Oracle Lab

Q1. Create the following tables

Customer

Column_name	Data type	<u>Size</u>	Constraint
SID	Varchar2	4	Primary Key
First_Name	Char	20	
Last_name	Char	20	

Orders

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

```
CREATE TABLE Customer (

SID VARCHAR(4) PRIMARY KEY,

First_name CHAR(20),

Last_name CHAR(20)
);
```

Output

SQL query successfully executed. However, the result set is empty.

```
CREATE TABLE Orders (
Order_ID VARCHAR2(4) PRIMARY KEY,
Order_date CHAR(20),
Customer_SID VARCHAR2(20),
Amount NUMBER CHECK (Amount > 20000),
FOREIGN KEY (Customer_SID) REFERENCES Customer(SID)
);
```

Output

SQL query successfully executed. However, the result set is empty.

Q2. Insert five records for each table

```
Insert into Customer(SID, First_name, Last_name) VALUES ("1","Elena","Greorge");
Insert into Customer(SID, First_name, Last_name) VALUES ("2","Adam","John");
Insert into Customer(SID, First_name, Last_name) VALUES ("3","Elijah","Mikaelson");
Insert into Customer(SID, First_name, Last_name) VALUES ("4","Stefen","Salvatore");
Insert into Customer(SID, First_name, Last_name) VALUES ("5","Klaus","Alexander");
```

Select * from Customer;

Output		
alb.	= .	
SID	First_name	Last_name
1	Elena	Greorge
2	Adam	John
3	Elijah	Mikaelson
4	Stefen	Salvatore
5	Klaus	Alexander

Insert into Orders(Order_ID,Order_date,Customer_SID,Amount) VALUES ("01","08-03","1",21000); Insert into Orders(Order_ID,Order_date,Customer_SID,Amount) VALUES ("02","15-03","2",23000); Insert into Orders(Order_ID,Order_date,Customer_SID,Amount) VALUES ("03","07-07","3",26000); Insert into Orders(Order_ID,Order_date,Customer_SID,Amount) VALUES ("04","14-07","4",25000); Insert into Orders(Order_ID,Order_date,Customer_SID,Amount) VALUES ("05","23-09","5",29000);

Select * from Orders;

Order_date	Customer_SID	Amount
08-03	1	21000
15-03	2	23000
07-07	3	26000
14-07	4	25000
23-09	5	29000
	08-03 15-03 07-07 14-07	08-03 1 15-03 2 07-07 3 14-07 4

Q3.Customer_SID column in the ORDERS table is a foreign key pointing to the SID column in the CUSTOMER table.

```
CREATE TABLE Orders (

Order_ID VARCHAR2(4) PRIMARY KEY,

Order_date CHAR(20),
```

```
Customer_SID VARCHAR2(20),

Amount NUMBER CHECK (Amount > 20000),

FOREIGN KEY (Customer_SID) REFERENCES Customer(SID)

);
```

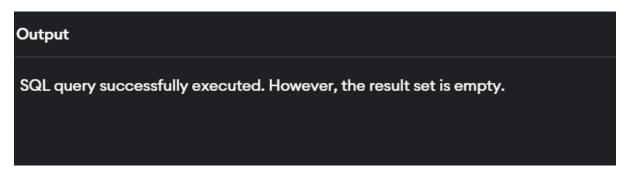
Q5. List the details of the customers along with the amount.

SELECT Customer.*, Orders.Amount FROM Customer INNER JOIN Orders ON Customer.SID = Orders.Customer_SID;

Output			
SID	First_name	Last_name	Amount
1	Elena	Greorge	21000
2	Adam	John	23000
3	Elijah	Mikaelson	26000
4	Stefen	Salvatore	25000
5	Klaus	Alexander	29000

Q6. List the customers whose names end with "s".

SELECT * FROM Customer WHERE last_name LIKE '%s';



Q7. List the orders where amount is between 21000 and 30000

Output			
Order_ID	Order_date	Customer_SID	Amount
01	08-03	1	21000
02	15-03	2	23000
03	07-07	3	26000
04	14-07	4	25000
05	23-09	5	29000

Q8. list the orders where amount is increased by 500 and replace with name "new amount".

SELECT *, (Amount + 500) AS New_Amount FROM Orders;

Output				
Order_ID	Order_date	Customer_SID	Amount	New_Amount
01	08-03	1	21000	21500
02	15-03	2	23000	23500
03	07-07	3	26000	26500
04	14-07	4	25000	25500
05	23-09	5	29000	29500

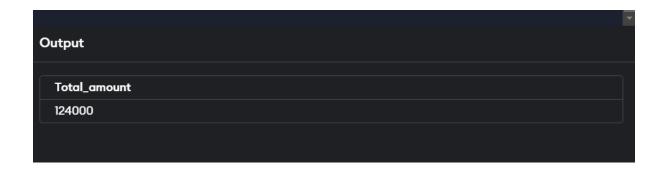
Q9.Display the order_id and total amount of orders

SELECT Order_id, SUM(Amount) AS Total_amount FROM Orders GROUP BY Order_id;

Order_ID	Total_amount
01	21000
02	23000
03	26000
04	25000
05	29000
	27000

Q10. Calculate the totaamount of orders that has more than 15000.

SELECT SUM(Amount) AS Total_amount FROM Orders WHERE Amount > 15000;



11. Display all the string functions used in SQL.

```
Select CONCAT('John','Doe') AS Concatenation;
Select LENGTH("Adam","John") AS Length;
Select LOWER("Adam","John") AS Lower;
Select UPPER("Adam","John") AS Upper;
Select SUBSTR("Adam","John") AS Substring;
Select TRIM("Adam","John") AS Trim;
Select LTRIM("Adam","John") AS LeftTrim;
Select RTRIM("Adam","John") AS RightTrim;
```

Q12. Create the following tables

```
CREATE TABLE Student (
ROllNumber VARCHAR2(20),
Name VARCHAR2(20),
Class VARCHAR2(20),
Marks NUMBER(6,2)
);
```

Output

SQL query successfully executed. However, the result set is empty.

```
INSERT INTO Student (RollNumber, Name, Class, Marks)
VALUES ('S001', 'John Doe', '10A', 85.50);
INSERT INTO Student (RollNumber, Name, Class, Marks)
VALUES ('S002', 'Alice Smith', '10B', 92.75);
INSERT INTO Student (RollNumber, Name, Class, Marks)
VALUES ('S003', 'Bob Johnson', '10A', 78.00);
INSERT INTO Student (RollNumber, Name, Class, Marks)
VALUES ('S004', 'Charlie Brown', '10C', 88.25);
INSERT INTO Student (RollNumber, Name, Class, Marks)
VALUES ('S005', 'Diana Clark', '10B', 91.00);
CREATE TABLE Student1 (
  R_no VARCHAR2(20),
  Name VARCHAR2(20),
  Class VARCHAR2(20),
  Marks NUMBER(6,2)
);
```

Output

SQL query successfully executed. However, the result set is empty.

```
INSERT INTO Student1 (R_no, Name, Class, Marks)
VALUES ('S006', 'Eva White', '10C', 79.50);
```

```
INSERT INTO Student1 (R_no, Name, Class, Marks)

VALUES ('S007', 'Frank Green', '10A', 94.00);

INSERT INTO Student1 (R_no, Name, Class, Marks)

VALUES ('S008', 'Grace Lee', '10B', 83.25);

INSERT INTO Student1 (R_no, Name, Class, Marks)

VALUES ('S009', 'Hannah Scott', '10C', 77.50);

INSERT INTO Student1 (R_no, Name, Class, Marks)

VALUES ('S010', 'Isaac Taylor', '10A', 90.00);
```

Q13. Display all the contents of student and student1 using union clause.

SELECT RollNumber AS R_no, Name, Class, Marks
FROM Student
UNION
SELECT R_no, Name, Class, Marks
FROM Student1;

O		ъ.	-	 я.
u	ш	п	0	ш

SQL query successfully executed. However, the result set is empty.

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

SELECT Student.name, Student1.name

FROM Student

INNER JOIN Student1 ON Student.RollNumber = Student1.R_no;

SELECT Student.name, Student1.name

FROM Student

RIGHT JOIN Student1 ON Student.student_RollNumber = Student1.Student.R_no

SELECT Student.name, Student1.name

FROM Student

LEFT JOIN Student1 ON Student.Student_RollNumber = Student1.Student_R_no;

SELECT Student.name, Student1.name

FROM Student

FULL JOIN Student1 ON Student.Student_RollNumber = Student1.Student_R_no;

Q15. To Write a PL/SQL block to calculate total salary of employee having employee number 100.

```
Run SQL
                       < Input
                                                                         0 6 :
EMPLOYEE [-]
                          -- Step 1: Create the EMPLOYEE table
   EMP_ID [int]
                          CREATE TABLE IF NOT EXISTS EMPLOYEE (
   EMP_NAME
                              EMP_ID INT PRIMARY KEY,
   [varchar(100)]
                              EMP_NAME VARCHAR(100),
  - SALARY [decimal(10, 2)]
                              SALARY DECIMAL(10, 2),
 - BONUS [decimal(10, 2)]
                              BONUS DECIMAL(10, 2)
                          );
                          -- Step 2: Insert sample data
Numbers [-]
                          INSERT INTO EMPLOYEE (EMP_ID, EMP_NAME, SALARY, BONUS) VALUES (100, 'John Doe',
                          50000, 5000);
   numl [int]
                          INSERT INTO EMPLOYEE (EMP_ID, EMP_NAME, SALARY, BONUS) VALUES (102, 'Alice
   num2[int]
                          Johnson', 70000, 7000);
   num3 [int]
                          INSERT INTO EMPLOYEE (EMP_ID, EMP_NAME, SALARY, BONUS) VALUES (103, 'Bob
                          Brown', 55000, 5500);
                          INSERT INTO EMPLOYEE (EMP_ID, EMP_NAME, SALARY, BONUS) VALUES (104, 'Charlie
                          Davis', 80000, 8000);
 INSERT INTO EMPLOYEE (EMP_ID, EMP_NAME, SALARY, BONUS) VALUES (105, 'Diana
 Prince', 90000, 9000);
 INSERT INTO EMPLOYEE (EMP_ID, EMP_NAME, SALARY, BONUS) VALUES (106, 'Evan
 Thomas', 45000, 4500);
 -- Step 3: Calculate total salary for employee with EMP ID = 100
 SELECT
      EMP ID,
      EMP NAME,
      (SALARY + BONUS) AS TOTAL_SALARY
 FROM
      EMPLOYEE
 WHERE
      EMP ID = 100;;
```

EMPLOYEE

EMP_ID	EMP_NAME	SALARY	BONUS
100	John Doe	50000	5000
102	Alice Johnson	70000	7000
103	Bob Brown	55000	5500
104	Charlie Davis	80000	8000
105	Diana Prince	90000	9000
106	Evan Thomas	45000	4500

EMP_ID	EMP_NAME	TOTAL_SALARY
100	John Doe	55000

Q16. To Write a PL/SQL code to find the greatest of three numbers.

```
Run SQL
                        < Input
EMPLOYEE [-]
                           -- Create a table to store three numbers
   EMP_ID [int]
                           CREATE TABLE IF NOT EXISTS Numbers (
   EMP_NAME
                               num1 INT,
   [varchar(100)]
                               num2 INT,
   SALARY [decimal(10, 2)]
                               num3 INT
  BONUS [decimal(10, 2)]
                           );
                           -- Insert sample data
Mumbers [-]
                           INSERT INTO Numbers (num1, num2, num3) VALUES (15, 25, 20);
   num1[int]
                           -- Use a CASE statement to find the greatest number
   num2[int]
                           SELECT
  num3 [int]
                               num1,
                               num2,
                               num3,
                               CASE
            WHEN num1 >= num2 AND num1 >= num3 THEN num1
            WHEN num2 >= num1 AND num2 >= num3 THEN num2
            ELSE num3
       END AS greatest_number
 FROM
       Numbers;
```

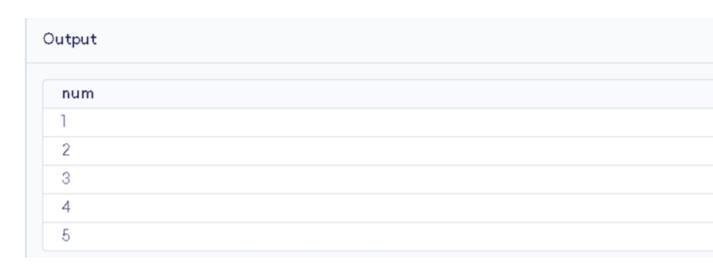
Numbers

num1	num2	num3
15	25	20

num1	num2	num3	greatest_number
15	25	20	25

Q17. To Write a PL/SQL code to find the greatest of three numbers.

```
C & Run So
                        < Input
EMPLOYEE [-]
                            -- Define the value of n
   EMP_ID [int]
                            WITH RECURSIVE numbers AS (
   EMP_NAME
                                SELECT 1 AS num -- Starting number
  [varchar(100)]
                                UNION ALL
 - SALARY [decimal(10, 2)]
                                SELECT num + 1 FROM numbers WHERE num < 5 -- Change 5 to any n value
 BONUS [decimal(10, 2)]
                            SELECT num FROM numbers;
Mumbers [-]
   num1[int]
   num2[int]
   num3 [int]
```



Q18. To Write a PL/SQL code to reverse a string using for loop.

```
C & :
                                                                                                 Run SQL
                         < Input
EMPLOYEE [-]
                             -- Input string to reverse
   EMP_ID [int]
                             WITH RECURSIVE reverse_string (original_str, reversed_str, position) AS (
   EMP_NAME
                               -- Initialize with the string, empty reversed string, and starting position
   [varchar(100)]
                               SELECT 'hello', '', LENGTH('hello')
   SALARY [decimal(10, 2)]
                               UNION ALL
   BONUS [decimal(10, 2)]
                               -- Concatenate the last character from the string to the reversed string
                               SELECT original_str, reversed_str || SUBSTR(original_str, position, 1),
                             position - 1
Mumbers [-]
                               FROM reverse_string
                               WHERE position > 0
   num1[int]
  num2[int]
                             -- Final output of the reversed string
  num3 [int]
                             SELECT reversed_str FROM reverse_string WHERE position = 0;
                            Output
```

```
Output

reversed_str
olleh
```

Q19. To Write a PL/SQL code to find the sum of n numbers.

```
C 6 :
                        < Input
                                                                                              Run SQL
EMPLOYEE [-]
                            -- Define the value of n
  EMP_ID [int]
                            WITH RECURSIVE sum_numbers (num, sum) AS (
   EMP_NAME
                             -- Starting with the first number and sum as 0
   [varchar(100)]
  SALARY [decimal(10, 2)]
                              SELECT 1, 1
                              UNION ALL
 BONUS [decimal(10, 2)]
                              -- Add the next number to the sum
                              SELECT num + 1, sum + (num + 1)
                              FROM sum_numbers
Mumbers [-]
                              WHERE num < 1011 -- Change 101 to any n value
   num1[int]
                            -- Final output of the sum
   num2[int]
                            SELECT sum FROM sum_numbers WHERE num = 101;
 - num3 [int]
```



Q20. To Consider a PL/SQL code to display the empno, ename, job of employees of department number 10.

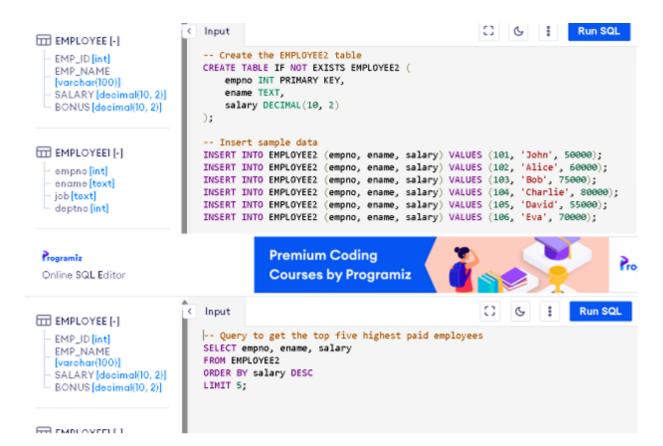
```
Run SQL
                                                                          C 6 :
                       < Input
EMPLOYEE [-]
                          -- Create the EMPLOYEE1 table
  EMP_ID [int]
                          CREATE TABLE IF NOT EXISTS EMPLOYEE1 (
  EMP_NAME
                              empno INT PRIMARY KEY,
  [varchar(100)]
                              ename TEXT,
  SALARY [decimal(10, 2)]
                              job TEXT,
 BONUS [decimal(10, 2)]
                              deptno INT
                          );
EMPLOYEEI [-]
                          -- Insert sample data
                          INSERT INTO EMPLOYEE1 (empno, ename, job, deptno) VALUES (101, 'John',
  empno [int]
                          'Manager', 10);
  ename [text]
                          INSERT INTO EMPLOYEE1 (empno, ename, job, deptno) VALUES (102, 'Alice',
  job [text]
  - deptno [int]
                          INSERT INTO EMPLOYEE1 (empno, ename, job, deptno) VALUES (103, 'Bob',
                          'Developer', 10);
  INSERT INTO EMPLOYEE1 (empno, ename, job, deptno) VALUES (104, 'Charlie',
  'Analyst', 10);
  -- Query to display empno, ename, and job for employees of department 10
  SELECT empno, ename, job
  FROM EMPLOYEE1
  WHERE deptno = 10;
```

EMPLOYEE1

empno	ename	job	deptno
101	John	Manager	10
102	Alice	Clerk	20
103	Bob	Developer	10
104	Charlie	Analyst	10

empno	ename	job		
101	John	Manager		
103	Bob	Developer		
104	Charlie	Analyst		

Q21. To Consider a PL/SQL code to display the employee number & name of top five highest paid employees.



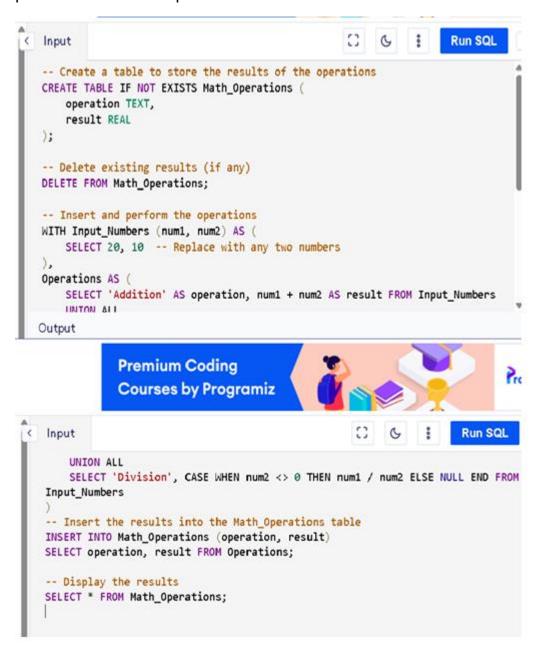
Output

empno	ename	salary
104	Charlie	80000
103	Bob	75000
106	Eva	70000
102	Alice	60000
105	David	55000

EMPLOYEE2

empno	ename	salary
101	John	50000
102	Alice	60000
103	Bob	75000
104	Charlie	80000
105	David	55000
106	Eva	70000

Q22. To Consider a PL/SQL procedure that accepts 2 numbers & return addition, subtraction, multiplication & division of two numbers using stored procedure AND local procedure.

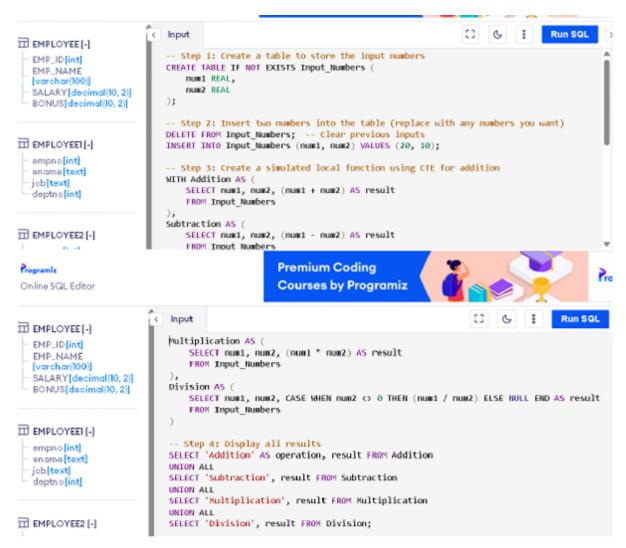


Math_Operations

operation	result
Addition	30
Subtraction	10
Multiplication	200
Division	2

operation	result
Addition	30
Subtraction	10
Multiplication	200
Division	2

Q23. To Consider a PL/SQL code that accepts 2 numbers & return addition, subtraction, multiplication & division of two numbers using stored functions and local function.



Input_Numbers

num1	num2
20	10

operation	result
Addition	30
Subtraction	10
Multiplication	200
Division	2

Q24.To Write a PL/SQL block to show the use of NO_DATA FOUND exception.

```
C G : Run SQL
                            < Input
EMPLOYEE[-]
                               -- Step 1: Create the EMPLOYEE4 table
   EMP_ID[int]
                               CREATE TABLE IF NOT EXISTS EMPLOYEES (
   EMP_NAME
                                  empno INT PRIMARY KEY,
   [varchar(100)]
                                   ename TEXT,
   SALARY[decimal(10, 2)]
 BONUS[decimal(10, 2)]
                                   salary DECIMAL(10, 2)
                               );
                                -- Step 2: Insert sample data
EMPLOYEET[-]
                               DELETE FROM EMPLOYEE3;
                               INSERT INTO EMPLOYEE3 (empno, ename, salary) VALUES (101, 'John', 50000); INSERT INTO EMPLOYEE3 (empno, ename, salary) VALUES (102, 'Alice', 60000); INSERT INTO EMPLOYEE3 (empno, ename, salary) VALUES (103, 'Bob', 75000);
   empno[int]
   ename [text]
   job[text]
   deptno[int]
                                - Step 3: Simulate NO_DATA_FOUND using a SELECT query
                               WITH Employee_Check AS (
                                   SELECT ename, salary
EMPLOYEE2 [-]
                                   FROM EMPLOYEE3
                                   WHERE emono = 999 -- This emono does not exist, simulating NO DATA FOUND
  -- Check if the query returned any results
 SELECT
       CASE
             WHEN EXISTS (SELECT 1 FROM Employee_Check)
             THEN (SELECT 'Employee Found: ' || ename || ', Salary: ' || salary FROM
 Employee Check)
             ELSE 'NO_DATA_FOUND: No employee found with the given employee number'
       END AS result;
```

EMPLOYEE3

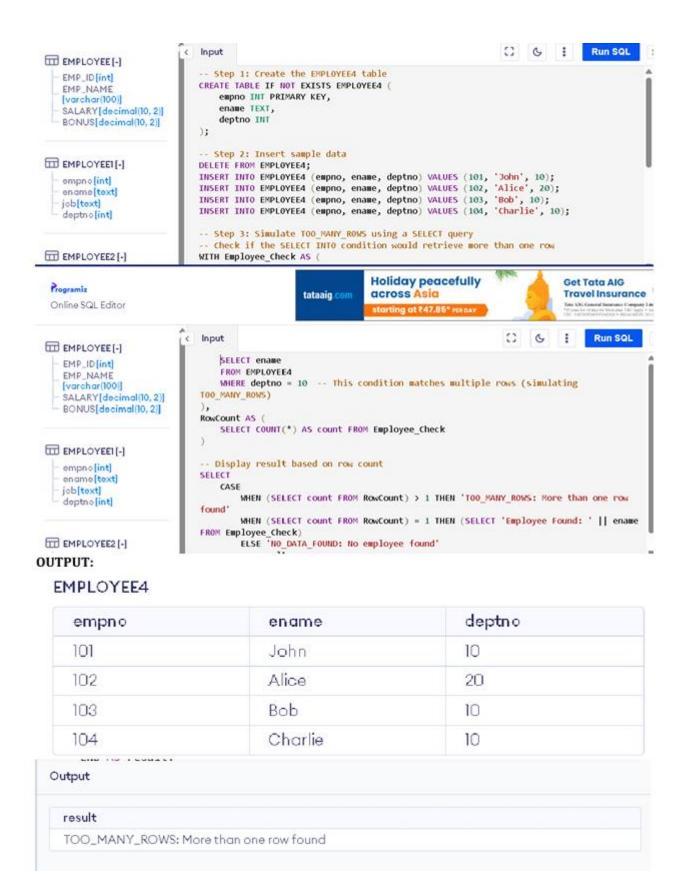
empno	ename	salary	
101	John	50000	
102	Alice	60000	
103	Bob	75000	

Output

result

NO_DATA_FOUND: No employee found with the given employee number

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



Q26. To Write a PL/SQL block to show the use of ZERO_DIVIDE exception.

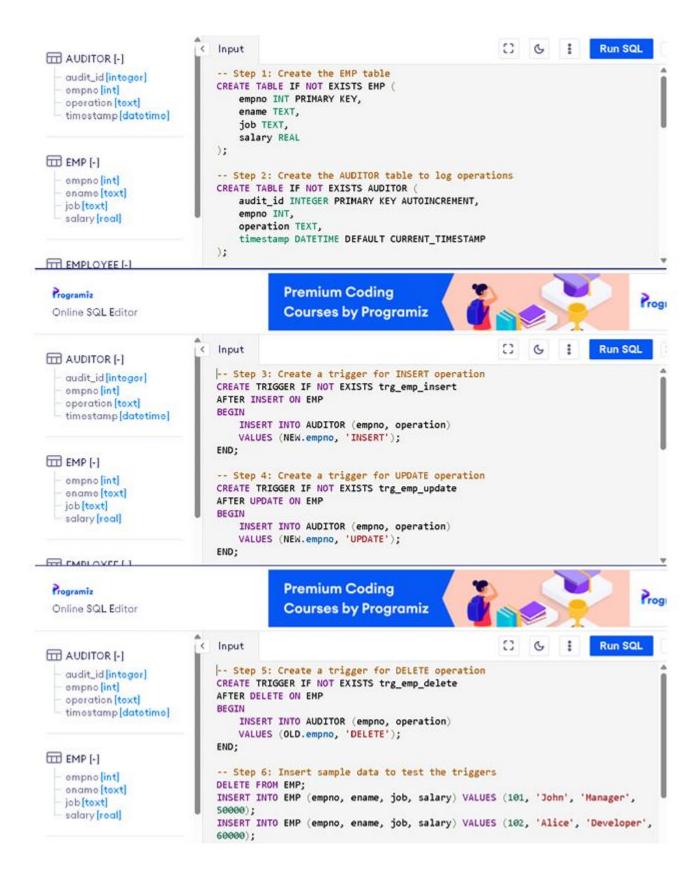
```
C & Run SQL
                           < Input
EMPLOYEE [-]
                              -- Step 1: Create a table to store the numbers
   EMP_ID (int)
                              CREATE TABLE IF NOT EXISTS Numbers (
   EMP_NAME
                                num1 REAL,
   [varchar(100)]
                                 num2 REAL
   SALARY [decimal(10, 2)]
   BONUS (decimal(10, 2))
                              -- Step 2: Insert sample data
                              DELETE FROM Numbers;
EMPLOYEEI [-]
                              INSERT INTO Numbers (num1, num2) VALUES (100, 0); -- Division by zero scenario
                              INSERT INTO Numbers (num1, num2) VALUES (200, 10); -- Normal division
   empno [int]
   ename [text]
                              -- Step 3: Simulate ZERO_DIVIDE using a SELECT query
   job (text)
                              SELECT
   deptno [int]
                                  num1,
                                  num2,
EMPLOYEE2 [-]
                                     WHEN num2 = 0 THEN 'ZERO_DIVIDE: Division by zero is not allowed'z
                                     ELSE 'Result: ' || (num1 / num2)
   empno [int]
                                  END AS result
   ename (text)
                              FROM Numbers;
 salary [decimal[10, 2]]
```

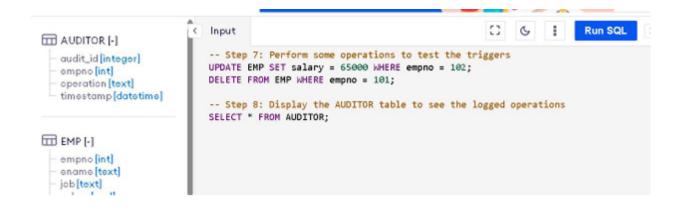
OUTPUT:

Numbers

numl		num2	num3
100		0	
200		10	
tput			
	num2	result	
numl	num2	result ZERO_DIVIDE: Division	by zero is not allowed

Q27. To create a trigger on the emp table, which store the empno& operation in the table auditor for each operation i.e. Insert, Update & Delete.





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			

2024-11-10 14:42:06

Q28. To create a trigger so that no operation can be performed on emp table.

DELETE

101

```
-- Step 1: Create the EMP table
  audit_id[integer]
                              CREATE TABLE IF NOT EXISTS EMP (
   empno [int]
                                  empno INT PRIMARY KEY,
   operation [text]
                                  ename TEXT,
   timestamp[datotime]
                                  job TEXT,
                                  salary REAL
                              );
-- Step 2: Create a trigger to block INSERT operation
   empno [int]
                              CREATE TRIGGER IF NOT EXISTS trg_block_insert
   ename [text]
                              BEFORE INSERT ON EMP
   job [text]
                              BEGIN
   salary [real]
                                  SELECT RAISE(ABORT, 'INSERT operation is not allowed on EMP table');
                              END;
                                           Premium Coding
 Programiz
                                                                                                          rogi
 Online SQL Editor
                                           Courses by Programiz
                                                                                                   Run SQL
                              Input
                                                                                       6
AUDITOR [-]
                               -- Step 3: Create a trigger to block UPDATE operation
   audit_id[integer]
                              CREATE TRIGGER IF NOT EXISTS trg_block_update
   empno [int]
                              BEFORE UPDATE ON EMP
   operation [text]
                              BEGIN
   timestamp[datetime]
                                  SELECT RAISE(ABORT, 'UPDATE operation is not allowed on EMP table');
-- Step 4: Create a trigger to block DELETE operation
                              CREATE TRIGGER IF NOT EXISTS trg_block_delete
   empno [int]
                              BEFORE DELETE ON EMP
   ename [text]
   job (text)
                                  SELECT RAISE(ABORT, 'DELETE operation is not allowed on EMP table');
   salary [real]
                              END;
                                           Premium Coding
 Programiz
                                                                                                           Prog
 Online SQL Editor
                                           Courses by Programiz
                                                                                       0
                                                                                                   Run SQL
                              Input
                                                                                             ŧ
AUDITOR [-]
                               -- Step 5: Attempt to perform some operations to see the triggers in action
   audit_id[integer]
   empno [int]
                               -- Attempt to insert a record (This should fail)
   operation [text]
                              INSERT INTO EMP (empno, ename, job, salary) VALUES (101, 'John', 'Manager',
   timestamp[datetime]
                              50000);
                               -- Attempt to update a record (This should fail)
UPDATE EMP SET salary = 60000 WHERE empno = 101;
   empno [int]
                               -- Attempt to delete a record (This should fail)
   ename [text]
                              DELETE FROM EMP WHERE empno = 101;
   job [text]
   salary [real]
```

Output

Error: INSERT operation is not allowed on EMP table