MANAV RACHNA INTERNATIONAL INSTITUTE OF RESEARCH AND STUDIES

FACULTY OF COMPUTER APPLICATIONS

PRACTICAL FILE

OF

ORACLE LAB

BCA 5th SEM-SECTION D (2024-2025)

SUBMITTED TO:

Ms. Neerja Negi

SUBMITTED BY:

NAME: Kapil sorout

ROLL NO:22/FCA/BCA(CS)/020

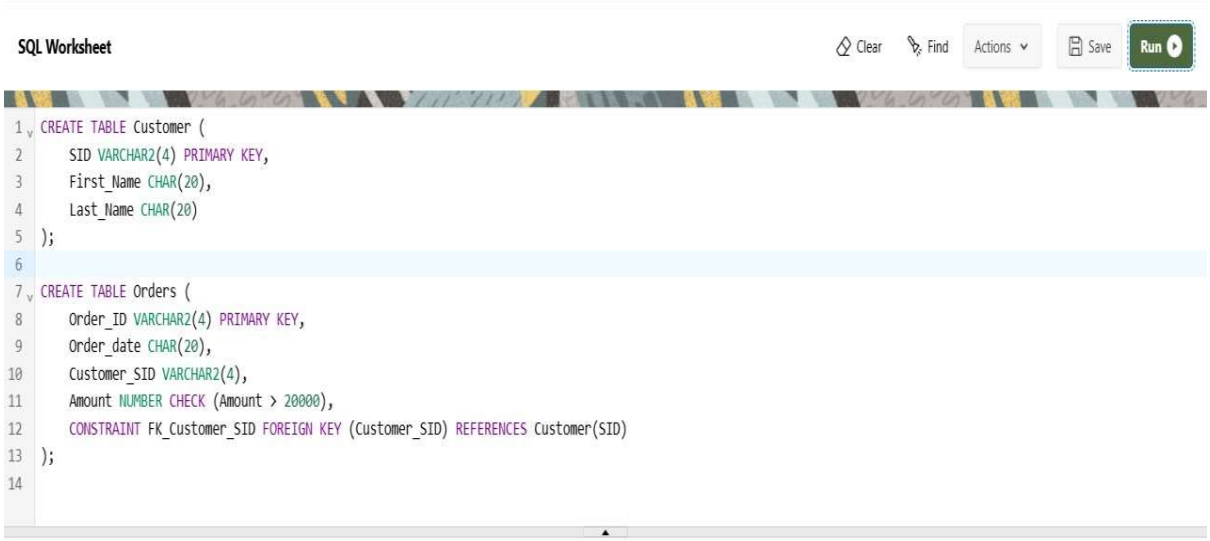
1. Create the following tables

Customer

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

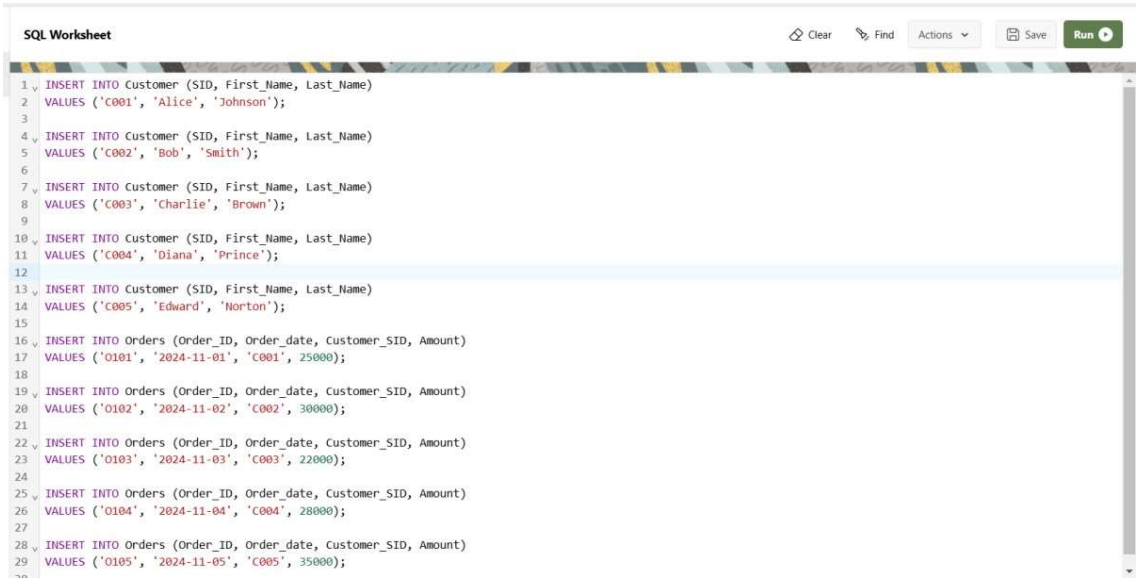
Orders

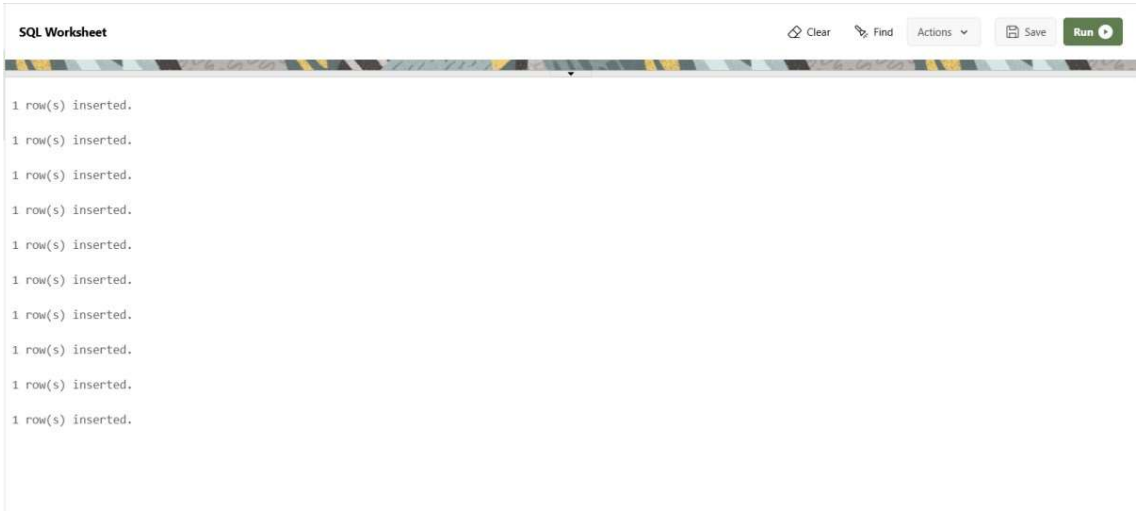
Column_name	Data type	Size	Constraint
Order_ID	Varchar2	4	Primary Key
Order_date	Char	20	
Customer_SID	Varchar2	20	Foreign Key
Amount	Number		Check > 20000



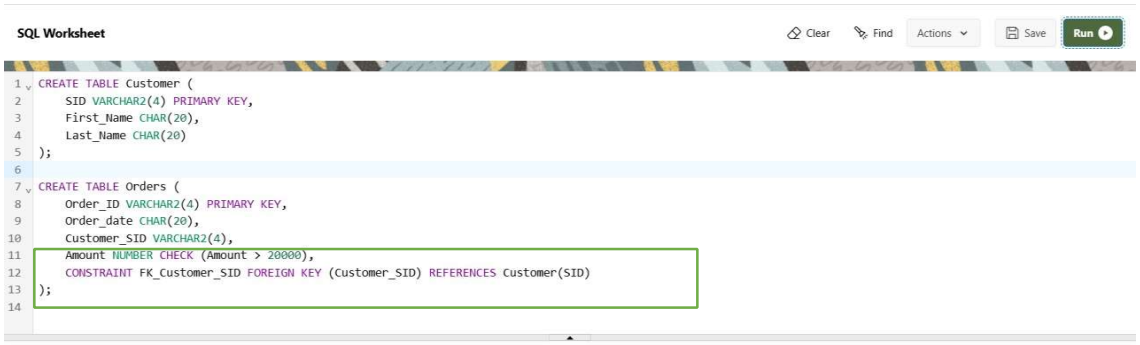

```
Table created.

Table created.
```

2. Insert five records for each table.



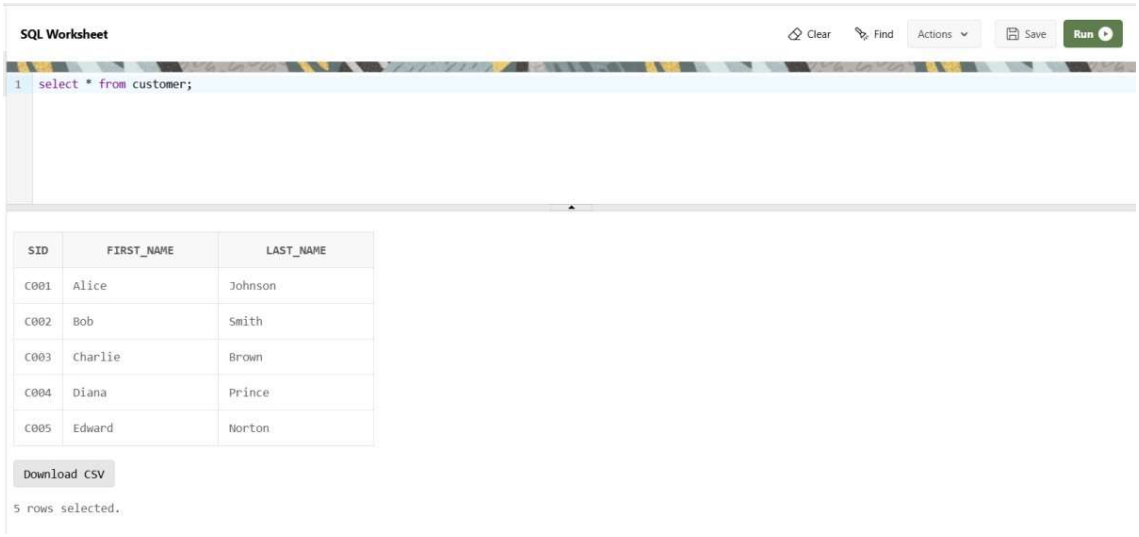


3. Customer_SID column in the ORDERS table is a foreign key pointing to the SID column in the CUSTOMER table.




```
Table created.

Table created.
```

4. Insert five records for both tables



SID	FIRST_NAME	LAST_NAME
C001	Alice	Johnson
C002	Bob	Smith
C003	Charlie	Brown
C004	Diana	Prince
C005	Edward	Norton

Download CSV

5 rows selected.

SQL Worksheet

Clear Find Actions Save Run

```
1  select * from orders;
```

ORDER_ID	ORDER_DATE	CUSTOMER_SID	AMOUNT
O101	2024-11-01	C001	25000
O102	2024-11-02	C002	30000
O103	2024-11-03	C003	22000
O104	2024-11-04	C004	28000
O105	2024-11-05	C005	35000

Download CSV

5 rows selected.

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

SQL Worksheet

Clear Find Actions Save Run

```
1  SELECT
2      c.First_Name,
3      c.Last_Name,
4      SUM(o.Amount) AS Total_Amount
5  FROM
6      Customer c
7  INNER JOIN Orders o ON c.SID = o.Customer_SID
8  GROUP BY
9      c.First_Name, c.Last_Name;
```

FIRST_NAME	LAST_NAME	TOTAL_AMOUNT
Diana	Prince	28000
Edward	Norton	35000
Charlie	Brown	22000
Bob	Smith	30000
Alice	Johnson	25000

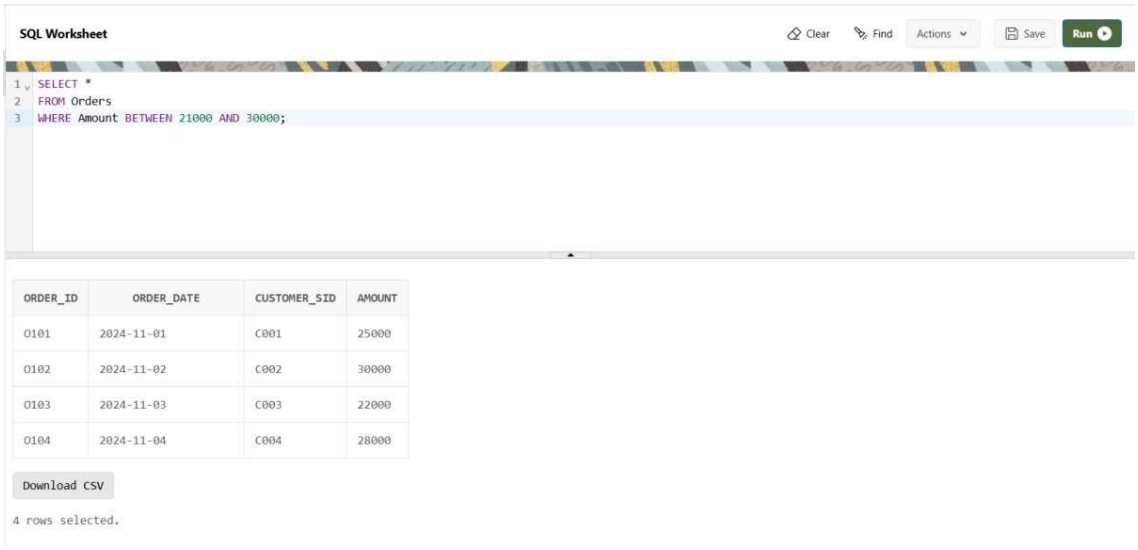
Download CSV

5 rows selected.

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

no data found

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

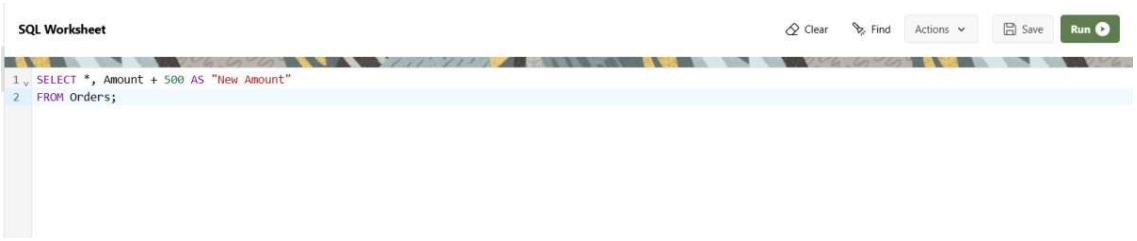


ORDER_ID	ORDER_DATE	CUSTOMER_SID	AMOUNT
O101	2024-11-01	C001	25000
O102	2024-11-02	C002	30000
O103	2024-11-03	C003	22000
O104	2024-11-04	C004	28000

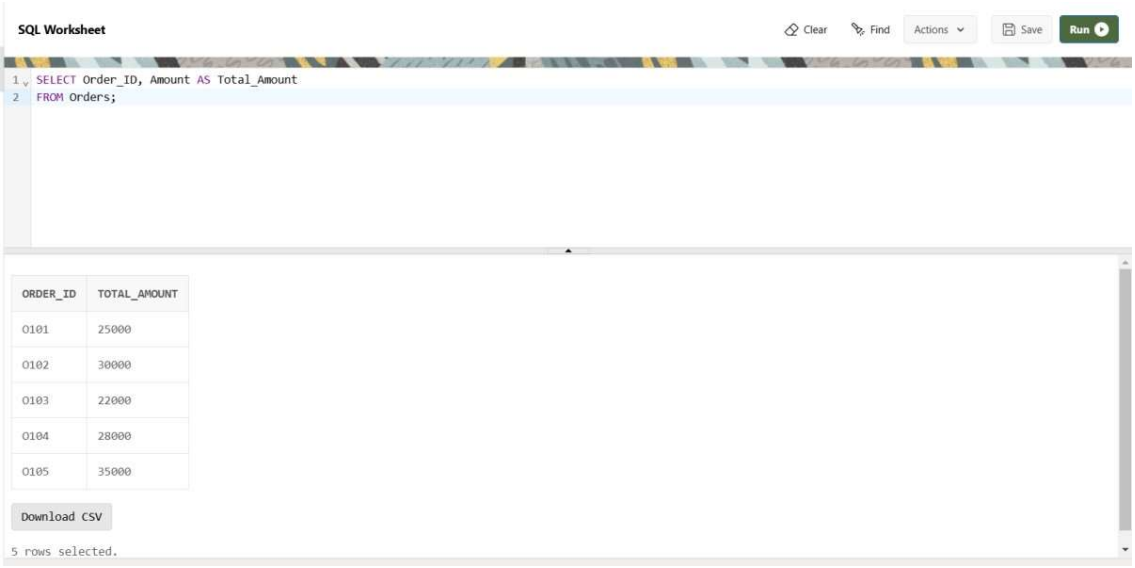
Download CSV

4 rows selected.

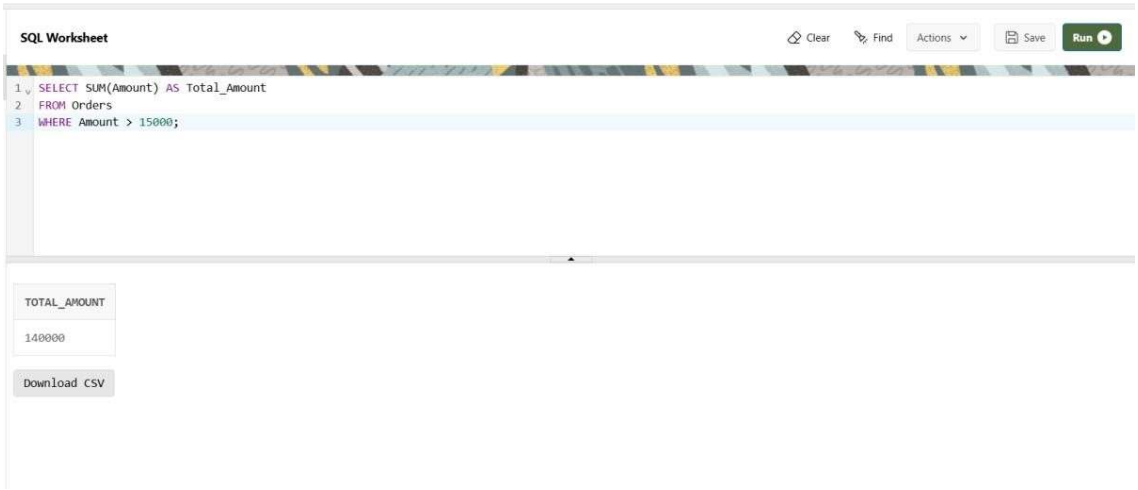
8. List the orders where amount is increased by 500 and replace with name “new amount”.



9. Display the order_id and total amount of orders



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



12. Create the following tables

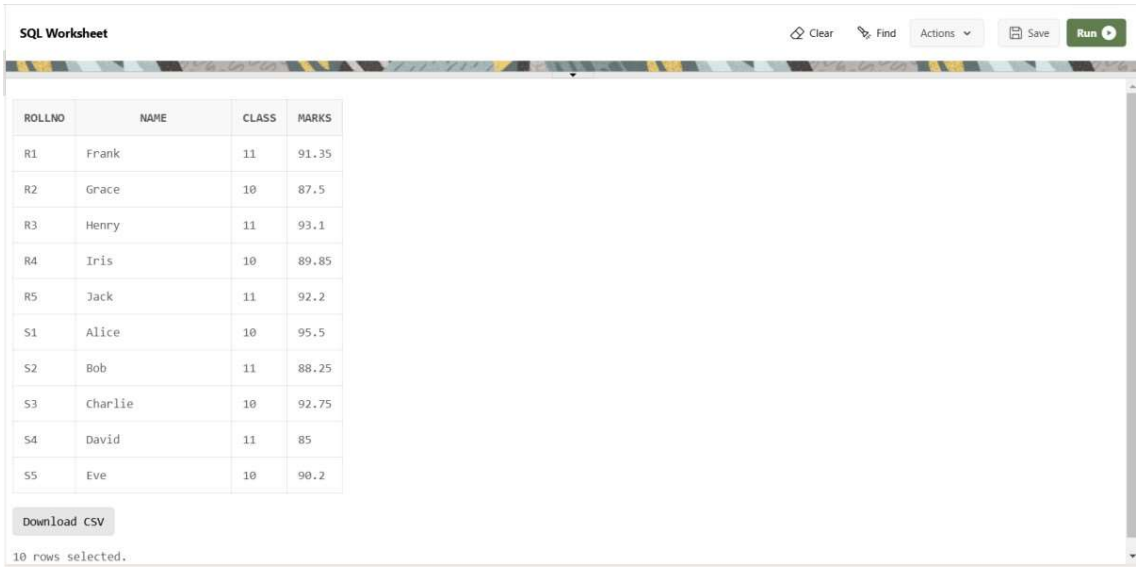
Student

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

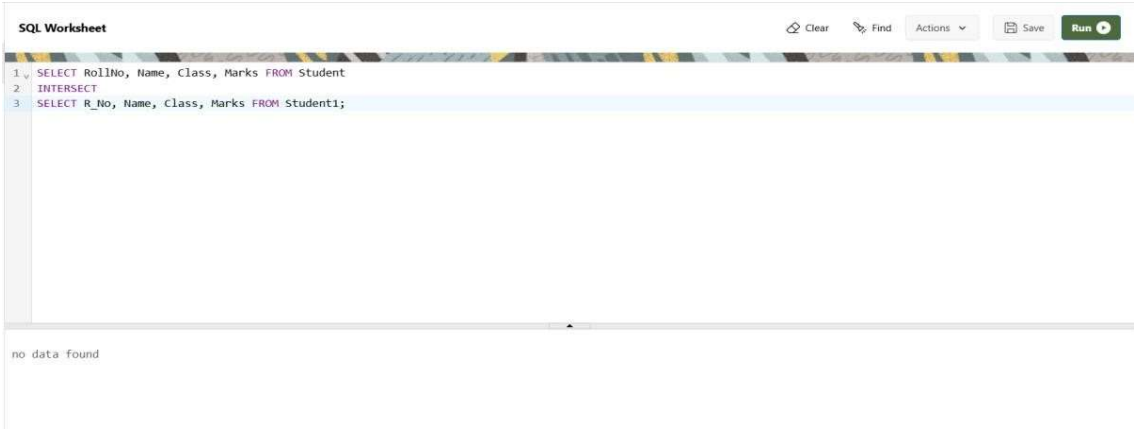
Student1

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

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



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



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

SQL Worksheet

Clear Find Actions Save Run

```
SELECT s.Name AS Student_Name, s1.Name AS Student1_Name
FROM Student s
LEFT JOIN Student1 s1 ON s.RollNo = s1.R_No;
```

STUDENT_NAME	STUDENT1_NAME
Bob	-
David	-
Eve	-
Alice	-
Charlie	-

Download CSV

5 rows selected.

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

```
CREATE TABLE employee (
    employee_id NUMBER PRIMARY KEY,
    first_name VARCHAR2(50),
    last_name VARCHAR2(50),
    job VARCHAR2(50),
    salary NUMBER(10, 2),
    department VARCHAR2(50)
);
```

Table created.

```sql
INSERT INTO employee (employee_id, first_name, last_name, job, salary, department_id)
VALUES (110, 'Isha', 'Kumar', 'IT_PROG', 62000, 10);

INSERT INTO employee (employee_id, first_name, last_name, job, salary, department_id)
VALUES (111, 'Ravi', 'Sharma', 'HR_REP', 56000, 20);

INSERT INTO employee (employee_id, first_name, last_name, job, salary, department_id)
VALUES (112, 'Priya', 'Singh', 'FIN_ANALYST', 71000, 30);

INSERT INTO employee (employee_id, first_name, last_name, job, salary, department_id)
VALUES (113, 'Amit', 'Patel', 'SALES_REP', 46000, 40);

INSERT INTO employee (employee_id, first_name, last_name, job, salary, department_id)
VALUES (114, 'Neha', 'Verma', 'MARKETING', 51000, 50);

INSERT INTO employee (employee_id, first_name, last_name, job, salary, department_id)
VALUES (115, 'Karan', 'Mehta', 'IT_PROG', 63000, 10);
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

```sql
DECLARE
    v_total_salary NUMBER;
BEGIN
    SELECT salary
    INTO v_total_salary
    FROM employees
    WHERE employee_id = 100;

    DBMS_OUTPUT.PUT_LINE('Total Salary of Employee 100: ' || v_total_salary);
END;
```

```
Statement processed.
Total Salary of Employee 100: 60000
```

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

```
DECLARE
   num1 NUMBER := 10;
   num2 NUMBER := 20;
   num3 NUMBER := 15;
   greatest NUMBER;
BEGIN
   -- Compare the three numbers to find the greatest
   IF num1 >= num2 AND num1 >= num3 THEN
      greatest := num1;
   ELSIF num2 >= num1 AND num2 >= num3 THEN
      greatest := num2;
   ELSE
      greatest := num3;
   END IF;

   -- Output the greatest number
   DBMS_OUTPUT.PUT_LINE('The greatest number is: ' || greatest);
END;
```

```
Statement processed.
The greatest number is: 20
```

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

```
DECLARE
   n NUMBER := 10; -- You can change this value to any positive number
BEGIN
   FOR i IN 1..n LOOP
      DBMS_OUTPUT.PUT_LINE(i);
   END LOOP;
END;
```

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

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

```
DECLARE
   original_string VARCHAR2(100) := 'Hello, World!'; -- Input string
   reversed_string VARCHAR2(100) := ''; -- Variable to store the reversed string
BEGIN
   FOR i IN REVERSE 1..LENGTH(original_string) LOOP
      reversed_string := reversed_string || SUBSTR(original_string, i, 1);
   END LOOP;

   DBMS_OUTPUT.PUT_LINE('Original String: ' || original_string);
   DBMS_OUTPUT.PUT_LINE('Reversed String: ' || reversed_string);
END;
```

```
Statement processed.
Original String: Hello, World!
Reversed String: !dlroW ,olleH
```

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

```
DECLARE
sumVal NUMBER;
n NUMBER;
i NUMBER;

FUNCTION Findmax(n IN NUMBER)
   RETURN NUMBER
IS
   sums NUMBER := 0;
BEGIN
   FOR i IN 1..n
   LOOP
   sums := sums + i*(i+1)/2;
   END LOOP;
   RETURN sums;
END;
BEGIN
   n := 8;
   sumVal := findmax(n);
   dbms_output.Put_line('Sum of numbers is ' || sumVal);
END;
```

```
Statement processed.
Sum of numbers is 120
```

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

```
DECLARE
    CURSOR emp_cursor IS
        SELECT employee_id, first_name, last_name, job
        FROM employee
        WHERE department_id = 10;

    v_empno employee.employee_id%TYPE;
    v_ename employee.first_name%TYPE;
    v_lname employee.last_name%TYPE;
    v_job employee.job%TYPE;
BEGIN
    OPEN emp_cursor;
    LOOP
        FETCH emp_cursor INTO v_empno, v_ename, v_lname, v_job;
        EXIT WHEN emp_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Emp No: ' || v_empno || ', Name: ' || v_ename || ' ' || v_lname || ', Job: ' ||
    END LOOP;
    CLOSE emp_cursor;
END;
```

```
Statement processed.
Emp No: 110, Name: Isha Kumar, Job: IT_PROG
Emp No: 115, Name: Karan Mehta, Job: IT_PROG
```

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

```
DECLARE
    CURSOR top_paid_emp_cursor IS
        SELECT employee_id, first_name, last_name
        FROM employee
        ORDER BY salary DESC
        FETCH FIRST 5 ROWS ONLY;

    v_empno employee.employee_id%TYPE;
    v_ename employee.first_name%TYPE;
    v_lname employee.last_name%TYPE;
BEGIN
    OPEN top_paid_emp_cursor;
    LOOP
        FETCH top_paid_emp_cursor INTO v_empno, v_ename, v_lname;
        EXIT WHEN top_paid_emp_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Emp No: ' || v_empno || ', Name: ' || v_ename || ' ' || v_lname);
    END LOOP;
    CLOSE top_paid_emp_cursor;
END;
```

```
Statement processed.
Emp No: 112, Name: Priya Singh
Emp No: 115, Name: Karan Mehta
Emp No: 110, Name: Isha Kumar
Emp No: 111, Name: Ravi Sharma
Emp No: 114, Name: Neha Verma
```

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

```
CREATE OR REPLACE PACKAGE math_operations AS
   PROCEDURE calculate_operations(num1 IN NUMBER, num2 IN NUMBER);
END math_operations;
```

```
Package created.
```

```
CREATE OR REPLACE PACKAGE BODY math_operations AS

   -- Local Procedure for Addition
   PROCEDURE add_numbers(num1 IN NUMBER, num2 IN NUMBER, result OUT NUMBER) IS
   BEGIN
      result := num1 + num2;
   END add_numbers;

   -- Local Procedure for Subtraction
   PROCEDURE subtract_numbers(num1 IN NUMBER, num2 IN NUMBER, result OUT NUMBER) IS
   BEGIN
      result := num1 - num2;
   END subtract_numbers;

   -- Local Procedure for Multiplication
   PROCEDURE multiply_numbers(num1 IN NUMBER, num2 IN NUMBER, result OUT NUMBER) IS
   BEGIN
      result := num1 * num2;
   END multiply_numbers;
```

```
    -- Local Procedure for Division
    PROCEDURE divide_numbers(num1 IN NUMBER, num2 IN NUMBER, result OUT NUMBER) IS
    BEGIN
        result := num1 / num2;
    END divide_numbers;

    -- Stored Procedure to Call Local Procedures
    PROCEDURE calculate_operations(num1 IN NUMBER, num2 IN NUMBER) IS
        add_result NUMBER;
        subtract_result NUMBER;
        multiply_result NUMBER;
        divide_result NUMBER;
    BEGIN
        add_numbers(num1, num2, add_result);
        subtract_numbers(num1, num2, subtract_result);
        multiply_numbers(num1, num2, multiply_result);
        divide_numbers(num1, num2, divide_result);

        DBMS_OUTPUT.PUT_LINE('Addition: ' || add_result);
        DBMS_OUTPUT.PUT_LINE('Subtraction: ' || subtract_result);
        DBMS_OUTPUT.PUT_LINE('Multiplication: ' || multiply_result);
        DBMS_OUTPUT.PUT_LINE('Division: ' || divide_result);
    END calculate_operations;

END math_operations;
```

```
'ackage Body created.
```

```
BEGIN
    math_operations.calculate_operations(10, 5);
END;
```

```
Statement processed.
Addition: 15
Subtraction: 5
Multiplication: 50
Division: 2
```

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

```
CREATE OR REPLACE PACKAGE BODY math_functions AS
   FUNCTION add_numbers(num1 IN NUMBER, num2 IN NUMBER) RETURN NUMBER IS
   BEGIN
      RETURN num1 + num2;
   END add_numbers;
   FUNCTION subtract_numbers(num1 IN NUMBER, num2 IN NUMBER) RETURN NUMBER IS
   BEGIN
      RETURN num1 - num2;
   END subtract_numbers;
   FUNCTION multiply_numbers(num1 IN NUMBER, num2 IN NUMBER) RETURN NUMBER IS
   BEGIN
      RETURN num1 * num2;
   END multiply_numbers;
   FUNCTION divide_numbers(num1 IN NUMBER, num2 IN NUMBER) RETURN NUMBER IS
   BEGIN
      RETURN num1 / num2;
   END divide_numbers;
   FUNCTION calculate_operations(num1 IN NUMBER, num2 IN NUMBER) RETURN VARCHAR2 IS
      add_result NUMBER;
      subtract_result NUMBER;
      multiply_result NUMBER;
      divide_result NUMBER;
      result_string VARCHAR2(200);
   BEGIN
      add_result := add_numbers(num1, num2);
      subtract_result := subtract_numbers(num1, num2);
      multiply_result := multiply_numbers(num1, num2);
      divide_result := divide_numbers(num1, num2);

      result_string := 'Addition: ' || add_result ||
                       ', Subtraction: ' || subtract_result ||
```

```
                    ', Subtraction: ' || subtract_result ||
                    ', Multiplication: ' || multiply_result ||
                    ', Division: ' || divide_result;
    RETURN result_string;
  END calculate_operations;

END math_functions;
```

```
ackage Body created.
```

```
DECLARE
  result VARCHAR2(200);
BEGIN
  result := math_functions.calculate_operations(10, 5);
  DBMS_OUTPUT.PUT_LINE(result);
END;
```

```
tatement processed.
ddition: 15, Subtraction: 5, Multiplication: 50, Division: 2
```

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

```
DECLARE
temp varchar(20);

BEGIN
SELECT g_id into temp from geeks where g_name='GeeksforGeeks';

exception
WHEN no_data_found THEN
        dbms_output.put_line('ERROR');
        dbms_output.put_line('there is no name as');
        dbms_output.put_line('GeeksforGeeks in geeks table');
end;
```

```
Statement processed.
ERROR
there is no name as
GeeksforGeeks in geeks table
```

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

```
DECLARE
temp varchar(20);

BEGIN

-- raises an exception as SELECT
-- into trying to return too many rows
SELECT g_name into temp from geeks;
dbms_output.put_line(temp);

EXCEPTION
WHEN too_many_rows THEN
        dbms_output.put_line('error trying to SELECT too many rows');

end;
```

```
Statement processed.
error trying to SELECT too many rows
```

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

```
DECLARE
a int:=10;
b int:=0;
answer int;

BEGIN
answer:=a/b;
dbms_output.put_line('the result after division is'||answer);

exception
WHEN zero_divide THEN
        dbms_output.put_line('dividing by zero please check the values again');
        dbms_output.put_line('the value of a is '||a);
        dbms_output.put_line('the value of b is '||b);
END;
```

```
Statement processed.
dividing by zero please check the values again
the value of a is 10
the value of b is 0
```