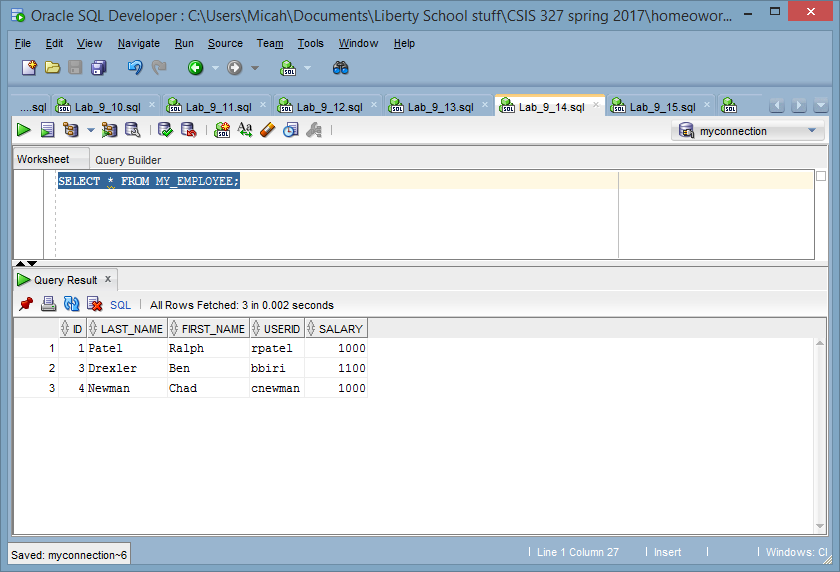
**FROM MY\_EMPLOYEE**

**WHERE FIRST\_NAME = 'Betty' and LAST\_NAME = 'Dancs';**



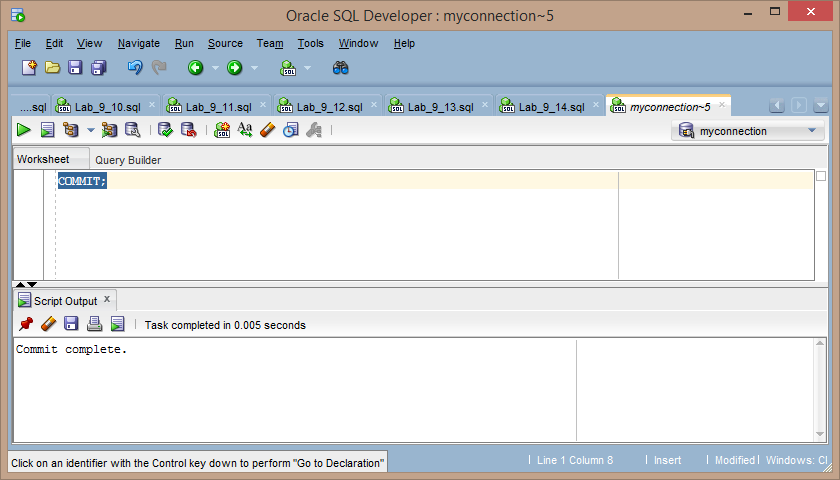
14) Confirm your changes to the table.

**SELECT \* FROM MY\_EMPLOYEE;**



15) Commit all pending changes. Control data transaction to the MY\_EMPLOYEE table.

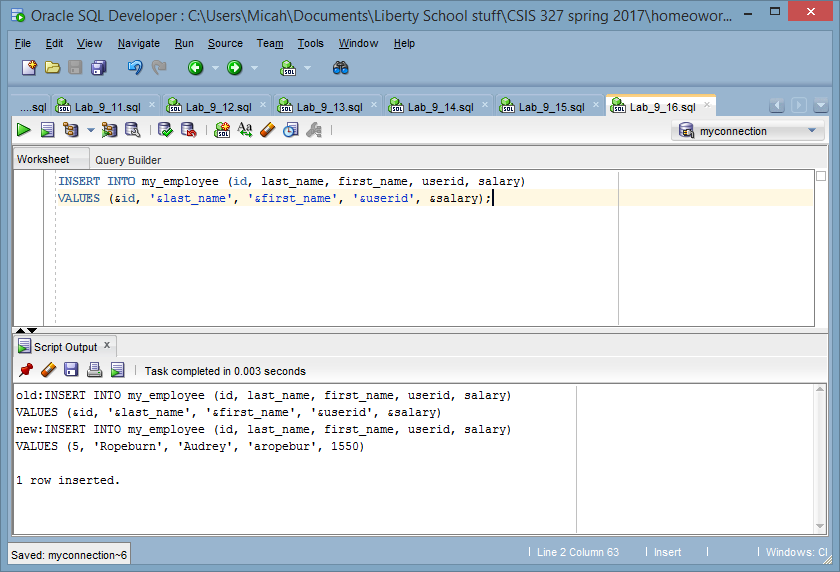
**COMMIT;**



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

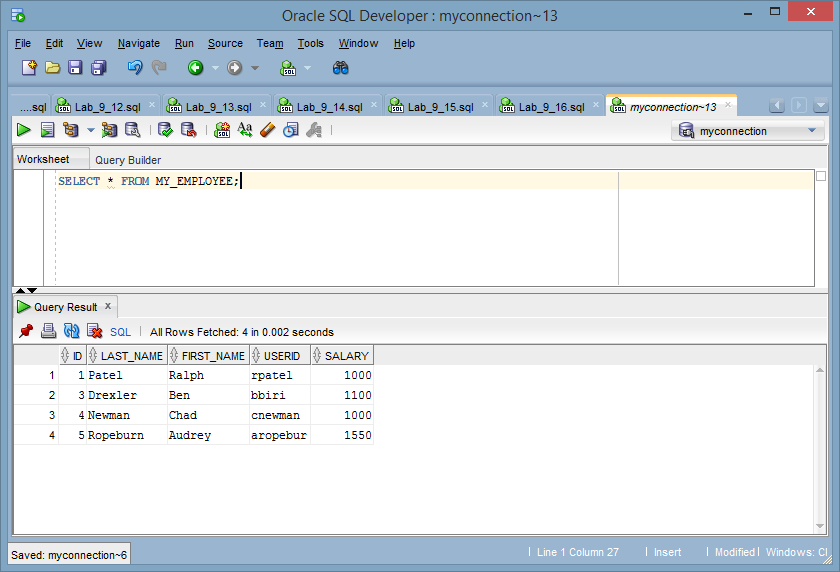
**INSERT INTO my\_employee (id, last\_name, first\_name, userid, salary)**

**VALUES (&id, '&last\_name', '&first\_name', '&userid', &salary);**



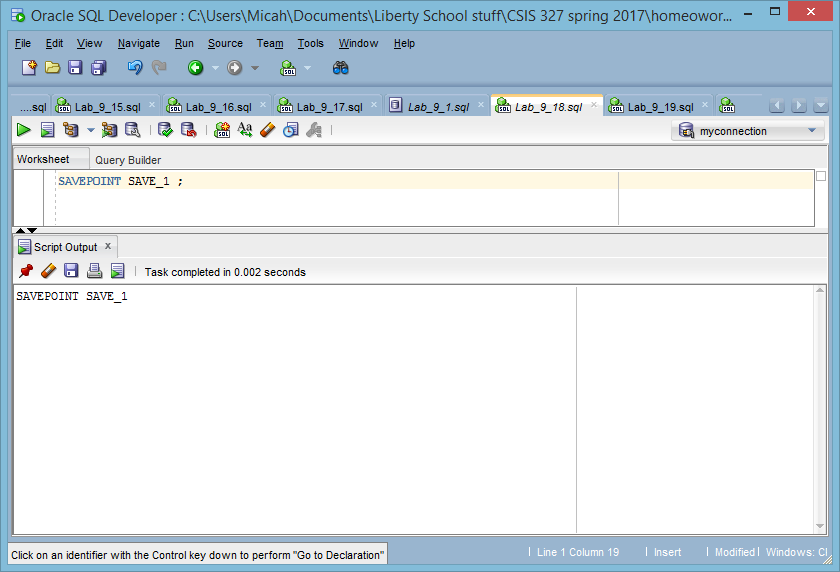
17) Confirm your addition to the table.

**SELECT \* FROM MY\_EMPLOYEE;**



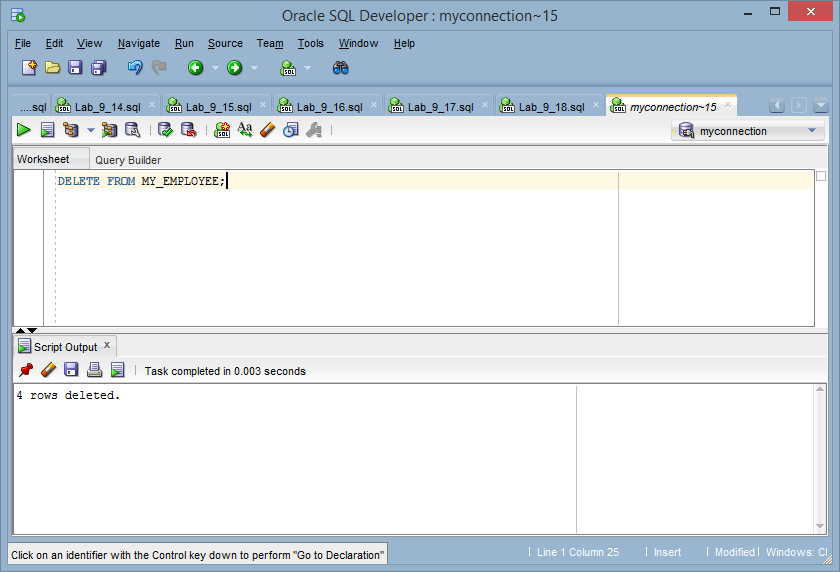
18) Mark an intermediate point in the processing of the transaction.

**SAVEPOINT step18;**



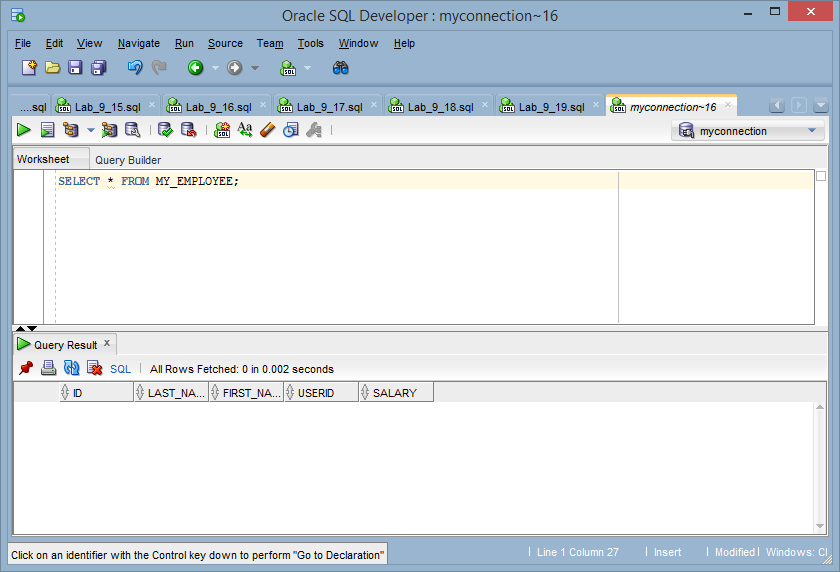
19) Delete all the rows from the MY\_EMPLOYEE table.

**DELETE FROM MY\_EMPLOYEE;**



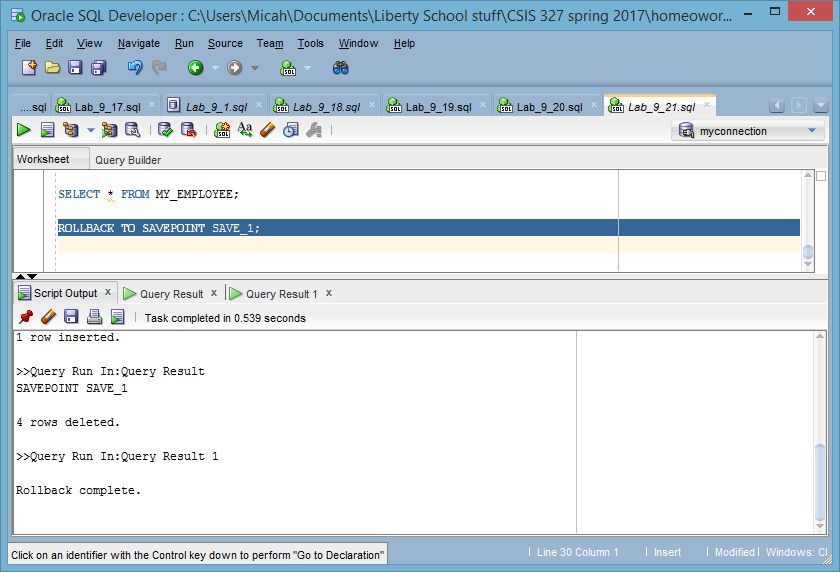
20) Confirm that the table is empty.

**SELECT \* FROM MY\_EMPLOYEE;**



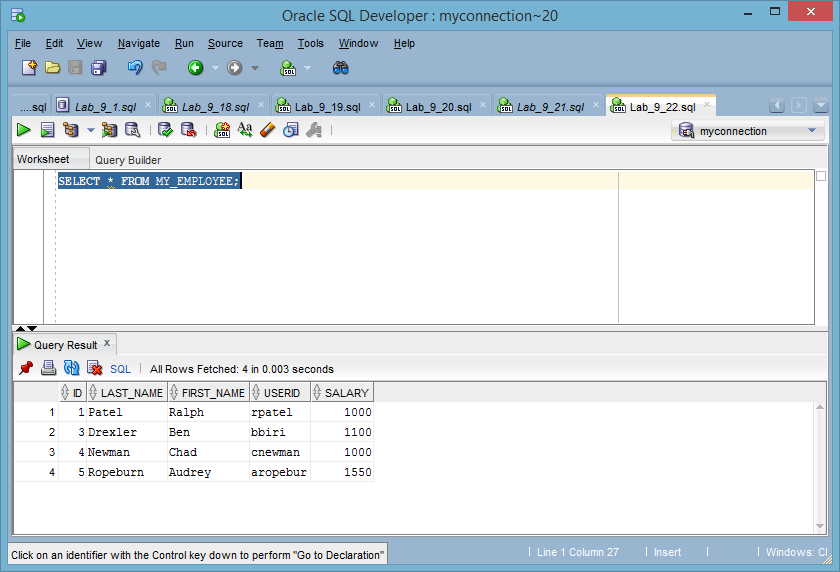
21) Discard the most recent DELETE operation without discarding the earlier INSERT operation.

**ROLLBACK TO SAVEPOINT SAVE\_1;**



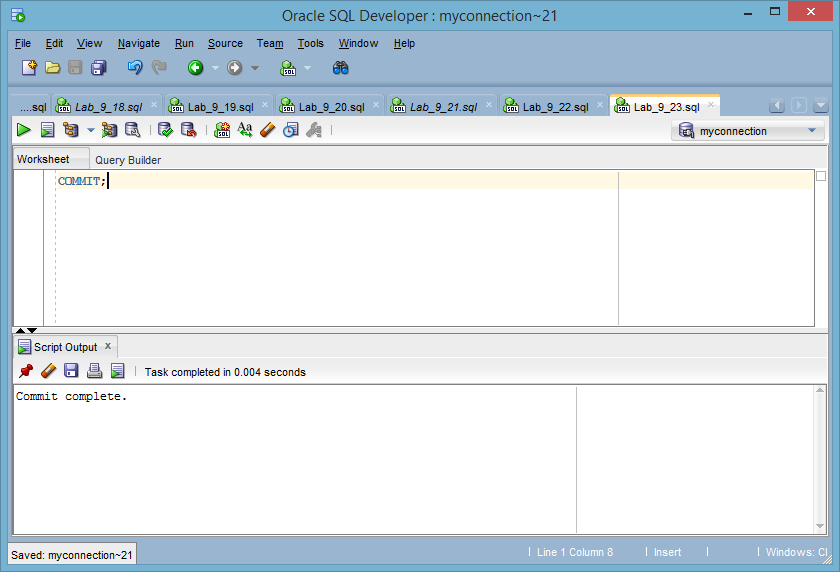
22) Confirm that the new row is still intact.

**SELECT \* FROM MY\_EMPLOYEE;**



23) Make the data addition permanent.

**COMMIT;**



24) Make a copy of the employees table and name it employees\_yourname. Replace “yourname” with your actual name. Use the employees\_yourname table for the subsequent exercises. Insert five (5) new rows. You can use any appropriate real or figurative people. Show a screenshot of the new individuals only.

**DROP TABLE employees\_micah;**

**CREATE TABLE employees\_micah**

**(id NUMBER(4) CONSTRAINT employees\_micah\_id\_nn NOT NULL,**

**last\_name VARCHAR2(25),**

**first\_name VARCHAR2(25),**

**userid VARCHAR2(8),**

**salary NUMBER(9,2));**

**INSERT INTO employees\_micah**

**SELECT \* FROM MY\_EMPLOYEE ;**

**INSERT INTO my\_employee (id, last\_name, first\_name, userid, salary)**

**VALUES (6, 'Shmoe', 'Joe', 'jshmoe', 5);**

**INSERT INTO my\_employee (id, last\_name, first\_name, userid, salary)**

**VALUES (7, 'Sam', 'Sammy', 'ssam', 9999);**

**INSERT INTO my\_employee (id, last\_name, first\_name, userid, salary)**

**VALUES (8, 'Mad', 'Vlad', 'vmad', 1234);**

**INSERT INTO my\_employee (id, last\_name, first\_name, userid, salary)**

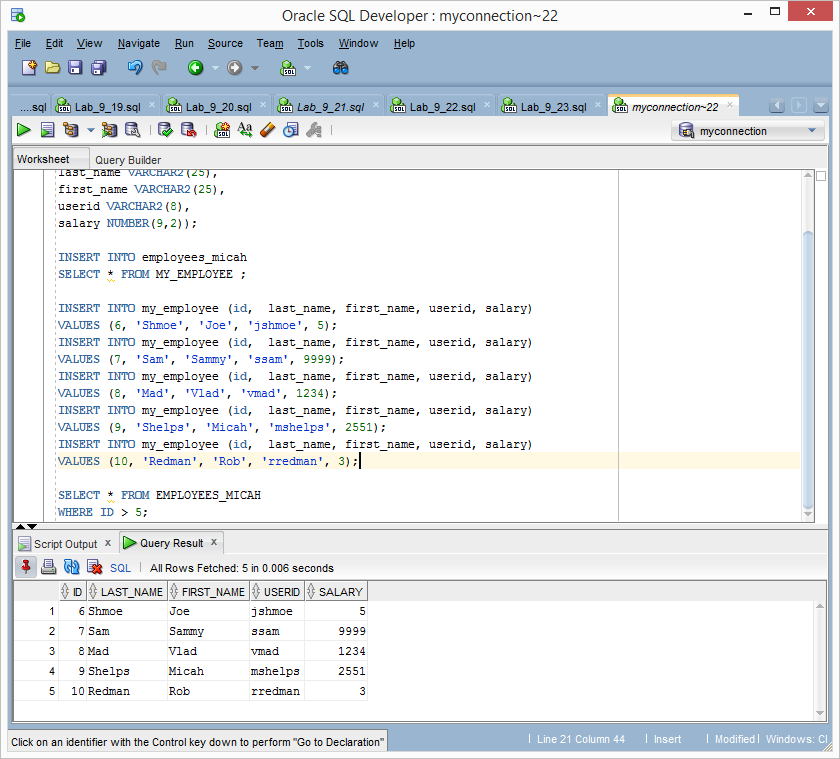
**VALUES (9, 'Shelps', 'Micah', 'mshelps', 2551);**

**INSERT INTO my\_employee (id, last\_name, first\_name, userid, salary)**

**VALUES (10, 'Redman', 'Rob', 'rredman', 3);**

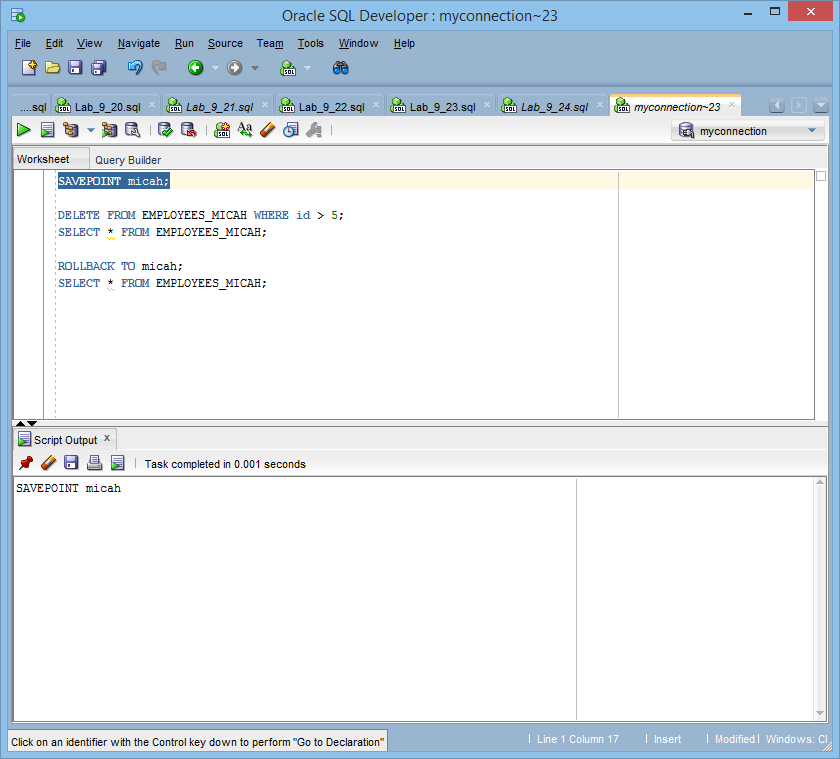
**SELECT \* FROM EMPLOYEES\_MICAH**

**WHERE ID > 5;**



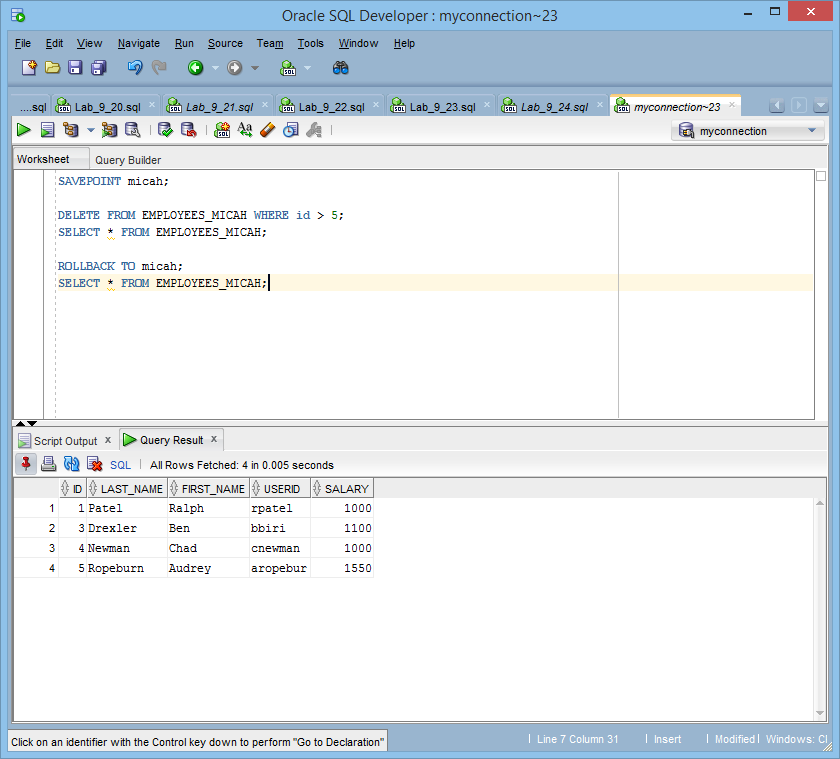
25) Create a new savepoint of the table. Delete only the new rows. Subsequently, rollback the transaction. Show a minimum of three (3) screenshots of the successful savepoint, confirmation of the removal of the rows using proper SQL, and the restoration following removal to the original savepoint.

**SAVEPOINT micah;**

****

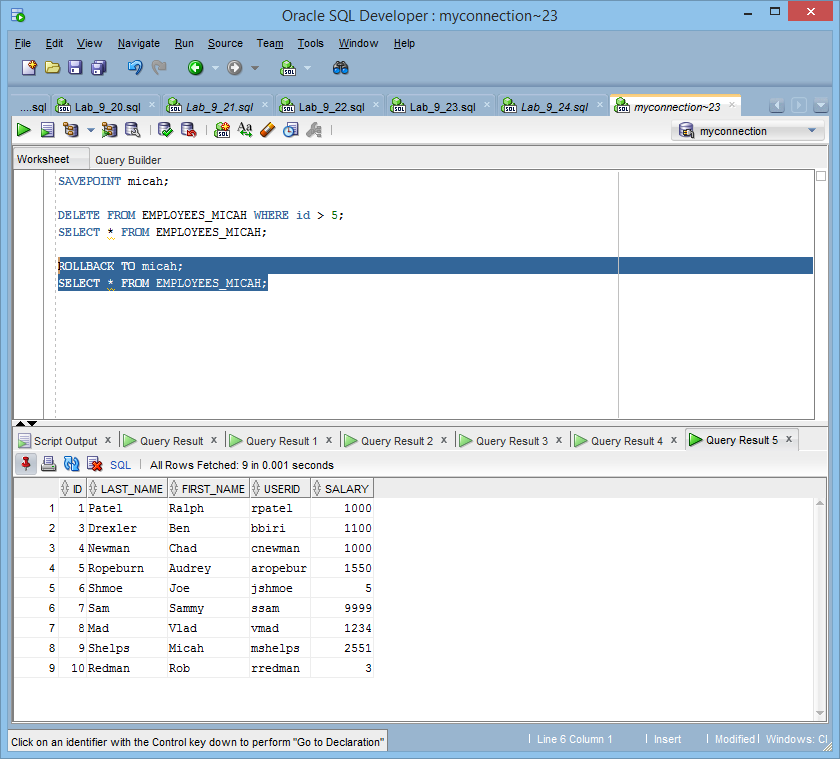
**DELETE FROM EMPLOYEES\_MICAH WHERE id > 5;**

**SELECT \* FROM EMPLOYEES\_MICAH;**

****

**ROLLBACK TO micah;**

**SELECT \* FROM EMPLOYEES\_MICAH;**



26) Using the crow’s foot methodology, design an entity relationship diagram in third normal form (3NF). Follow appropriate ERD notation rules. This should include entities, attributes, relationships, descriptions, inside and outside cardinality, primary keys, and foreign keys. At the minimum, the business is requesting that the following data be stored for a new cookbook application:

* Management of kitchen inventory
* Management of cooking recipes
* Comparison of cooking recipes to necessary inventory to fulfill the recipe’s requirements

