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	Archit Saxena	
Roll No	22/FCA/BCA(AIML)/043	
Programme	Bachelor of Computer Applications	
Semester	5 th Semester	
Section	D	
Department	Computer Applications	
Batch	2022-25	
Submitted To		
Faculty Name	Mrs. Neerja Negi	



SCHOOL OF COMPUTER APPLICATIONS

AIM: Create the following table.

Customer

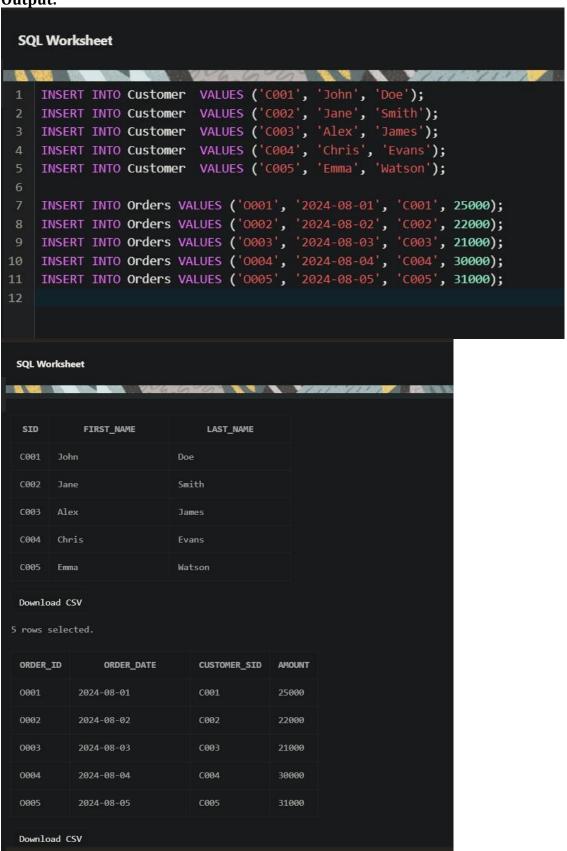
Column_name	Data type	<u>Size</u>	Constraint
SID	Varchar2	4	Primary Key
First_Name	Char	20	
Last_name	Char	20	

Orders

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

```
SQL Worksheet
 1 v CREATE TABLE Customer
        (
        SID VARCHAR2(4) PRIMARY KEY,
        First_Name CHAR(20),
        Last name CHAR(20)
6 );
8 v CREATE TABLE Orders
        (
        Order_ID VARCHAR2(4) PRIMARY KEY,
10
        Order date CHAR(20),
11
        Customer_SID VARCHAR2(4),
12
        Amount NUMBER CHECK (Amount > 20000),
13
        FOREIGN KEY (Customer SID) REFERENCES Customer(SID)
15
    );
16
Table created.
Table created.
```

AIM: Insert 5 records for each table.



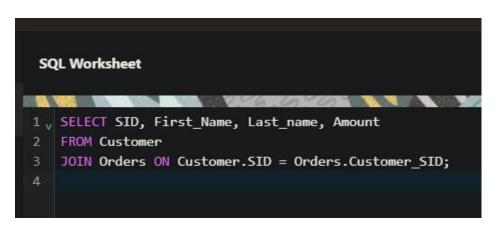
AIM: Customer SID column in the ORDERS table is a foreign key pointing to the SID column in the CUSTOMER table.

Output:

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

EXERCISE 4

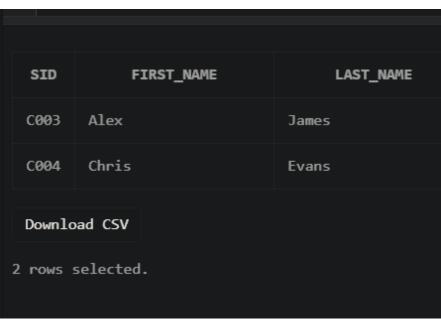
AIM: List the details of the customers along with the amount.



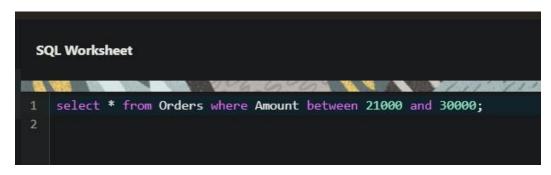
SID	FIRST_NAME	LAST_NAME	AMOUNT
C001	John	Doe	25000
C002	Jane	Smith	22000
C003	Alex	James	21000
C004	Chris	Evans	30000
C005	Етта	Watson	31000
Downlo	ad CSV		

AIM: List the customers whose names end with "s".





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



ORDER_ID ORDER_DATE CUSTOMER_SID AMOUNT 0001 2024-08-01 C001 25000 0002 2024-08-02 C002 22000 0003 2024-08-03 C003 21000 0004 2024-08-04 C004 30000 Download CSV 4 rows selected.				
0001 2024-08-01 C001 25000 0002 2024-08-02 C002 22000 0003 2024-08-03 C003 21000 0004 2024-08-04 C004 30000 Download CSV				
0002 2024-08-02 C002 22000 0003 2024-08-03 C003 21000 0004 2024-08-04 C004 30000 Download CSV	ORDER_ID	ORDER_DATE	CUSTOMER_SID	AMOUNT
0003 2024-08-03 C003 21000 0004 2024-08-04 C004 30000 Download CSV	0001	2024-08-01	C001	25000
0004 2024-08-04 C004 30000 Download CSV	0002	2024-08-02	C002	22000
Download CSV	0003	2024-08-03	C003	21000
	0004	2024-08-04	C004	30000
1 rows selected.	Download (csv		
4 rows selected.				
	4 rows sele	cted.		

 $\boldsymbol{\text{AIM:}}$ List the orders where amount is increased by 500 and replace with name "new amount".

```
SQL Worksheet

1 update Orders set Amount = Amount + 500;
2
3 select Order_ID, Amount as "New Amount" from Orders;
4
```



AIM: Display the order_id and total amount of orders. **Output:**

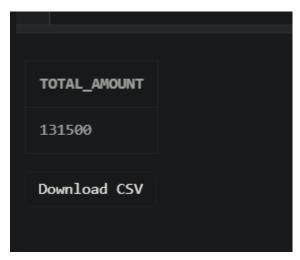
```
SQL Worksheet

1 v SELECT Order_ID, SUM(Amount) AS Total_Amount
2 FROM Orders
3 GROUP BY Order_ID;
4
```



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





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	<u>Size</u>	Constraint
RollNo	Varchar2	20	Primary Key
Name	Char	20	
Class	Varchar2	20	
Marks	Number	6,2	

Student1

Column_name	Data type	<u>Size</u>	Constraint
R_No	Varchar2	20	Primary Key
Name	Char	20	
Class	Varchar2	20	
Marks	Number	6,2	

```
SQL Worksheet
1, create table Student
    (
        RollNo varchar(20) primary key,
        Name char(20),
        Class varchar(20),
        Marks number(6,2)
    );
9 v create table Student1
        R_No varchar(20) primary key,
        Name char(20),
        Class varchar(20),
        Marks number(6,2)
    );
Table created.
Table created.
```

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 5 records into the Student table

INSERT INTO Student (RollNo, Name, Class, Marks)

VALUES

('S101', 'John', '10th', 85.50),
('S102', 'Alice', '11th', 90.00),
('S103', 'Bob', '12th', 75.75),
('S104', 'Charlie', '10th', 88.00),
('S105', 'David', '11th', 92.50);

-- Insert 5 records into the Student1 table (with some common entries)

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

VALUES

('S201', 'Eve', '10th', 80.25),
('S202', 'Frank', '12th', 70.50),
('S103', 'Bob', '12th', 75.75), -- Common entry with Student
('S104', 'Charlie', '10th', 88.00), -- Common entry with Student
('S104', 'Charlie', '10th', 88.00), -- Common entry with Student
('S205', 'Isaac', '12th', 85.00);
```

Student

RollNo	Name	Class	Marks
S101	John	10th	85.5
S102	Alice	11th	90
S103	Bob	12th	75.75
S104	Charlie	10th	88
\$105	David	llth	92.5

Student1

R_No	Name	Class	Marks
S201	Eve	10th	80.25
S202	Frank	12th	70.5
S103	Bob	12th	75.75
S104	Charlie	10th	88
S205	Isaac	12th	85

Now union:

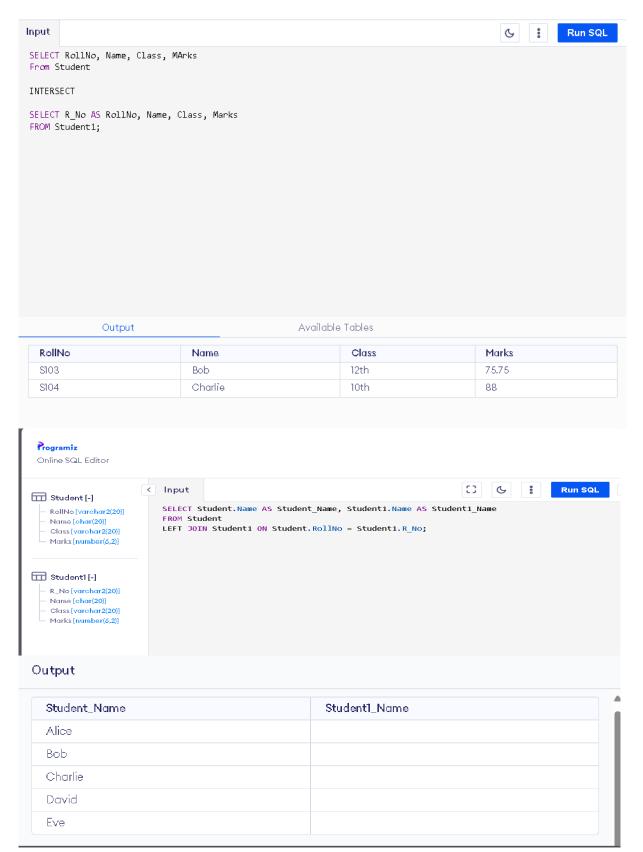
Programiz

Online SQL Editor



RollNo	Name	Class	Marks
S101	John	10th	85.5
S102	Alice	11th	90
S103	Bob	12th	75.75
S104	Charlie	10th	88
S105	David	11th	92.5
S201	Eve	10th	80.25
S202	Frank	12th	70.5
S205	Isaac	12th	85

EXERCISE 13AIM: Find out the intersection of student and student1 tables.



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

```
-- INNER JOIN to display names of students from both tables where there's a match
SELECT S.Name AS Student_Name, S1.Name AS Student1_Name
FROM Student S
INNER JOIN Student1 S1
ON S.Name = S1.Name;

Output

Student_Name

Bob
Bob
Charlie
Charlie
```

LEFT JOIN AND RIGHT JOIN

```
-- LEFT JOIN to display names from Student and corresponding names from Student1 (if any)

SELECT S.Name AS Student_Name, S1.Name AS Student1_Name

FROM Student S

LEFT JOIN Student1 S1

ON S.Name = S1.Name;

-- RIGHT JOIN simulation: Swap tables and use LEFT JOIN to simulate RIGHT JOIN

SELECT S.Name AS Student_Name, S1.Name AS Student1_Name

FROM Student S

LEFT JOIN Student1 S1

ON S.Name = S1.Name;
```

Student_Name	Student1_Name
John	
Alice	
Bob	Bob
Charlie	Charlie
David	

```
-- FULL JOIN simulation: Combine LEFT JOIN and RIGHT JOIN results

SELECT S.Name AS Student_Name, S1.Name AS Student1_Name

FROM Student S

LEFT JOIN Student1 S1

ON S.Name = S1.Name

UNION

SELECT S.Name AS Student_Name, S1.Name AS Student1_Name

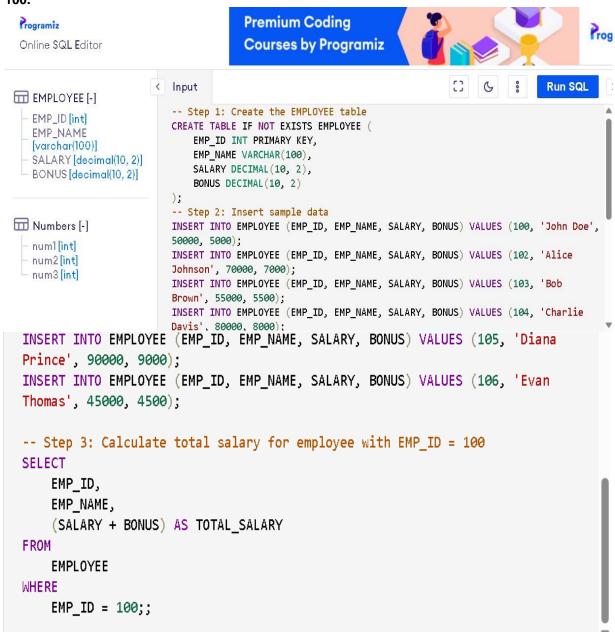
FROM Student S

LEFT JOIN Student1 S1

ON S.Name = S1.Name;
```

Student_Name	Student1_Name
Alice	
Bob	Bob
Charlie	Charlie
David	
John	

Exercise 15
AIM: To Write a PL/SQL block to calculate total salary of employee having employee number 100.



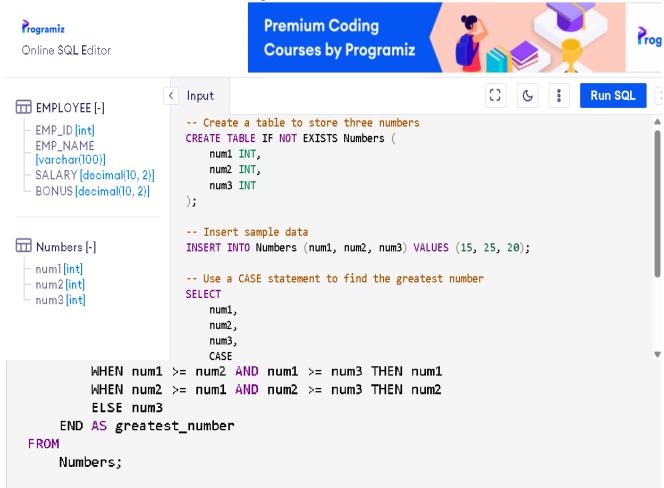
OUTPUT:

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

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



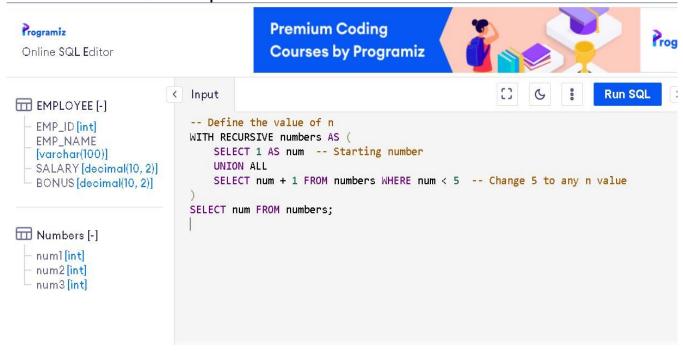
OUTPUT:

Numbers

num1	num2	num3
15	25	20

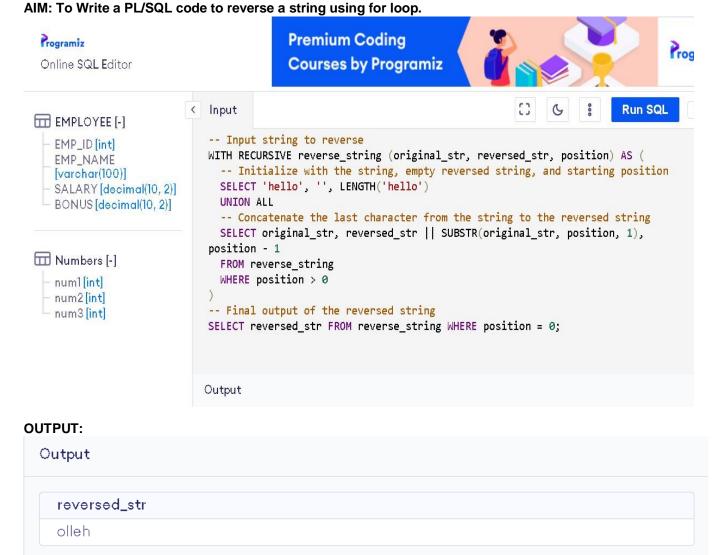
num1	num2	num3	greatest_number
15	25	20	25

EXERCISE 17 AIM: To Write a PL/SQL code to print the numbers from 1 to n.



OUTPUT:

Output			
num			
1			
2			
3			
4			
5			



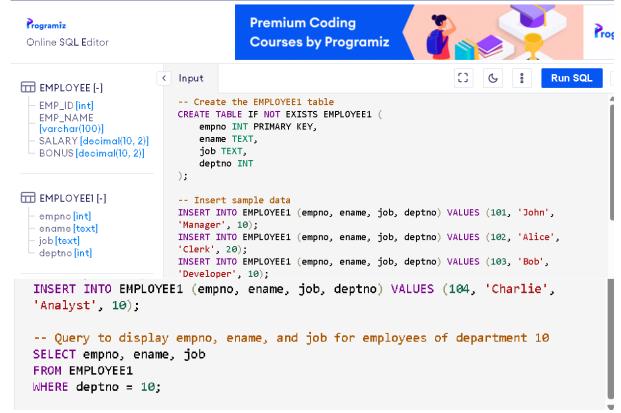
EXERCISE 19 AIM: To Write a PL/SQL code to find the sum of n numbers.



OUTPUT:

Output			
sum			
5151			

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



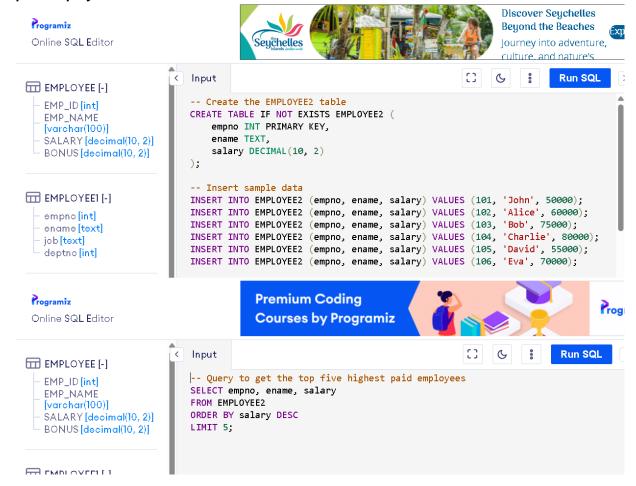
OUTPUT:

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	

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



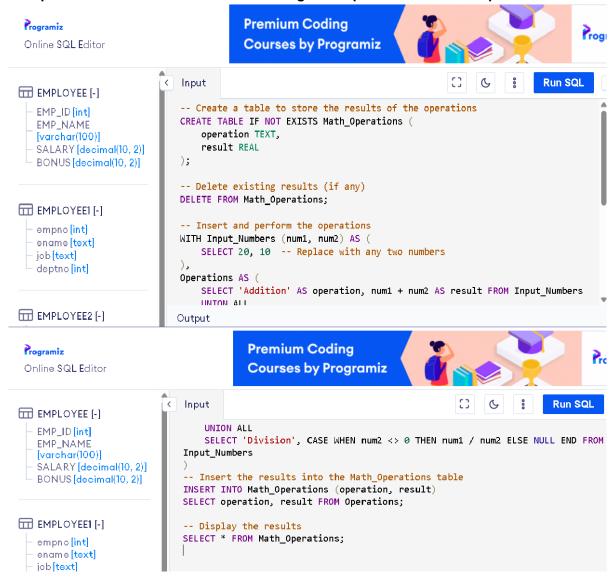
OUTPUT:

EMPLOYEE2

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

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

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.



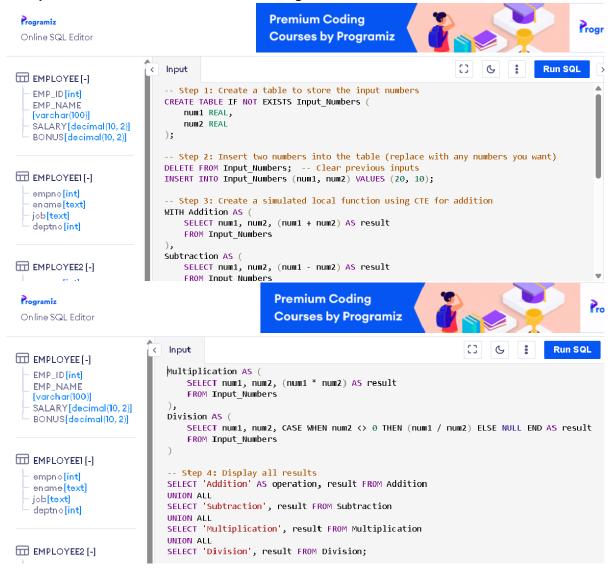
OUTPUT:

Math_Operations

operation	result
Addition	30
Subtraction	10
M ultiplication	200
Division	2

operation	result
Addition	30
Subtraction	10
Multiplication	200
Division	2

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.



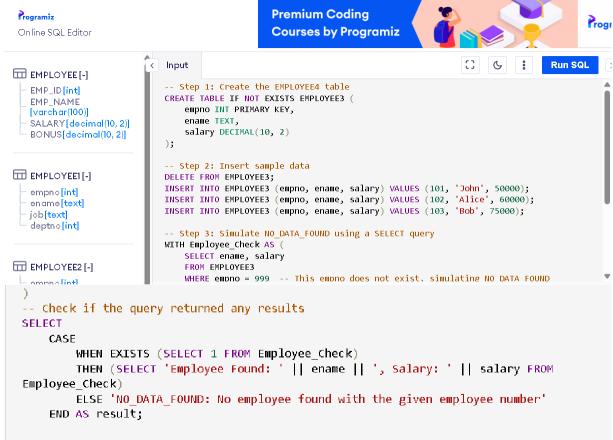
OUTPUT:

Input_Numbers

num1	num2
20	10

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.



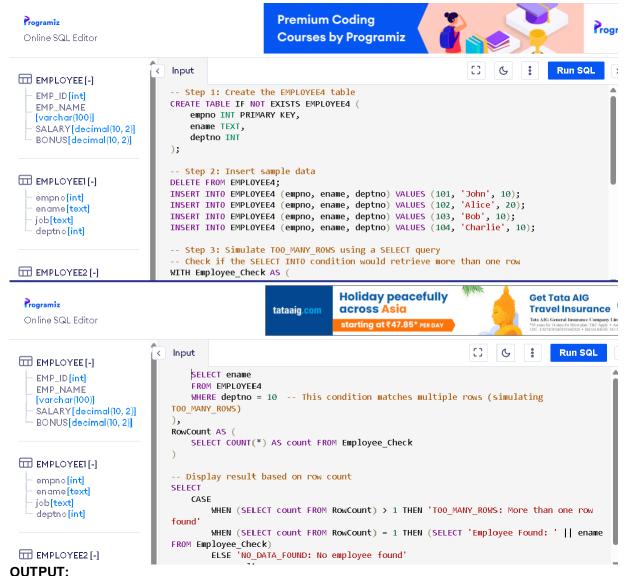
OUTPUT:

EMPLOYEE3

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

Presult NO_DATA_FOUND: No employee found with the given employee number

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



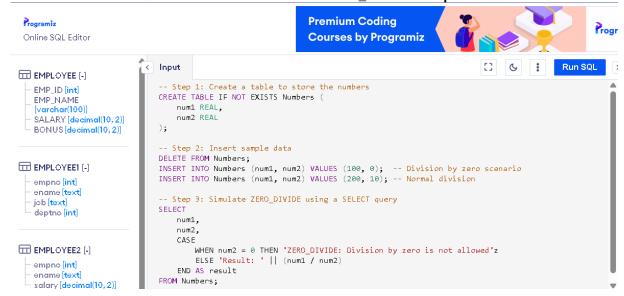
3011 01.

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.



OUTPUT:

Numbers

num1	num2	num3
100	0	
200	10	

num1	num2	result
100	0	ZERO_DIVIDE: Division by zero is not allowed
200	10	Result: 20

AIM: 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.





OUTPUT:

AUDITOR

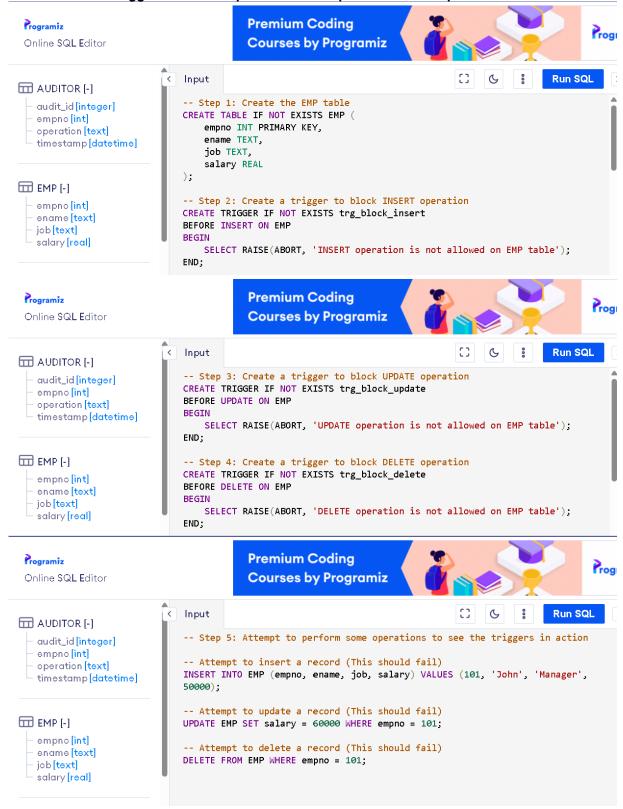
audit_id	empno	operation	timestamp
]	101	INSERT	2024-11-10 14:42:06
2	102	INSERT	2024-11-10 14:42:06
3	102	UPDATE	2024-11-10 14:42:06
4	101	DELETE	2024-11-10 14:42:06

EMP

empno	ename	job	salary
102	Alice	Developer	65000

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

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



OUTPUT:

Output

Error: INSERT operation is not allowed on EMP table