RDBMS LAB (BCA-DS-552)

Manav Rachna International Institute of Research and Studies

School of Computer Applications

Department of Computer Applications

Submitted By	
Student Name	Kunal Jha
Roll No	22/FCA/BCA(DS&BDA)/007
Program	Bachelor of Computer Applications
Semester	5th Semester
Section/Group	E/A
Department	Computer Applications
Batch	2022-25
Submitted To	
Faculty Name	Iram Fatima

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

SCHOOL OF COMPUTER APPLICATIONS

S. No.	Date	Aim of the Experiment	Signature/Date	Grade
		SQL		
1		Create the following tables and insert five records for each table. And make Customer_SID column in the ORDERS table a foreign key pointing to the SID column in the CUSTOMER table.		
1(a)		List the details of the customers along with the amount.		
1(b)		List the customers whose names end with "s".		
1(c)		List the orders where amount is between 2000 and 3000.		
1(d)		Calculate the total amount of orders that has more than 2500.		
1(e)		List the orders where amount is increased by 500 and replace with name "New Amount".		
2		Create the tables student and student1 and insert values.		
2(a)		Display all the contents of student and student1 using union clause.		
2(b)		Find out the intersection of student and student1 tables.		
2(c)		Display the names of student and student1 tables using left, right, inner and full join.		
3		Display all the string functions used in SQL.		
		TRIGGERS		
1		Create a trigger on the emp table, which store the empno & operation in the table auditor for each operation i.e. Insert, Update & Delete.		
2		Create a trigger so that no operation can be performed on emp table.		
		PL/SQL		
1		Write a PL/SQL block to calculate total salary of employee having employee number 100.		
2		Write a PL/SQL code to find the greatest of three numbers.		
3		Write a PL/SQL code to print the numbers from 1 to n.		

S. No.	Date	Aim of the Experiment	Signature/Date	Grade
4		Write a PL/SQL code to reverse a string using for loop.		
5		Write a PL/SQL code to find the sum of n numbers.		
6		Write a PL/SQL code to display the empno, ename, job of employees of department number 10.		
7		Write a PL/SQL code to display the employee number & name of top five highest paid employees.		
8		Write a PL/SQL procedure that accepts 2 numbers & return addition, subtraction, multiplication & division of two numbers using stored procedure AND local procedure.		
9		Write a PL/SQL block to show the use of NO_DATA FOUND exception.		
10		Write a PL/SQL block to show the use of TOO_MANY ROWS exception.		
11		Write a PL/SQL block to show the use of ZERO_DIVIDE exception.		

SQL

EXPERIMENT 1:

Create the following tables and insert five records for each table. And make Customer_SID column in the ORDERS table a foreign key pointing to the SID column in the CUSTOMER table.

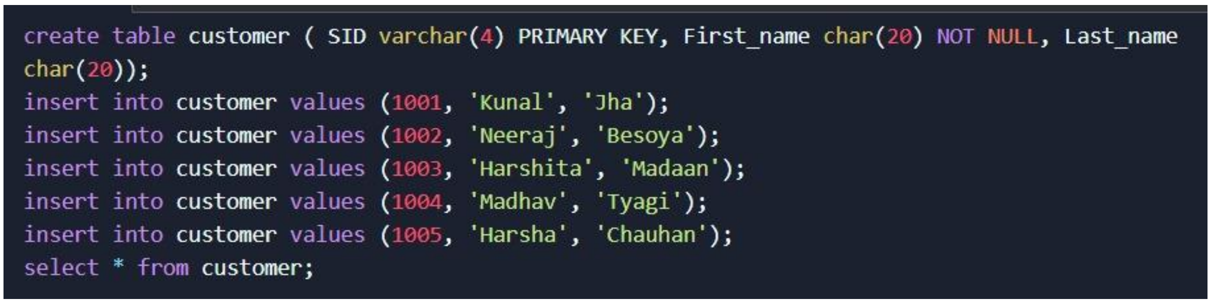
Customer

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

Orders

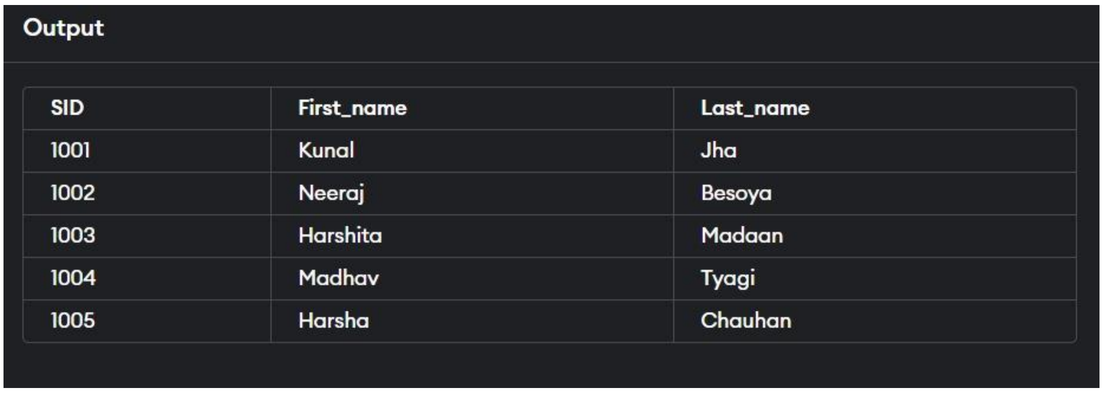
Column_name	**Data type**	**Size**	**Constraint**
Order_ID	Varchar2	4	Primary Key
Order_date	Char	20	
Customer_SID	Varchar2	20	Foreign Key
Amount	Number		Check > 20000

1. Customer table:



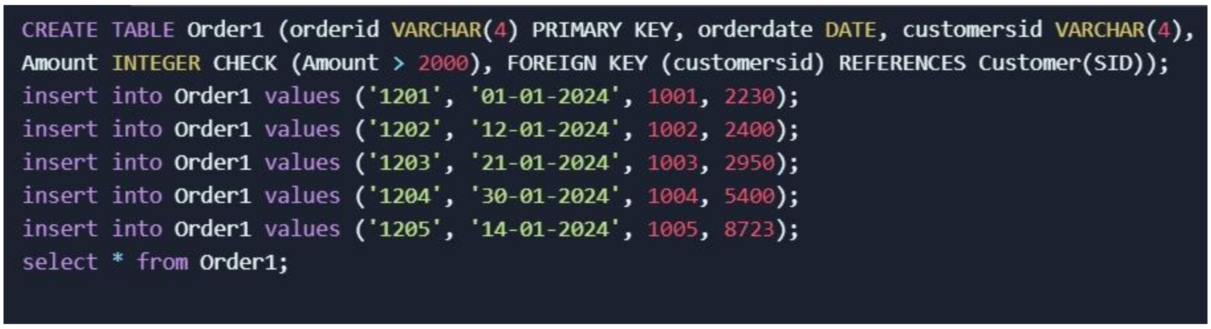
```sql
create table customer ( SID varchar(4) PRIMARY KEY, First_name char(20) NOT NULL, Last_name
char(20));
insert into customer values (1001, 'Kunal', 'Jha');
insert into customer values (1002, 'Neeraj', 'Besoya');
insert into customer values (1003, 'Harshita', 'Madaan');
insert into customer values (1004, 'Madhav', 'Tyagi');
insert into customer values (1005, 'Harsha', 'Chauhan');
select * from customer;
```

OUTPUT:



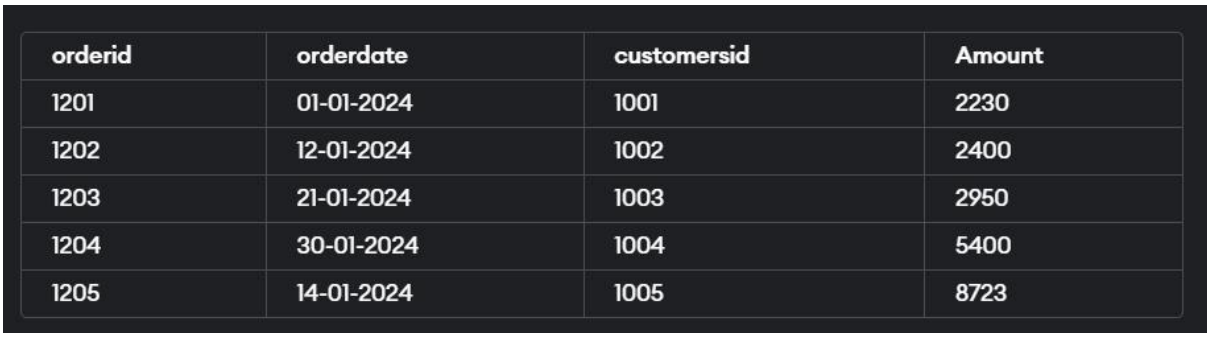
```
Output

SID     First_name     Last_name
1001    Kunal          Jha
1002    Neeraj         Besoya
1003    Harshita       Madaan
1004    Madhav         Tyagi
1005    Harsha         Chauhan
```

2. Order table:



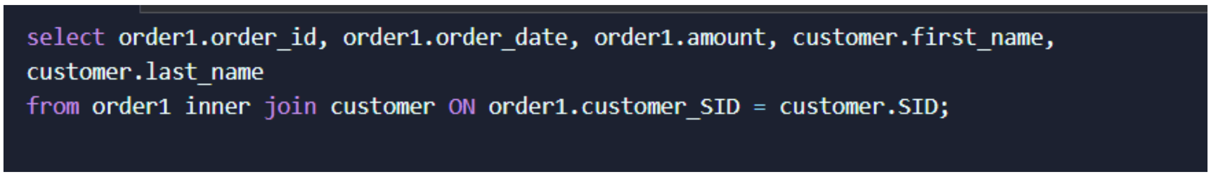
```
CREATE TABLE Order1 (orderid VARCHAR(4) PRIMARY KEY, orderdate DATE, customersid VARCHAR(4),
Amount INTEGER CHECK (Amount > 2000), FOREIGN KEY (customersid) REFERENCES Customer(SID));
insert into Order1 values ('1201', '01-01-2024', 1001, 2230);
insert into Order1 values ('1202', '12-01-2024', 1002, 2400);
insert into Order1 values ('1203', '21-01-2024', 1003, 2950);
insert into Order1 values ('1204', '30-01-2024', 1004, 5400);
insert into Order1 values ('1205', '14-01-2024', 1005, 8723);
select * from Order1;
```

OUTPUT:



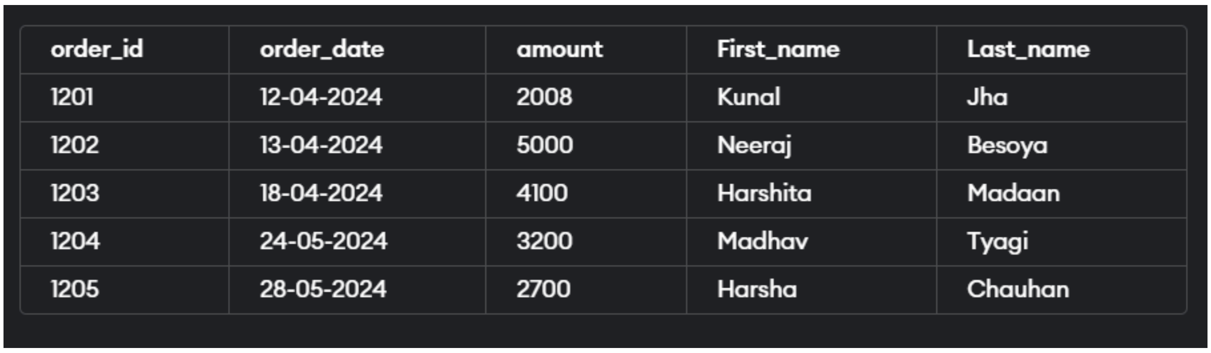
orderid	orderdate	customersid	Amount
1201	01-01-2024	1001	2230
1202	12-01-2024	1002	2400
1203	21-01-2024	1003	2950
1204	30-01-2024	1004	5400
1205	14-01-2024	1005	8723

- Joining of above two tables:



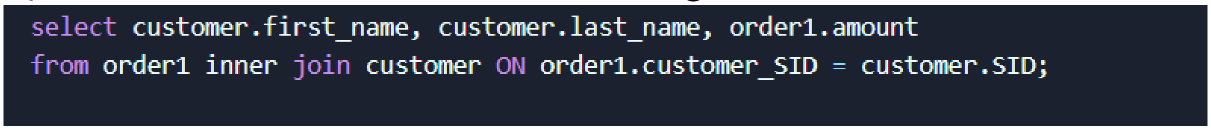
```
select order1.order_id, order1.order_date, order1.amount, customer.first_name,
customer.last_name
from order1 inner join customer ON order1.customer_SID = customer.SID;
```

OUTPUT:



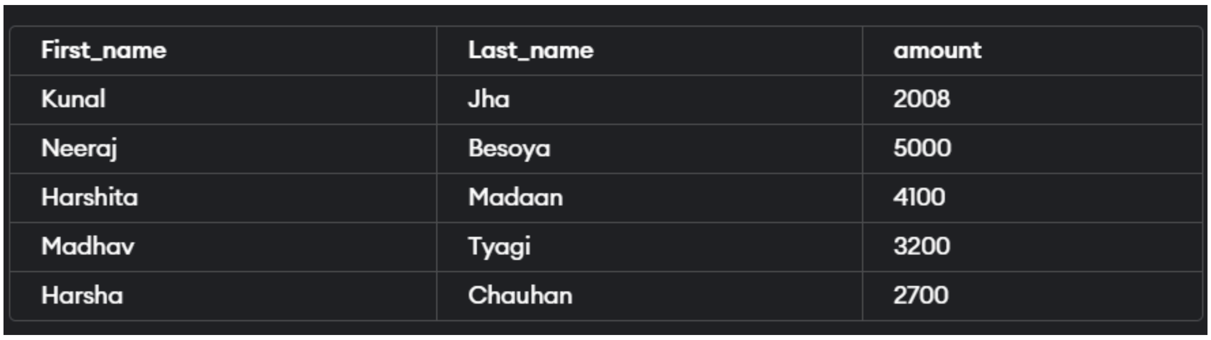
order_id	order_date	amount	First_name	Last_name
1201	12-04-2024	2008	Kunal	Jha
1202	13-04-2024	5000	Neeraj	Besoya
1203	18-04-2024	4100	Harshita	Madaan
1204	24-05-2024	3200	Madhav	Tyagi
1205	28-05-2024	2700	Harsha	Chauhan

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



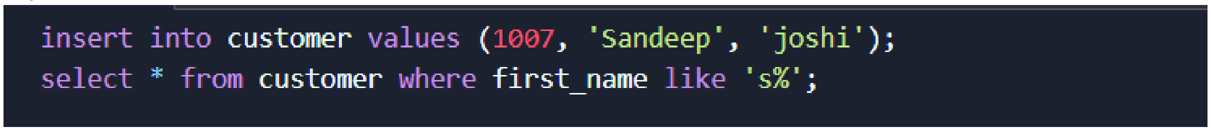
```sql
select customer.first_name, customer.last_name, order1.amount
from order1 inner join customer ON order1.customer_SID = customer.SID;
```

OUTPUT:



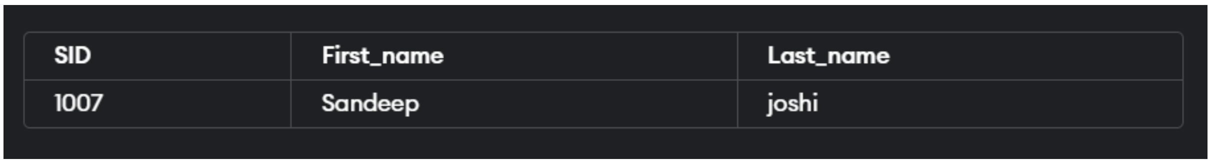
First_name	Last_name	amount
Kunal	Jha	2008
Neeraj	Besoya	5000
Harshita	Madaan	4100
Madhav	Tyagi	3200
Harsha	Chauhan	2700

Q. List the customers whose names end with “s”.



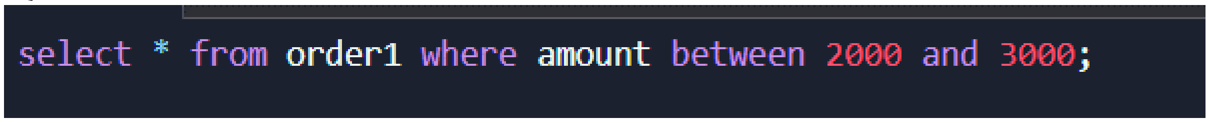
```sql
insert into customer values (1007, 'Sandeep', 'joshi');
select * from customer where first_name like 's%';
```

OUTPUT:



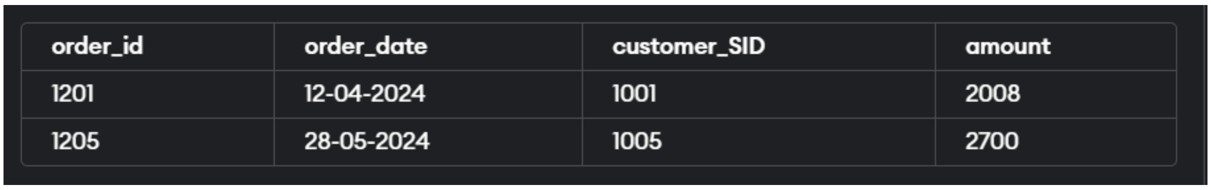
SID	First_name	Last_name
1007	Sandeep	joshi

Q. List the orders where amount is between 2000 and 3000.



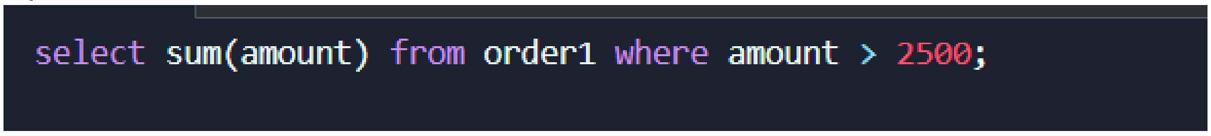
```sql
select * from order1 where amount between 2000 and 3000;
```

OUTPUT:



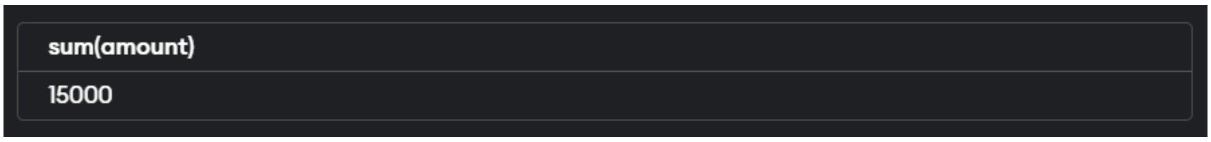
order_id	order_date	customer_SID	amount
1201	12-04-2024	1001	2008
1205	28-05-2024	1005	2700

Q. Calculate the total amount of orders that has more than 2500.



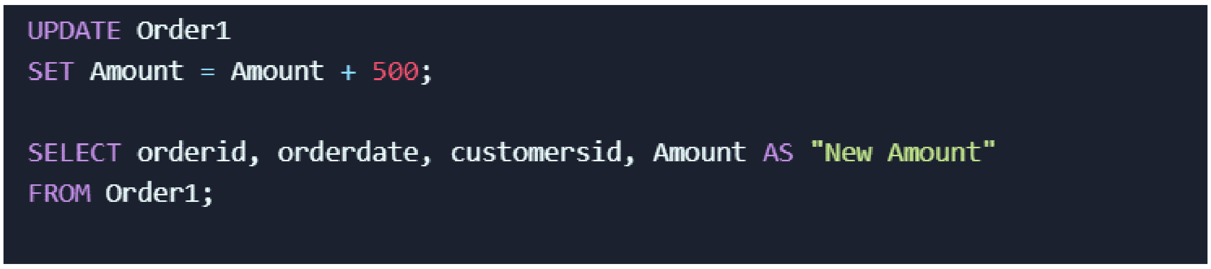
```sql
select sum(amount) from order1 where amount > 2500;
```

OUTPUT:



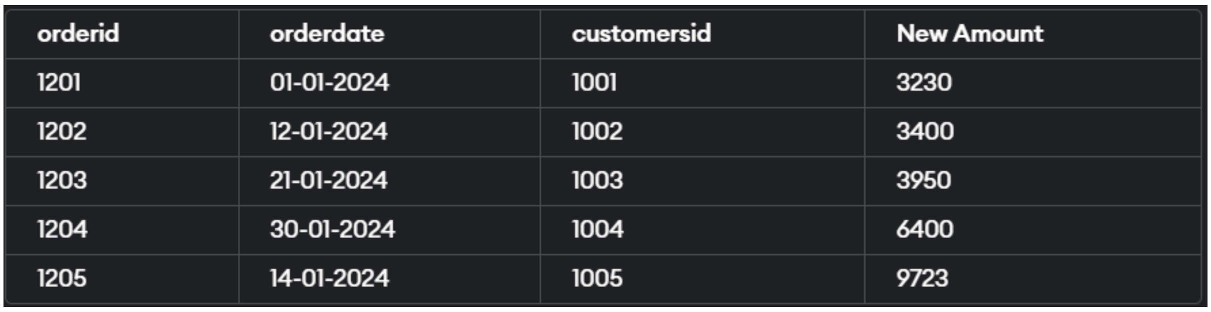
sum(amount)
15000

Q. List the orders where amount is increased by 500 and replace with name “New Amount”.



```sql
UPDATE Order1
SET Amount = Amount + 500;

SELECT orderid, orderdate, customersid, Amount AS "New Amount"
FROM Order1;
```

OUTPUT:



orderid	orderdate	customersid	New Amount
1201	01-01-2024	1001	3230
1202	12-01-2024	1002	3400
1203	21-01-2024	1003	3950
1204	30-01-2024	1004	6400
1205	14-01-2024	1005	9723

<u>EXPERIMENT 2</u>:

Create the following tables and insert values:

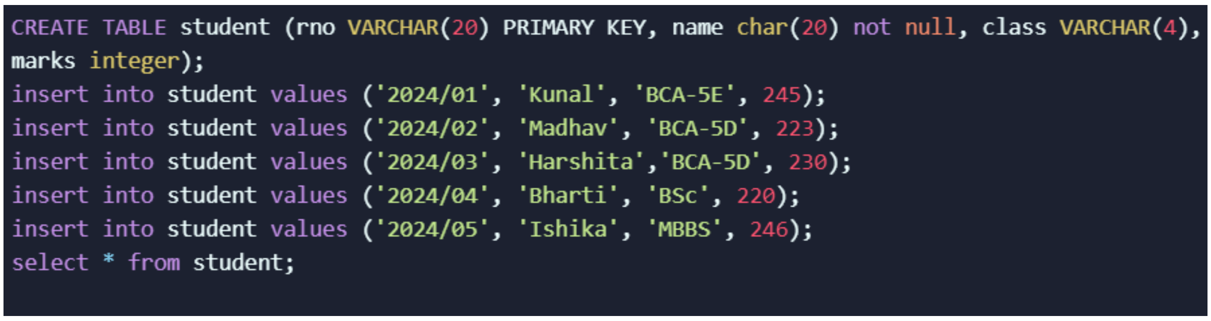
<u>Student</u>

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

<u>Student1</u>

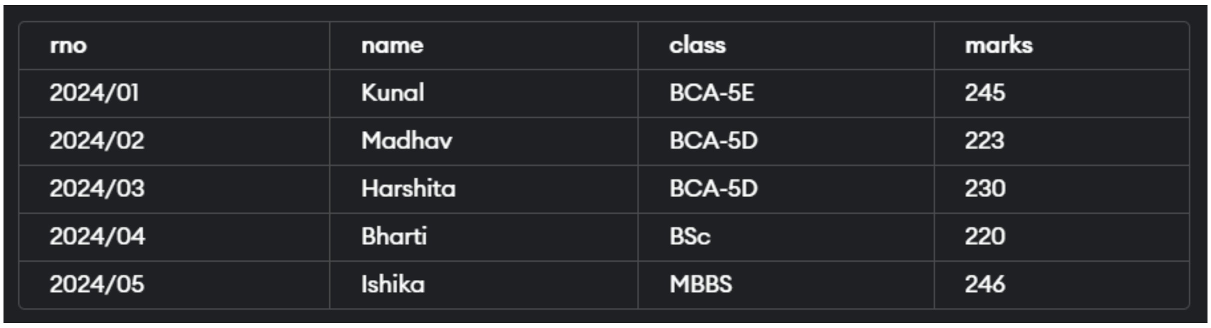
Column_name	**Data type**	**Size**	**Constraint**
R_No	Varchar2	20	Primary Key
Name	Char	20	
Class	Varchar2	20	
Marks	Number	6,2	

1. Student table:



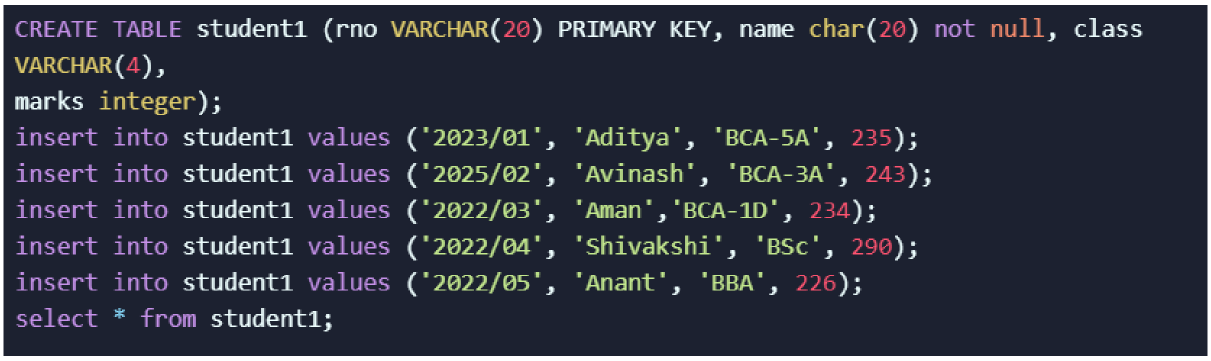
```sql
CREATE TABLE student (rno VARCHAR(20) PRIMARY KEY, name char(20) not null, class VARCHAR(4),
marks integer);
insert into student values ('2024/01', 'Kunal', 'BCA-5E', 245);
insert into student values ('2024/02', 'Madhav', 'BCA-5D', 223);
insert into student values ('2024/03', 'Harshita','BCA-5D', 230);
insert into student values ('2024/04', 'Bharti', 'BSc', 220);
insert into student values ('2024/05', 'Ishika', 'MBBS', 246);
select * from student;
```

<u>OUTPUT</u>:



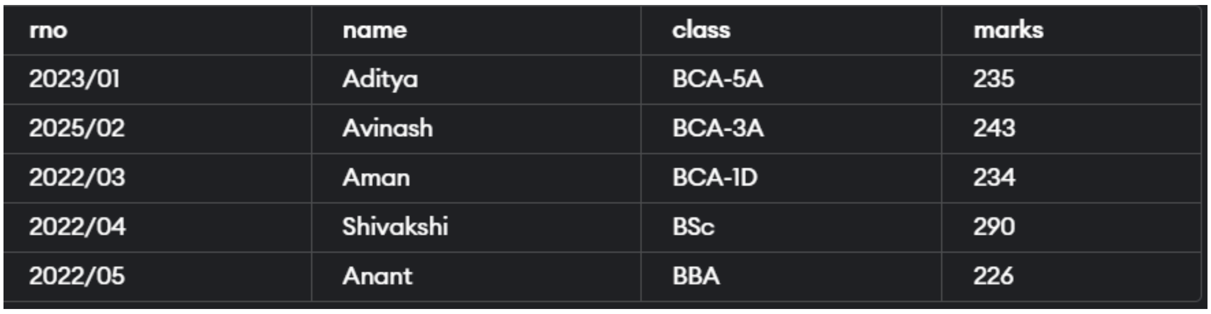
rno	name	class	marks
2024/01	Kunal	BCA-5E	245
2024/02	Madhav	BCA-5D	223
2024/03	Harshita	BCA-5D	230
2024/04	Bharti	BSc	220
2024/05	Ishika	MBBS	246

2. Student1 table:



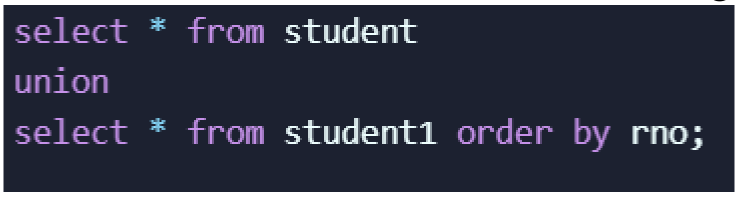
```
CREATE TABLE student1 (rno VARCHAR(20) PRIMARY KEY, name char(20) not null, class
VARCHAR(4),
marks integer);
insert into student1 values ('2023/01', 'Aditya', 'BCA-5A', 235);
insert into student1 values ('2025/02', 'Avinash', 'BCA-3A', 243);
insert into student1 values ('2022/03', 'Aman','BCA-1D', 234);
insert into student1 values ('2022/04', 'Shivakshi', 'BSc', 290);
insert into student1 values ('2022/05', 'Anant', 'BBA', 226);
select * from student1;
```

OUTPUT:



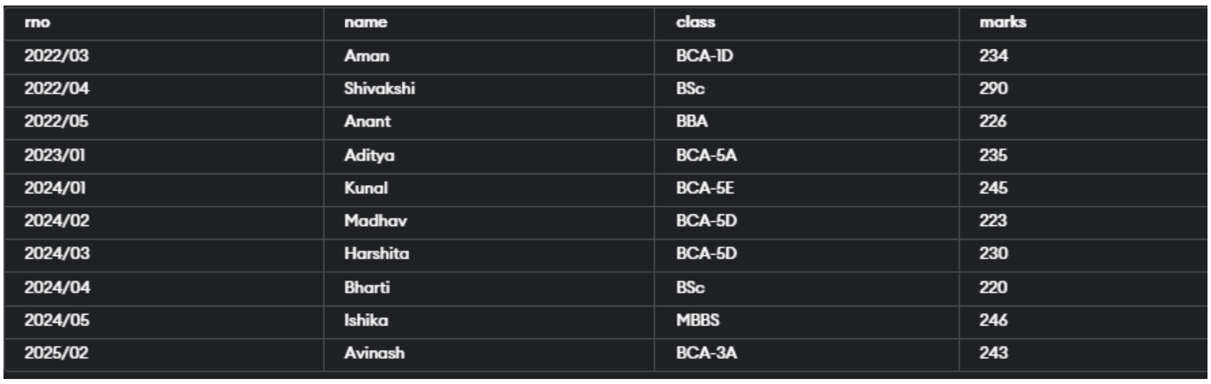
rno	name	class	marks
2023/01	Aditya	BCA-5A	235
2025/02	Avinash	BCA-3A	243
2022/03	Aman	BCA-1D	234
2022/04	Shivakshi	BSc	290
2022/05	Anant	BBA	226

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



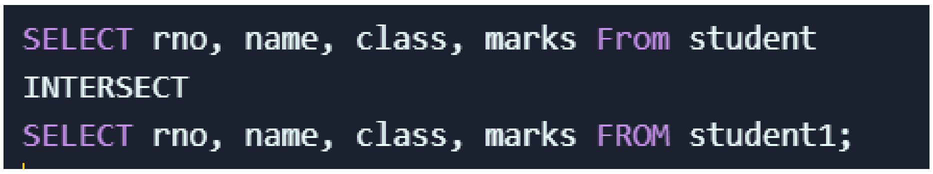
```
select * from student
union
select * from student1 order by rno;
```

OUTPUT:



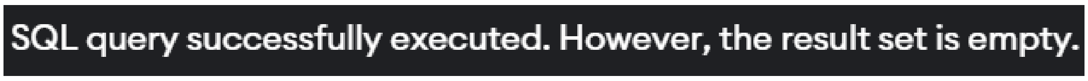
rno	name	class	marks
2022/03	Aman	BCA-1D	234
2022/04	Shivakshi	BSc	290
2022/05	Anant	BBA	226
2023/01	Aditya	BCA-5A	235
2024/01	Kunal	BCA-5E	245
2024/02	Madhav	BCA-5D	223
2024/03	Harshita	BCA-5D	230
2024/04	Bharti	BSc	220
2024/05	Ishika	MBBS	246
2025/02	Avinash	BCA-3A	243

Q. Find out the intersection of student and student1 tables.



```sql
SELECT rno, name, class, marks From student
INTERSECT
SELECT rno, name, class, marks FROM student1;
```

<u>OUTPUT</u>:

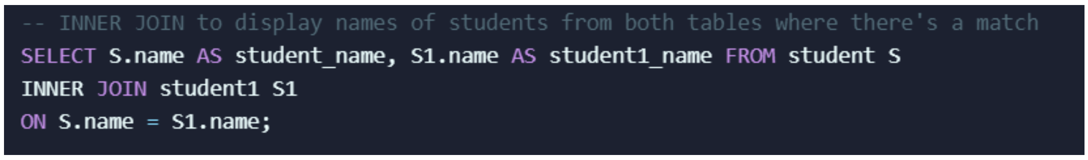


```
SQL query successfully executed. However, the result set is empty.
```

*Since there is no common entry between student and student1; the result set comes as empty.

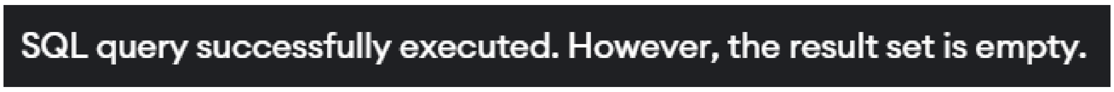
Q. Display the names of student and student1 tables using left, right, inner and full join.

1. Inner Join:



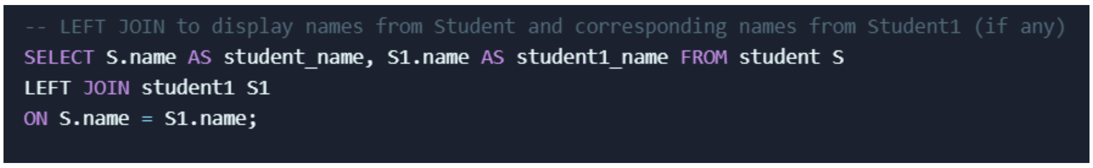
```sql
-- INNER JOIN to display names of students from both tables where there's a match
SELECT S.name AS student_name, S1.name AS student1_name FROM student S
INNER JOIN student1 S1
ON S.name = S1.name;
```

<u>OUTPUT</u>:



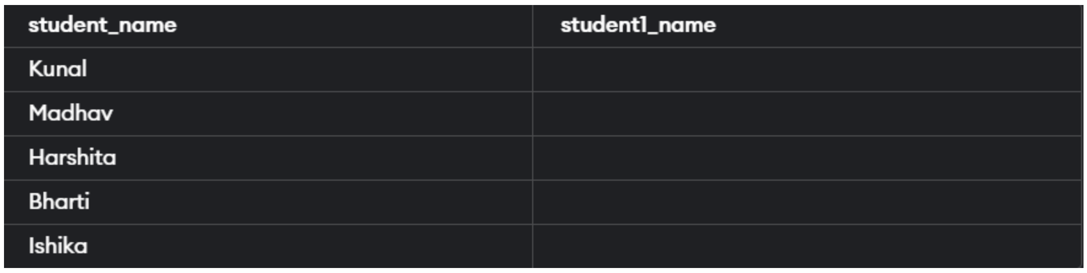
```
SQL query successfully executed. However, the result set is empty.
```

2. Left Join:



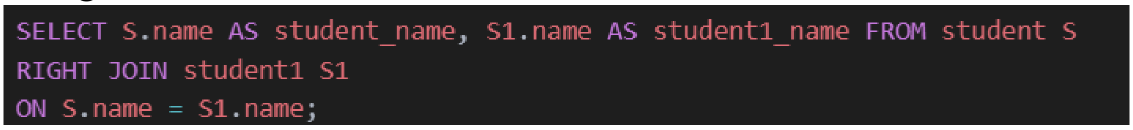
```sql
-- LEFT JOIN to display names from Student and corresponding names from Student1 (if any)
SELECT S.name AS student_name, S1.name AS student1_name FROM student S
LEFT JOIN student1 S1
ON S.name = S1.name;
```

<u>OUTPUT</u>:



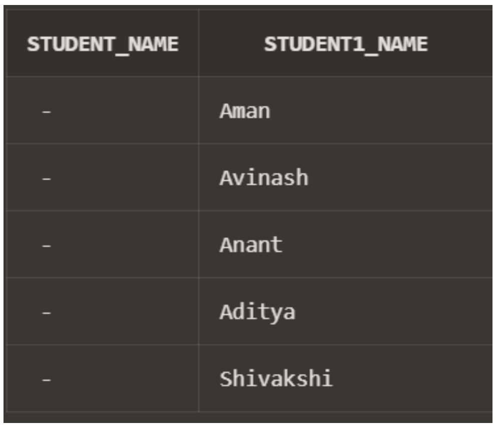
student_name	student1_name
Kunal	
Madhav	
Harshita	
Bharti	
Ishika	

3. Right Join:



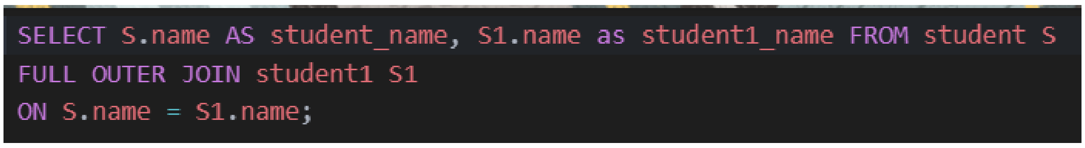
```
SELECT S.name AS student_name, S1.name AS student1_name FROM student S
RIGHT JOIN student1 S1
ON S.name = S1.name;
```

OUTPUT:



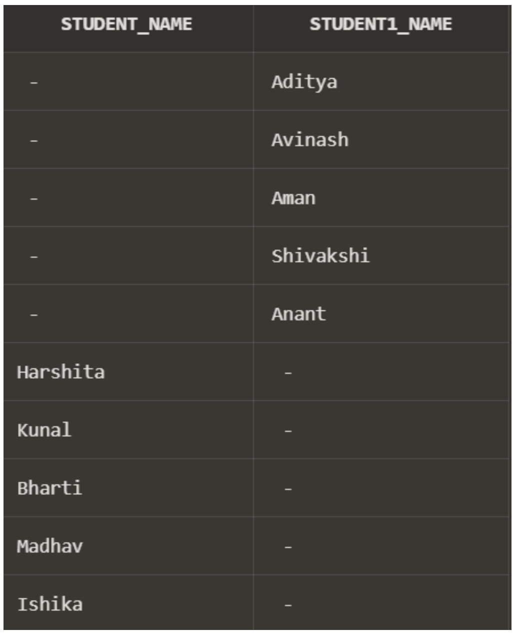
STUDENT_NAME	STUDENT1_NAME
-	Aman
-	Avinash
-	Anant
-	Aditya
-	Shivakshi

4. Full Join:



```
SELECT S.name AS student_name, S1.name as student1_name FROM student S
FULL OUTER JOIN student1 S1
ON S.name = S1.name;
```

OUTPUT:



STUDENT_NAME	STUDENT1_NAME
-	Aditya
-	Avinash
-	Aman
-	Shivakshi
-	Anant
Harshita	-
Kunal	-
Bharti	-
Madhav	-
Ishika	-

EXPERIMENT 3:

Display all the string functions used in SQL.

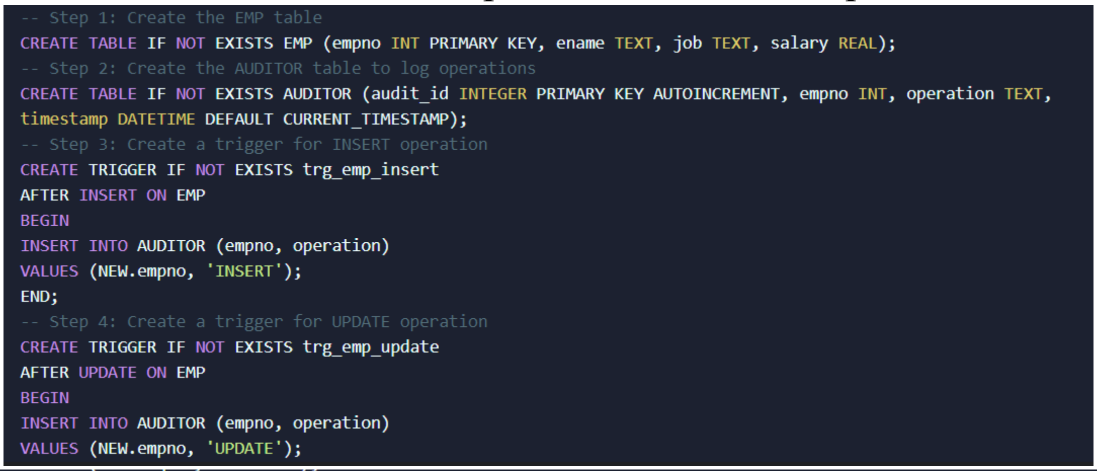
OUTPUT:

```
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;
```

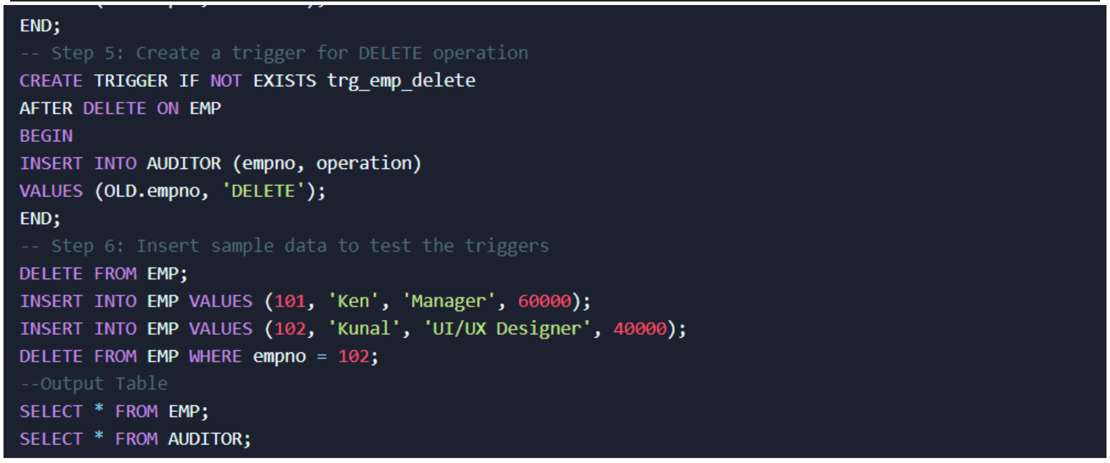
TRIGGERS

EXPERIMENT 1:

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

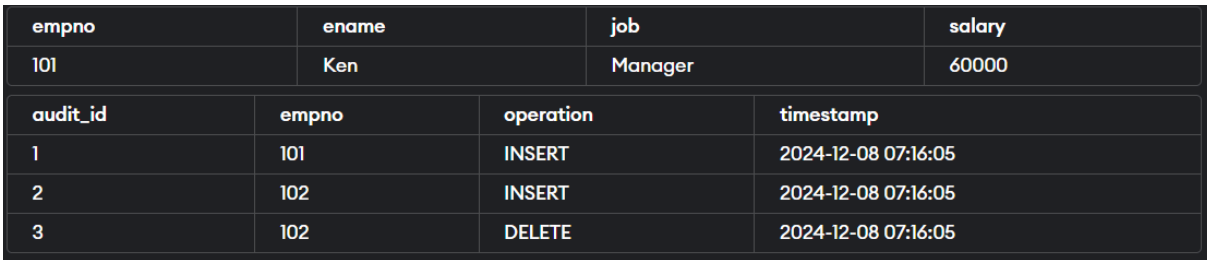


```
-- Step 1: Create the EMP table
CREATE TABLE IF NOT EXISTS EMP (empno INT PRIMARY KEY, ename TEXT, job TEXT, salary REAL);
-- Step 2: Create the AUDITOR table to log operations
CREATE TABLE IF NOT EXISTS AUDITOR (audit_id INTEGER PRIMARY KEY AUTOINCREMENT, empno INT, operation TEXT,
timestamp DATETIME 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 VALUES (101, 'Ken', 'Manager', 60000);
INSERT INTO EMP VALUES (102, 'Kunal', 'UI/UX Designer', 40000);
DELETE FROM EMP WHERE empno = 102;
--Output Table
SELECT * FROM EMP;
SELECT * FROM AUDITOR;
```

OUTPUT:

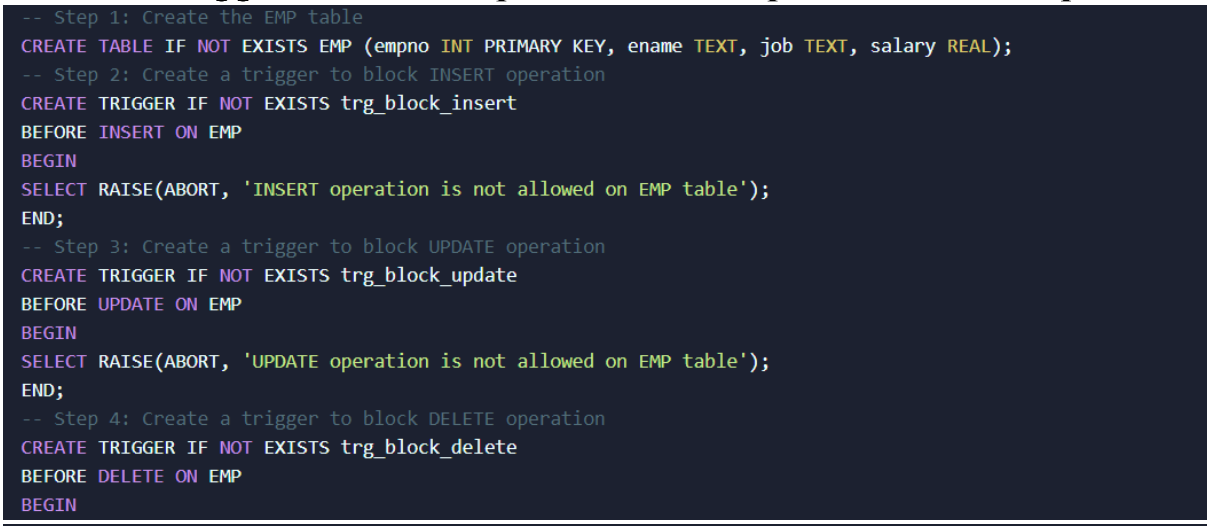


empno	ename	job	salary
101	Ken	Manager	60000

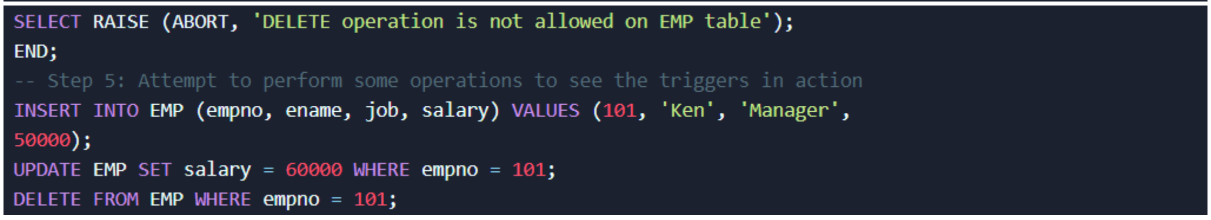
audit_id	empno	operation	timestamp
1	101	INSERT	2024-12-08 07:16:05
2	102	INSERT	2024-12-08 07:16:05
3	102	DELETE	2024-12-08 07:16:05

EXPERIMENT 2:

Create a trigger so that no operation can be performed on emp table.

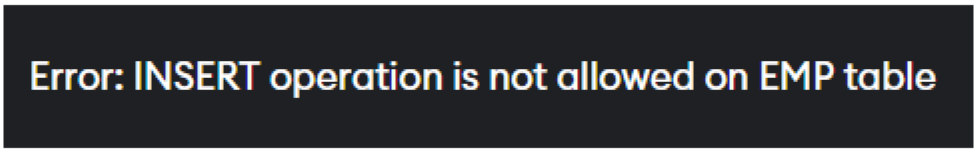


```sql
-- Step 1: Create the EMP table
CREATE TABLE IF NOT EXISTS EMP (empno INT PRIMARY KEY, ename TEXT, job TEXT, salary REAL);
-- Step 2: Create a trigger to block INSERT operation
CREATE TRIGGER IF NOT EXISTS trg_block_insert
BEFORE INSERT ON EMP
BEGIN
SELECT RAISE(ABORT, 'INSERT operation is not allowed on EMP table');
END;
-- 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
```



```sql
SELECT RAISE (ABORT, 'DELETE operation is not allowed on EMP table');
END;
-- Step 5: Attempt to perform some operations to see the triggers in action
INSERT INTO EMP (empno, ename, job, salary) VALUES (101, 'Ken', 'Manager',
50000);
UPDATE EMP SET salary = 60000 WHERE empno = 101;
DELETE FROM EMP WHERE empno = 101;
```

OUTPUT:



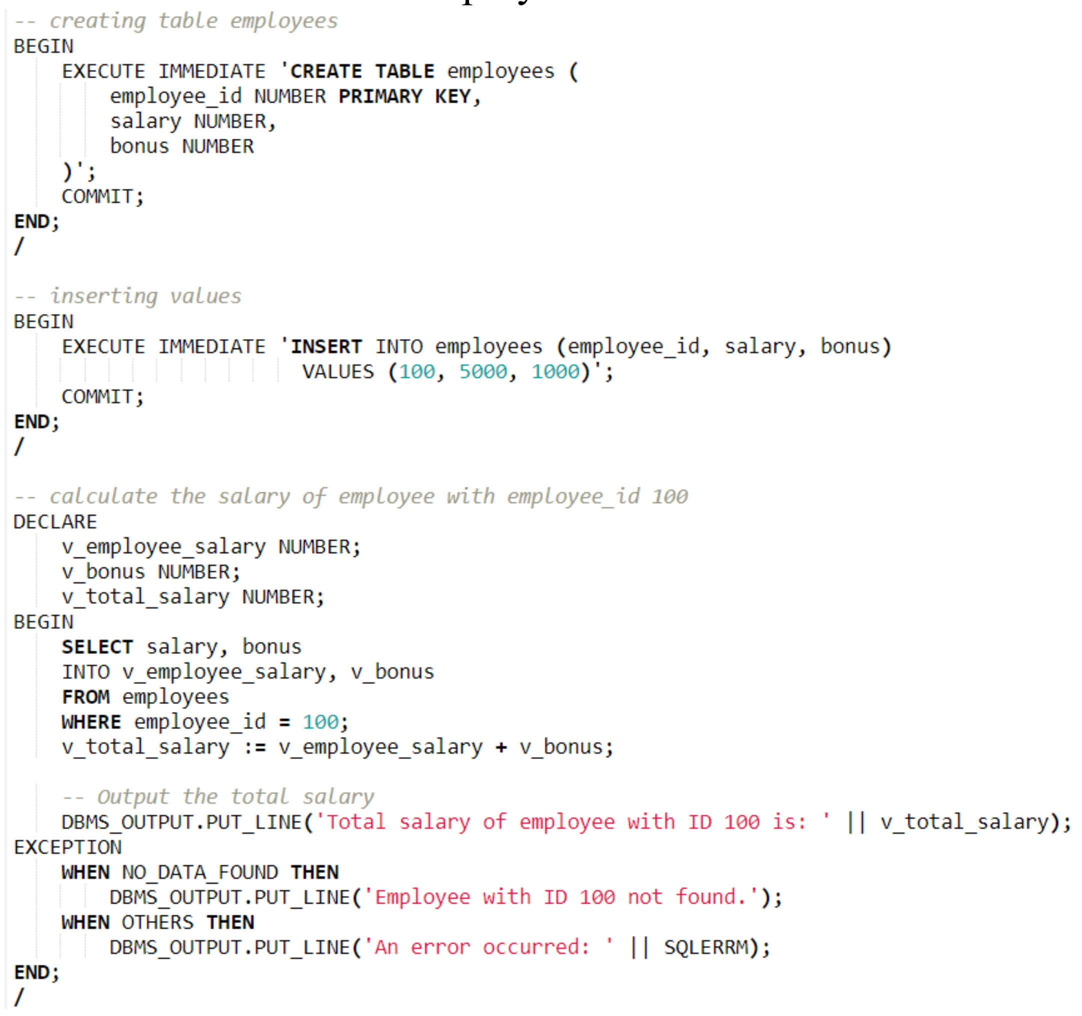
```
Error: INSERT operation is not allowed on EMP table
```

*Since script has raised an abort function, the code will terminate once insert trigger is triggered.

PL/SQL

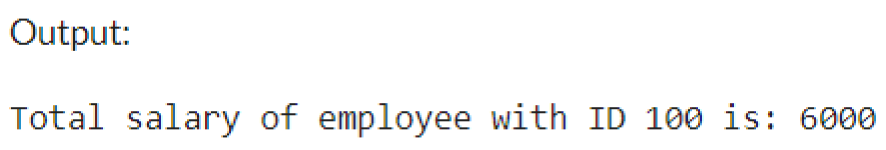
EXPERIMENT 1:

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



```sql
-- creating table employees
BEGIN
    EXECUTE IMMEDIATE 'CREATE TABLE employees (
        employee_id NUMBER PRIMARY KEY,
        salary NUMBER,
        bonus NUMBER
    )';
    COMMIT;
END;
/

-- inserting values
BEGIN
    EXECUTE IMMEDIATE 'INSERT INTO employees (employee_id, salary, bonus)
                        VALUES (100, 5000, 1000)';
    COMMIT;
END;
/

-- calculate the salary of employee with employee_id 100
DECLARE
    v_employee_salary NUMBER;
    v_bonus NUMBER;
    v_total_salary NUMBER;
BEGIN
    SELECT salary, bonus
    INTO v_employee_salary, v_bonus
    FROM employees
    WHERE employee_id = 100;
    v_total_salary := v_employee_salary + v_bonus;

    -- Output the total salary
    DBMS_OUTPUT.PUT_LINE('Total salary of employee with ID 100 is: ' || v_total_salary);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Employee with ID 100 not found.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/
```

OUTPUT:

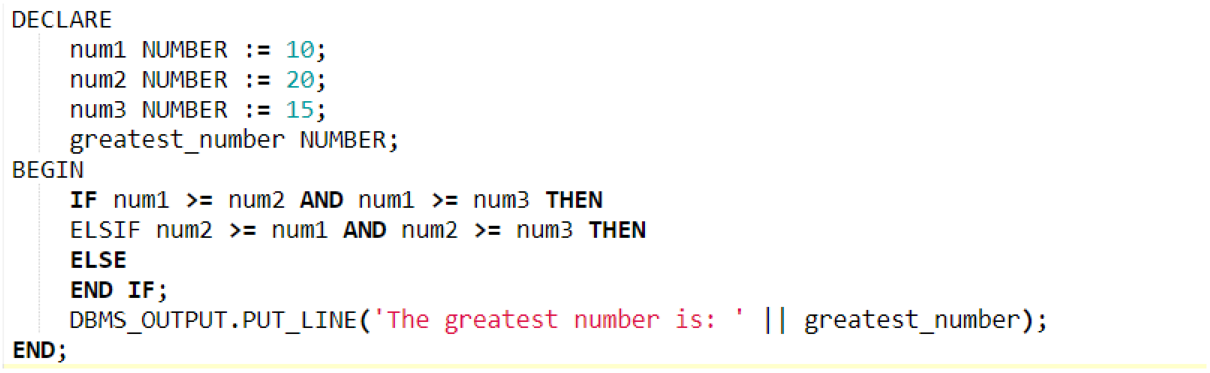


Output:

```
Total salary of employee with ID 100 is: 6000
```

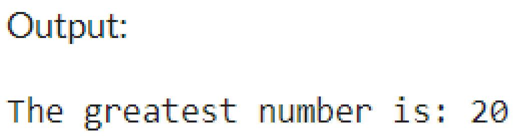
EXPERIMENT 2:

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



```
DECLARE
    num1 NUMBER := 10;
    num2 NUMBER := 20;
    num3 NUMBER := 15;
    greatest_number NUMBER;
BEGIN
    IF num1 >= num2 AND num1 >= num3 THEN
    ELSIF num2 >= num1 AND num2 >= num3 THEN
    ELSE
    END IF;
    DBMS_OUTPUT.PUT_LINE('The greatest number is: ' || greatest_number);
END;
```

OUTPUT:

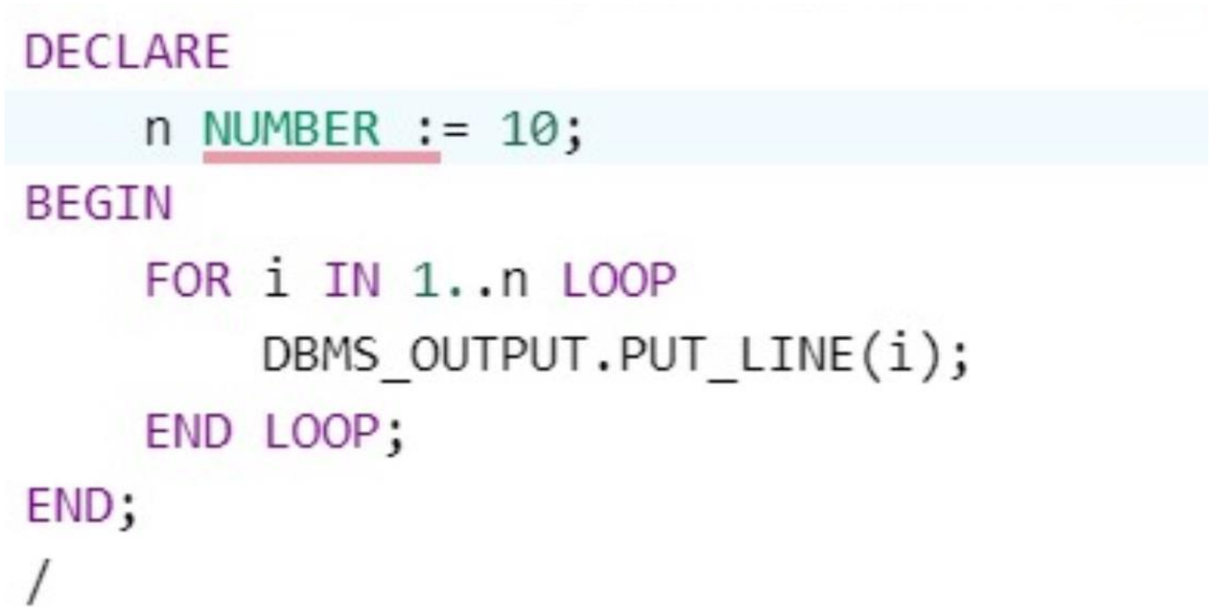


Output:

```
The greatest number is: 20
```

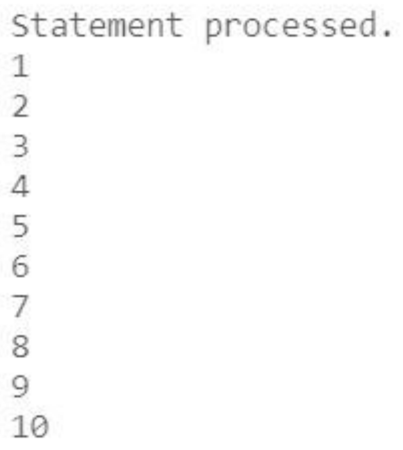
EXPERIMENT 3:

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



```
DECLARE
    n NUMBER := 10;
BEGIN
    FOR i IN 1..n LOOP
        DBMS_OUTPUT.PUT_LINE(i);
    END LOOP;
END;
/
```

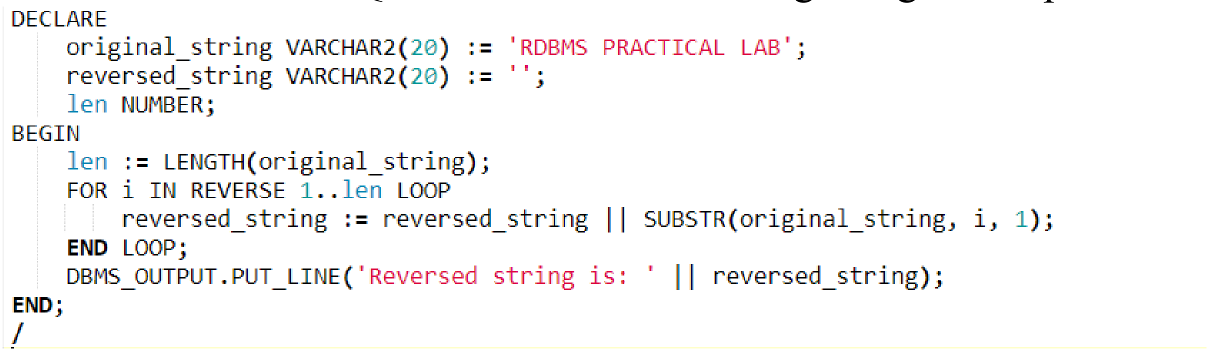
OUTPUT:



```
Statement processed.
1
2
3
4
5
6
7
8
9
10
```

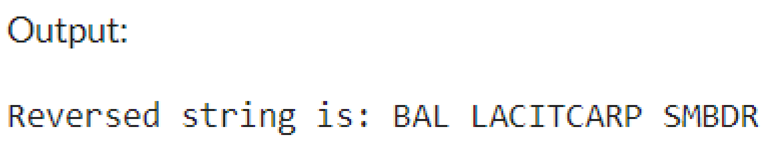
EXPERIMENT 4:

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



```
DECLARE
    original_string VARCHAR2(20) := 'RDBMS PRACTICAL LAB';
    reversed_string VARCHAR2(20) := '';
    len NUMBER;
BEGIN
    len := LENGTH(original_string);
    FOR i IN REVERSE 1..len LOOP
        reversed_string := reversed_string || SUBSTR(original_string, i, 1);
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('Reversed string is: ' || reversed_string);
END;
/
```

OUTPUT:

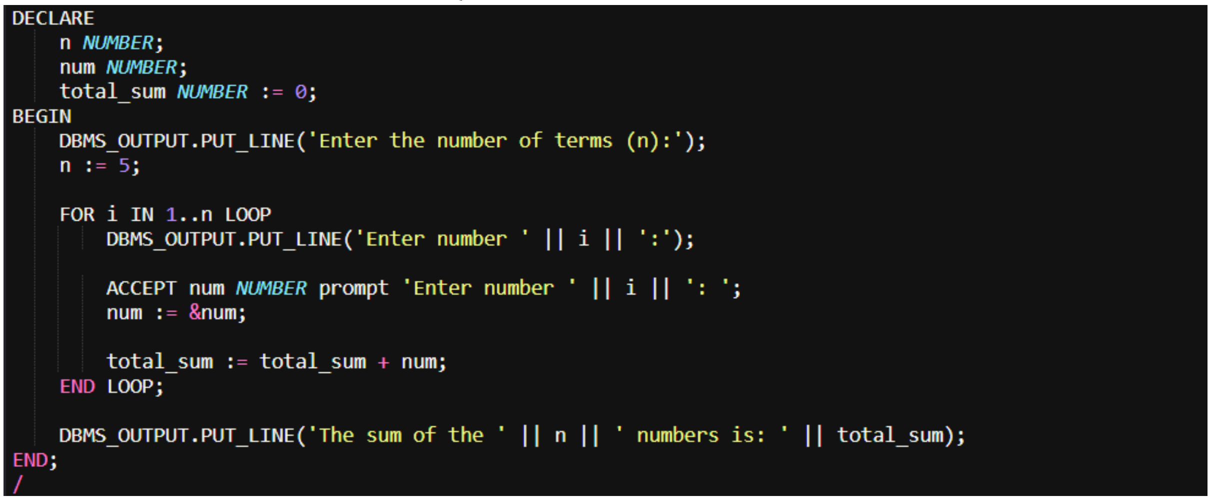


Output:

```
Reversed string is: BAL LACITCARP SMBDR
```

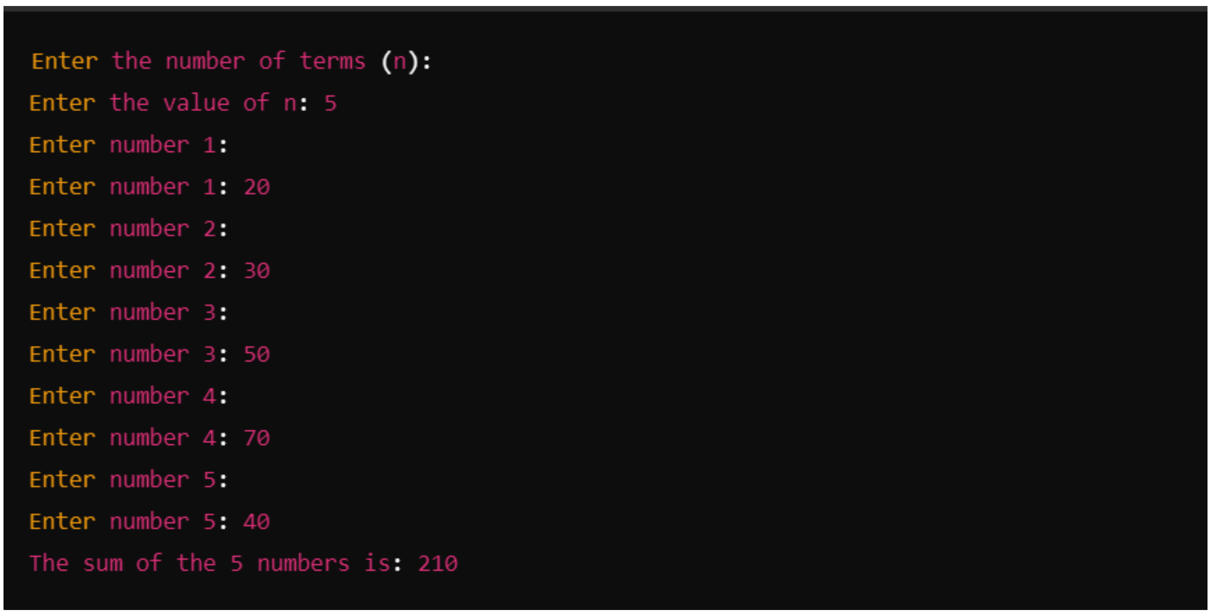
EXPERIMENT 5:

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



```
DECLARE
    n NUMBER;
    num NUMBER;
    total_sum NUMBER := 0;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Enter the number of terms (n):');
    n := 5;

    FOR i IN 1..n LOOP
        DBMS_OUTPUT.PUT_LINE('Enter number ' || i || ':');

        ACCEPT num NUMBER prompt 'Enter number ' || i || ': ';
        num := &num;

        total_sum := total_sum + num;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('The sum of the ' || n || ' numbers is: ' || total_sum);
END;
/
```

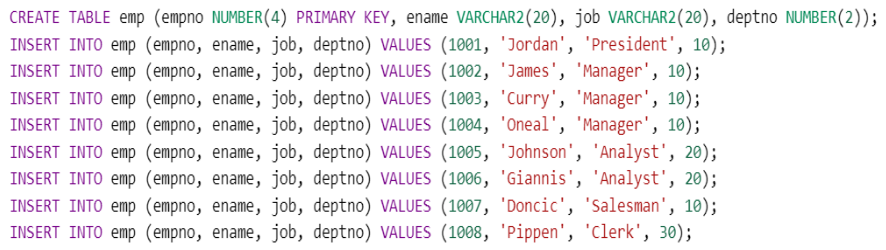
OUTPUT:



```
Enter the number of terms (n):
Enter the value of n: 5
Enter number 1:
Enter number 1: 20
Enter number 2:
Enter number 2: 30
Enter number 3:
Enter number 3: 50
Enter number 4:
Enter number 4: 70
Enter number 5:
Enter number 5: 40
The sum of the 5 numbers is: 210
```

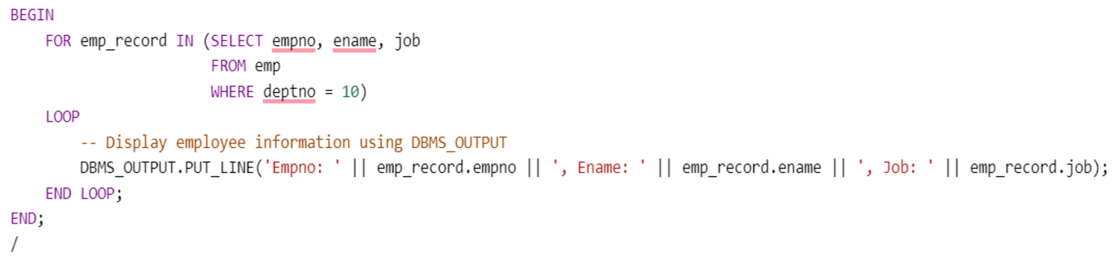
<u>EXPERIMENT 6</u>:

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



```
CREATE TABLE emp (empno NUMBER(4) PRIMARY KEY, ename VARCHAR2(20), job VARCHAR2(20), deptno NUMBER(2));
INSERT INTO emp (empno, ename, job, deptno) VALUES (1001, 'Jordan', 'President', 10);
INSERT INTO emp (empno, ename, job, deptno) VALUES (1002, 'James', 'Manager', 10);
INSERT INTO emp (empno, ename, job, deptno) VALUES (1003, 'Curry', 'Manager', 10);
INSERT INTO emp (empno, ename, job, deptno) VALUES (1004, 'Oneal', 'Manager', 10);
INSERT INTO emp (empno, ename, job, deptno) VALUES (1005, 'Johnson', 'Analyst', 20);
INSERT INTO emp (empno, ename, job, deptno) VALUES (1006, 'Giannis', 'Analyst', 20);
INSERT INTO emp (empno, ename, job, deptno) VALUES (1007, 'Doncic', 'Salesman', 10);
INSERT INTO emp (empno, ename, job, deptno) VALUES (1008, 'Pippen', 'Clerk', 30);
```

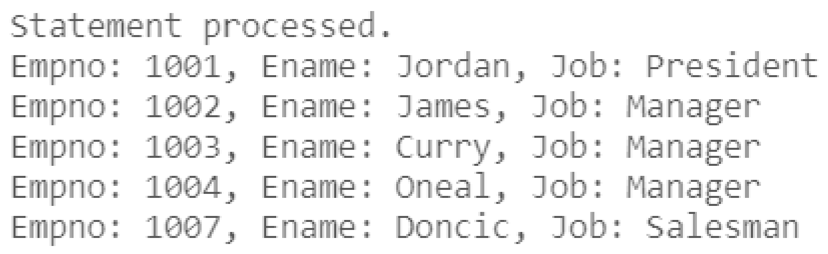
- Creation of table emp



```
BEGIN
    FOR emp_record IN (SELECT empno, ename, job
                       FROM emp
                       WHERE deptno = 10)
    LOOP
        -- Display employee information using DBMS_OUTPUT
        DBMS_OUTPUT.PUT_LINE('Empno: ' || emp_record.empno || ', Ename: ' || emp_record.ename || ', Job: ' || emp_record.job);
    END LOOP;
END;
/
```

- PL/SQL script to display empno, ename, job of employees with department number 10.

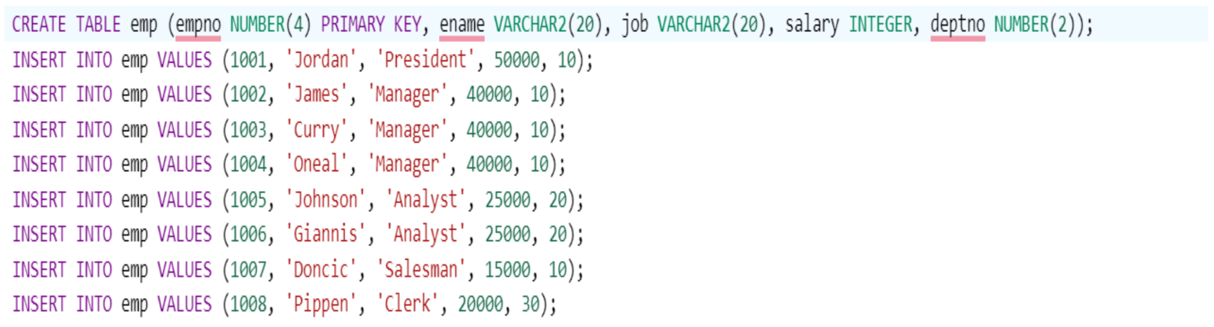
<u>OUTPUT</u>:



```
Statement processed.
Empno: 1001, Ename: Jordan, Job: President
Empno: 1002, Ename: James, Job: Manager
Empno: 1003, Ename: Curry, Job: Manager
Empno: 1004, Ename: Oneal, Job: Manager
Empno: 1007, Ename: Doncic, Job: Salesman
```

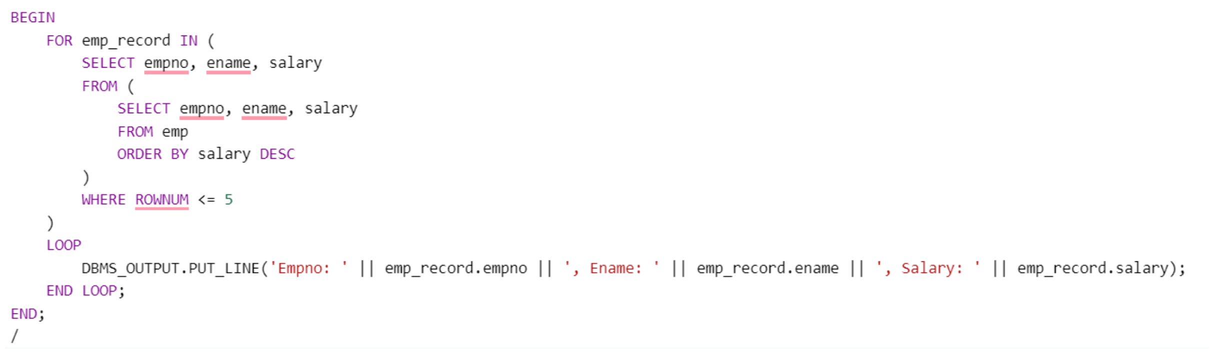
EXPERIMENT 7:

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



```
CREATE TABLE emp (empno NUMBER(4) PRIMARY KEY, ename VARCHAR2(20), job VARCHAR2(20), salary INTEGER, deptno NUMBER(2));
INSERT INTO emp VALUES (1001, 'Jordan', 'President', 50000, 10);
INSERT INTO emp VALUES (1002, 'James', 'Manager', 40000, 10);
INSERT INTO emp VALUES (1003, 'Curry', 'Manager', 40000, 10);
INSERT INTO emp VALUES (1004, 'Oneal', 'Manager', 40000, 10);
INSERT INTO emp VALUES (1005, 'Johnson', 'Analyst', 25000, 20);
INSERT INTO emp VALUES (1006, 'Giannis', 'Analyst', 25000, 20);
INSERT INTO emp VALUES (1007, 'Doncic', 'Salesman', 15000, 10);
INSERT INTO emp VALUES (1008, 'Pippen', 'Clerk', 20000, 30);
```

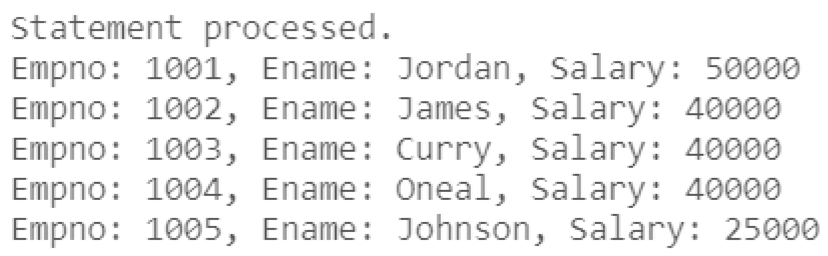
- Creation of table emp



```
BEGIN
    FOR emp_record IN (
        SELECT empno, ename, salary
        FROM (
            SELECT empno, ename, salary
            FROM emp
            ORDER BY salary DESC
        )
        WHERE ROWNUM <= 5
    )
    LOOP
        DBMS_OUTPUT.PUT_LINE('Empno: ' || emp_record.empno || ', Ename: ' || emp_record.ename || ', Salary: ' || emp_record.salary);
    END LOOP;
END;
/
```

- PL/SQL script to display top 5 highest paid employees.

OUPUT:



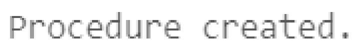
```
Statement processed.
Empno: 1001, Ename: Jordan, Salary: 50000
Empno: 1002, Ename: James, Salary: 40000
Empno: 1003, Ename: Curry, Salary: 40000
Empno: 1004, Ename: Oneal, Salary: 40000
Empno: 1005, Ename: Johnson, Salary: 25000
```

EXPERIMENT 8:

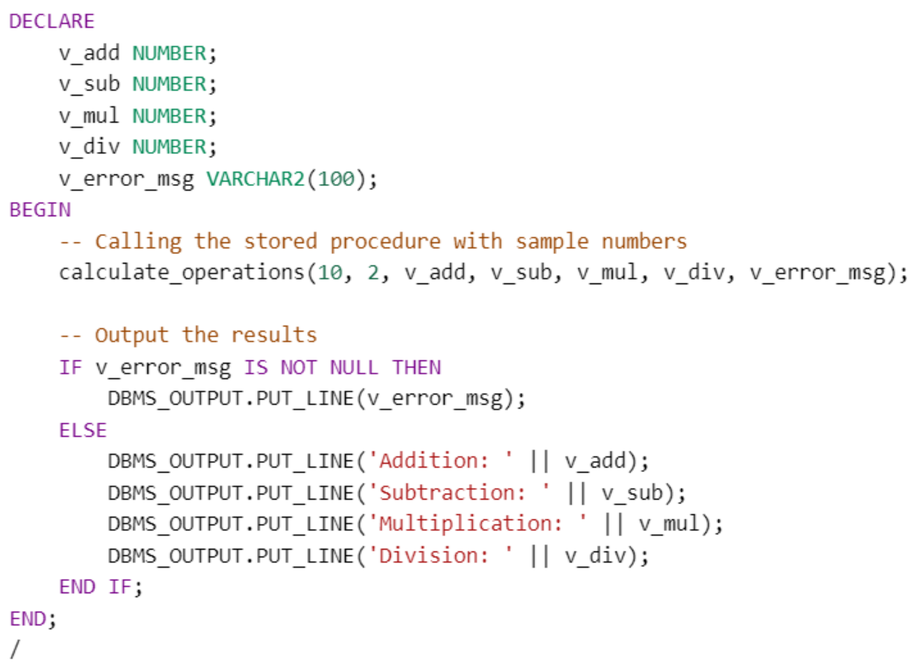
Write a PL/SQL procedure that accepts 2 numbers & return addition, subtraction, multiplication & division of two numbers using stored procedure AND local procedure.

```
-- Main procedure to accept two numbers and perform operations
CREATE OR REPLACE PROCEDURE calculate_operations(
    p_num1 IN NUMBER,
    p_num2 IN NUMBER,
    p_add OUT NUMBER,
    p_sub OUT NUMBER,
    p_mul OUT NUMBER,
    p_div OUT NUMBER,
    p_error_msg OUT VARCHAR2
) IS

    -- Local procedure to perform calculations
    PROCEDURE perform_calculations(
        num1 IN NUMBER,
        num2 IN NUMBER,
        add_result OUT NUMBER,
        sub_result OUT NUMBER,
        mul_result OUT NUMBER,
        div_result OUT NUMBER,
        error_msg OUT VARCHAR2
    ) IS
    BEGIN
        add_result := num1 + num2;
        sub_result := num1 - num2;
        mul_result := num1 * num2;
        IF num2 = 0 THEN
            div_result := NULL;
            error_msg := 'Error: Division by zero is not allowed';
        ELSE
            div_result := num1 / num2;
            error_msg := NULL;
        END IF;
    END perform_calculations;

BEGIN
    perform_calculations(p_num1, p_num2, p_add, p_sub, p_mul, p_div, p_error_msg);
    IF p_error_msg IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE(p_error_msg);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Addition: ' || p_add);
        DBMS_OUTPUT.PUT_LINE('Subtraction: ' || p_sub);
        DBMS_OUTPUT.PUT_LINE('Multiplication: ' || p_mul);
        DBMS_OUTPUT.PUT_LINE('Division: ' || p_div);
    END IF;
END calculate_operations;
/
```

- Creation of procedure



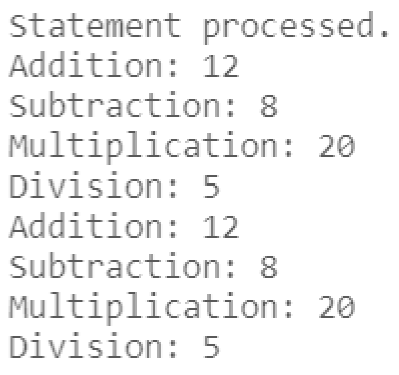
```
Procedure created.
```



```sql
DECLARE
    v_add NUMBER;
    v_sub NUMBER;
    v_mul NUMBER;
    v_div NUMBER;
    v_error_msg VARCHAR2(100);
BEGIN
    -- Calling the stored procedure with sample numbers
    calculate_operations(10, 2, v_add, v_sub, v_mul, v_div, v_error_msg);

    -- Output the results
    IF v_error_msg IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE(v_error_msg);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Addition: ' || v_add);
        DBMS_OUTPUT.PUT_LINE('Subtraction: ' || v_sub);
        DBMS_OUTPUT.PUT_LINE('Multiplication: ' || v_mul);
        DBMS_OUTPUT.PUT_LINE('Division: ' || v_div);
    END IF;
END;
/
```

- PL/SQL script to call the calculation_operations() procedure

<u>OUTPUT</u>:

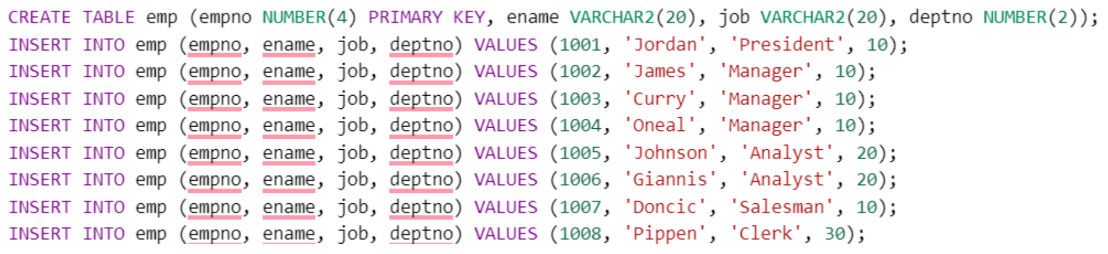


```
Statement processed.
Addition: 12
Subtraction: 8
Multiplication: 20
Division: 5
Addition: 12
Subtraction: 8
Multiplication: 20
Division: 5
```

- Output after passing 10 and 2 in the procedure

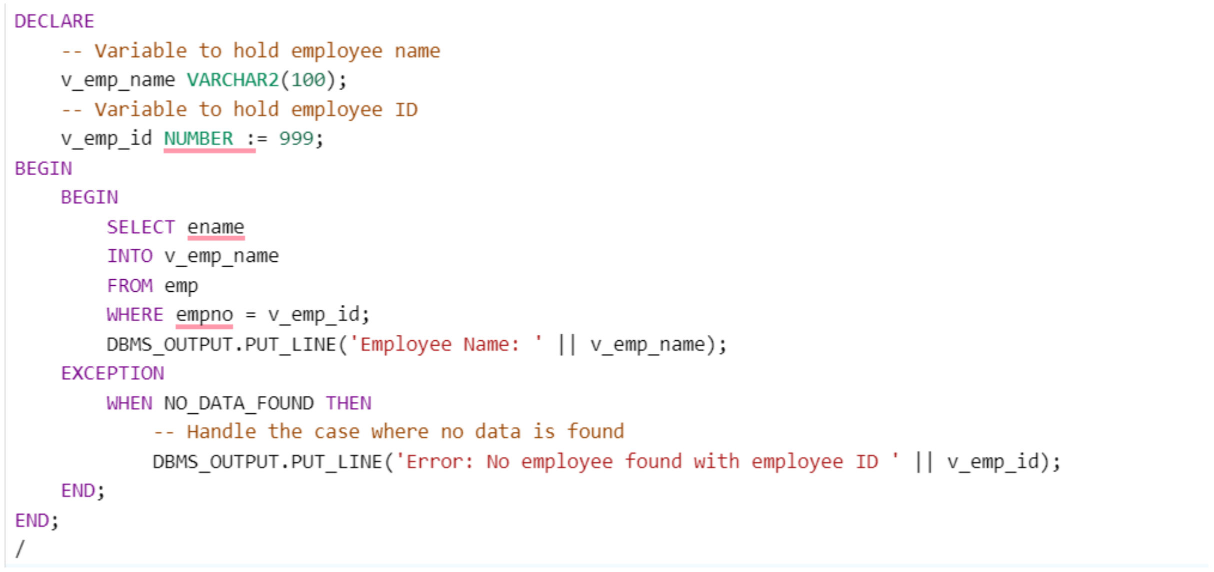
EXPERIMENT 9:

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



```sql
CREATE TABLE emp (empno NUMBER(4) PRIMARY KEY, ename VARCHAR2(20), job VARCHAR2(20), deptno NUMBER(2));
INSERT INTO emp (empno, ename, job, deptno) VALUES (1001, 'Jordan', 'President', 10);
INSERT INTO emp (empno, ename, job, deptno) VALUES (1002, 'James', 'Manager', 10);
INSERT INTO emp (empno, ename, job, deptno) VALUES (1003, 'Curry', 'Manager', 10);
INSERT INTO emp (empno, ename, job, deptno) VALUES (1004, 'Oneal', 'Manager', 10);
INSERT INTO emp (empno, ename, job, deptno) VALUES (1005, 'Johnson', 'Analyst', 20);
INSERT INTO emp (empno, ename, job, deptno) VALUES (1006, 'Giannis', 'Analyst', 20);
INSERT INTO emp (empno, ename, job, deptno) VALUES (1007, 'Doncic', 'Salesman', 10);
INSERT INTO emp (empno, ename, job, deptno) VALUES (1008, 'Pippen', 'Clerk', 30);
```

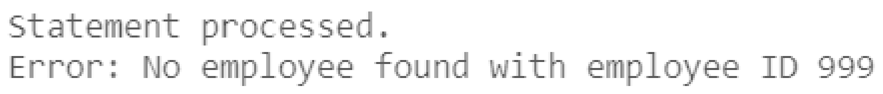
- Creation of table for searching data



```sql
DECLARE
    -- Variable to hold employee name
    v_emp_name VARCHAR2(100);
    -- Variable to hold employee ID
    v_emp_id NUMBER := 999;
BEGIN
    BEGIN
        SELECT ename
        INTO v_emp_name
        FROM emp
        WHERE empno = v_emp_id;
        DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_emp_name);
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            -- Handle the case where no data is found
            DBMS_OUTPUT.PUT_LINE('Error: No employee found with employee ID ' || v_emp_id);
    END;
END;
/
```

- PL/SQL scrip to create the NO_DATA_FOUND exception

OUTPUT:



```
Statement processed.
Error: No employee found with employee ID 999
```

EXPERIMENT 10:

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

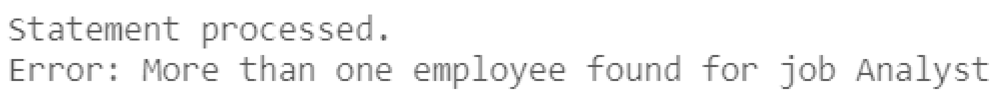
```sql
CREATE TABLE emp (empno NUMBER(4) PRIMARY KEY, ename VARCHAR2(20), job VARCHAR2(20), deptno NUMBER(2));
INSERT INTO emp (empno, ename, job, deptno) VALUES (1001, 'Jordan', 'President', 10);
INSERT INTO emp (empno, ename, job, deptno) VALUES (1002, 'James', 'Manager', 10);
INSERT INTO emp (empno, ename, job, deptno) VALUES (1003, 'Curry', 'Manager', 10);
INSERT INTO emp (empno, ename, job, deptno) VALUES (1004, 'Oneal', 'Manager', 10);
INSERT INTO emp (empno, ename, job, deptno) VALUES (1005, 'Johnson', 'Analyst', 20);
INSERT INTO emp (empno, ename, job, deptno) VALUES (1006, 'Giannis', 'Analyst', 20);
INSERT INTO emp (empno, ename, job, deptno) VALUES (1007, 'Doncic', 'Salesman', 10);
INSERT INTO emp (empno, ename, job, deptno) VALUES (1008, 'Pippen', 'Clerk', 30);
```

- Creation of table

```sql
DECLARE
    v_emp_name VARCHAR2(50);
    v_job VARCHAR2(50) := 'Analyst';
BEGIN
    BEGIN
        SELECT ename
        INTO v_emp_name
        FROM emp
        WHERE job = v_job;
        DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_emp_name);
    EXCEPTION
        WHEN TOO_MANY_ROWS THEN
            -- Handle the case where multiple rows are returned
            DBMS_OUTPUT.PUT_LINE('Error: More than one employee found for job ' || v_job);
    END;
END;
/
```

- PL/SQL script to create the TOO_MANY_ROWS exception

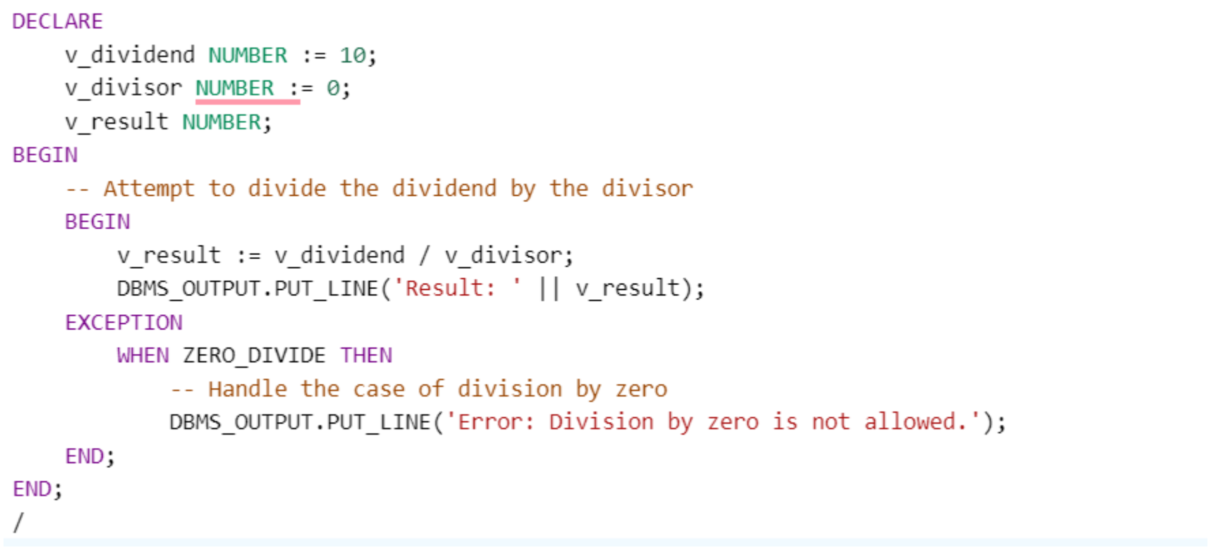
OUTPUT:



```
Statement processed.
Error: More than one employee found for job Analyst
```

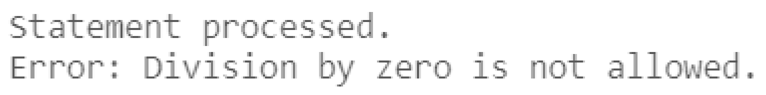
EXPERIMENT 11:

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



```
DECLARE
    v_dividend NUMBER := 10;
    v_divisor NUMBER := 0;
    v_result NUMBER;
BEGIN
    -- Attempt to divide the dividend by the divisor
    BEGIN
        v_result := v_dividend / v_divisor;
        DBMS_OUTPUT.PUT_LINE('Result: ' || v_result);
    EXCEPTION
        WHEN ZERO_DIVIDE THEN
            -- Handle the case of division by zero
            DBMS_OUTPUT.PUT_LINE('Error: Division by zero is not allowed.');
    END;
END;
/
```

OUTPUT:



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

- 10 divided by 0 will throw the Zero Divide error