**Database Programming (4203\_2)**

**Exercises 8-11**

**Name: Sakshi Jain Student ID: c0753352**

**Instruction: Use Oracle HR sample database to answer exercise 10 and 11.**

Exercise 8 (1.2%)

Create PL/SQL block that has two variables of number datatype, assign numerical values to the two variables, then compare these two numbers, if the first number greater than the second number, display the message “ First Number is greater than Second Number” , if the two numbers are equal, display “Two numbers are equal”, lastly, if the first number less than second number, display “First Number less the Second Number”.

Note: you need to include all the three conditions statements

SET SERVEROUTPUT ON

DECLARE

num1 number;

num2 number;

BEGIN

num1 := &number;

num2 := &number;

DBMS\_OUTPUT.PUT\_LINE('you entered first number as: ' || num1);

DBMS\_OUTPUT.PUT\_LINE('you entered second number as: ' || num2);

IF num1 > num2 THEN

DBMS\_OUTPUT.PUT\_LINE('First number is greater than second number');

ELSIF num1 < num2 THEN

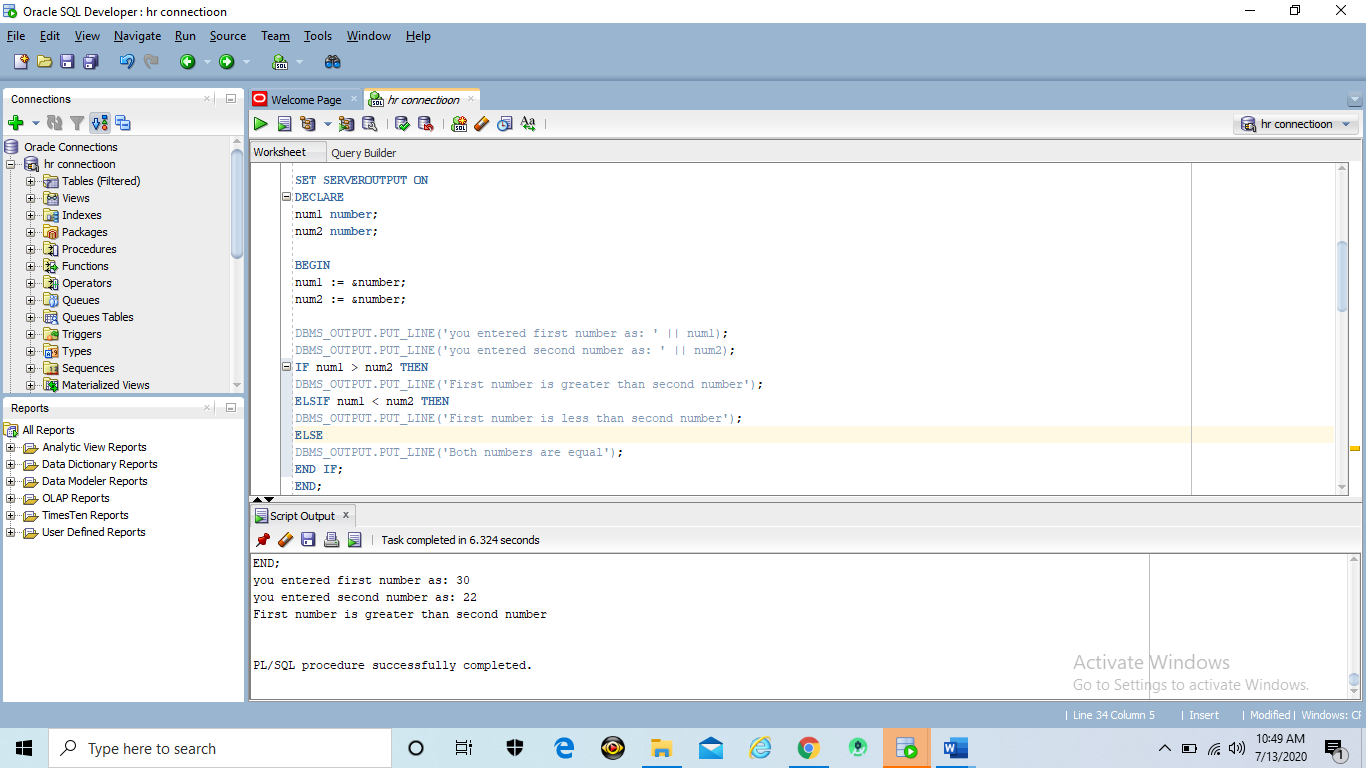
DBMS\_OUTPUT.PUT\_LINE('First number is less than second number');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Both numbers are equal');

END IF;

END;



DECLARE

num1 NUMBER(6,0) := 30;

num2 NUMBER(6,0) := 50;

BEGIN

IF num1 > num2 THEN

DBMS\_OUTPUT.PUT\_LINE('First number is greater than second number');

ELSIF num1 < num2 THEN

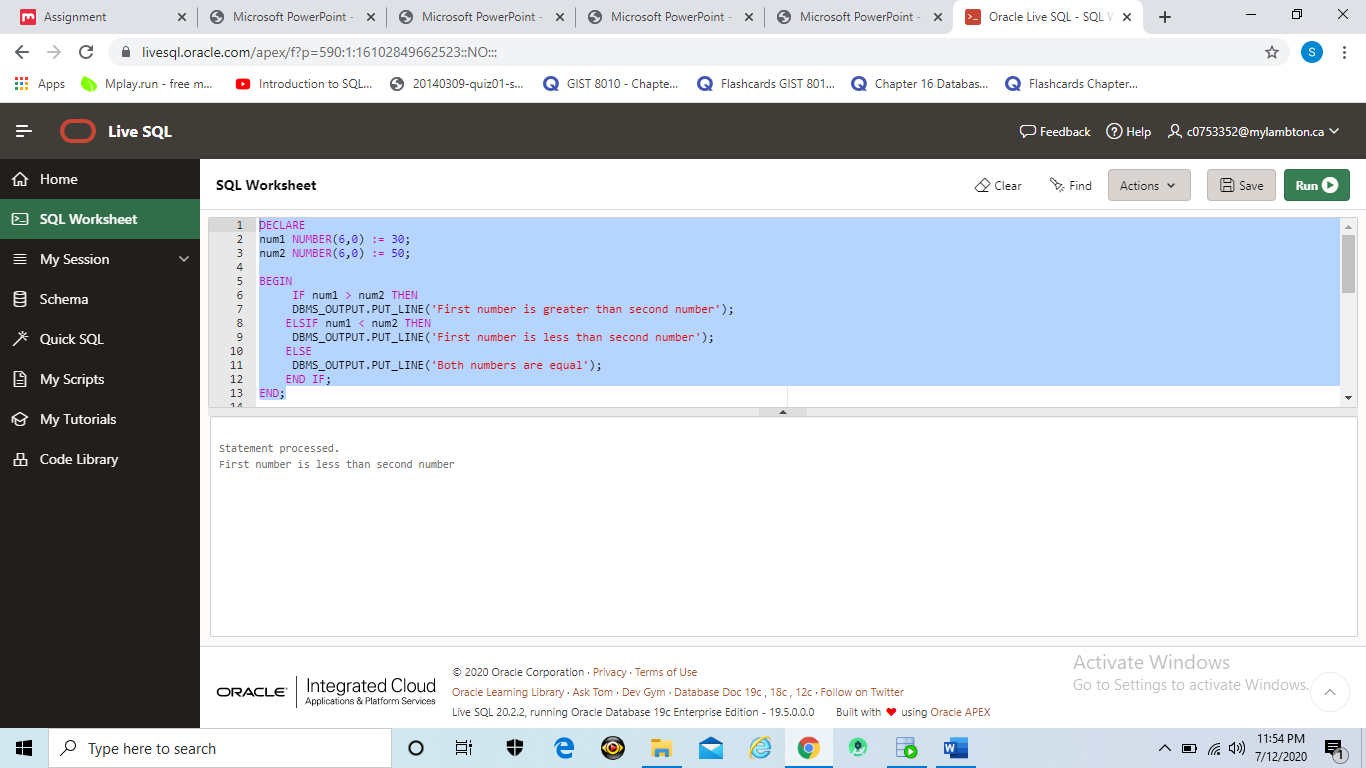
DBMS\_OUTPUT.PUT\_LINE('First number is less than second number');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Both numbers are equal');

END IF;

END;



Exercise 9 (1.2%)

**Instruction: Include FULL PAGE screenshot that will display the code and the output on SQL Developer.**

Create PL/SQL blocks that incorporate loops and conditional control structures. This practice tests your understanding of various IF statements and LOOP constructs.

1. Execute the following SQL:

DROP TABLE mails;

CREATE TABLE mails (inbox number);

1. Write a PL/SQL block to insert numbers into the mails table.
2. Insert the numbers 5 through 15, excluding 7 and 9.
3. Commit before the end of the block.
4. Execute a SELECT statement to verify that your PL/SQL block worked.

CREATE TABLE hr.mails(inbox number);

DECLARE

i NUMBER := 5;

BEGIN

LOOP

if i not in(7,9) then

insert into mails values(i);

end if;

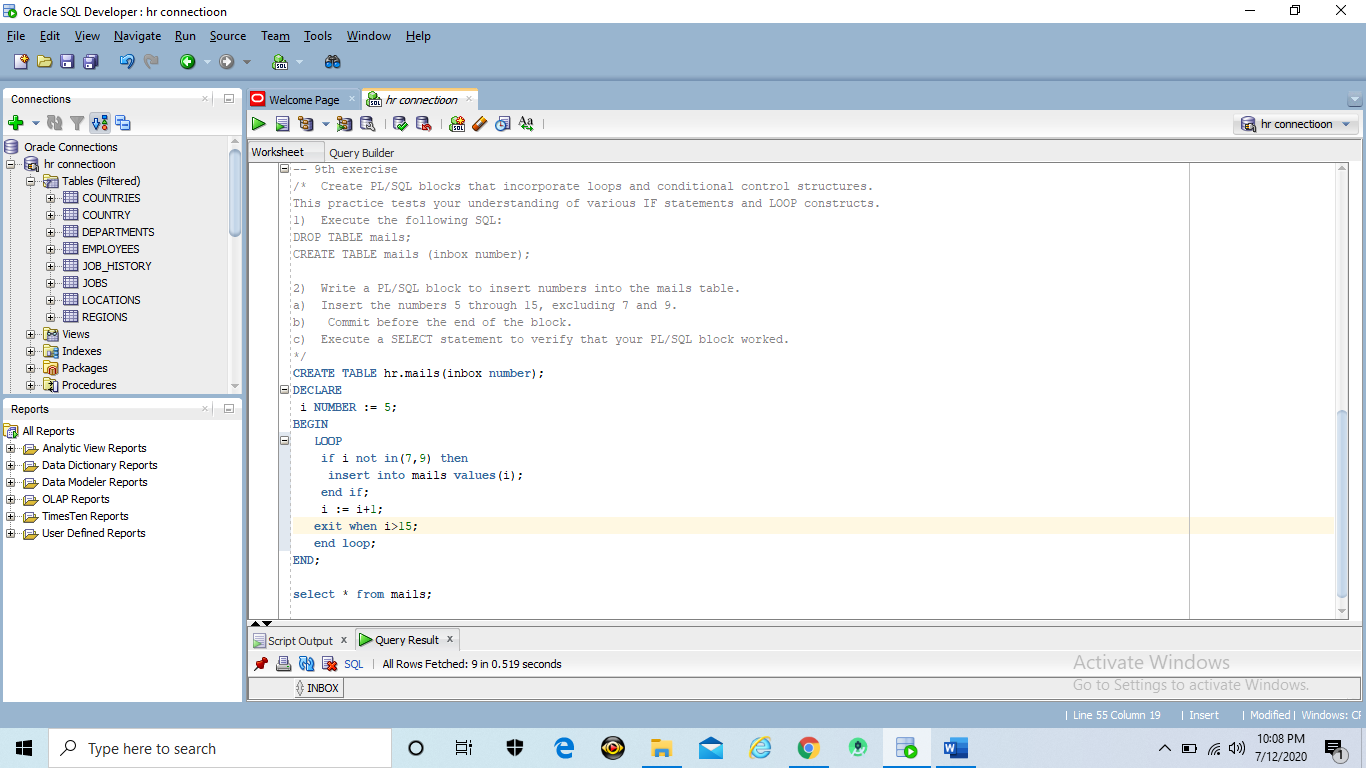
i := i+1;

exit when i>15;

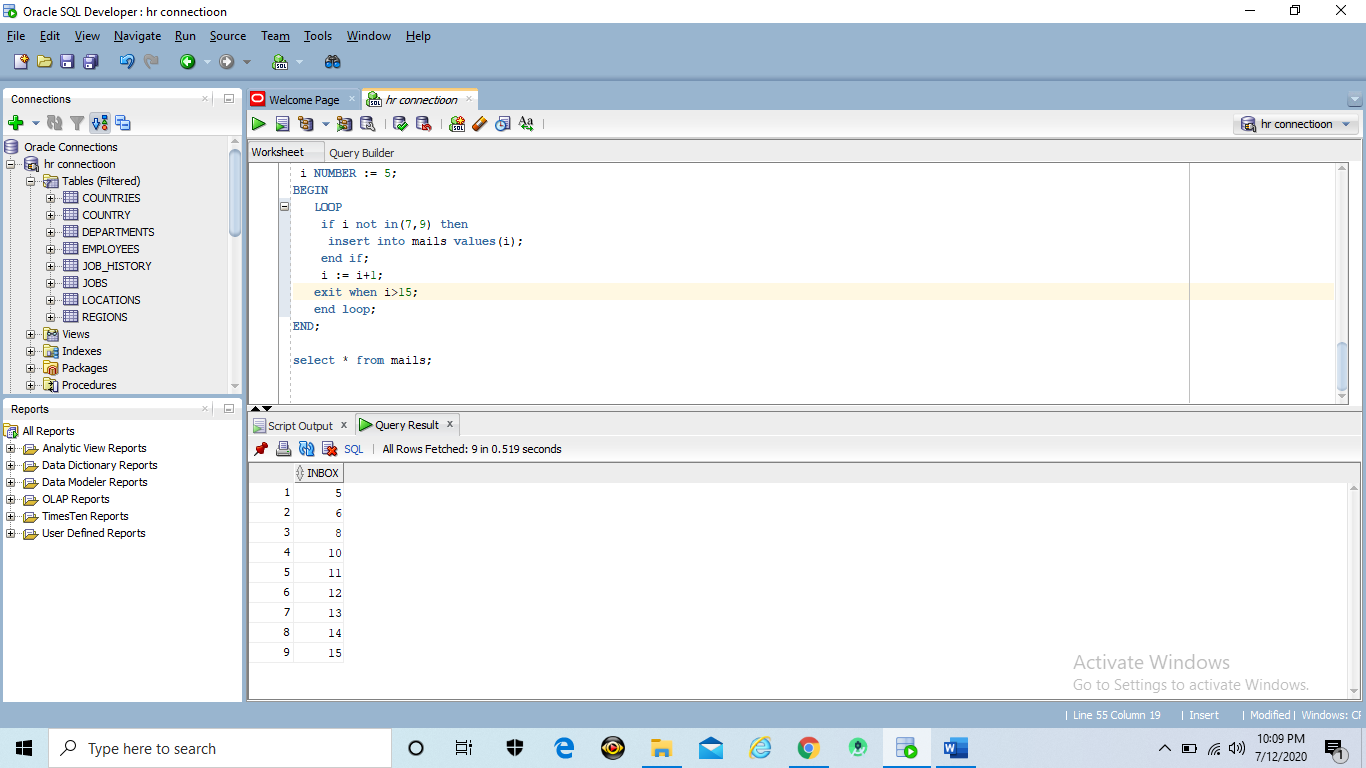
end loop;

END;

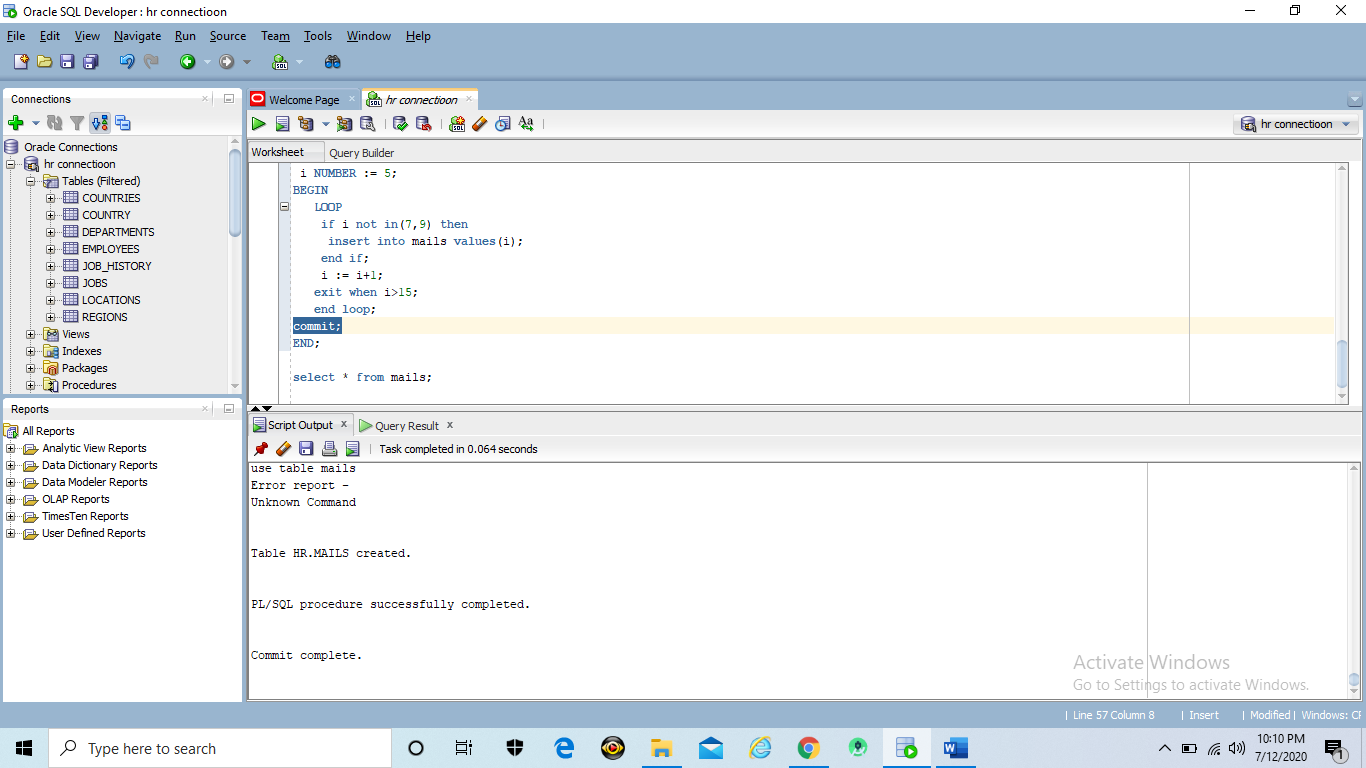
select \* from mails;



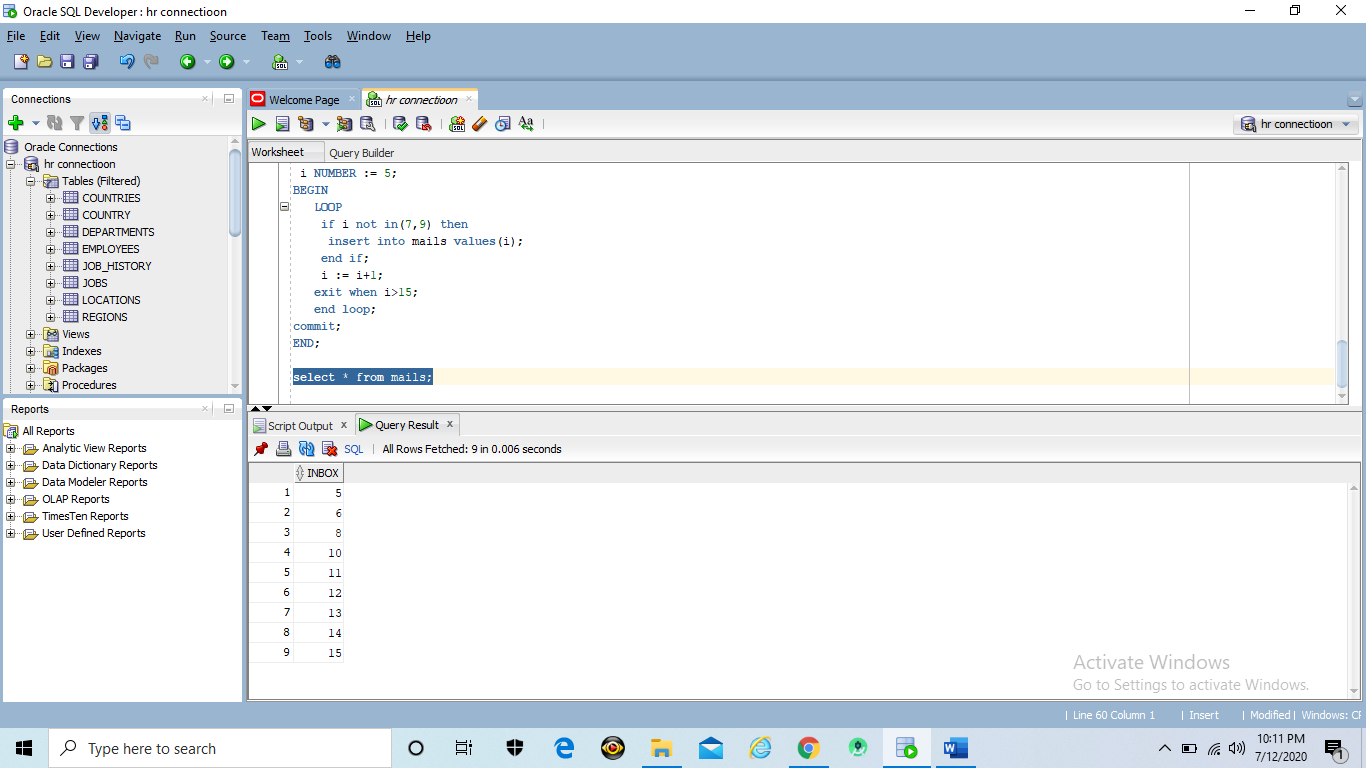
Output



After Commit

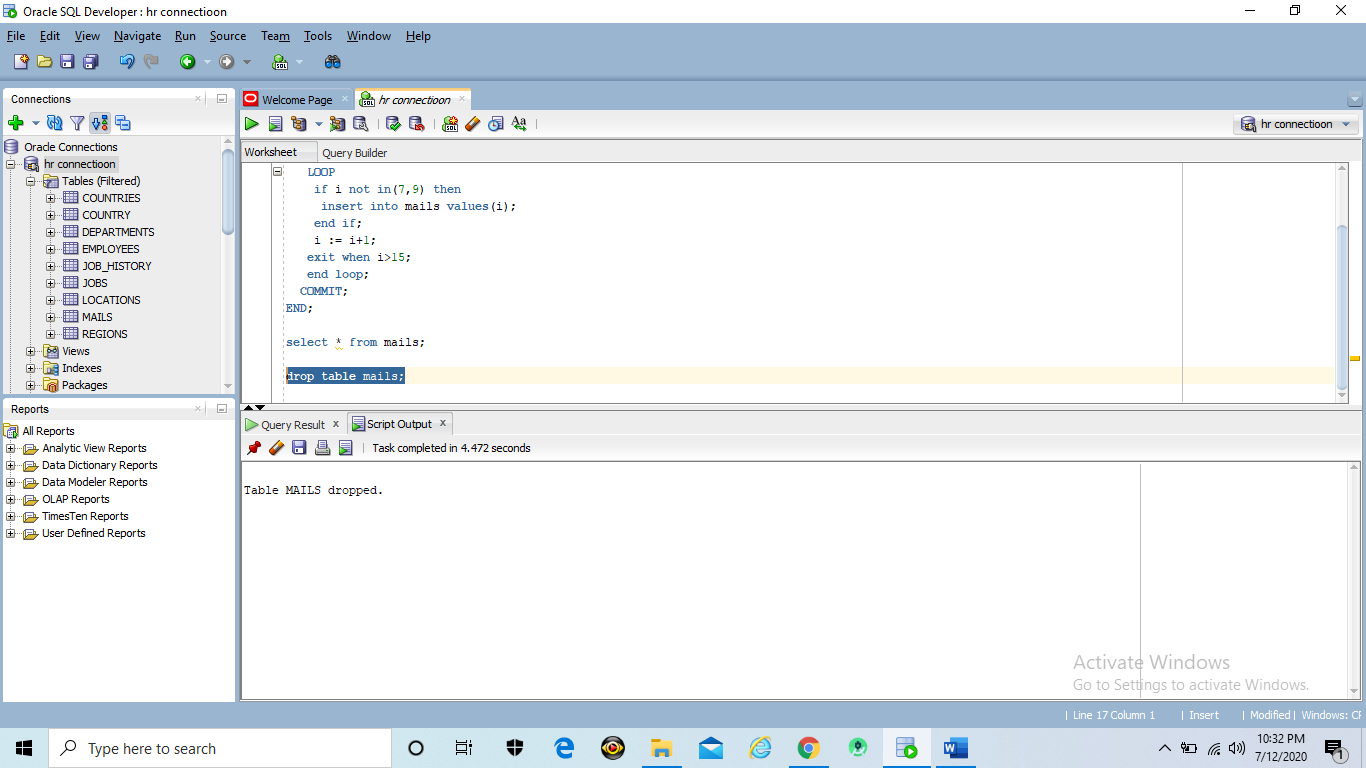


Select statement



Drop table

drop table mails;



Exercise 10 (1.2%)

**Instruction: Include FULL PAGE screenshot that will display the code and the output on SQL Developer.**

Write a PL/SQL block to print information about a given country.

a) Declare a PL/SQL record based on the structure of the departments table.

b) Declare a variable v\_department\_id. Assign 60 to v\_department\_id.

c) In the declarative section, use the %ROWTYPE attribute and declare the v\_departments\_record variable of type departments.

d) In the executable section, get all the information from the departments table by using v\_department\_id. Display selected information about the country.

The sample output is as follows

department Id: 60 department Name: IT Manager ID:103 Location\_id: 1400

SET SERVEROUTPUT ON

DECLARE

v\_department\_id NUMBER := 60;

v\_departments\_record hr.departments%ROWTYPE;

BEGIN

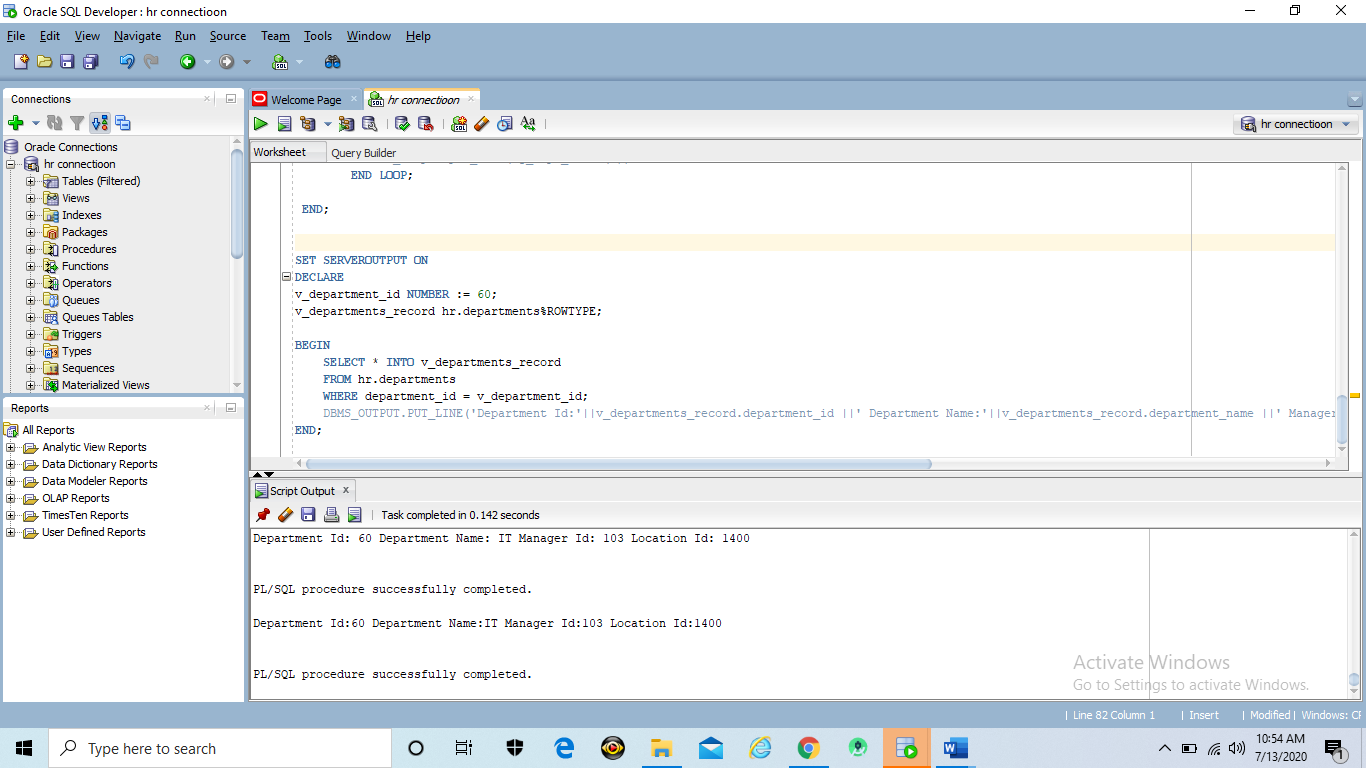
SELECT \* INTO v\_departments\_record

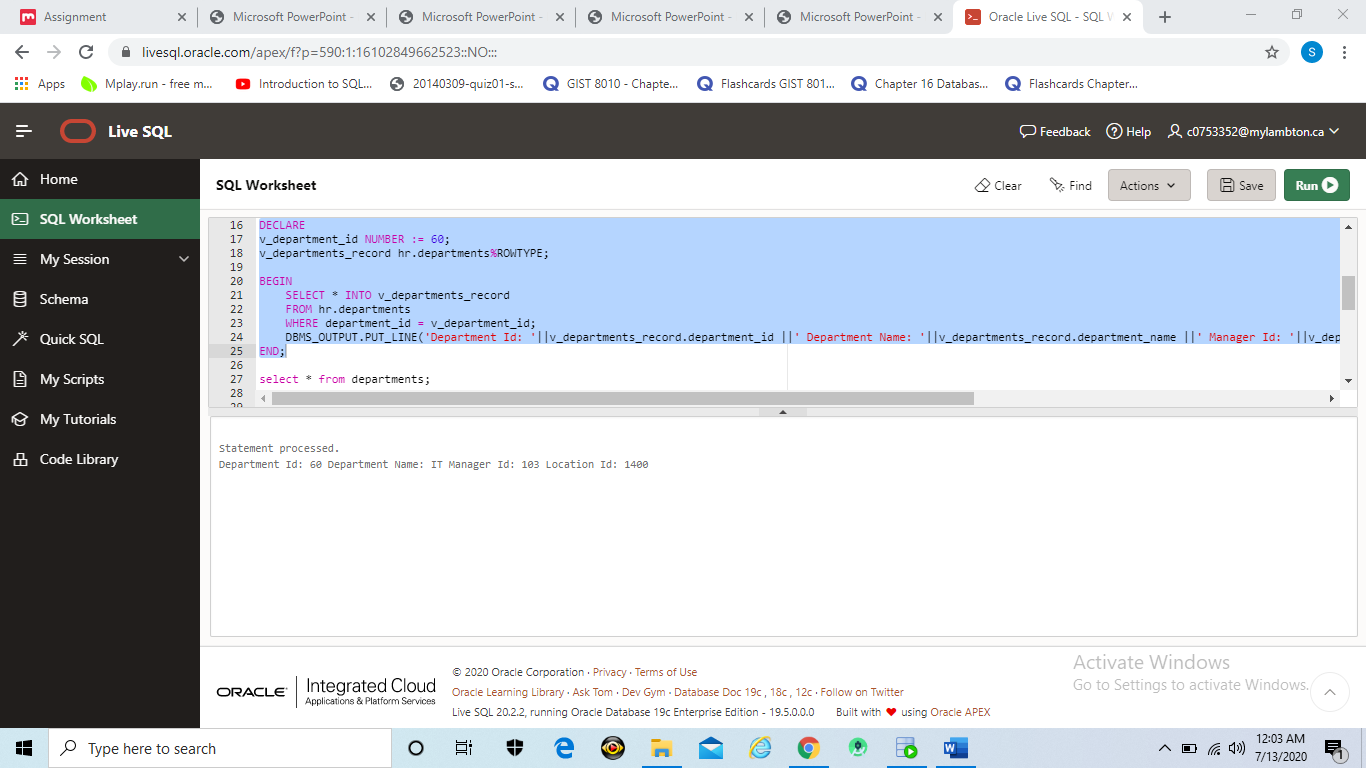
FROM hr.departments

WHERE department\_id = v\_department\_id;

DBMS\_OUTPUT.PUT\_LINE('Department Id:'||v\_departments\_record.department\_id ||' Department Name:'||v\_departments\_record.department\_name ||' Manager Id:'||v\_departments\_record.manager\_id||' Location Id:'|| v\_departments\_record.location\_id);

END;



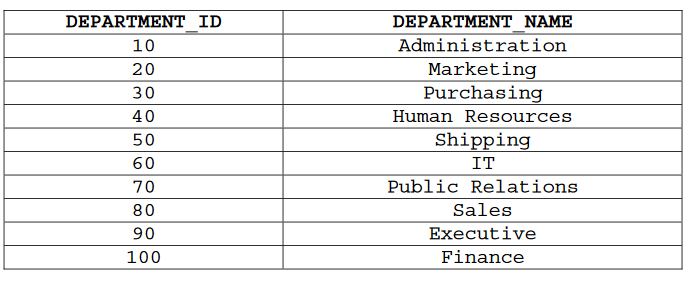


Exercise 11 (1.2%)

**Instruction: Include FULL PAGE screenshot that will display the code and the output on SQL Developer.**

Create a PL/SQL block to retrieve the names of some departments from the departments table and print each department name on the screen, incorporating an associative array.

1. Declare an INDEXBY table dept\_table\_type of type departments.department\_name. Declare a variable my\_dept\_table of type dept\_table\_type to temporarily store the names of the departments.
2. Declare two variables: f\_loop\_count and v\_deptno of type NUMBER. Assign 10 to f\_loop\_count and 0 to v\_deptno.
3. Using a loop, retrieve the names of 10 departments and store the names in the associative array. Start with department\_id 10. Increase v\_deptno by 10 for every loop iteration. The following table shows the department\_id for which you should retrieve the department\_name.



SET SERVEROUTPUT ON

DECLARE

TYPE dept\_table\_type IS TABLE OF departments%ROWTYPE INDEX BY PLS\_INTEGER;

my\_dept\_table dept\_table\_type;

f\_loop\_count NUMBER (2):= 10;

v\_deptno NUMBER (4) := 0;

BEGIN

for i IN 1..f\_loop\_count LOOP

v\_deptno := v\_deptno + 10;

SELECT \* INTO my\_dept\_table(i) FROM departments WHERE department\_id = v\_deptno;

END LOOP;

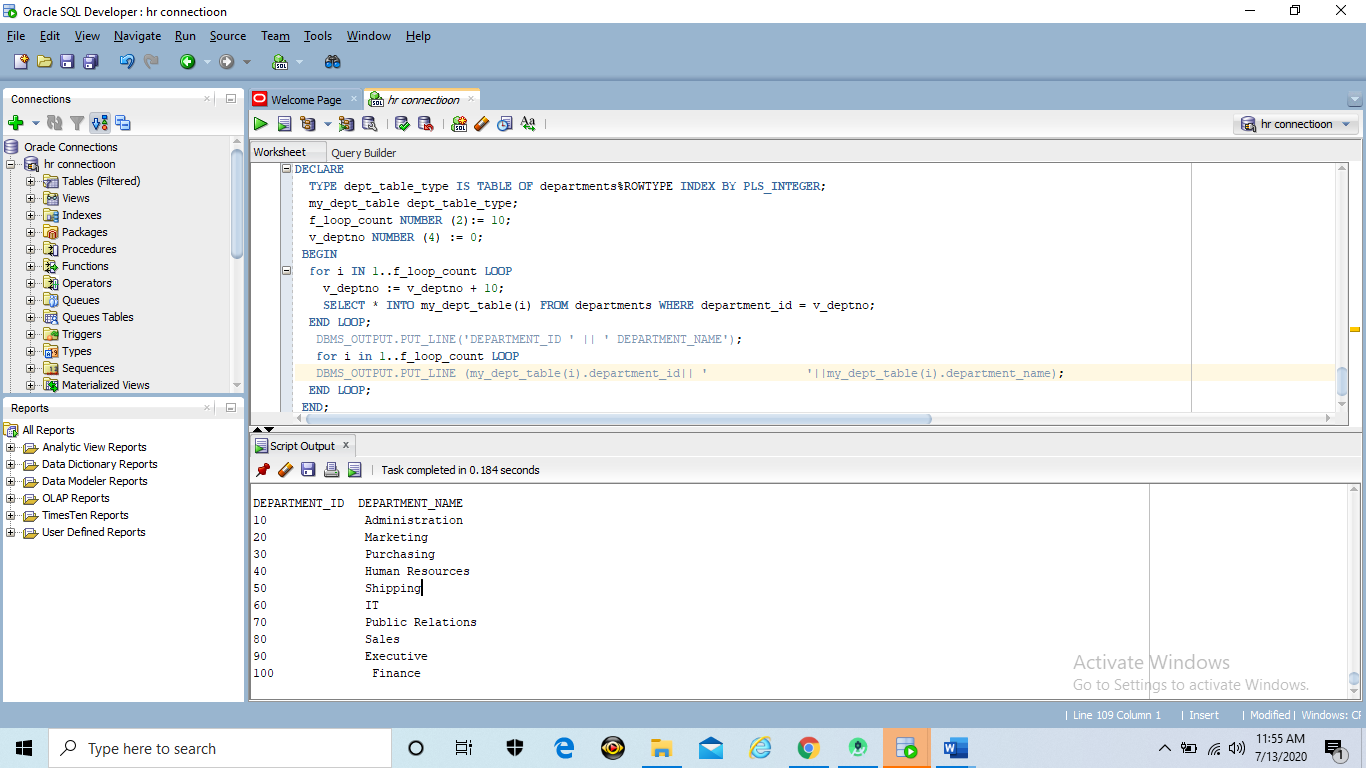
DBMS\_OUTPUT.PUT\_LINE('DEPARTMENT\_ID ' || ' DEPARTMENT\_NAME');

for i in 1..f\_loop\_count LOOP

DBMS\_OUTPUT.PUT\_LINE (my\_dept\_table(i).department\_id|| ' '||my\_dept\_table(i).department\_name);

END LOOP;

END;



1. Using another loop, retrieve the department names from the associative array and display them.

The output will look like below table:



SET SERVEROUTPUT ON

DECLARE

TYPE dept\_table\_type IS TABLE OF hr.departments.department\_name%TYPE INDEX BY PLS\_INTEGER;

my\_dept\_table dept\_table\_type;

f\_loop\_count NUMBER := 10;

v\_deptno NUMBER := 0;

BEGIN

for i IN 1..f\_loop\_count LOOP

v\_deptno := v\_deptno + 10;

SELECT department\_name INTO my\_dept\_table(i) FROM hr.departments WHERE department\_id = v\_deptno;

END LOOP;

for i in 1..f\_loop\_count LOOP

DBMS\_OUTPUT.PUT\_LINE(my\_dept\_table(i));

END LOOP;

END;

