ORACLE LAB

BCA-DS-552

Manav Rachna International Institute of Research and Studies

School of Computer Applications

Department of Computer Applications

Submitted By	
Student Name	Dheeraj Jamwal
Roll No	22/FCA/BCA(AIML)/011
Programme	Bachelor of Computer Applications
Semester	5th Semester
Section	E
Department	Computer Applications
Batch	2022-25
Submitted To	
Faculty Name	Ms. Iram Fatima

MANAV RACHNA
॥vidyanatariksha॥
Manav Rachna International Institute of Research and Studies
Deemed to be University under section 3 of the UGC Act, 1956

SCHOOL OF COMPUTER APPLICATIONS

EXERCISE 1

AIM: Create the following table.

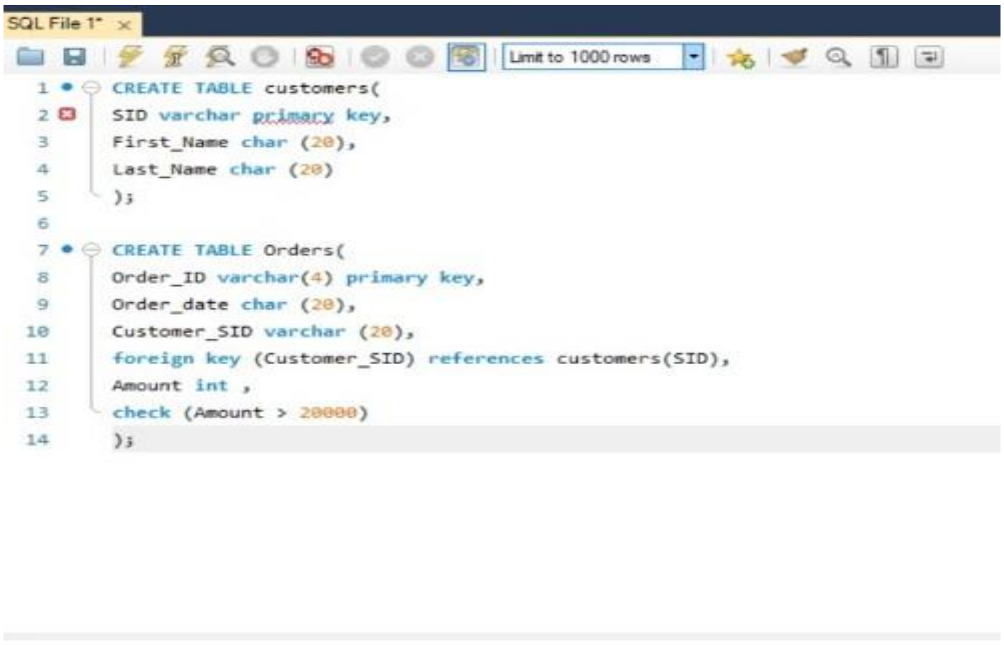
Customer

Column_name	**Data type**	**Size**	**Constraint**
SID	Varchar2	4	Primary Key
First_Name	Char	20	
Last_name	Char	20	

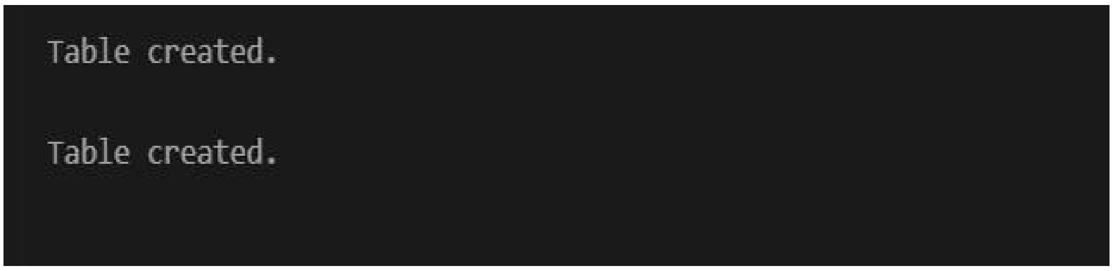
Orders

Column_name	**Data type**	**Size**	**Constraint**
Order_ID	Varchar2	4	Primary Key
Order_date	Char	20	
Customer_SID	Varchar2	20	Foreign Key
Amount	Number		Check > 20000

Output:



```
CREATE TABLE customers(
SID varchar primary key,
First_Name char (20),
Last_Name char (20)
);

CREATE TABLE Orders(
Order_ID varchar(4) primary key,
Order_date char (20),
Customer_SID varchar (20),
foreign key (Customer_SID) references customers(SID),
Amount int ,
check (Amount > 20000)
);
```

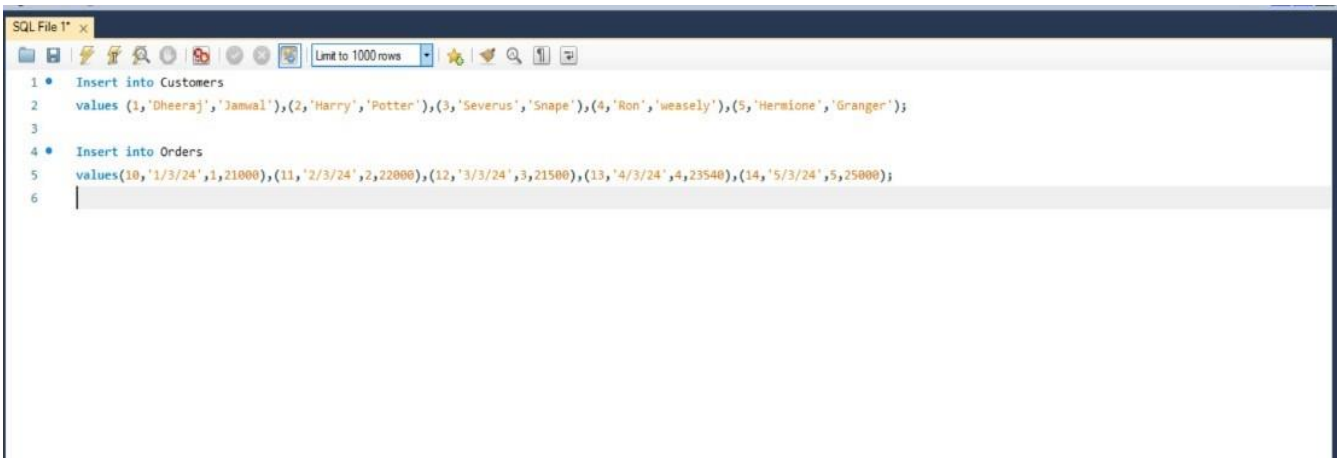


```
Table created.

Table created.
```

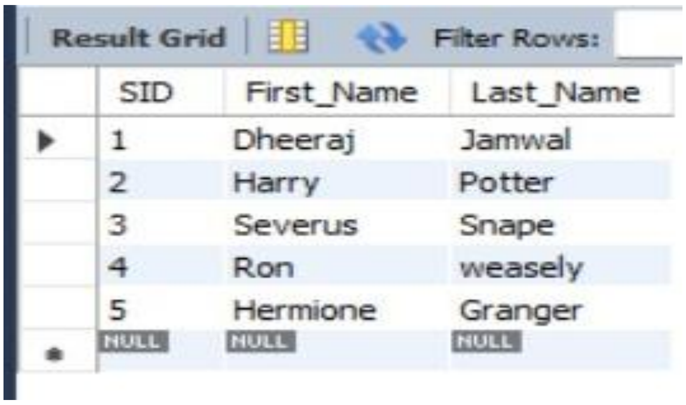
EXERCISE 2

AIM: Insert 5 records for each table.

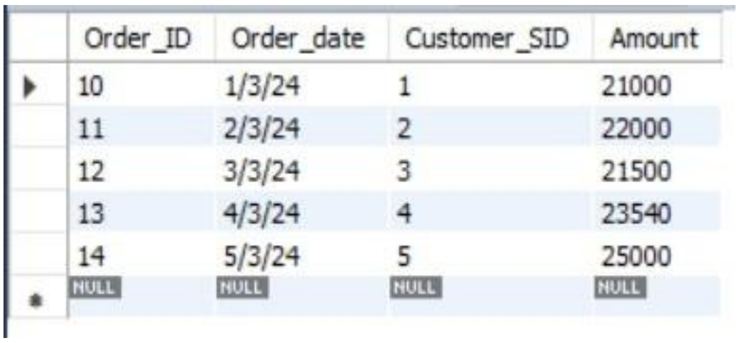
Output:



```
Insert into Customers
values (1,'Dheeraj','Jamwal'),(2,'Harry','Potter'),(3,'Severus','Snape'),(4,'Ron','weasely'),(5,'Hermione','Granger');

Insert into Orders
values(10,'1/3/24',1,21000),(11,'2/3/24',2,22000),(12,'3/3/24',3,21500),(13,'4/3/24',4,23540),(14,'5/3/24',5,25000);
```



SID	First_Name	Last_Name
1	Dheeraj	Jamwal
2	Harry	Potter
3	Severus	Snape
4	Ron	weasely
5	Hermione	Granger
NULL	NULL	NULL

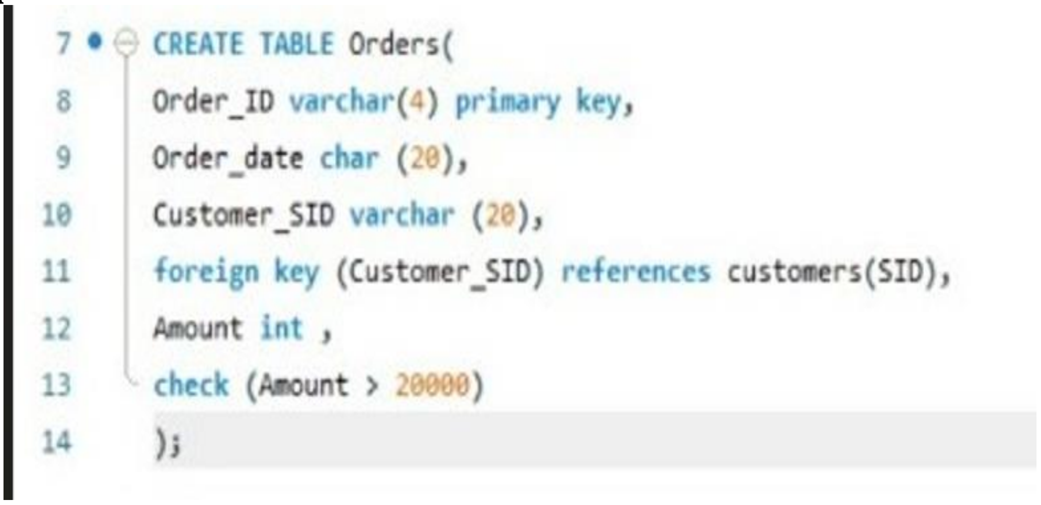


Order_ID	Order_date	Customer_SID	Amount
10	1/3/24	1	21000
11	2/3/24	2	22000
12	3/3/24	3	21500
13	4/3/24	4	23540
14	5/3/24	5	25000
NULL	NULL	NULL	NULL

EXERCISE 3

AIM: Customer SID column in the ORDERS table is a foreign key pointing to the SIDcolumn in the CUSTOMER table.

Output:

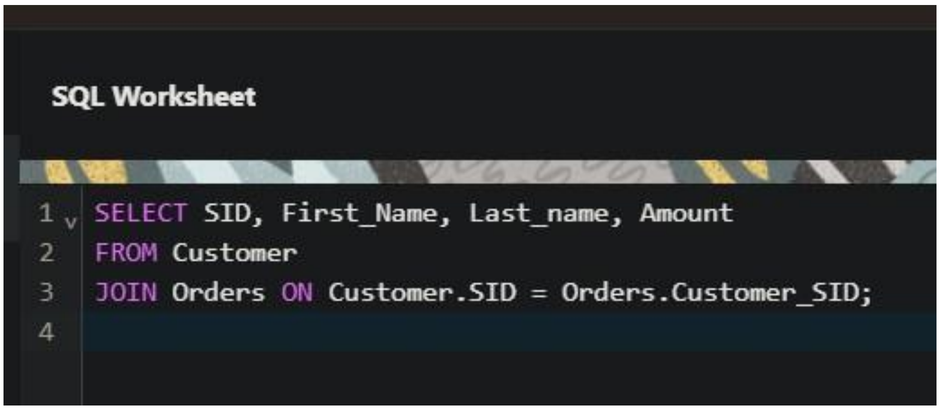


```
CREATE TABLE Orders(
Order_ID varchar(4) primary key,
Order_date char (20),
Customer_SID varchar (20),
foreign key (Customer_SID) references customers(SID),
Amount int ,
check (Amount > 20000)
);
```

EXERCISE 4

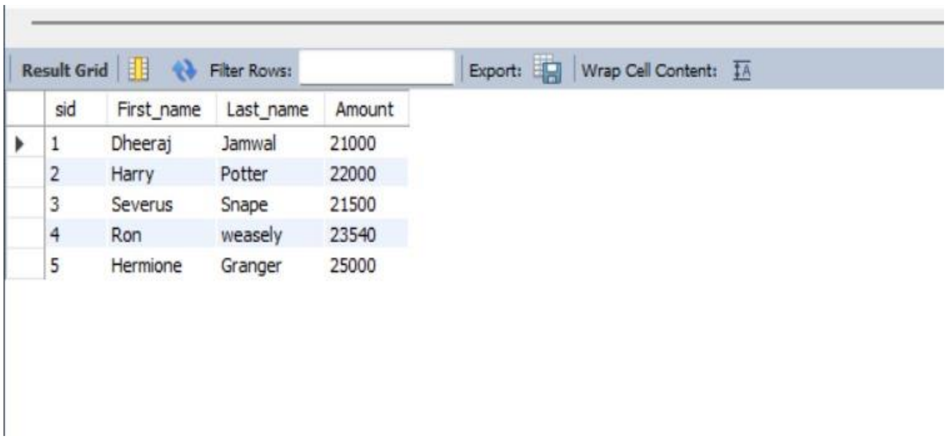
AIM: List the details of the customers along with the amount.

Output:



SQL Worksheet

```
SELECT SID, First_Name, Last_name, Amount
FROM Customer
JOIN Orders ON Customer.SID = Orders.Customer_SID;
```

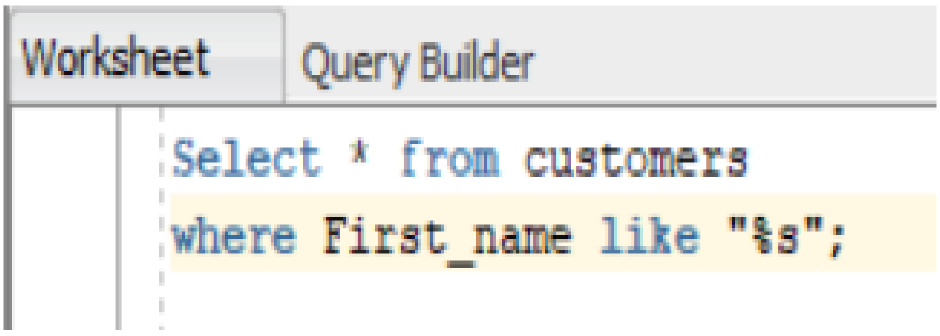


sid	First_name	Last_name	Amount
1	Dheeraj	Jamwal	21000
2	Harry	Potter	22000
3	Severus	Snape	21500
4	Ron	weasely	23540
5	Hermione	Granger	25000

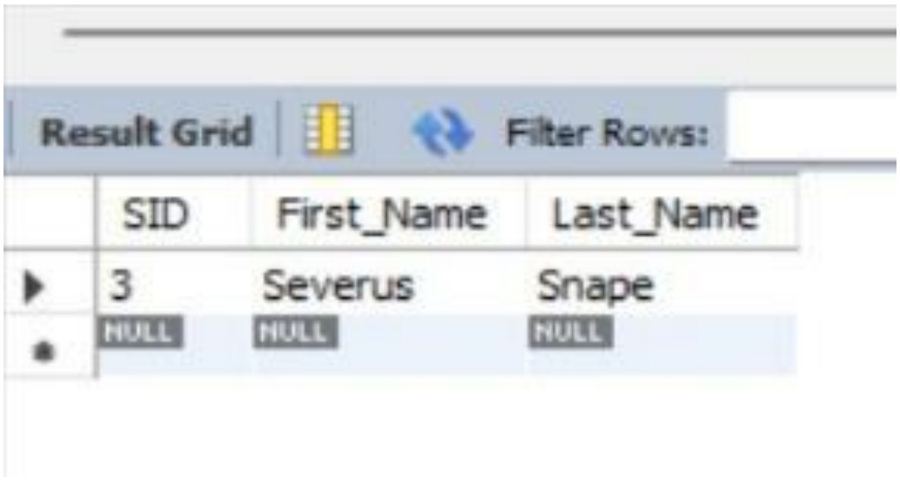
EXERCISE 5

AIM: List the customers whose names end with “s”.

Output:



```
Select * from customers
where First_name like "%s";
```

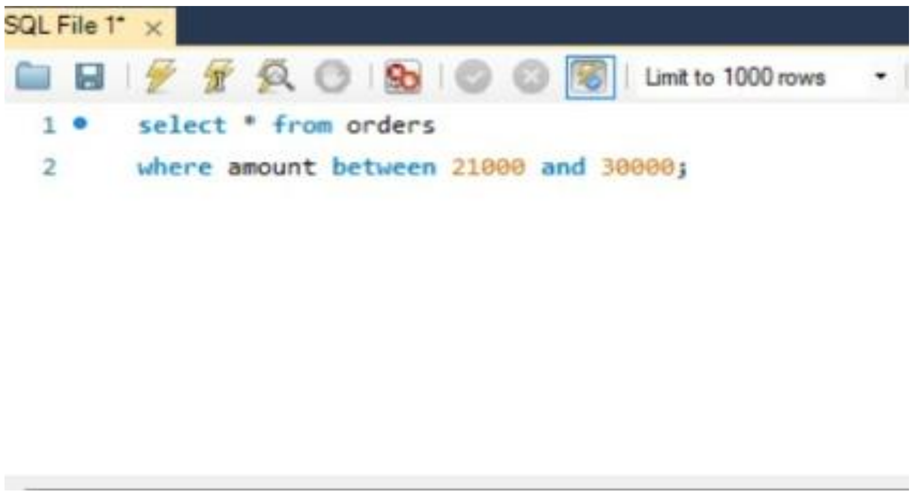


SID	First_Name	Last_Name
3	Severus	Snape
NULL	NULL	NULL

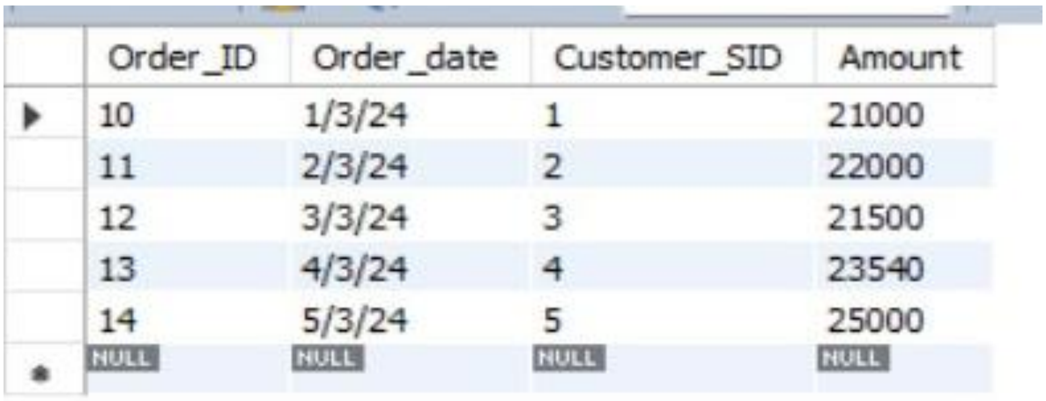
EXERCISE 6

AIM: List the orders where amount is between 21000 and 30000

Output:



```sql
select * from orders
where amount between 21000 and 30000;
```

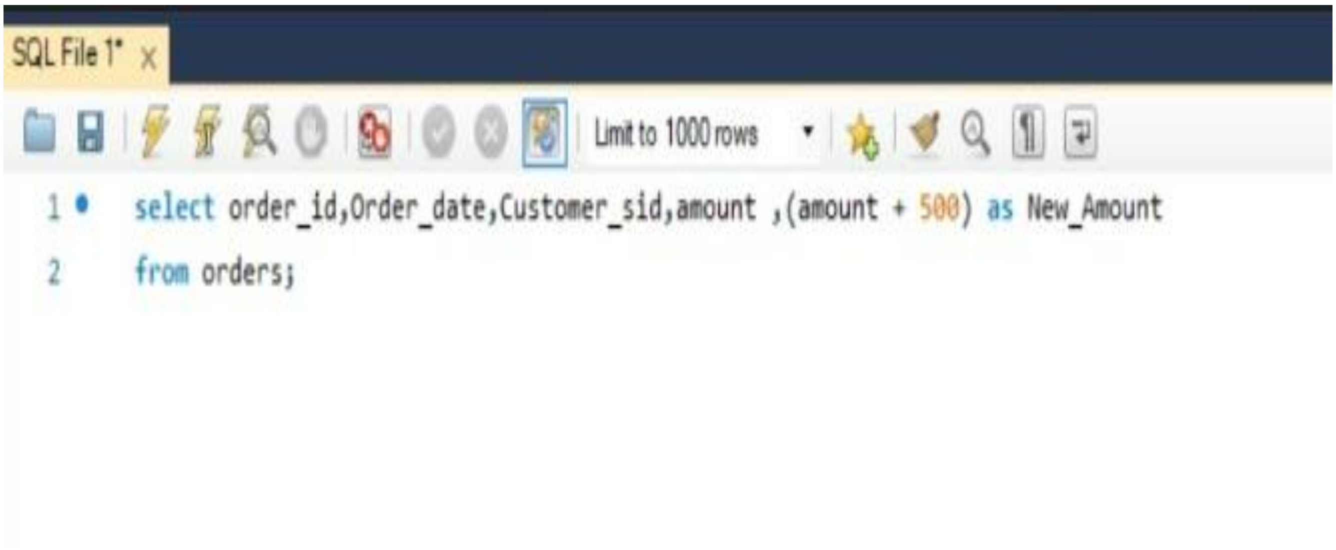


	Order_ID	Order_date	Customer_SID	Amount
▶	10	1/3/24	1	21000
	11	2/3/24	2	22000
	12	3/3/24	3	21500
	13	4/3/24	4	23540
	14	5/3/24	5	25000
*	NULL	NULL	NULL	NULL

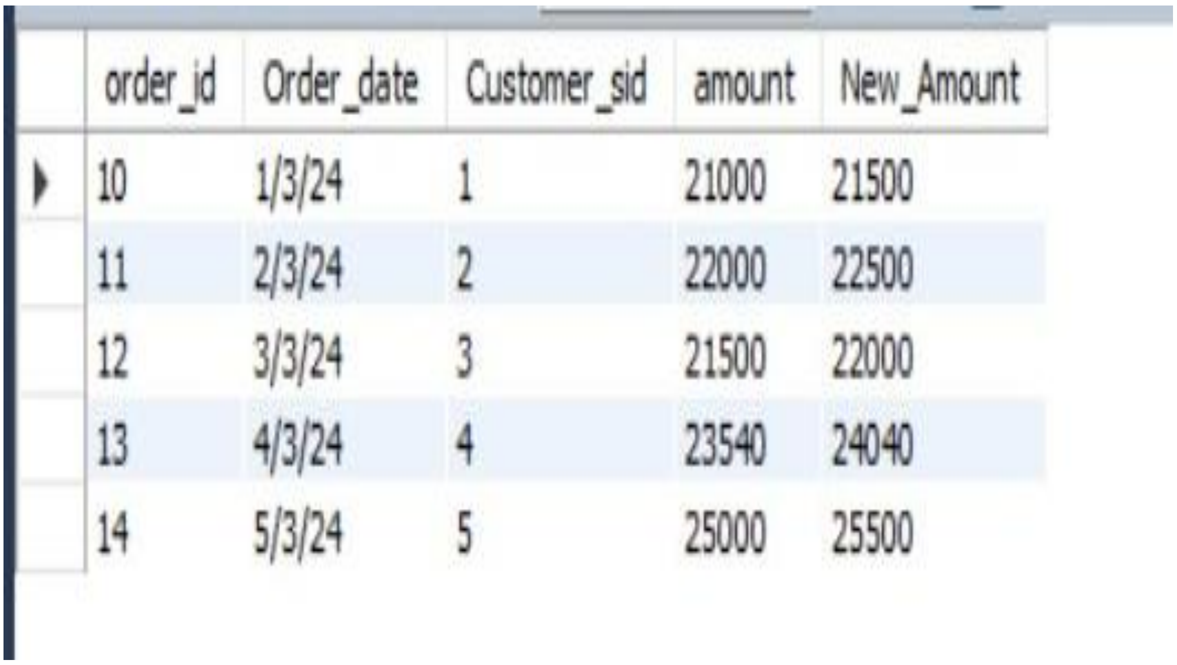
EXERCISE 7

AIM: List the orders where amount is increased by 500 and replace with name “newamount”.

Output:



```sql
select order_id,Order_date,Customer_sid,amount ,(amount + 500) as New_Amount
from orders;
```

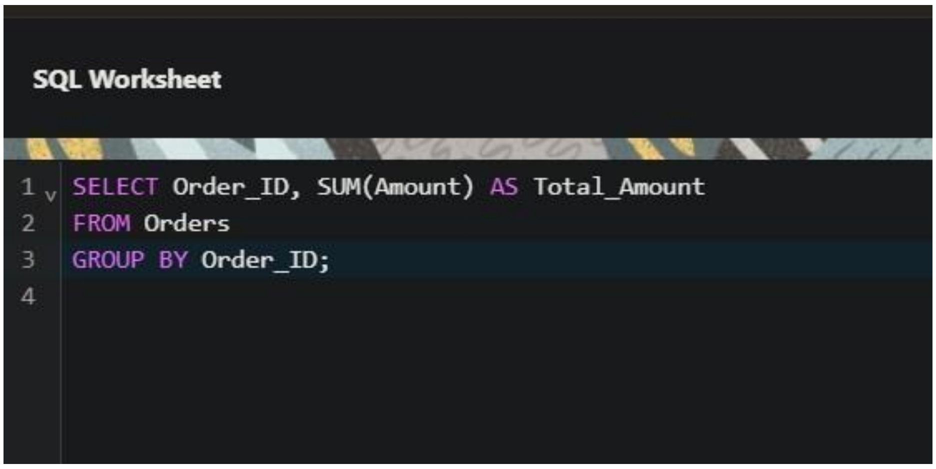


order_id	Order_date	Customer_sid	amount	New_Amount
10	1/3/24	1	21000	21500
11	2/3/24	2	22000	22500
12	3/3/24	3	21500	22000
13	4/3/24	4	23540	24040
14	5/3/24	5	25000	25500

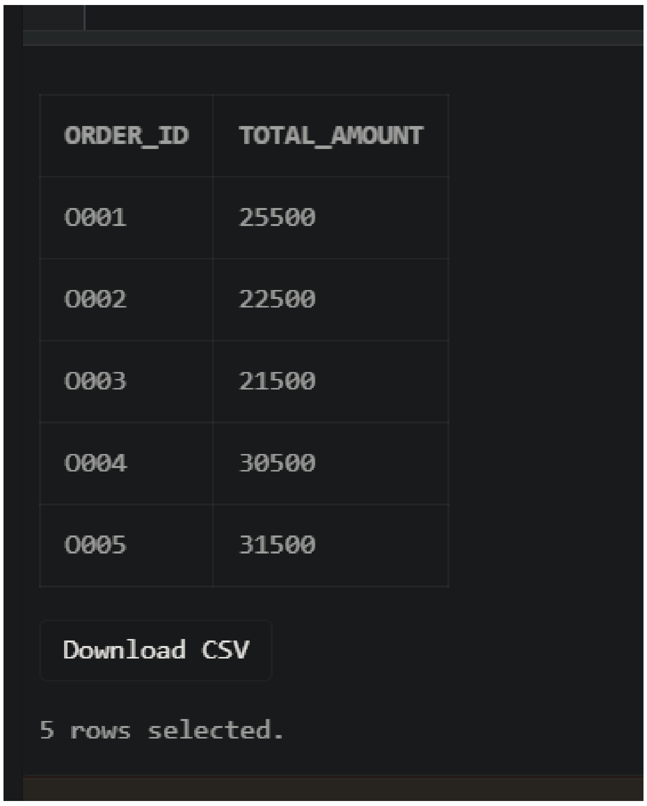
EXERCISE 8

AIM: Display the order_id and total amount of orders.

Output:



```sql
SELECT Order_ID, SUM(Amount) AS Total_Amount
FROM Orders
GROUP BY Order_ID;
```



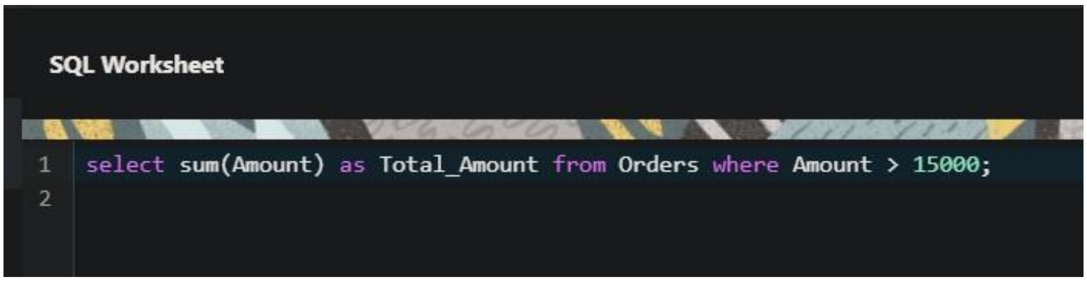
ORDER_ID	TOTAL_AMOUNT
O001	25500
O002	22500
O003	21500
O004	30500
O005	31500

```
5 rows selected.
```

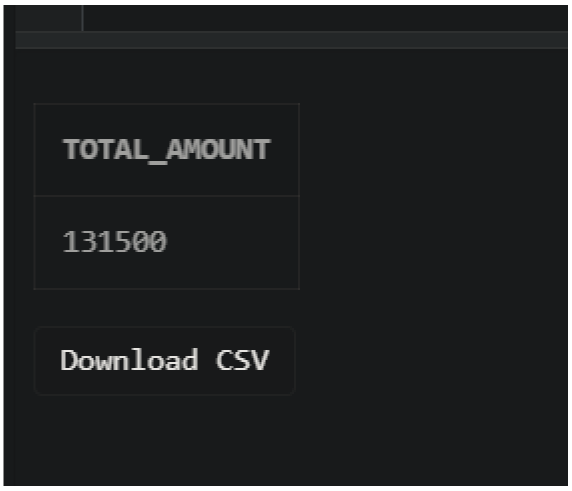
EXERCISE 9

AIM: Calculate the total amount of orders that has more than 15000.

Output:



```sql
select sum(Amount) as Total_Amount from Orders where Amount > 15000;
```



```
TOTAL_AMOUNT
131500
```

EXERCISE 10

AIM: Display all the string functions used in SQL.

Output:

```
SELECT

   LOWER('ORACLE') AS "Lowercase",        -- Converts string to lowercase

   UPPER('oracle') AS "Uppercase",        -- Converts string to uppercase

   SUBSTR('ORACLE', 2, 3) AS "Substring", -- Extracts substring

   LENGTH('ORACLE') AS "Length”,         -- Returns length of string

   INSTR('ORACLE', 'A') AS "Position", -- Returns position of a character

   LPAD('123', 5, '0') AS "Left Padding", -- Pads a string on the left

   RPAD('123', 5, '0') AS "Right Padding",-- Pads a string on the right

   TRIM('O' FROM 'ORACLE') AS "Trimmed"  -- Trims a specified character

FROM DUAL;
```

EXERCISE 11

AIM: Create the following tables.

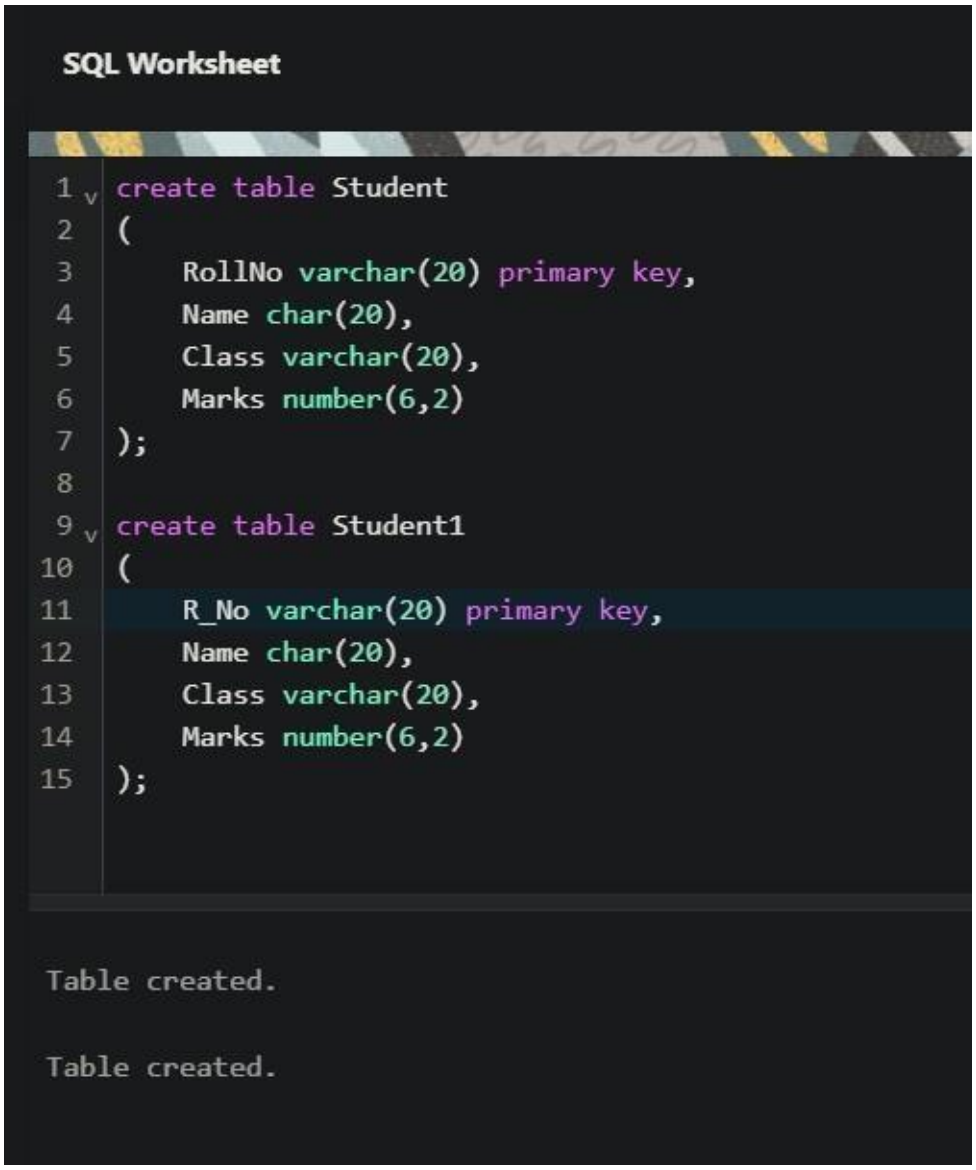
Student

Column_name	**Data type**	**Size**	**Constraint**
RollNo	Varchar2	20	Primary Key
Name	Char	20	
Class	Varchar2	20	
Marks	Number	6,2	

Student1

Column_name	**Data type**	**Size**	**Constraint**
R_No	Varchar2	20	Primary Key
Name	Char	20	
Class	Varchar2	20	
Marks	Number	6,2	

Output:

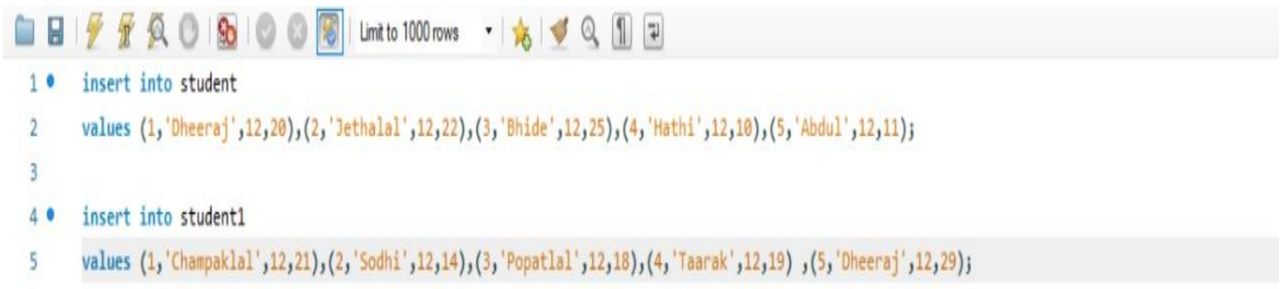


```
create table Student
(
    RollNo varchar(20) primary key,
    Name char(20),
    Class varchar(20),
    Marks number(6,2)
);

create table Student1
(
    R_No varchar(20) primary key,
    Name char(20),
    Class varchar(20),
    Marks number(6,2)
);
```

```
Table created.


Table created.
```

EXERCISE 12

AIM: Display all the contents of student and student1 using union clause.First insert 5 records in each table i.e. Student and Student1



```
insert into student
values (1,'Dheeraj',12,20),(2,'Jethalal',12,22),(3,'Bhide',12,25),(4,'Hathi',12,10),(5,'Abdul',12,11);

insert into student1
values (1,'Champaklal',12,21),(2,'Sodhi',12,14),(3,'Popatlal',12,18),(4,'Taarak',12,19) ,(5,'Dheeraj',12,29);
```

Output:

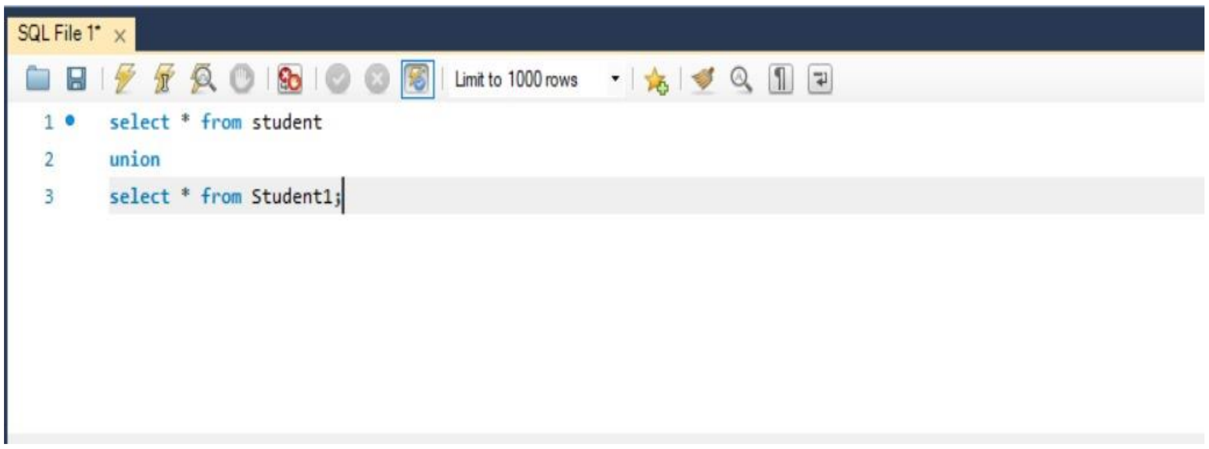
Student

RollNo	Name	class	marks
1	Dheeraj	12	20
2	Jethalal	12	22
3	Bhide	12	25
4	Hathi	12	10
5	Abdul	12	11

Student1

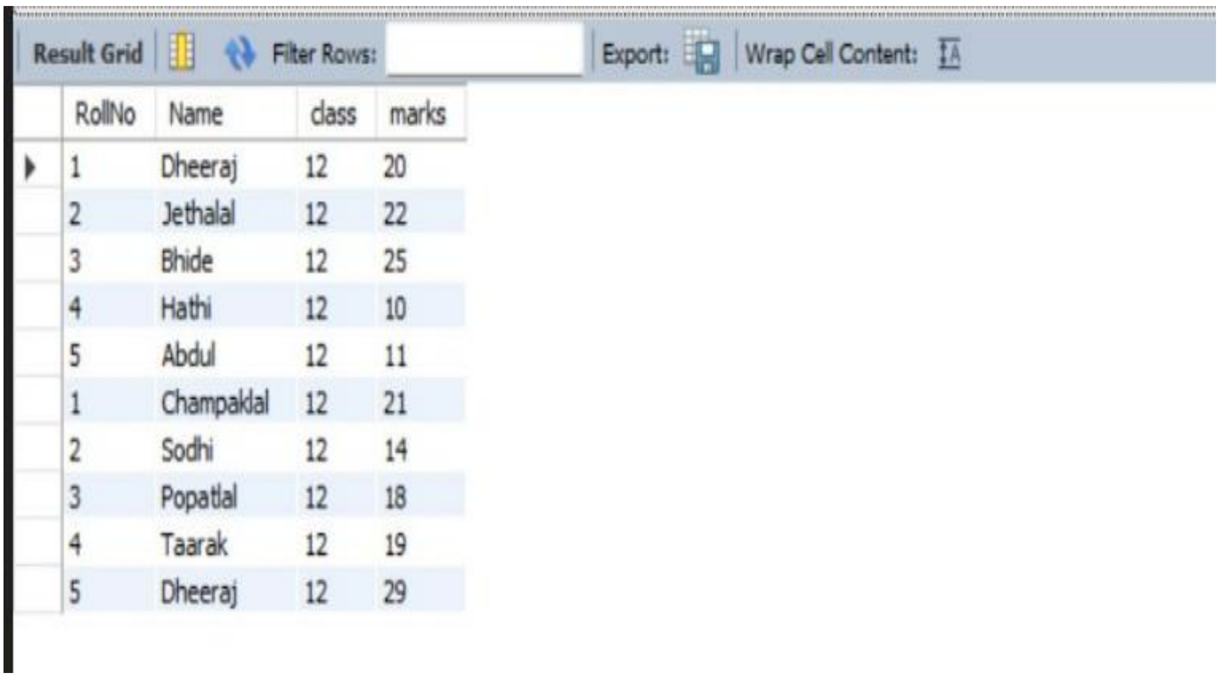
R_No	Name	class	marks
1	Champaklal	12	21
2	Sodhi	12	14
3	Popatlal	12	18
4	Taarak	12	19
5	Dheeraj	12	29

Now union:



```
select * from student
union
select * from Student1;
```

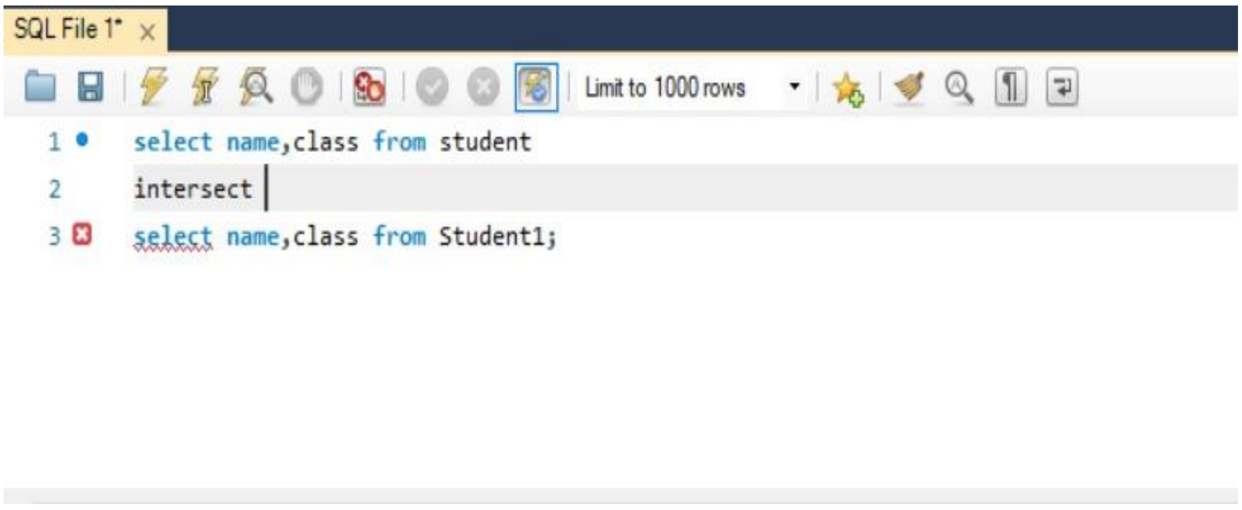
Output:



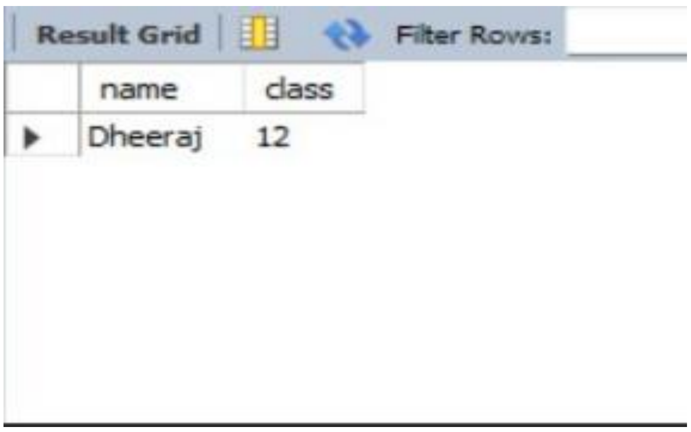
RollNo	Name	class	marks
1	Dheeraj	12	20
2	Jethalal	12	22
3	Bhide	12	25
4	Hathi	12	10
5	Abdul	12	11
1	Champaklal	12	21
2	Sodhi	12	14
3	Popatlal	12	18
4	Taarak	12	19
5	Dheeraj	12	29

EXERCISE 13

AIM: Find out the intersection of student and student1 tables.



```
select name,class from student
intersect
select name,class from Student1;
```

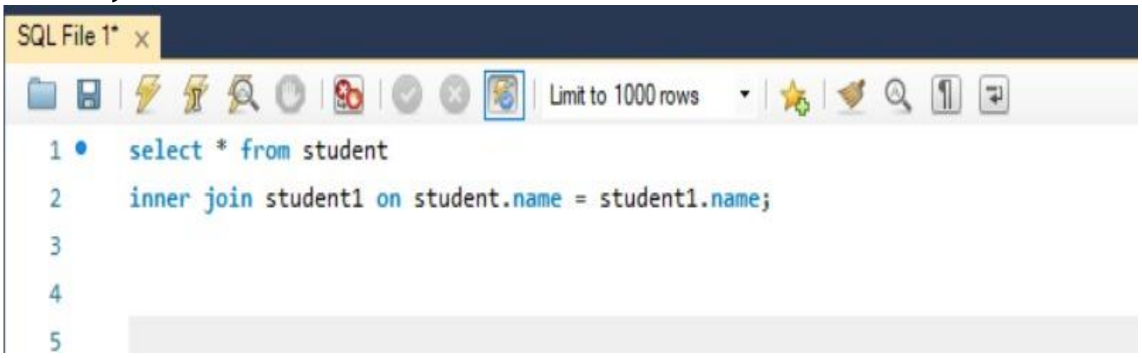


name	class
Dheeraj	12

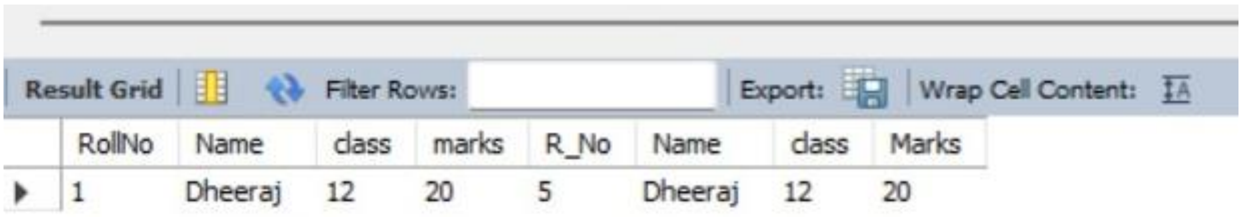
EXERCISE 14

AIM: Display the names of student and student1 tables using left, right, inner and full join.

INNER JOIN

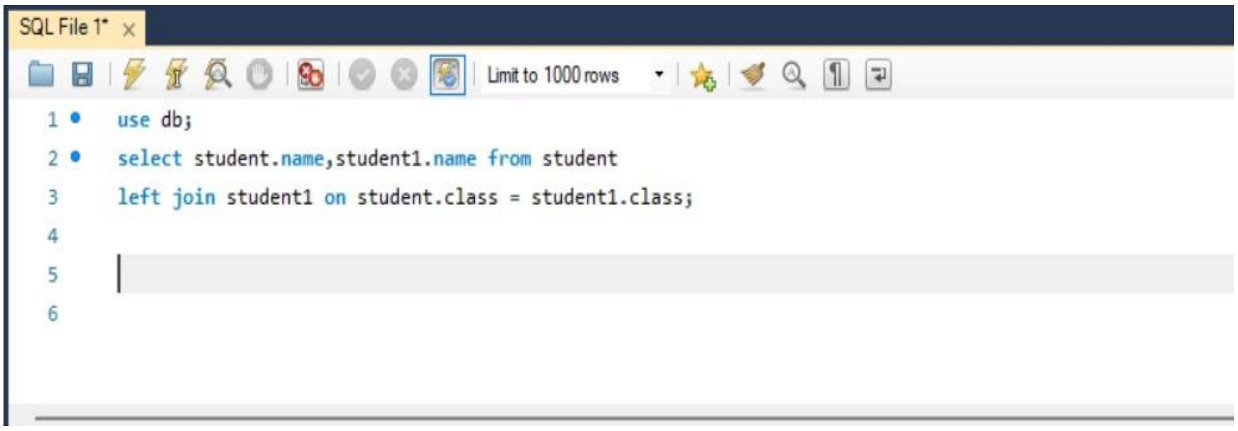


```
select * from student
inner join student1 on student.name = student1.name;
```

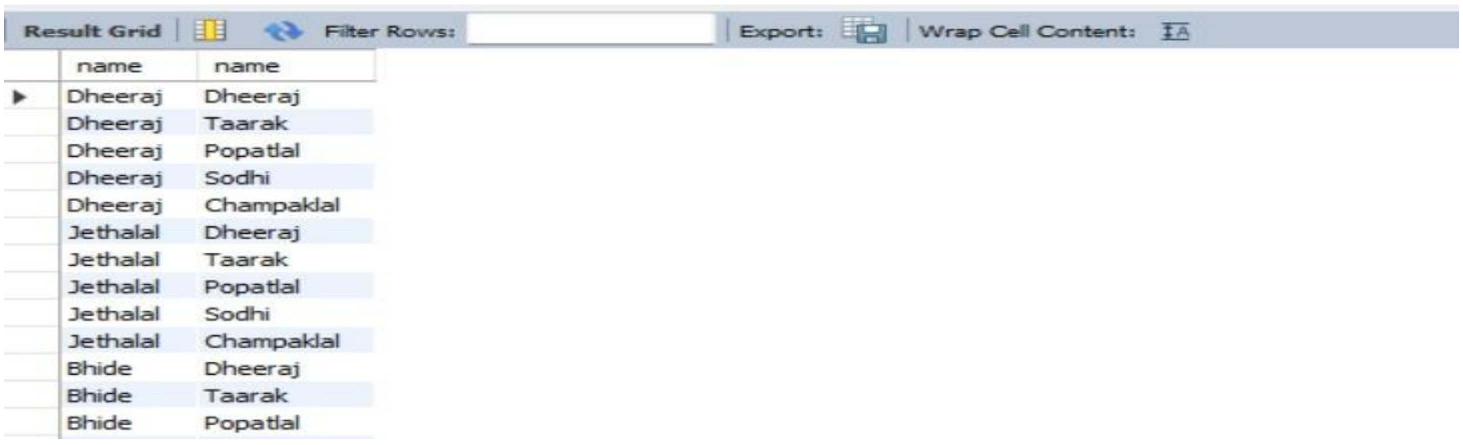


RollNo	Name	class	marks	R_No	Name	class	Marks
1	Dheeraj	12	20	5	Dheeraj	12	20

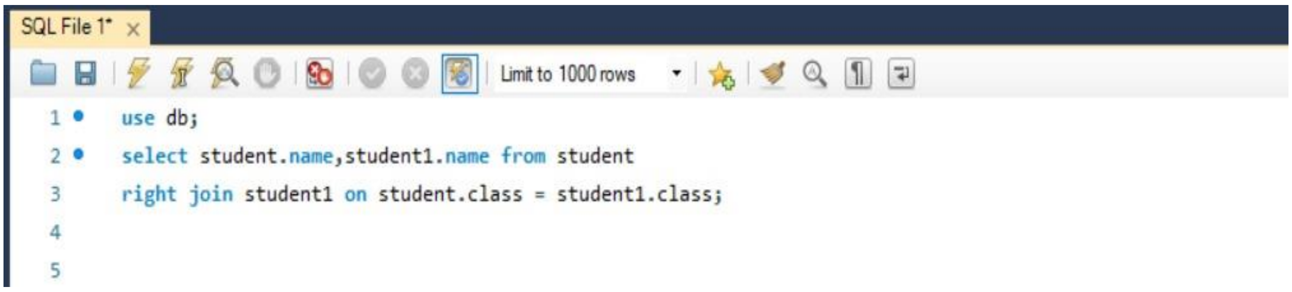
LEFT JOIN AND RIGHT JOIN



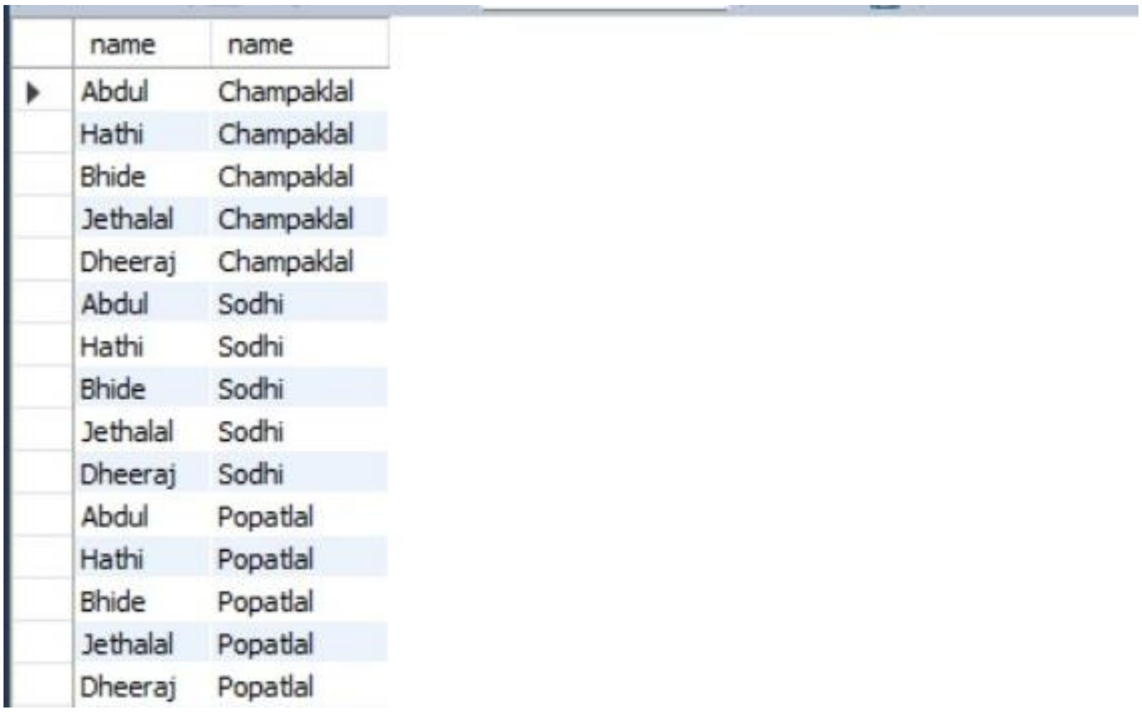
```
use db;
select student.name,student1.name from student
left join student1 on student.class = student1.class;
```



name	name
Dheeraj	Dheeraj
Dheeraj	Taarak
Dheeraj	Popatlal
Dheeraj	Sodhi
Dheeraj	Champaklal
Jethalal	Dheeraj
Jethalal	Taarak
Jethalal	Popatlal
Jethalal	Sodhi
Jethalal	Champaklal
Bhide	Dheeraj
Bhide	Taarak
Bhide	Popatlal

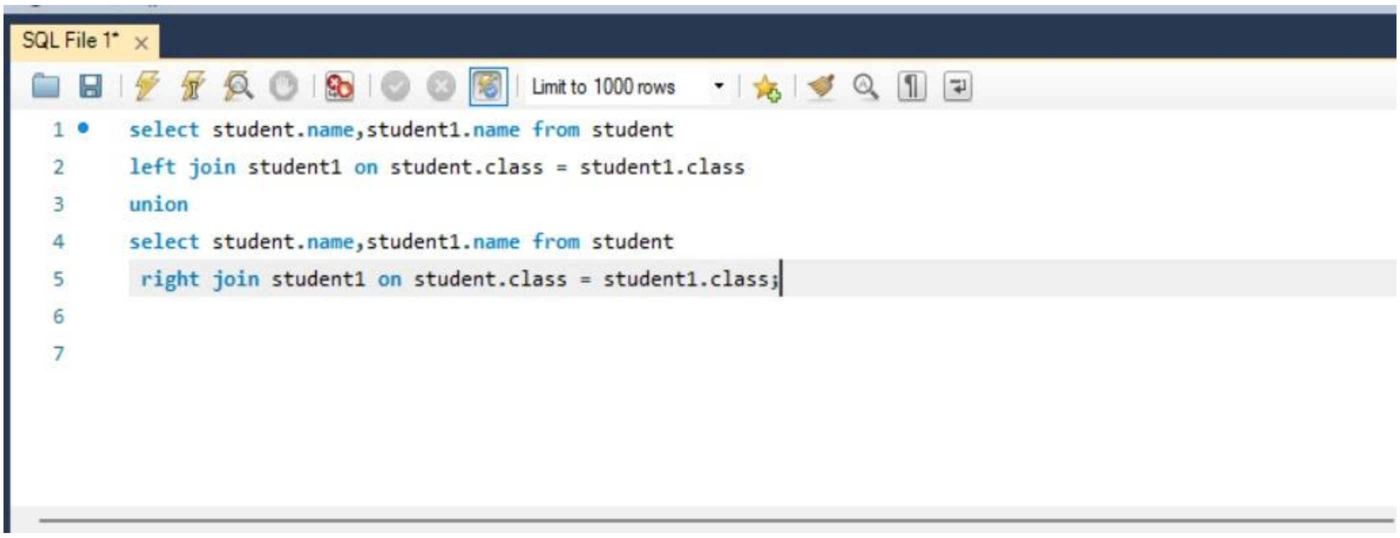


```
use db;
select student.name,student1.name from student
right join student1 on student.class = student1.class;
```

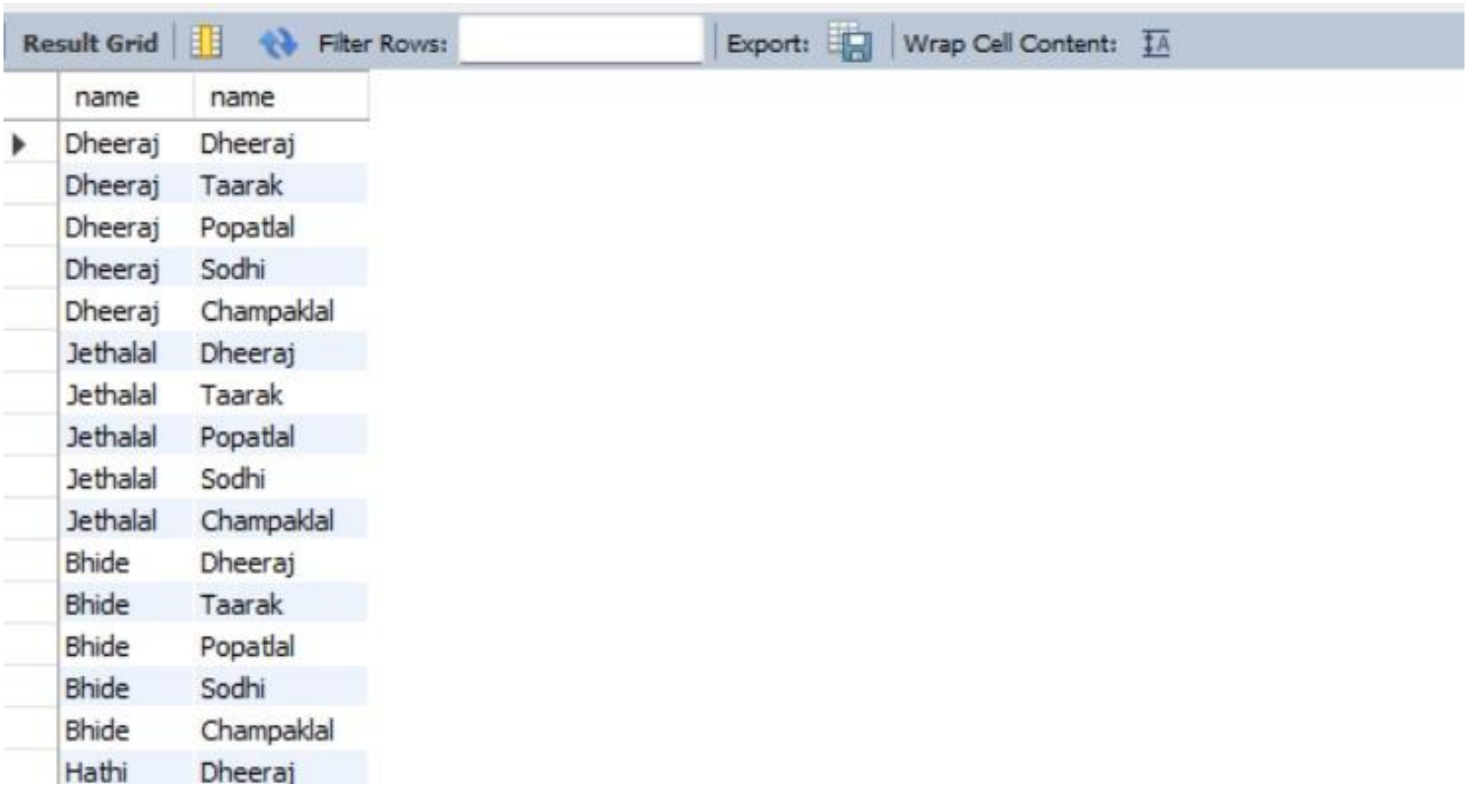


name	name
Abdul	Champaklal
Hathi	Champaklal
Bhide	Champaklal
Jethalal	Champaklal
Dheeraj	Champaklal
Abdul	Sodhi
Hathi	Sodhi
Bhide	Sodhi
Jethalal	Sodhi
Dheeraj	Sodhi
Abdul	Popatlal
Hathi	Popatlal
Bhide	Popatlal
Jethalal	Popatlal
Dheeraj	Popatlal

FULL JOIN



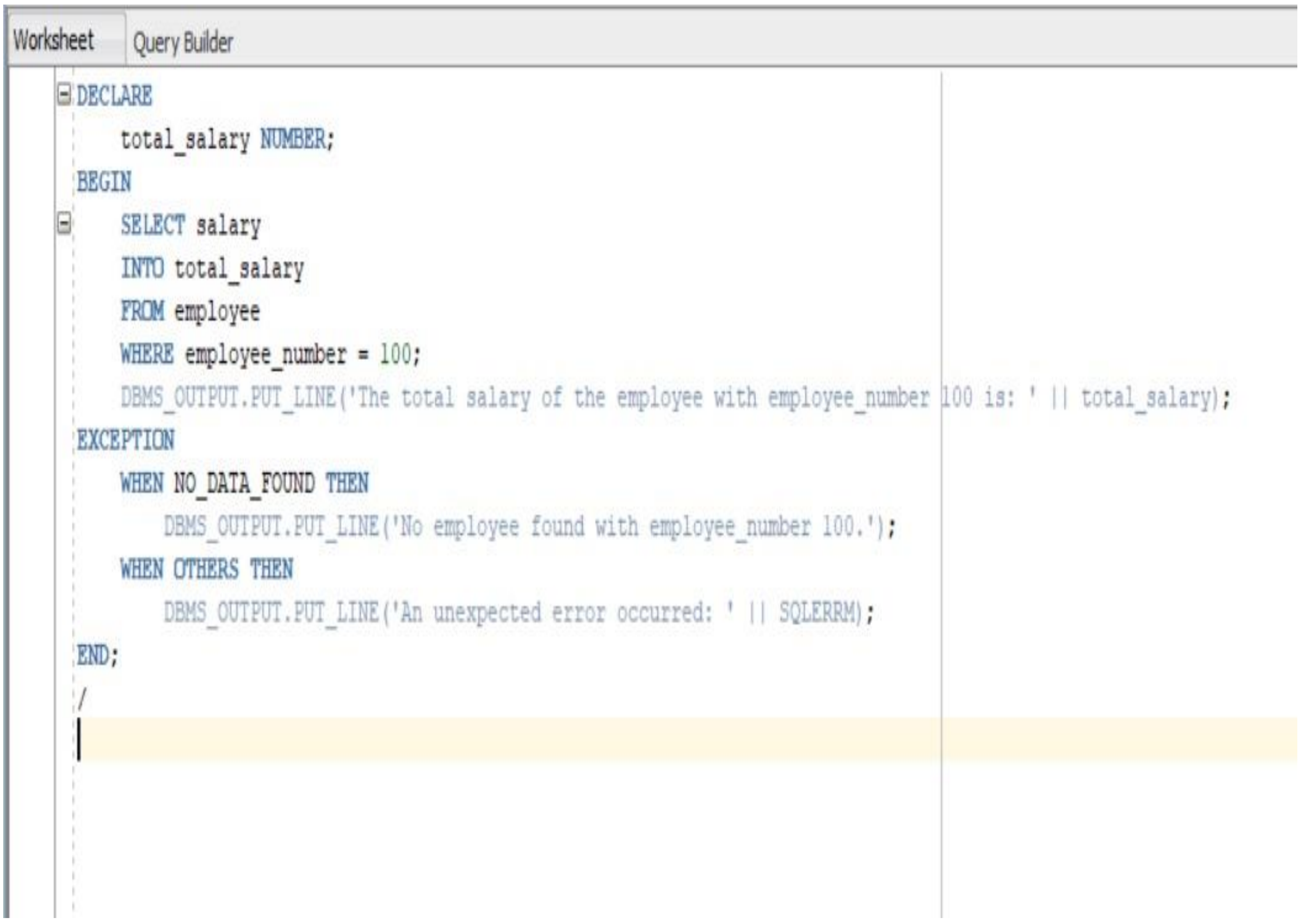
```
select student.name,student1.name from student
left join student1 on student.class = student1.class
union
select student.name,student1.name from student
 right join student1 on student.class = student1.class;
```



name	name
Dheeraj	Dheeraj
Dheeraj	Taarak
Dheeraj	Popatlal
Dheeraj	Sodhi
Dheeraj	Champaklal
Jethalal	Dheeraj
Jethalal	Taarak
Jethalal	Popatlal
Jethalal	Sodhi
Jethalal	Champaklal
Bhide	Dheeraj
Bhide	Taarak
Bhide	Popatlal
Bhide	Sodhi
Bhide	Champaklal
Hathi	Dheeraj

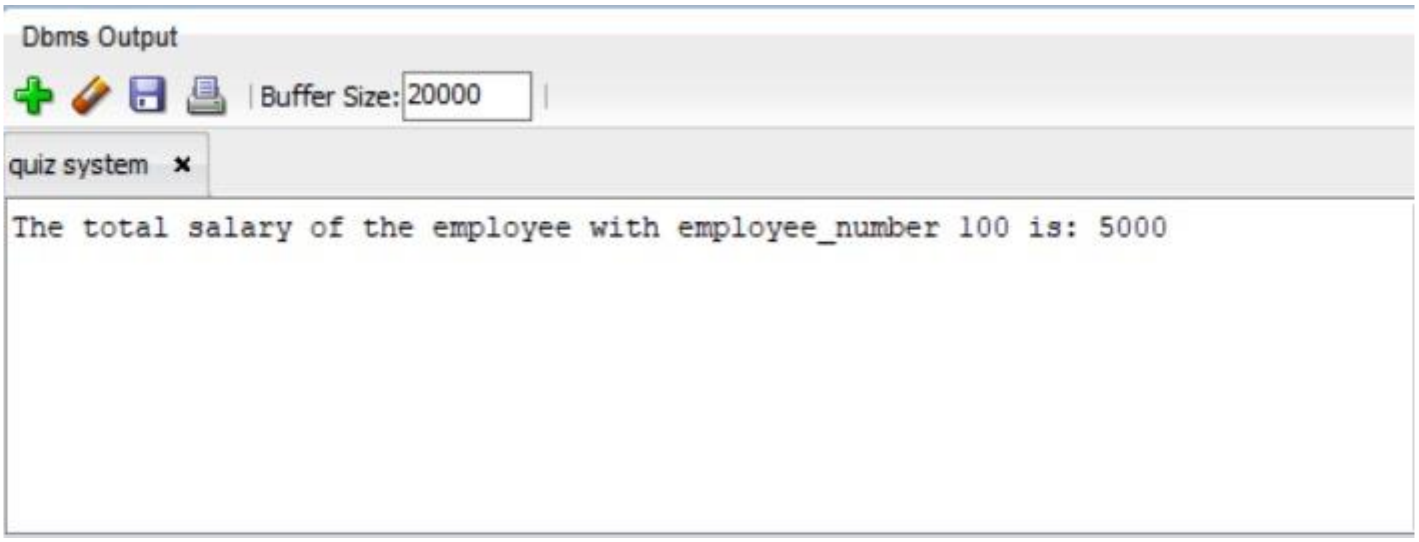
Exercise 15

AIM: To Write a PL/SQL block to calculate total salary of employee having employee number100.



```
DECLARE
    total_salary NUMBER;
BEGIN
    SELECT salary
    INTO total_salary
    FROM employee
    WHERE employee_number = 100;
    DBMS_OUTPUT.PUT_LINE('The total salary of the employee with employee_number 100 is: ' || total_salary);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No employee found with employee_number 100.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/
```

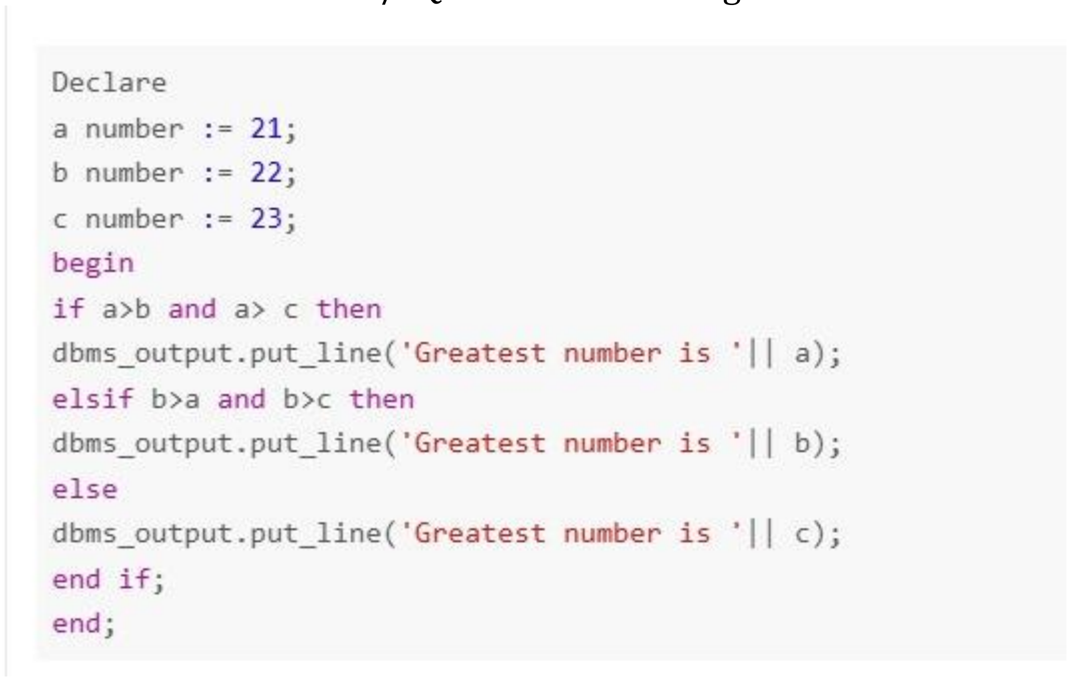
OUTPUT:



```
The total salary of the employee with employee_number 100 is: 5000
```

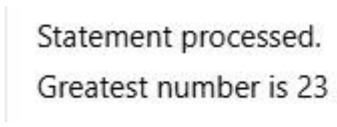
EXERCISE 16

AIM: To Write a PL/SQL code to find the greatest of three numbers.



```
Declare
a number := 21;
b number := 22;
c number := 23;
begin
if a>b and a> c then
dbms_output.put_line('Greatest number is '|| a);
elsif b>a and b>c then
dbms_output.put_line('Greatest number is '|| b);
else
dbms_output.put_line('Greatest number is '|| c);
end if;
end;
```

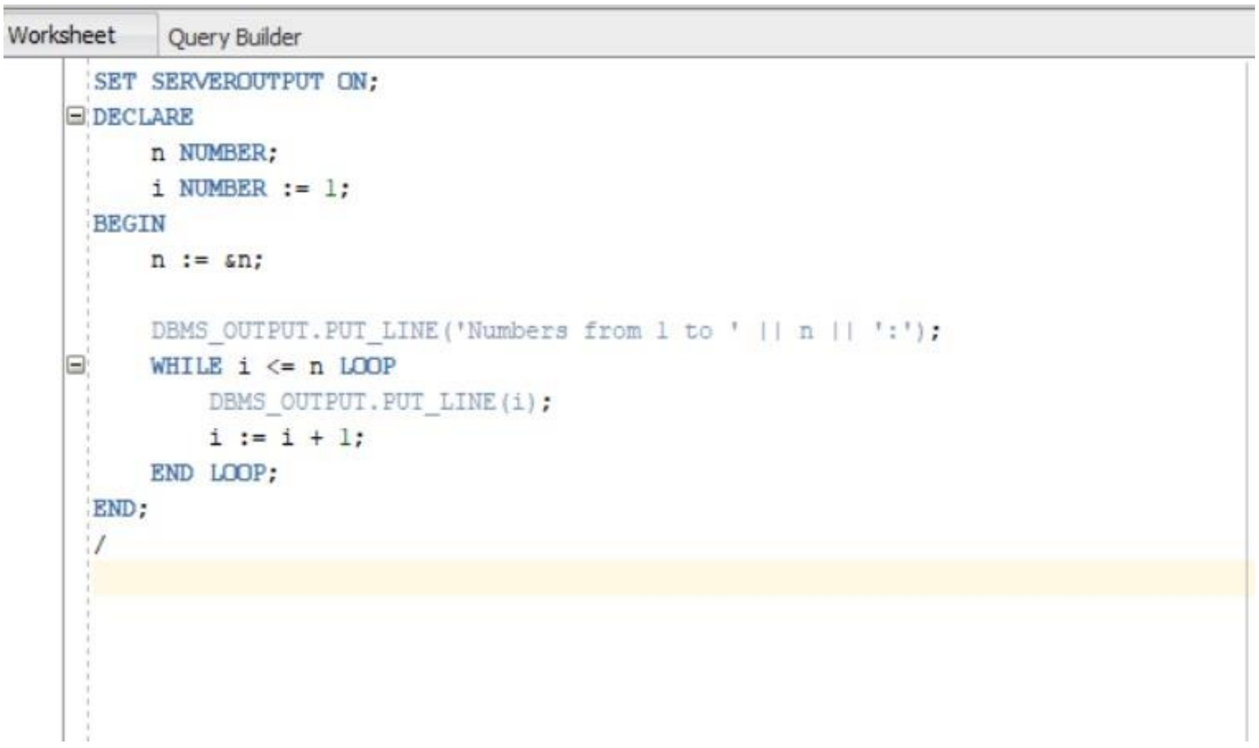
OUTPUT:



```
Statement processed.
Greatest number is 23
```

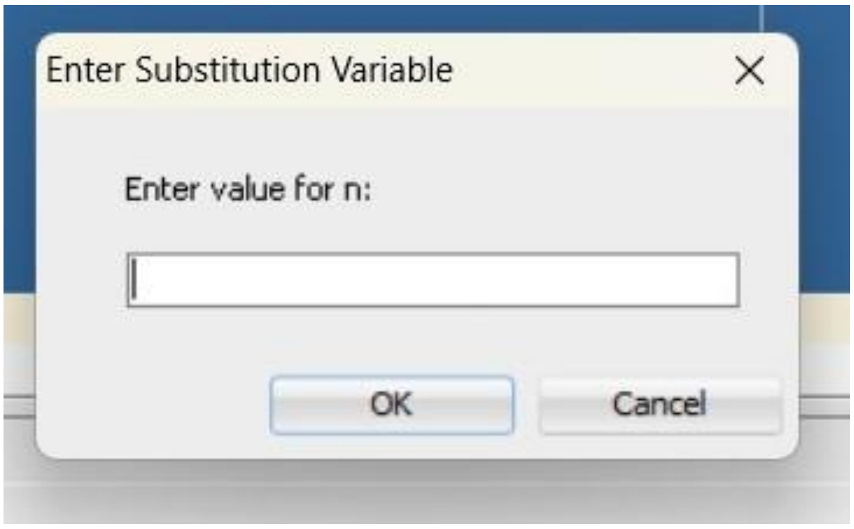
EXERCISE 17

AIM: To Write a PL/SQL code to print the numbers from 1 to n.



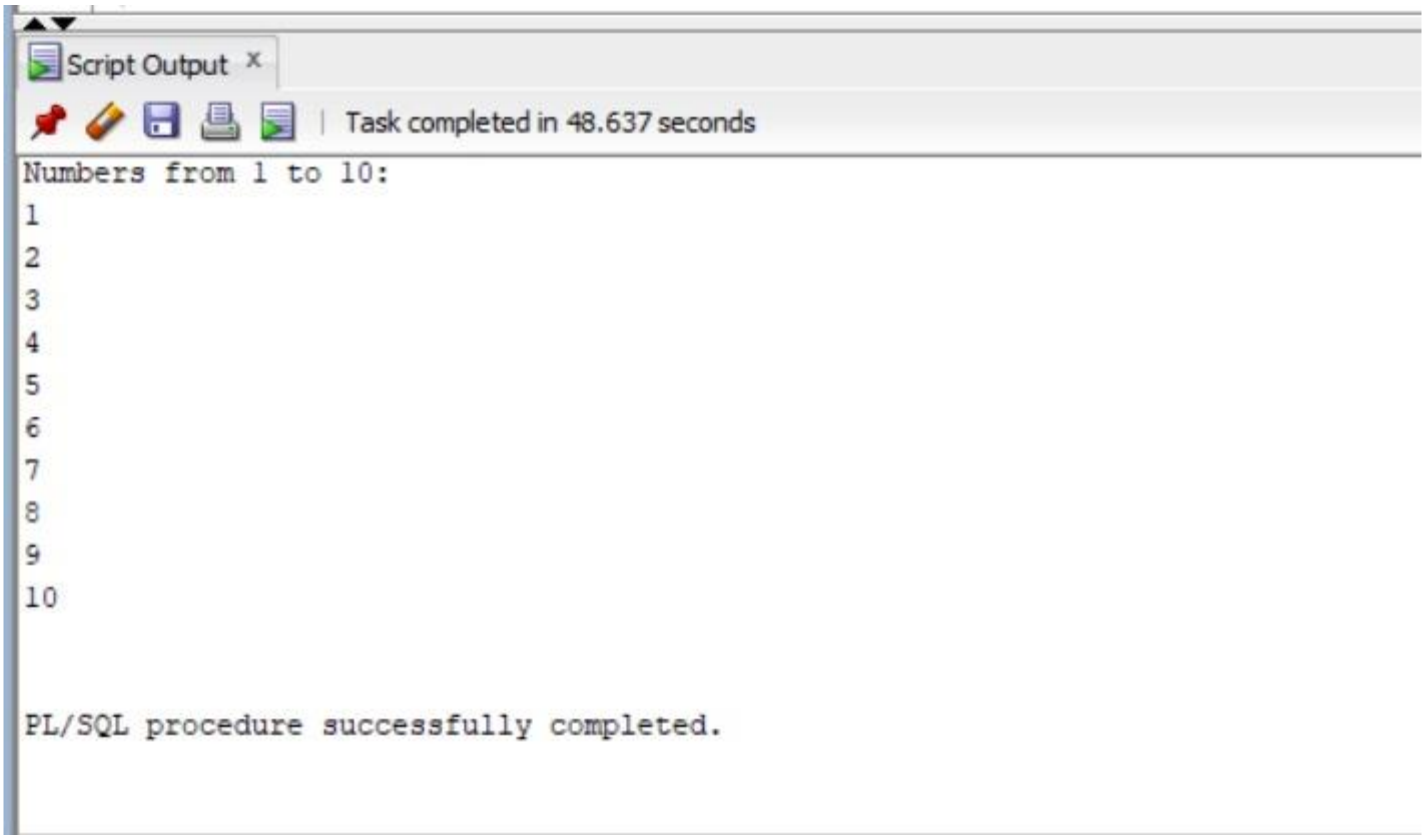
```
SET SERVEROUTPUT ON;
DECLARE
    n NUMBER;
    i NUMBER := 1;
BEGIN
    n := &n;

    DBMS_OUTPUT.PUT_LINE('Numbers from 1 to ' || n || ':');
    WHILE i <= n LOOP
        DBMS_OUTPUT.PUT_LINE(i);
        i := i + 1;
    END LOOP;
END;
/
```

OUTPUT:



Enter Substitution Variable

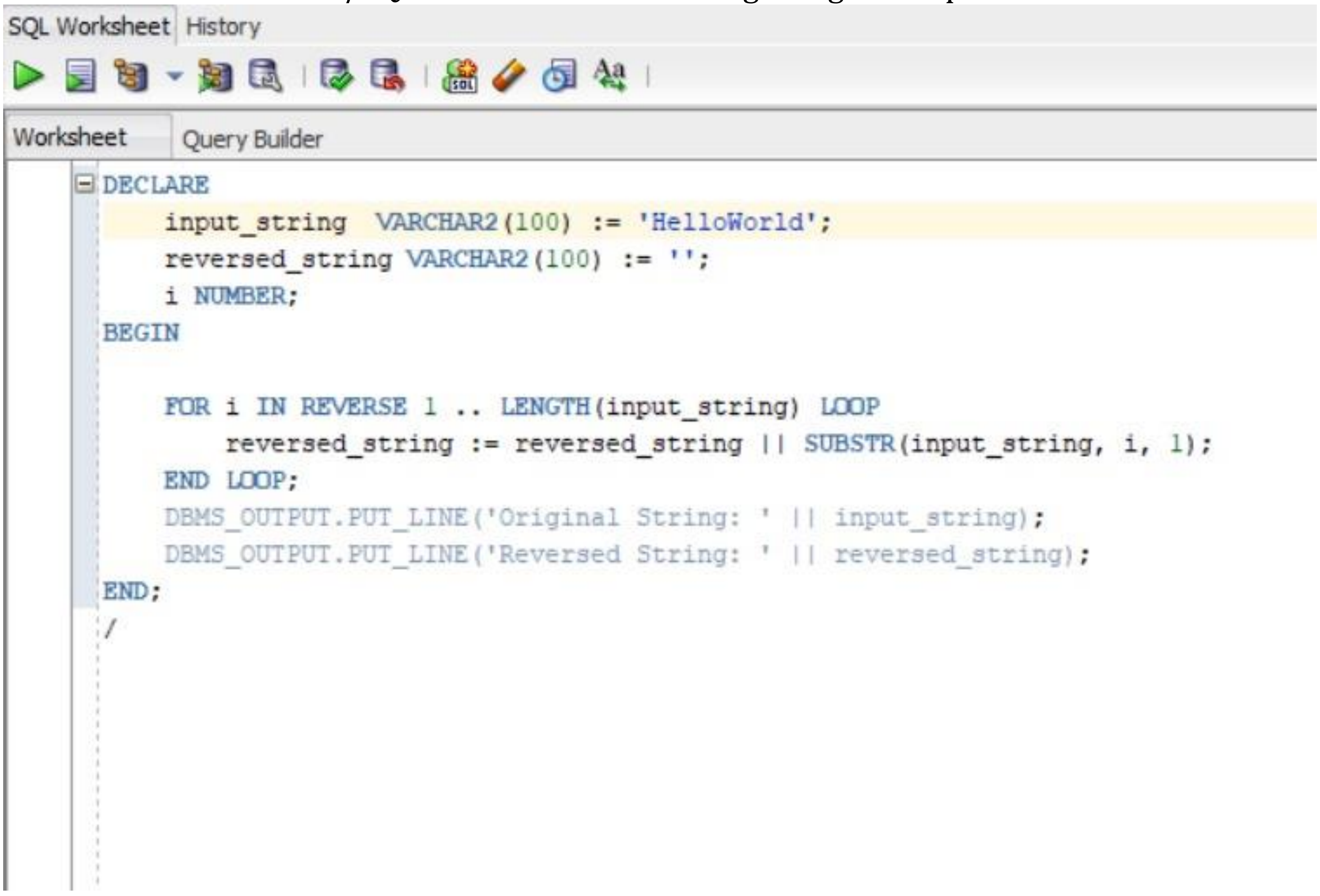
Enter value for n:



```
Numbers from 1 to 10:
1
2
3
4
5
6
7
8
9
10


PL/SQL procedure successfully completed.
```

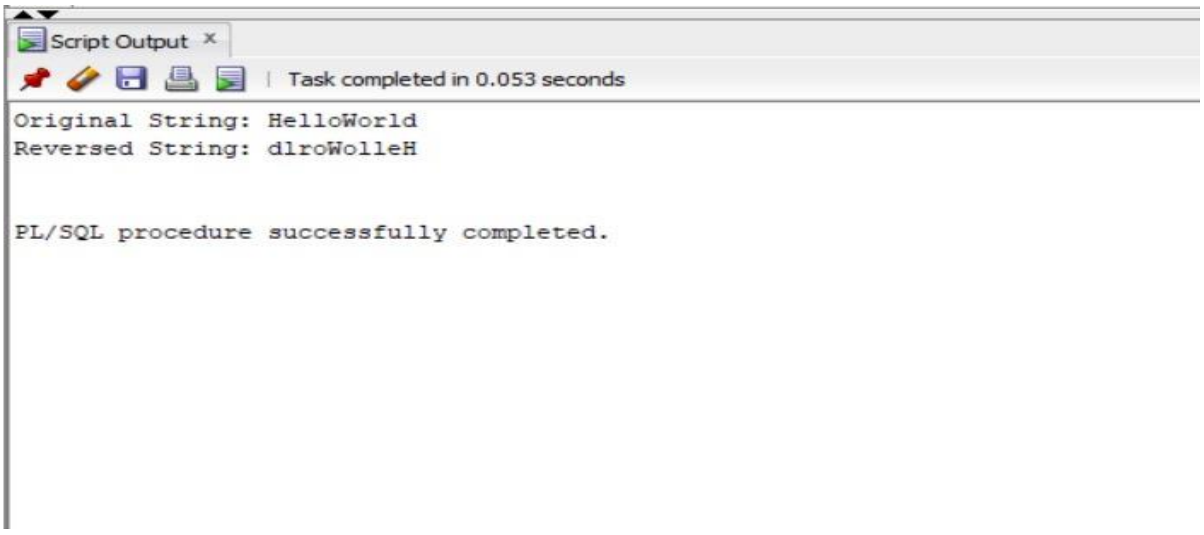
EXERCISE 18

AIM: To Write a PL/SQL code to reverse a string using for loop.



```
DECLARE
    input_string  VARCHAR2(100) := 'HelloWorld';
    reversed_string VARCHAR2(100) := '';
    i NUMBER;
BEGIN

    FOR i IN REVERSE 1 .. LENGTH(input_string) LOOP
        reversed_string := reversed_string || SUBSTR(input_string, i, 1);
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('Original String: ' || input_string);
    DBMS_OUTPUT.PUT_LINE('Reversed String: ' || reversed_string);
END;
/
```

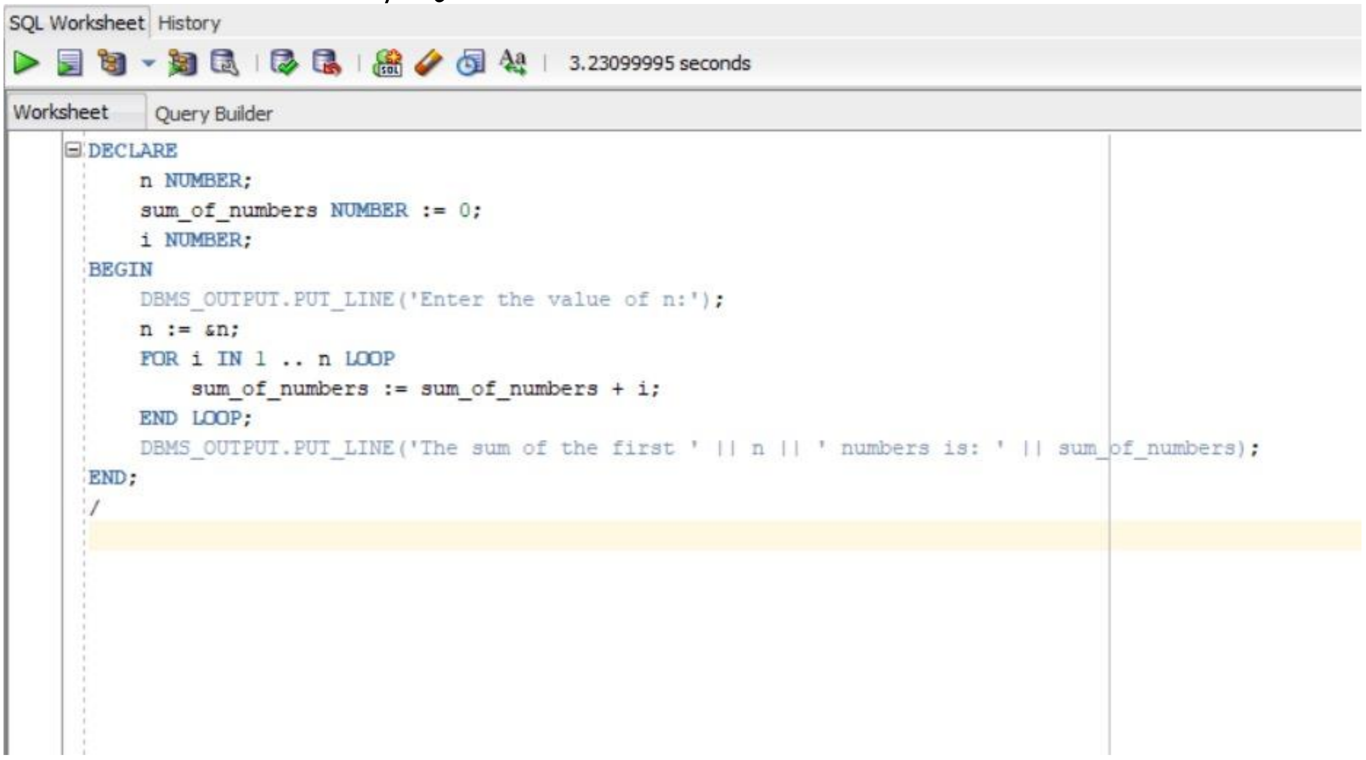
OUTPUT:



```
Original String: HelloWorld
Reversed String: dlroWolleH


PL/SQL procedure successfully completed.
```

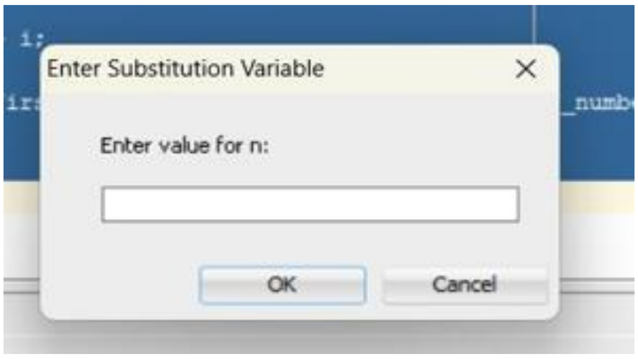
EXERCISE 19

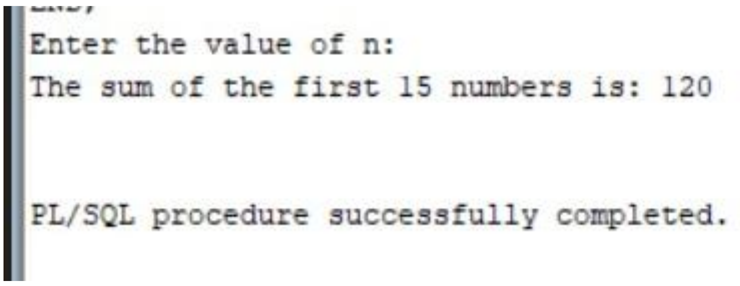
AIM: To Write a PL/SQL code to find the sum of n numbers.



```
DECLARE
    n NUMBER;
    sum_of_numbers NUMBER := 0;
    i NUMBER;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Enter the value of n:');
    n := &n;
    FOR i IN 1 .. n LOOP
        sum_of_numbers := sum_of_numbers + i;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('The sum of the first ' || n || ' numbers is: ' || sum_of_numbers);
END;
/
```

OUTPUT:

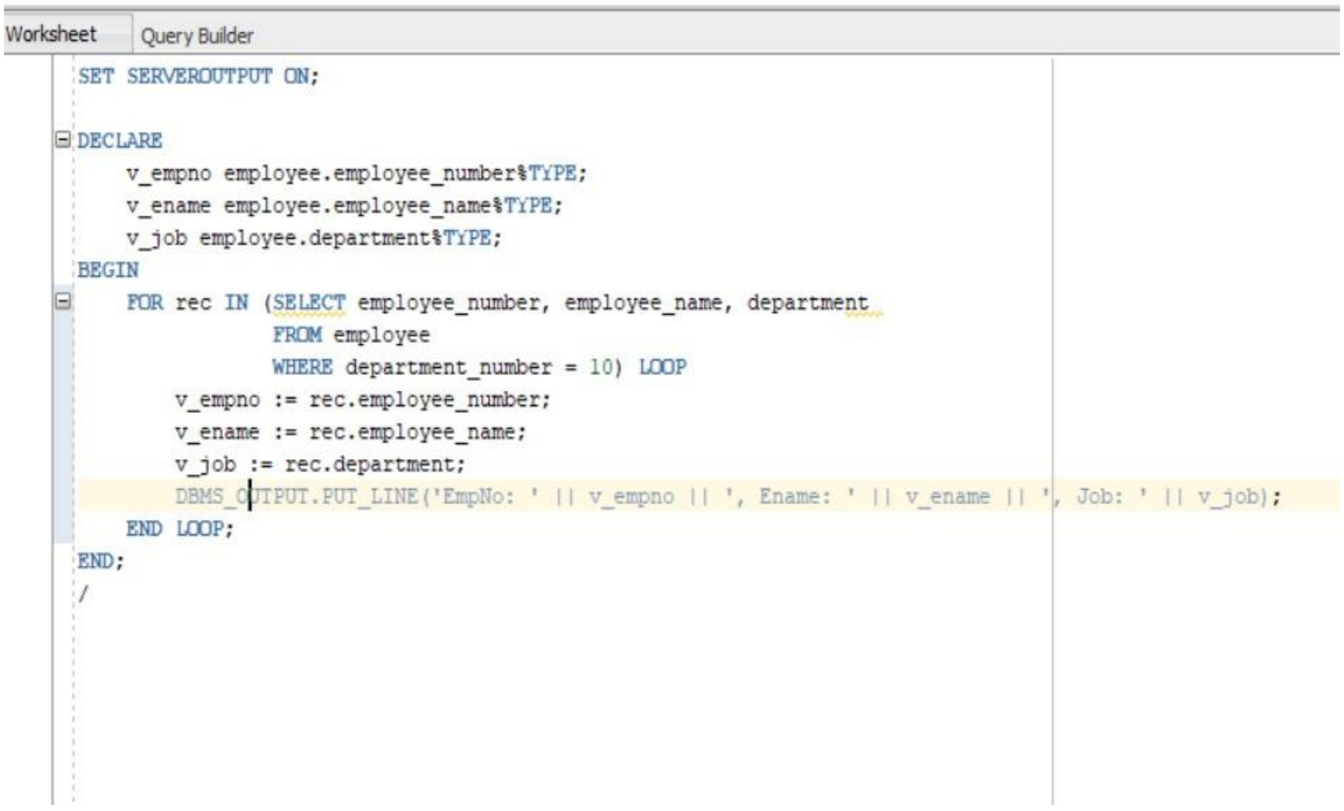




```
Enter the value of n:
The sum of the first 15 numbers is: 120


PL/SQL procedure successfully completed.
```

EXERCISE 20

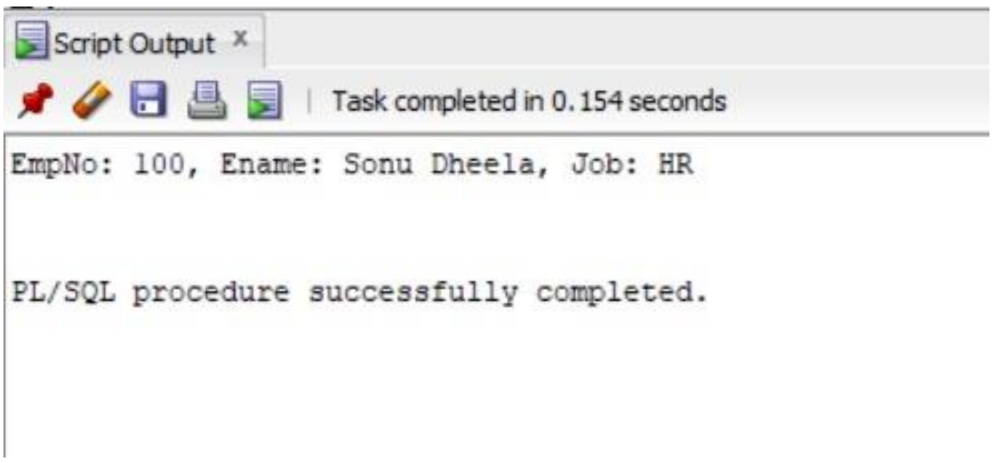
AIM: To Consider a PL/SQL code to display the empno, ename, job of



```
SET SERVEROUTPUT ON;

DECLARE
    v_empno employee.employee_number%TYPE;
    v_ename employee.employee_name%TYPE;
    v_job employee.department%TYPE;
BEGIN
    FOR rec IN (SELECT employee_number, employee_name, department
                FROM employee
                WHERE department_number = 10) LOOP
        v_empno := rec.employee_number;
        v_ename := rec.employee_name;
        v_job := rec.department;
        DBMS_OUTPUT.PUT_LINE('EmpNo: ' || v_empno || ', Ename: ' || v_ename || ', Job: ' || v_job);
    END LOOP;
END;
/
```

employees of department number 10.

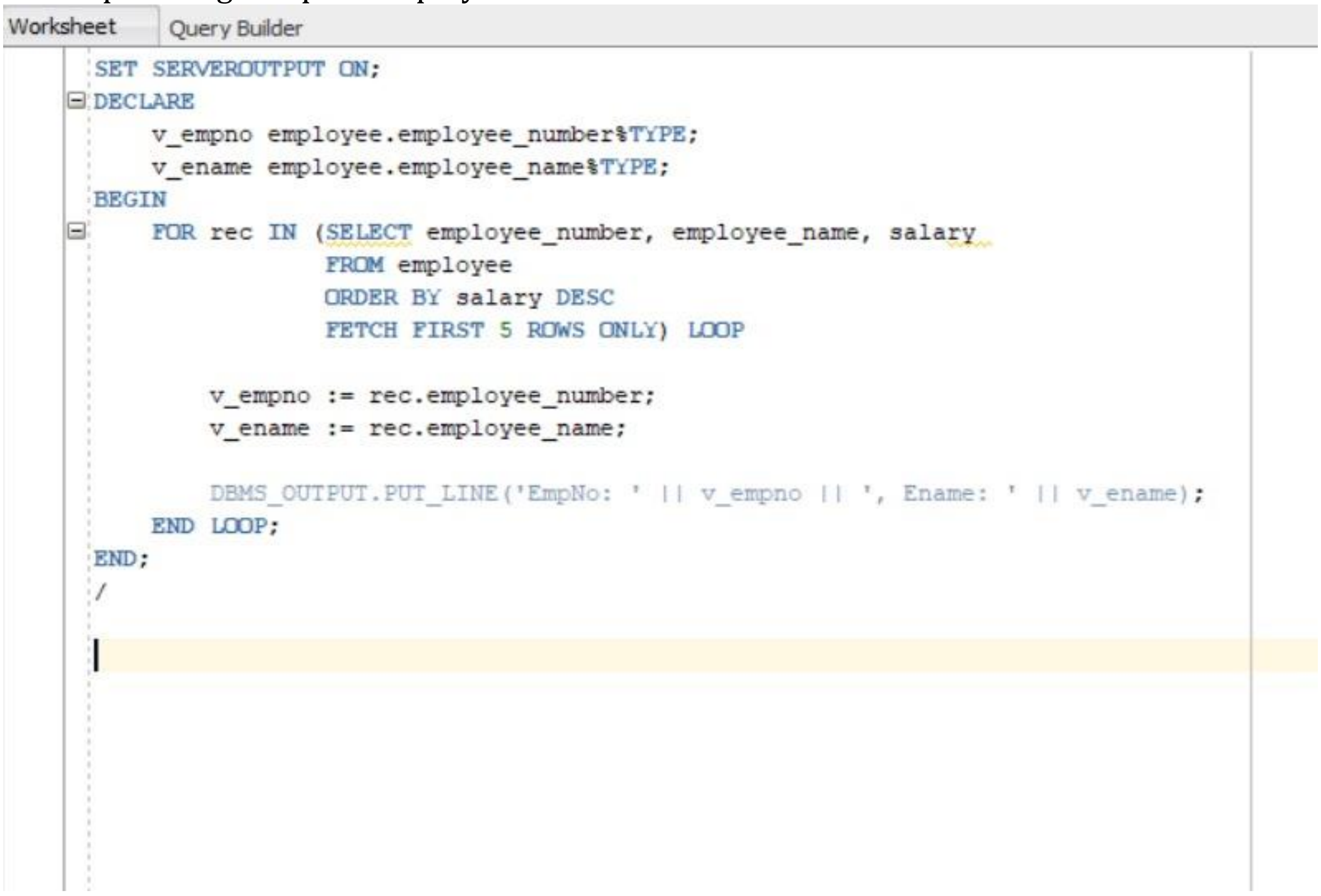
OUTPUT:



```
EmpNo: 100, Ename: Sonu Dheela, Job: HR


PL/SQL procedure successfully completed.
```

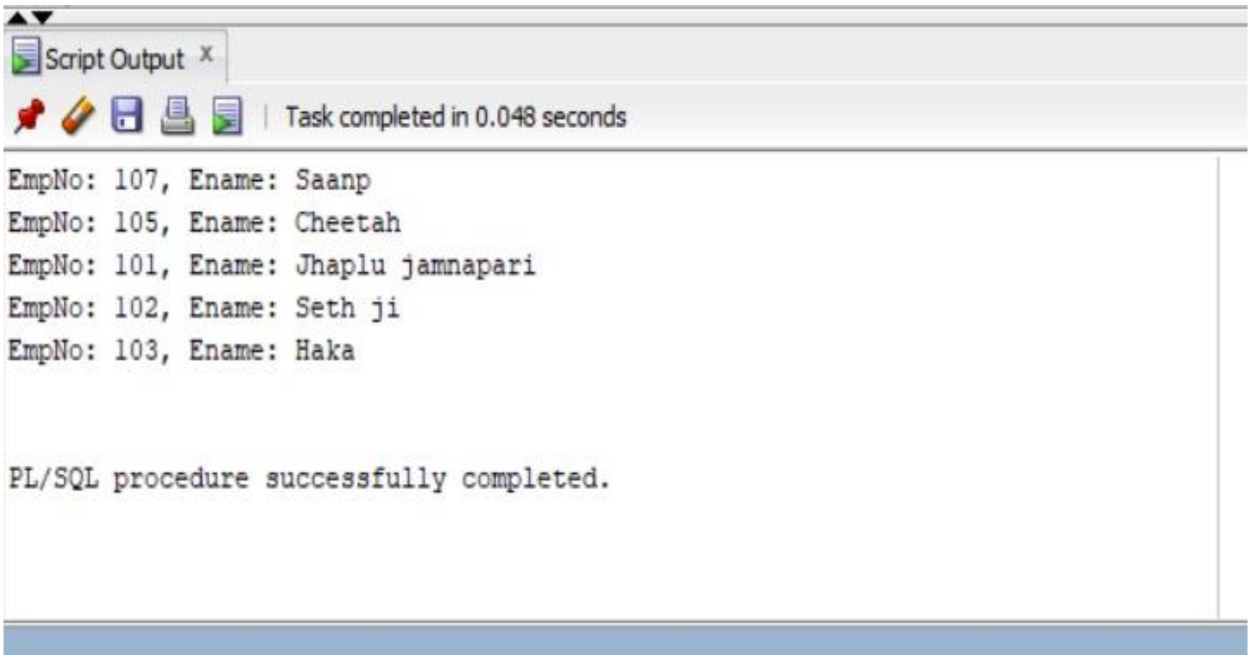
EXERCISE 21

AIM: To Consider a PL/SQL code to display the employee number & name of top five highest paid employees.



```
SET SERVEROUTPUT ON;
DECLARE
    v_empno employee.employee_number%TYPE;
    v_ename employee.employee_name%TYPE;
BEGIN
    FOR rec IN (SELECT employee_number, employee_name, salary
                FROM employee
                ORDER BY salary DESC
                FETCH FIRST 5 ROWS ONLY) LOOP

        v_empno := rec.employee_number;
        v_ename := rec.employee_name;

        DBMS_OUTPUT.PUT_LINE('EmpNo: ' || v_empno || ', Ename: ' || v_ename);
    END LOOP;
END;
/
```

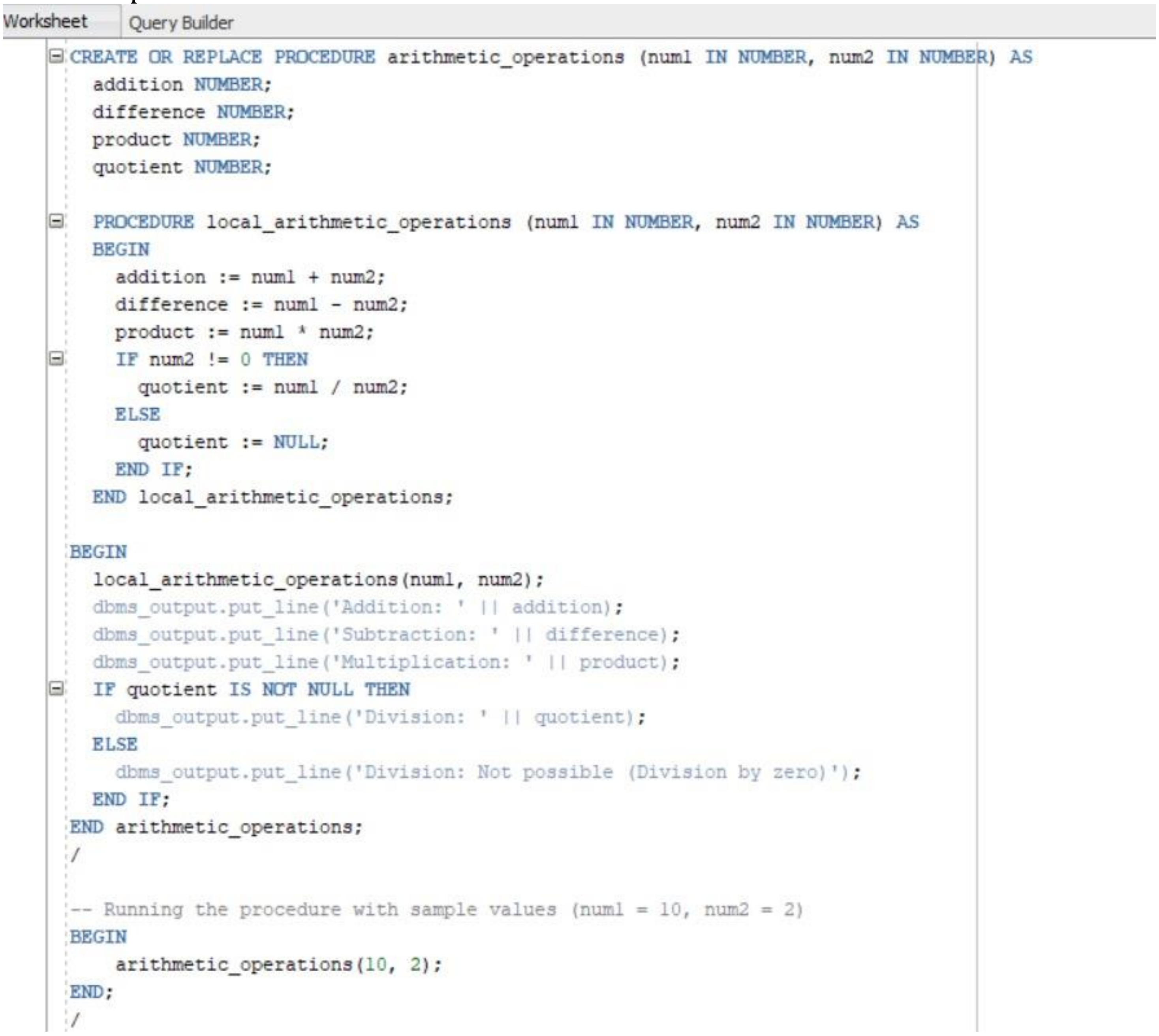
OUTPUT:



```
EmpNo: 107, Ename: Saanp
EmpNo: 105, Ename: Cheetah
EmpNo: 101, Ename: Jhaplu jamnapari
EmpNo: 102, Ename: Seth ji
EmpNo: 103, Ename: Haka


PL/SQL procedure successfully completed.
```

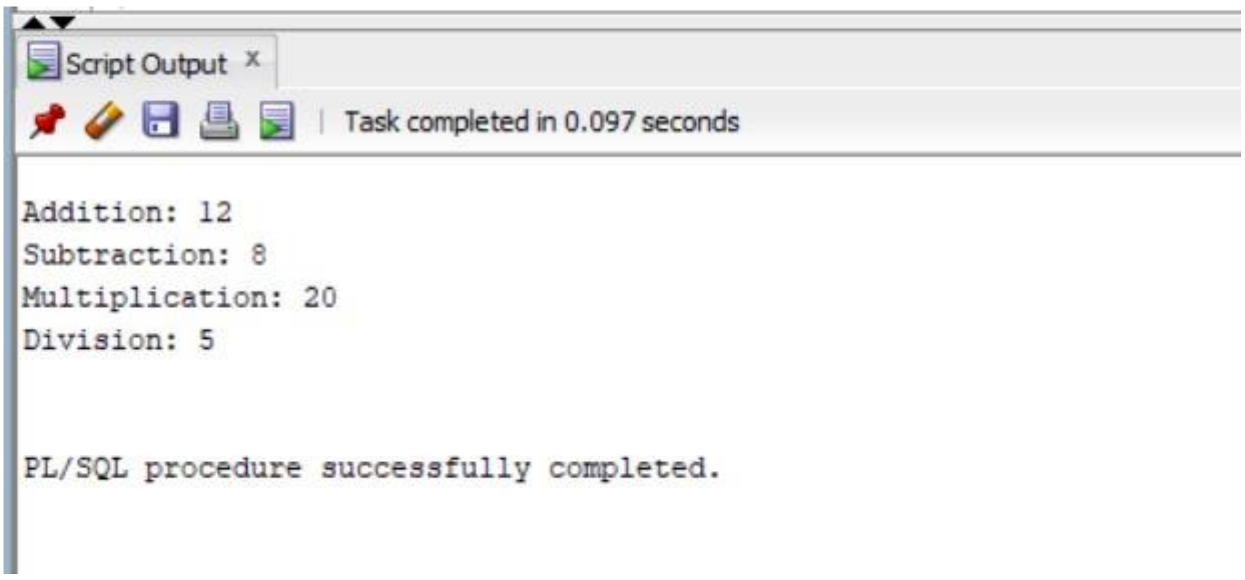
EXERCISE 22

AIM: To Consider a PL/SQL procedure that accepts 2 numbers & return addition, subtraction, multiplication & division of two numbers using stored procedure AND local procedure.



```sql
CREATE OR REPLACE PROCEDURE arithmetic_operations (num1 IN NUMBER, num2 IN NUMBER) AS
  addition NUMBER;
  difference NUMBER;
  product NUMBER;
  quotient NUMBER;

  PROCEDURE local_arithmetic_operations (num1 IN NUMBER, num2 IN NUMBER) AS
  BEGIN
    addition := num1 + num2;
    difference := num1 - num2;
    product := num1 * num2;
    IF num2 != 0 THEN
      quotient := num1 / num2;
    ELSE
      quotient := NULL;
    END IF;
  END local_arithmetic_operations;

BEGIN
  local_arithmetic_operations(num1, num2);
  dbms_output.put_line('Addition: ' || addition);
  dbms_output.put_line('Subtraction: ' || difference);
  dbms_output.put_line('Multiplication: ' || product);
  IF quotient IS NOT NULL THEN
    dbms_output.put_line('Division: ' || quotient);
  ELSE
    dbms_output.put_line('Division: Not possible (Division by zero)');
  END IF;
END arithmetic_operations;
/

-- Running the procedure with sample values (num1 = 10, num2 = 2)
BEGIN
    arithmetic_operations(10, 2);
END;
/
```

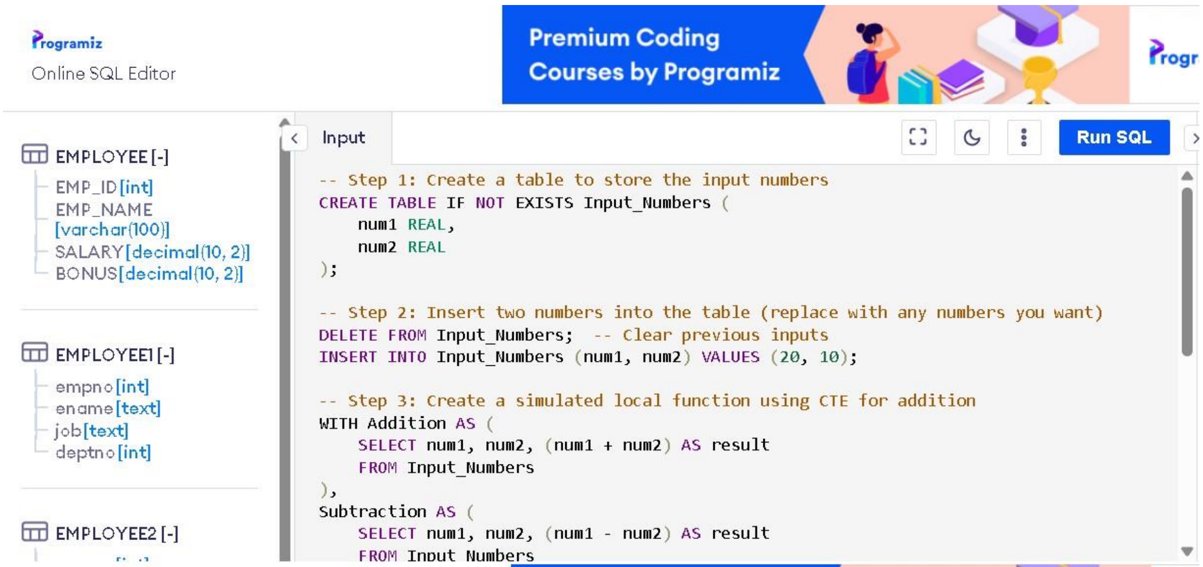
OUTPUT:



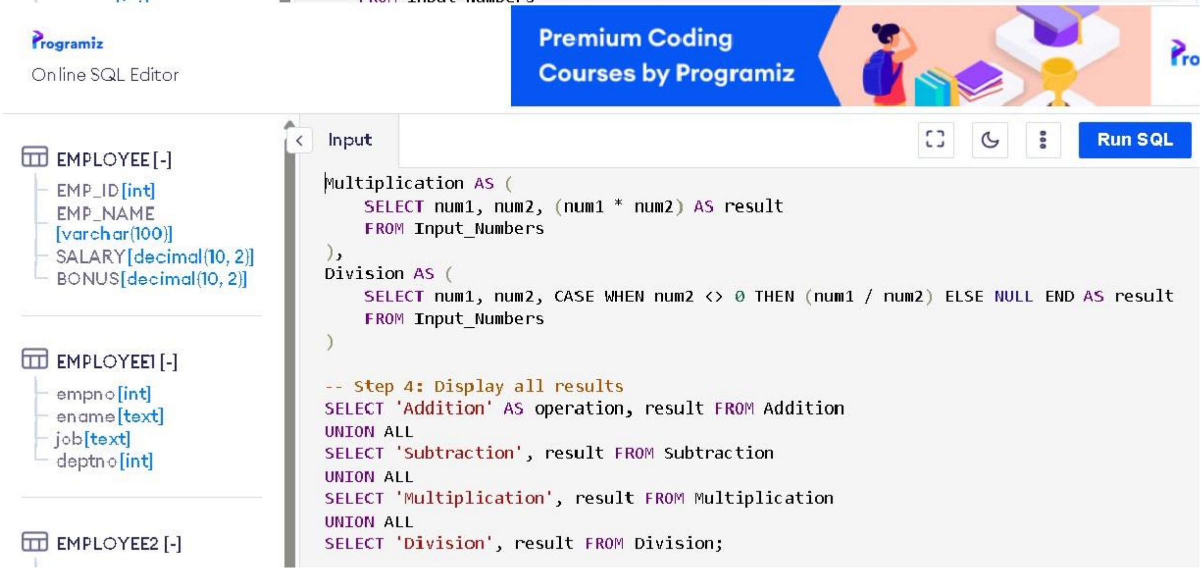
```
Addition: 12
Subtraction: 8
Multiplication: 20
Division: 5


PL/SQL procedure successfully completed.
```

EXERCISE 23

AIM: To Consider a PL/SQL code that accepts 2 numbers & return addition, subtraction,multiplication & division of two numbers using stored functions and local function.

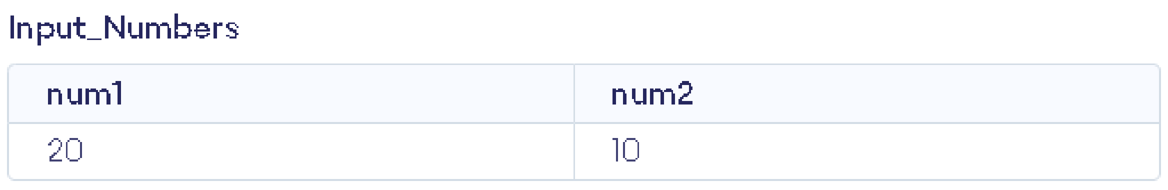


```sql
-- Step 1: Create a table to store the input numbers
CREATE TABLE IF NOT EXISTS Input_Numbers (
    num1 REAL,
    num2 REAL
);

-- Step 2: Insert two numbers into the table (replace with any numbers you want)
DELETE FROM Input_Numbers;  -- Clear previous inputs
INSERT INTO Input_Numbers (num1, num2) VALUES (20, 10);

-- Step 3: Create a simulated local function using CTE for addition
WITH Addition AS (
    SELECT num1, num2, (num1 + num2) AS result
    FROM Input_Numbers
),
Subtraction AS (
    SELECT num1, num2, (num1 - num2) AS result
    FROM Input_Numbers
```



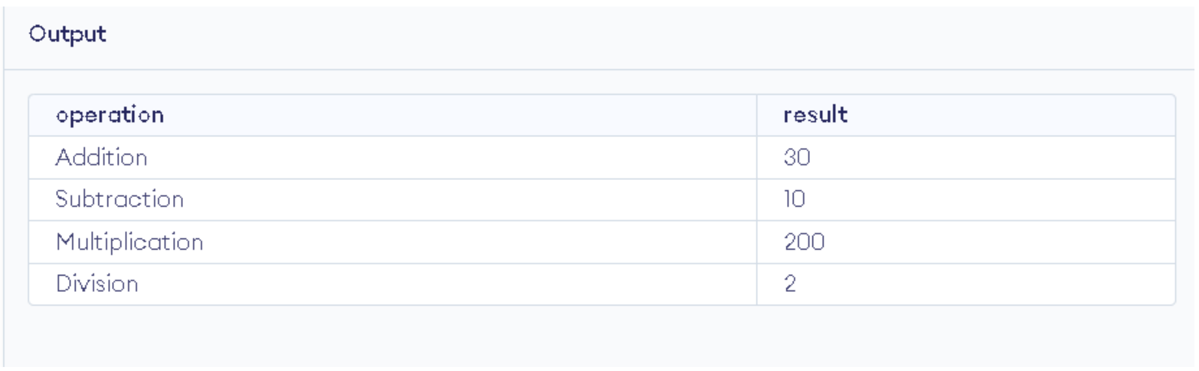
```sql
Multiplication AS (
    SELECT num1, num2, (num1 * num2) AS result
    FROM Input_Numbers
),
Division AS (
    SELECT num1, num2, CASE WHEN num2 <> 0 THEN (num1 / num2) ELSE NULL END AS result
    FROM Input_Numbers
)

-- Step 4: Display all results
SELECT 'Addition' AS operation, result FROM Addition
UNION ALL
SELECT 'Subtraction', result FROM Subtraction
UNION ALL
SELECT 'Multiplication', result FROM Multiplication
UNION ALL
SELECT 'Division', result FROM Division;
```

OUTPUT:



Input_Numbers

num1	num2
20	10

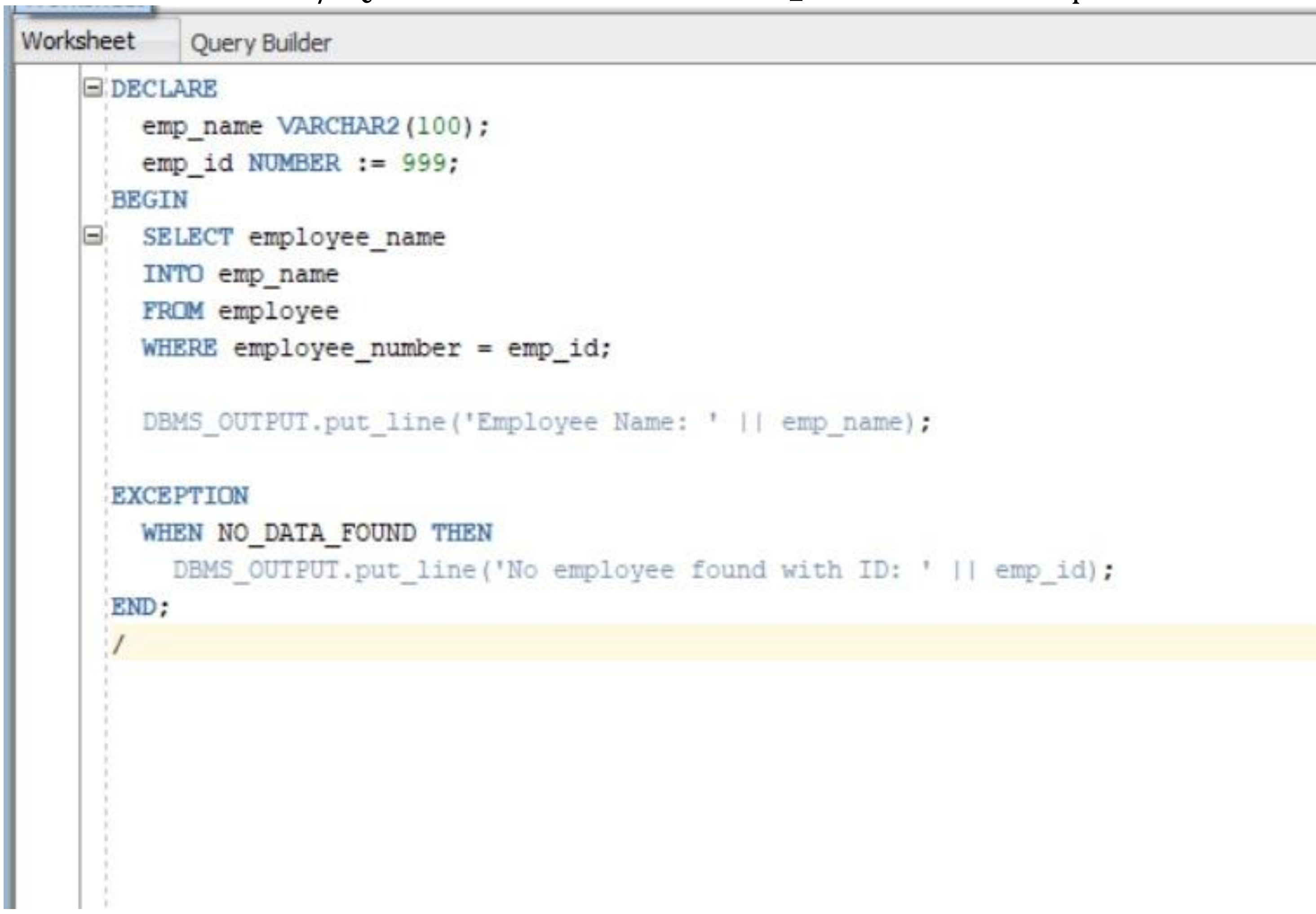


Output

operation	result
Addition	30
Subtraction	10
Multiplication	200
Division	2

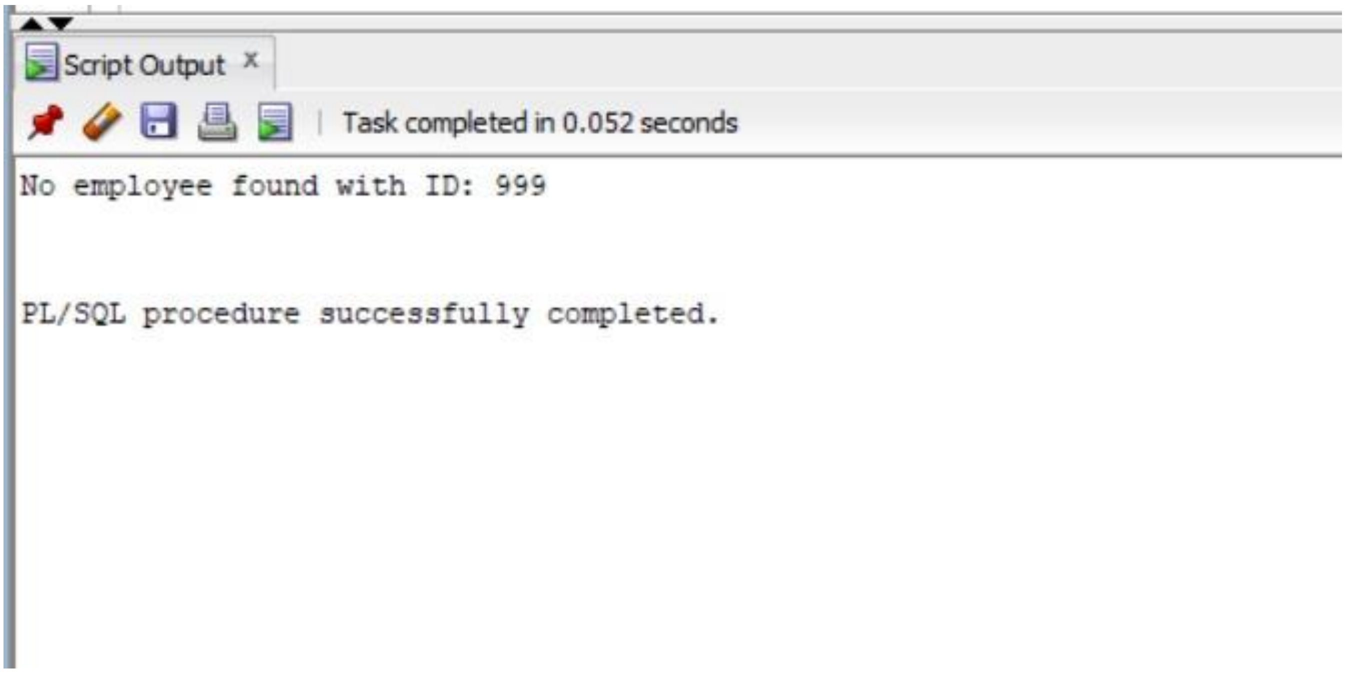
EXERCISE 24

AIM: To Write a PL/SQL block to show the use of NO_DATA FOUND exception.



```
DECLARE
  emp_name VARCHAR2(100);
  emp_id NUMBER := 999;
BEGIN
  SELECT employee_name
  INTO emp_name
  FROM employee
  WHERE employee_number = emp_id;

  DBMS_OUTPUT.put_line('Employee Name: ' || emp_name);

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.put_line('No employee found with ID: ' || emp_id);
END;
/
```

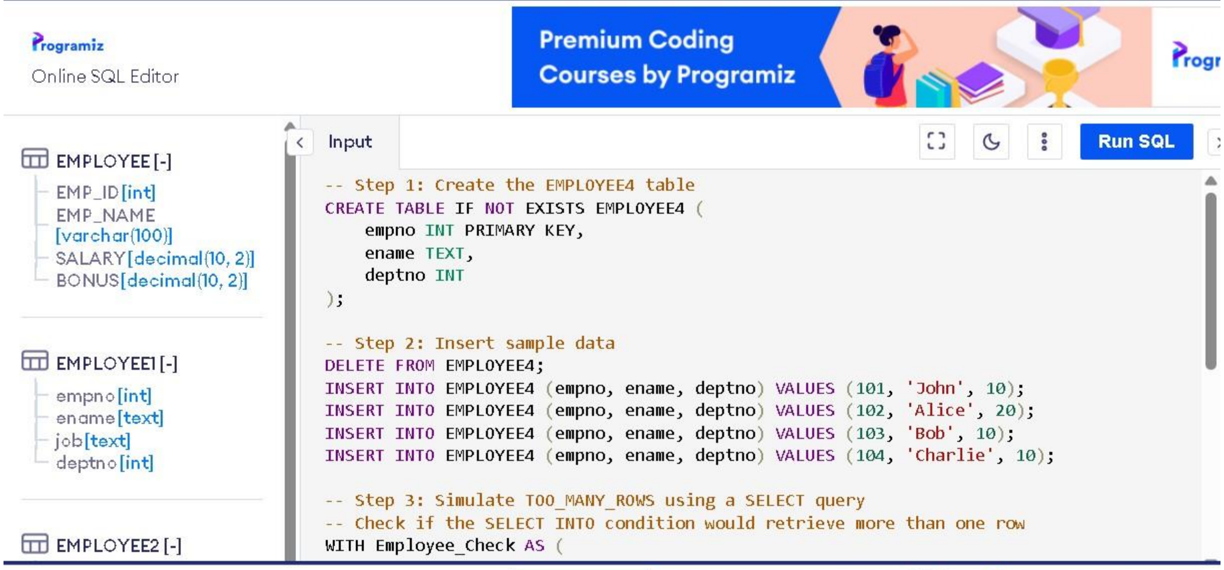
OUTPUT:



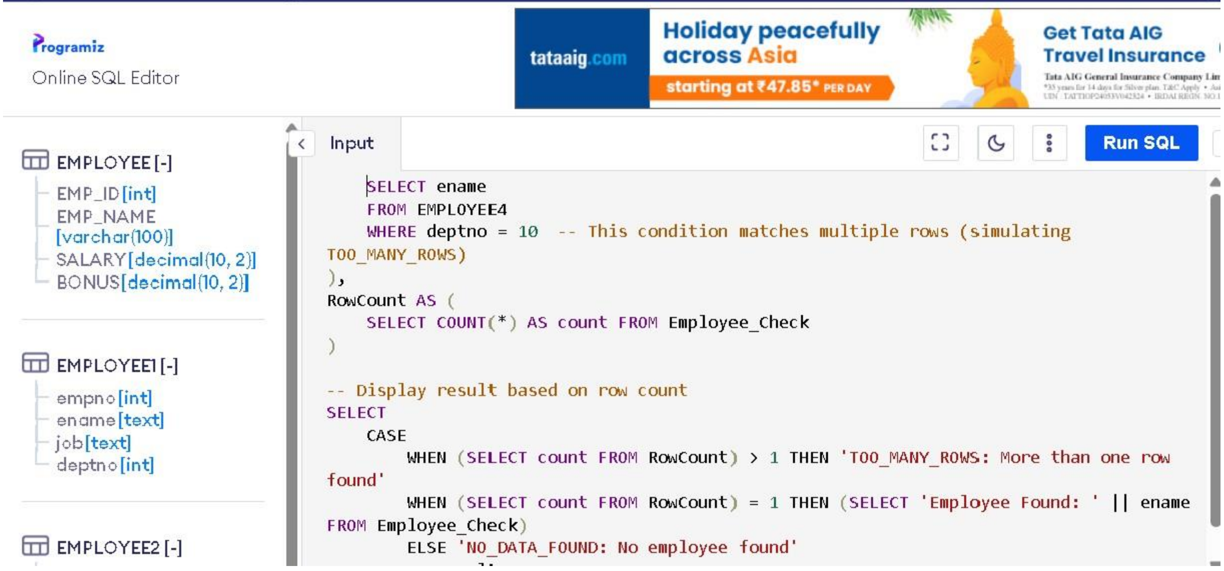
```
No employee found with ID: 999


PL/SQL procedure successfully completed.
```

EXERCISE 25

AIM: To Write a PL/SQL block to show the use of TOO_MANY ROWS exception.

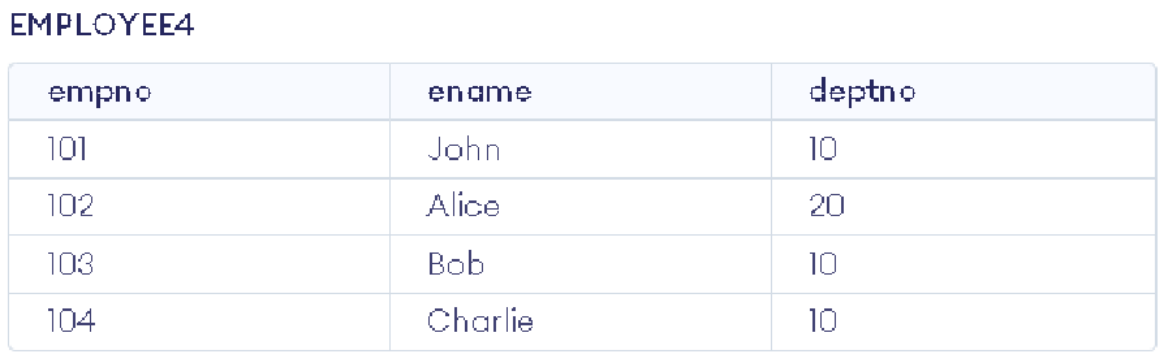


```
-- Step 1: Create the EMPLOYEE4 table
CREATE TABLE IF NOT EXISTS EMPLOYEE4 (
    empno INT PRIMARY KEY,
    ename TEXT,
    deptno INT
);

-- Step 2: Insert sample data
DELETE FROM EMPLOYEE4;
INSERT INTO EMPLOYEE4 (empno, ename, deptno) VALUES (101, 'John', 10);
INSERT INTO EMPLOYEE4 (empno, ename, deptno) VALUES (102, 'Alice', 20);
INSERT INTO EMPLOYEE4 (empno, ename, deptno) VALUES (103, 'Bob', 10);
INSERT INTO EMPLOYEE4 (empno, ename, deptno) VALUES (104, 'Charlie', 10);

-- Step 3: Simulate TOO_MANY_ROWS using a SELECT query
-- Check if the SELECT INTO condition would retrieve more than one row
WITH Employee_Check AS (
```



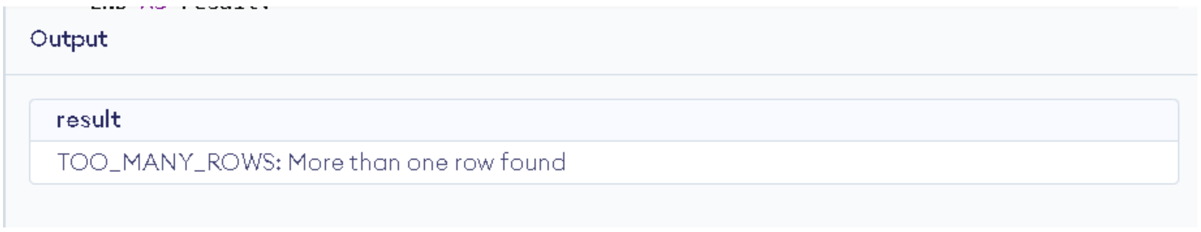
```
    SELECT ename
    FROM EMPLOYEE4
    WHERE deptno = 10  -- This condition matches multiple rows (simulating
TOO_MANY_ROWS)
),
RowCount AS (
    SELECT COUNT(*) AS count FROM Employee_Check
)

-- Display result based on row count
SELECT
    CASE
        WHEN (SELECT count FROM RowCount) > 1 THEN 'TOO_MANY_ROWS: More than one row
found'
        WHEN (SELECT count FROM RowCount) = 1 THEN (SELECT 'Employee Found: ' || ename
FROM Employee_Check)
        ELSE 'NO_DATA_FOUND: No employee found'
```

OUTPUT:



EMPLOYEE4

empno	ename	deptno
101	John	10
102	Alice	20
103	Bob	10
104	Charlie	10

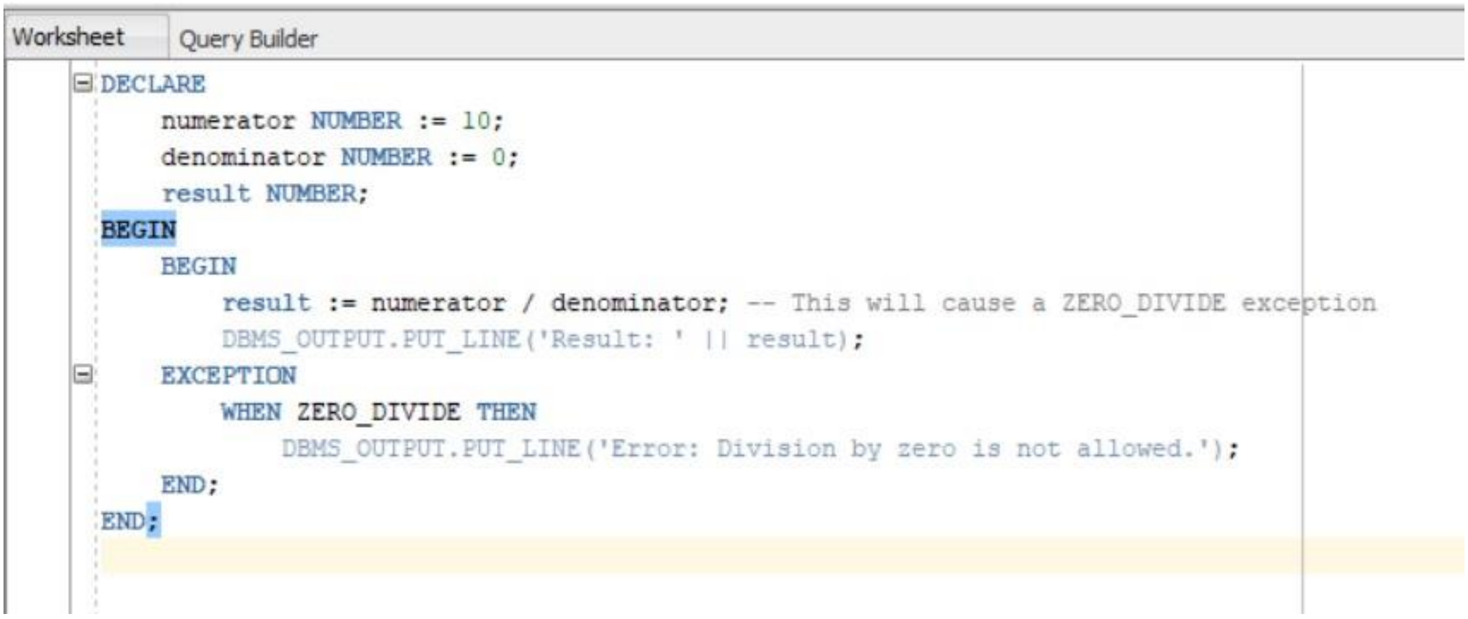


Output

```
result
TOO_MANY_ROWS: More than one row found
```

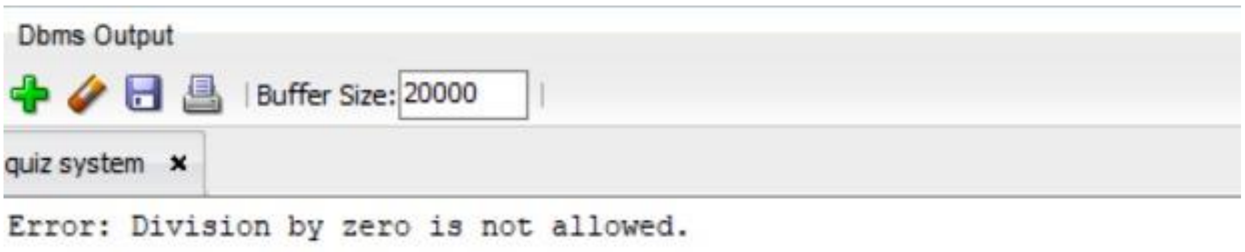
EXERCISE 26

AIM: To Write a PL/SQL block to show the use of ZERO_DIVIDE exception.



```
DECLARE
    numerator NUMBER := 10;
    denominator NUMBER := 0;
    result NUMBER;
BEGIN
    BEGIN
        result := numerator / denominator; -- This will cause a ZERO_DIVIDE exception
        DBMS_OUTPUT.PUT_LINE('Result: ' || result);
    EXCEPTION
        WHEN ZERO_DIVIDE THEN
            DBMS_OUTPUT.PUT_LINE('Error: Division by zero is not allowed.');
    END;
END;
```

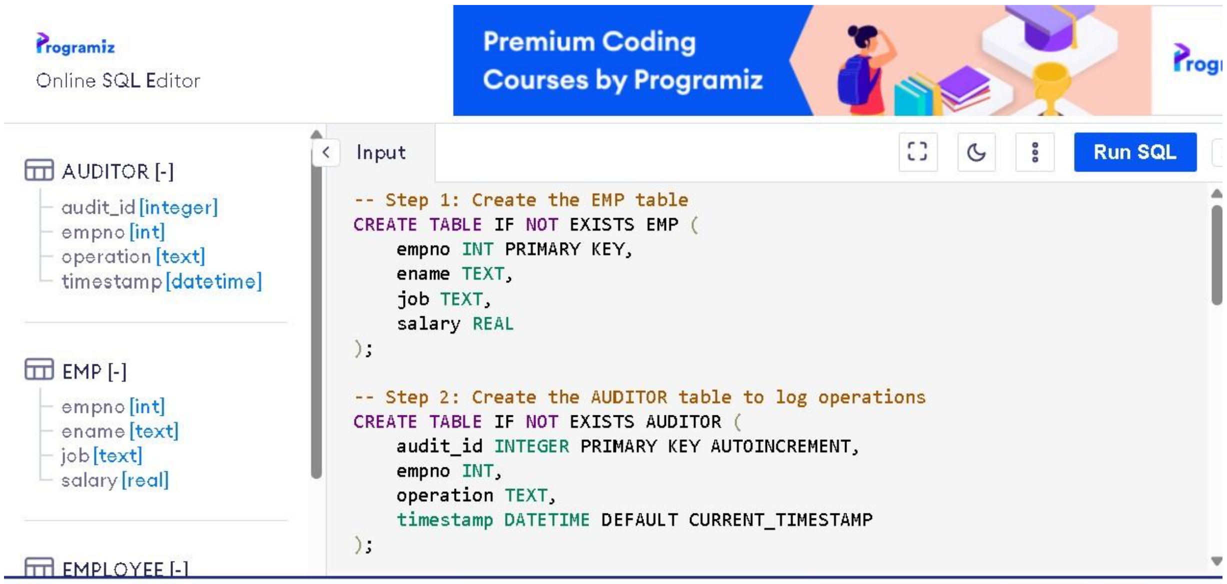
OUTPUT:



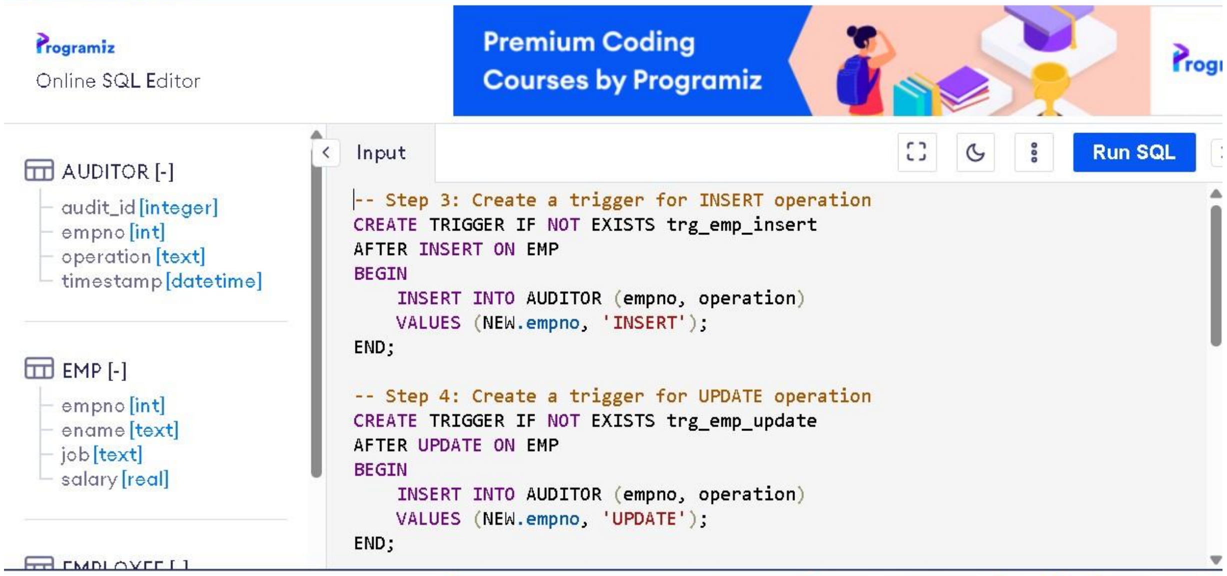
```
Error: Division by zero is not allowed.
```

EXERCISE 27

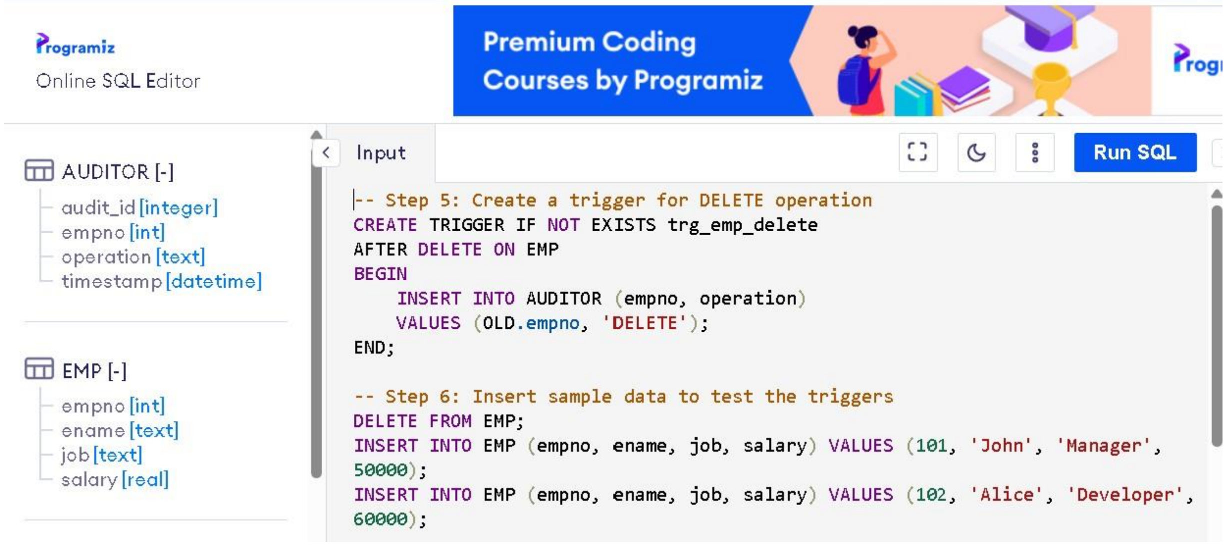
AIM: To create a trigger on the emp table, which store the empno& operation in the table auditorfor each operation i.e. Insert, Update & Delete.



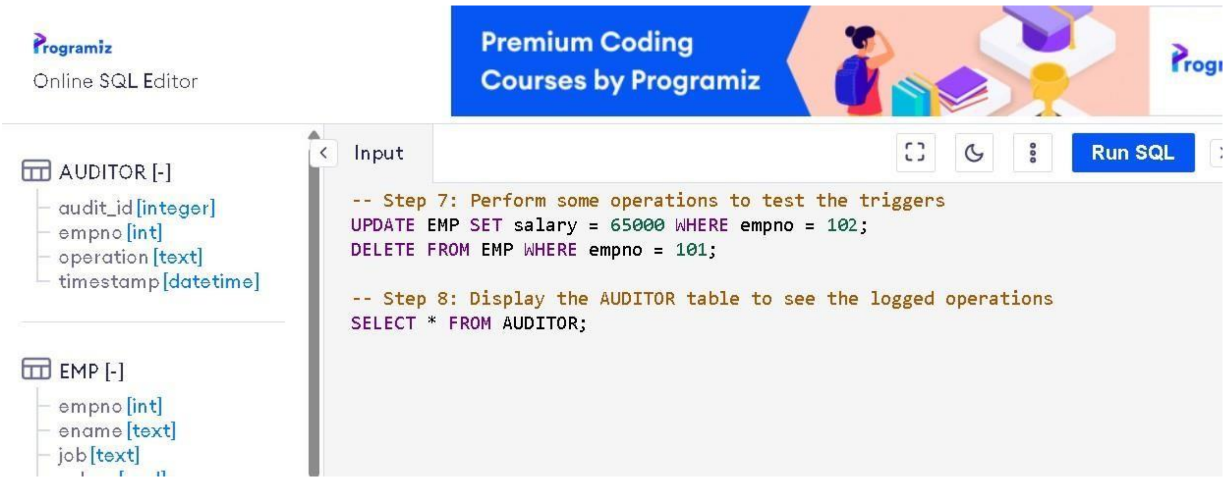
```
-- Step 1: Create the EMP table
CREATE TABLE IF NOT EXISTS EMP (
    empno INT PRIMARY KEY,
    ename TEXT,
    job TEXT,
    salary REAL
);

-- Step 2: Create the AUDITOR table to log operations
CREATE TABLE IF NOT EXISTS AUDITOR (
    audit_id INTEGER PRIMARY KEY AUTOINCREMENT,
    empno INT,
    operation TEXT,
    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
);
```



```
-- Step 3: Create a trigger for INSERT operation
CREATE TRIGGER IF NOT EXISTS trg_emp_insert
AFTER INSERT ON EMP
BEGIN
    INSERT INTO AUDITOR (empno, operation)
    VALUES (NEW.empno, 'INSERT');
END;

-- Step 4: Create a trigger for UPDATE operation
CREATE TRIGGER IF NOT EXISTS trg_emp_update
AFTER UPDATE ON EMP
BEGIN
    INSERT INTO AUDITOR (empno, operation)
    VALUES (NEW.empno, 'UPDATE');
END;
```

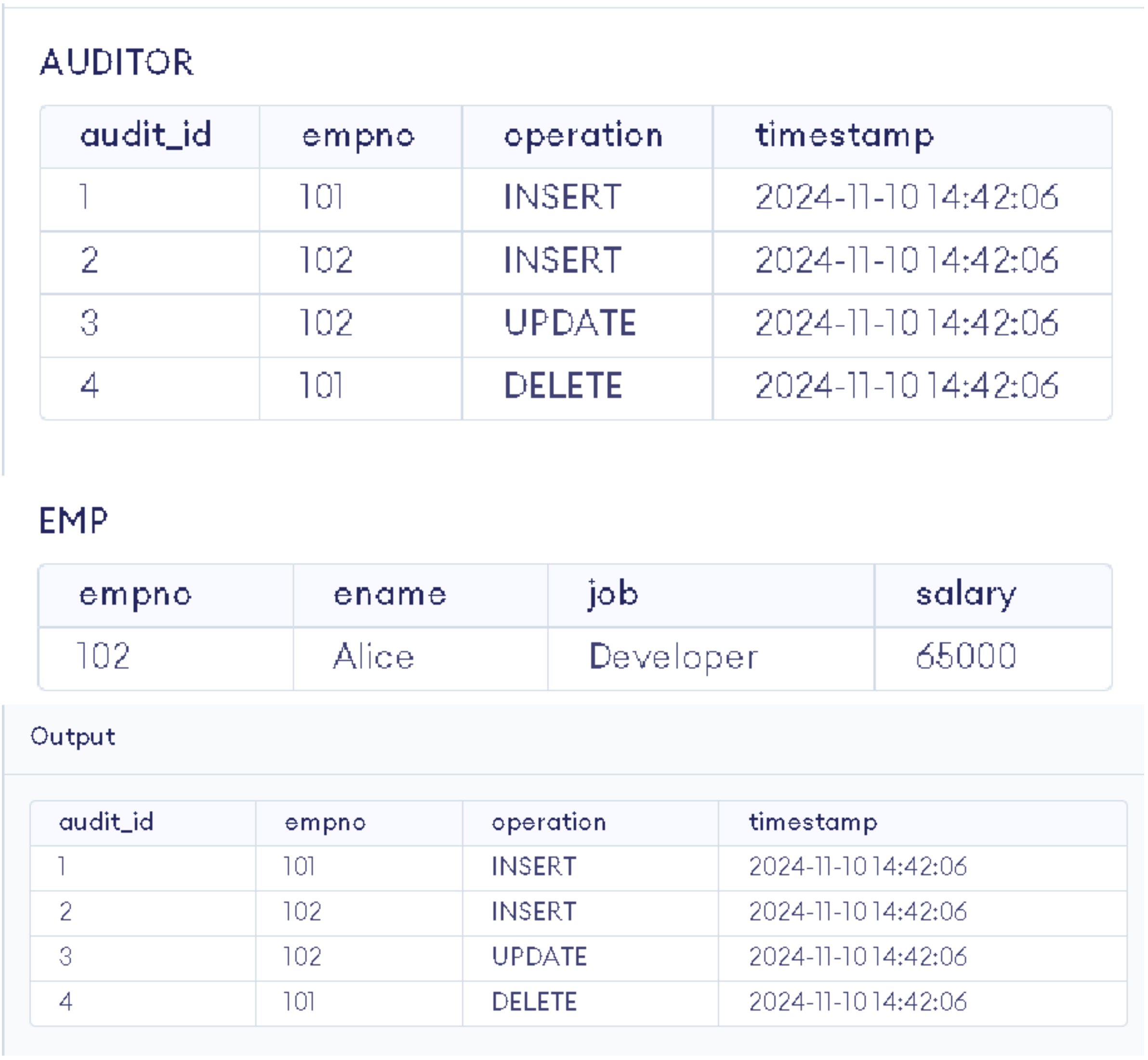


```
-- Step 5: Create a trigger for DELETE operation
CREATE TRIGGER IF NOT EXISTS trg_emp_delete
AFTER DELETE ON EMP
BEGIN
    INSERT INTO AUDITOR (empno, operation)
    VALUES (OLD.empno, 'DELETE');
END;

-- Step 6: Insert sample data to test the triggers
DELETE FROM EMP;
INSERT INTO EMP (empno, ename, job, salary) VALUES (101, 'John', 'Manager',
50000);
INSERT INTO EMP (empno, ename, job, salary) VALUES (102, 'Alice', 'Developer',
60000);
```



```
-- Step 7: Perform some operations to test the triggers
UPDATE EMP SET salary = 65000 WHERE empno = 102;
DELETE FROM EMP WHERE empno = 101;

-- Step 8: Display the AUDITOR table to see the logged operations
SELECT * FROM AUDITOR;
```

OUTPUT:



AUDITOR

audit_id	empno	operation	timestamp
1	101	INSERT	2024-11-10 14:42:06
2	102	INSERT	2024-11-10 14:42:06
3	102	UPDATE	2024-11-10 14:42:06
4	101	DELETE	2024-11-10 14:42:06

EMP

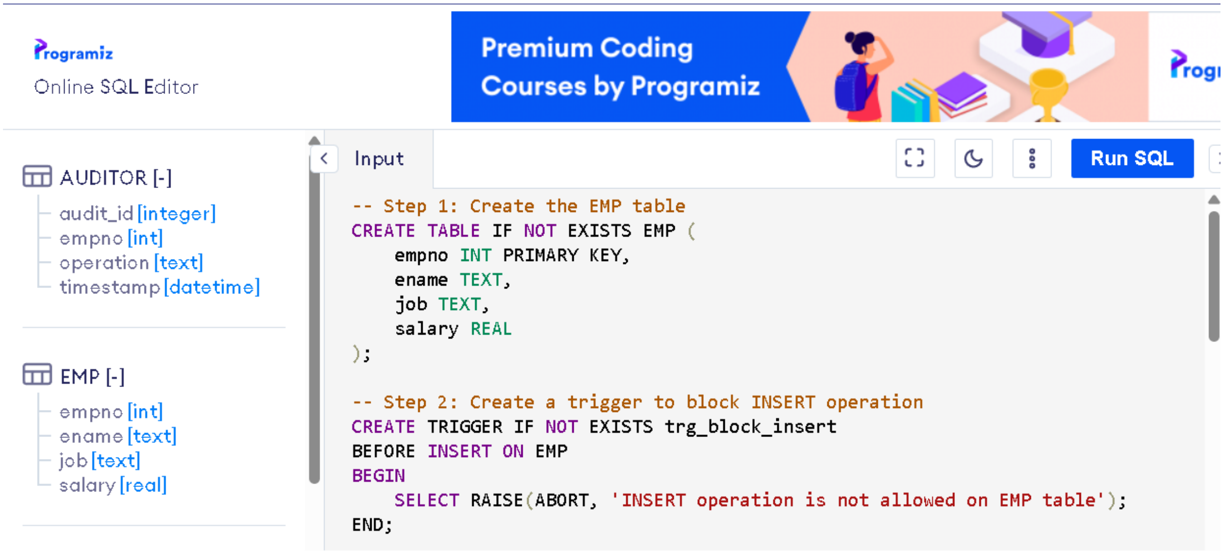
empno	ename	job	salary
102	Alice	Developer	65000

Output

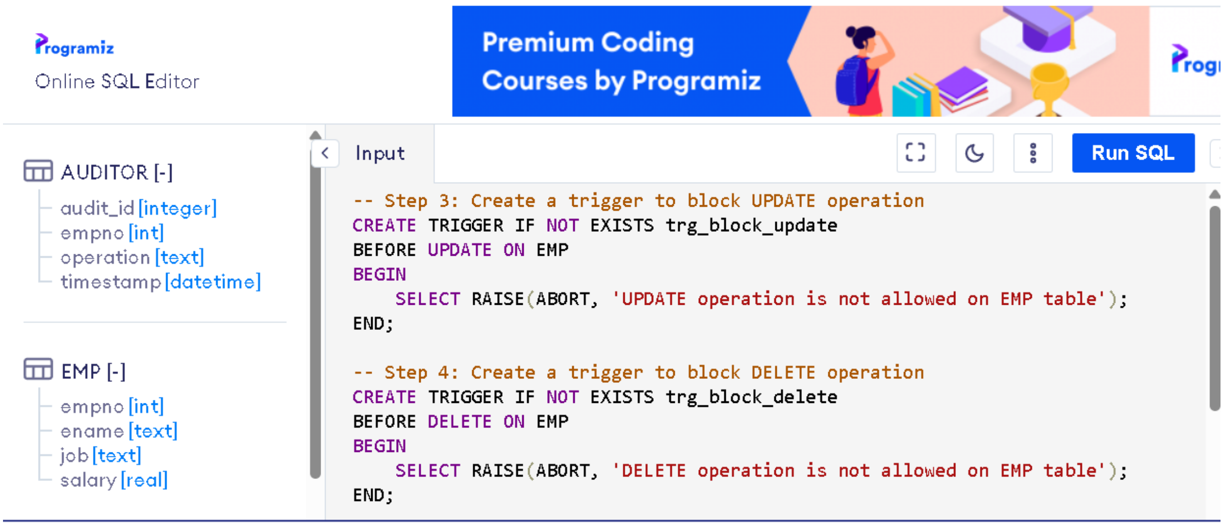
audit_id	empno	operation	timestamp
1	101	INSERT	2024-11-10 14:42:06
2	102	INSERT	2024-11-10 14:42:06
3	102	UPDATE	2024-11-10 14:42:06
4	101	DELETE	2024-11-10 14:42:06

EXERCISE 28

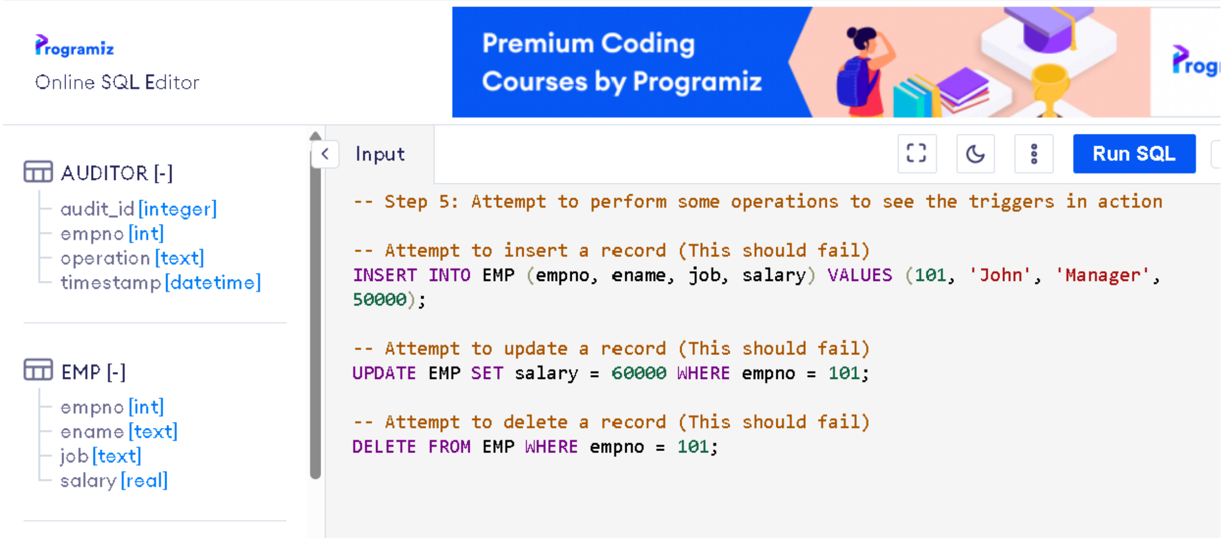
AIM: To create a trigger so that no operation can be performed on emp table.



```sql
-- Step 1: Create the EMP table
CREATE TABLE IF NOT EXISTS EMP (
    empno INT PRIMARY KEY,
    ename TEXT,
    job TEXT,
    salary REAL
);

-- Step 2: Create a trigger to block INSERT operation
CREATE TRIGGER IF NOT EXISTS trg_block_insert
BEFORE INSERT ON EMP
BEGIN
    SELECT RAISE(ABORT, 'INSERT operation is not allowed on EMP table');
END;
```

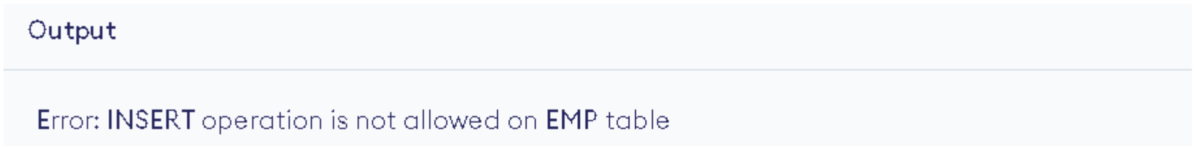


```sql
-- Step 3: Create a trigger to block UPDATE operation
CREATE TRIGGER IF NOT EXISTS trg_block_update
BEFORE UPDATE ON EMP
BEGIN
    SELECT RAISE(ABORT, 'UPDATE operation is not allowed on EMP table');
END;

-- Step 4: Create a trigger to block DELETE operation
CREATE TRIGGER IF NOT EXISTS trg_block_delete
BEFORE DELETE ON EMP
BEGIN
    SELECT RAISE(ABORT, 'DELETE operation is not allowed on EMP table');
END;
```



```sql
-- Step 5: Attempt to perform some operations to see the triggers in action

-- Attempt to insert a record (This should fail)
INSERT INTO EMP (empno, ename, job, salary) VALUES (101, 'John', 'Manager',
50000);

-- Attempt to update a record (This should fail)
UPDATE EMP SET salary = 60000 WHERE empno = 101;

-- Attempt to delete a record (This should fail)
DELETE FROM EMP WHERE empno = 101;
```

OUTPUT:



Output

```
Error: INSERT operation is not allowed on EMP table
```