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:

SUBMITTED BY:

Ms. Neerja Negi

NAME: Yashwant Verma

ROLL NO:22/FCA/BCA(CS)/049

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

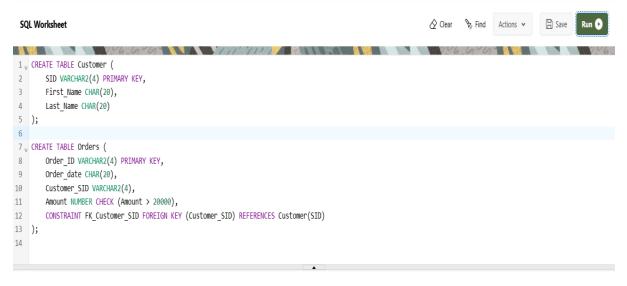
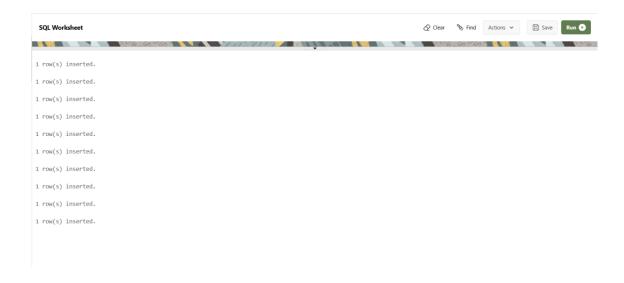


Table created.

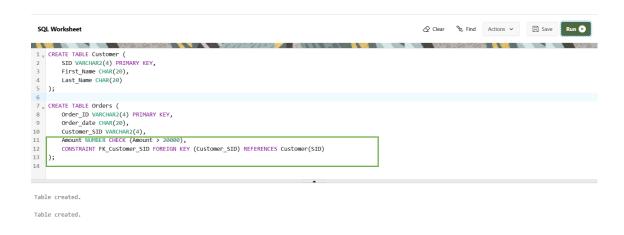
Table created.

2. Insert five records for each table.

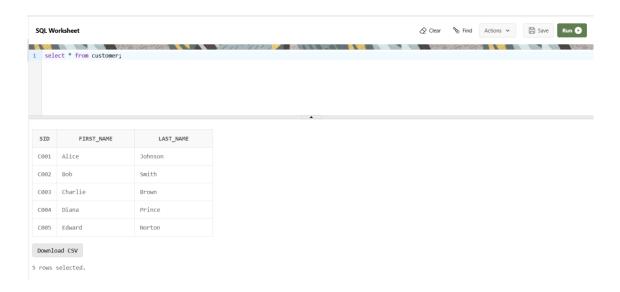
```
| SQL Worksheet | SQL Worksheet | SQL North | Square | S
```

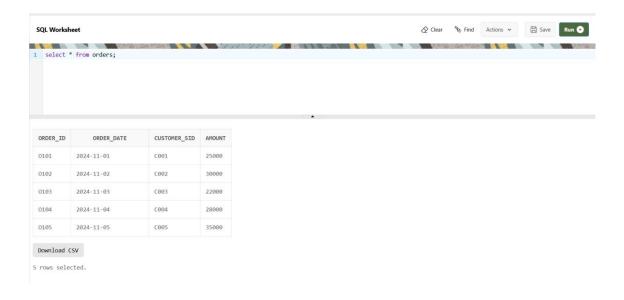


3. Customer_SID column in the ORDERS table is a foreign key pointing to the SID column in the CUSTOMER table.

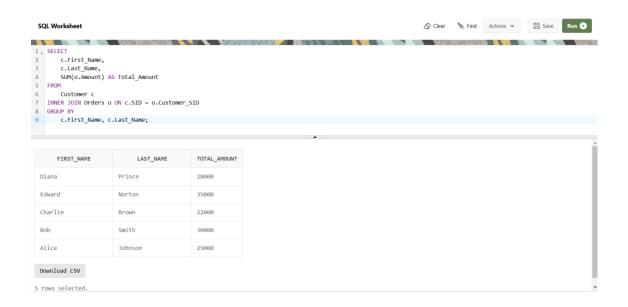


4. Insert five records for both tables





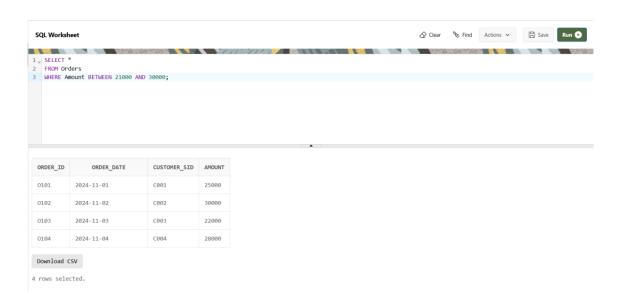
5. List the details of the customers along with the amount.



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



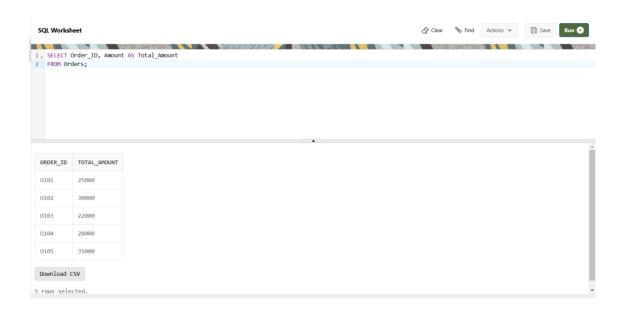
7. List the orders where amount is between 21000 and 30000



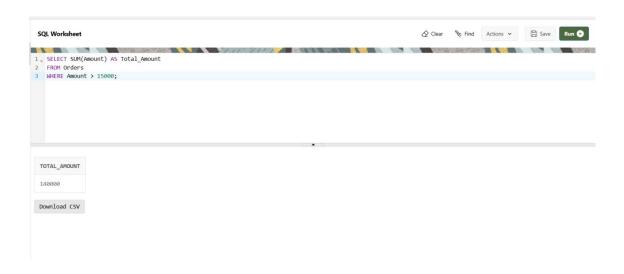
8. List the orders where amount is increased by 500 and replace with name "new amount".



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

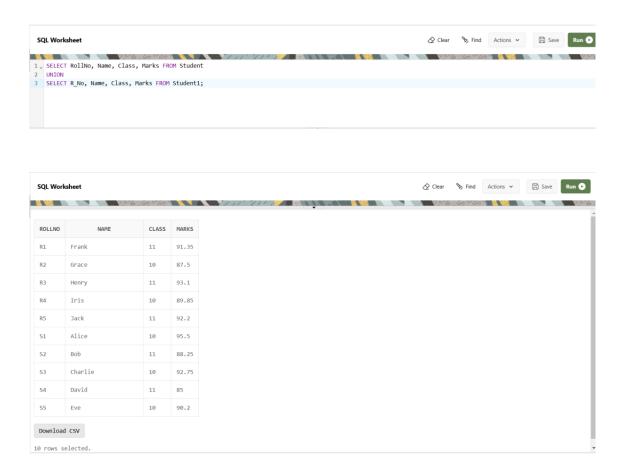
Student

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

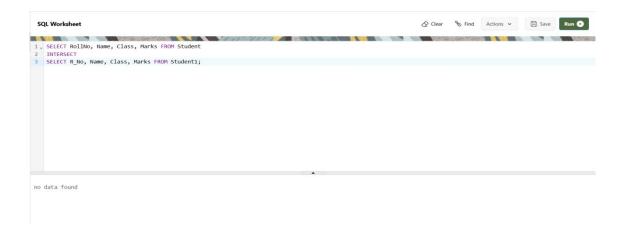
Student1

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

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.



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

```
1 v CREATE TABLE employee (
2
        employee_id NUMBER PRIMARY KEY,
3
        first_name VARCHAR2(50),
        last name VARCHAR2(50),
4
        job VARCHAR2(50),
5
        salary NUMBER(10, 2),
6
7
        department VARCHAR2(50)
8
   );
9
Table created.
```

```
1, INSERT INTO employee (employee id, first name, last name, job, salary, department id)
   VALUES (110, 'Isha', 'Kumar', 'IT_PROG', 62000, 10);
4 v INSERT INTO employee (employee_id, first_name, last_name, job, salary, department_id)
   VALUES (111, 'Ravi', 'Sharma', 'HR_REP', 56000, 20);
7 v INSERT INTO employee (employee_id, first_name, last_name, job, salary, department_id)
   VALUES (112, 'Priya', 'Singh', 'FIN_ANALYST', 71000, 30);
8
10 , INSERT INTO employee (employee_id, first_name, last_name, job, salary, department_id)
11 VALUES (113, 'Amit', 'Patel', 'SALES_REP', 46000, 40);
12
13 , INSERT INTO employee (employee_id, first_name, last_name, job, salary, department_id)
14 VALUES (114, 'Neha', 'Verma', 'MARKETING', 51000, 50);
15
16 , INSERT INTO employee (employee_id, first_name, last_name, job, salary, department_id)
17 VALUES (115, 'Karan', 'Mehta', 'IT_PROG', 63000, 10);
18
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 v DECLARE
2
         v total salary NUMBER;
3 , BEGIN
4
         SELECT salary
5
         INTO v_total_salary
         FROM employees
6
7
         WHERE employee id = 100;
8
         DBMS_OUTPUT.PUT_LINE('Total Salary of Employee 100: ' || v_total_salary);
9
10
    END;
11
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
   DBHS_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.

```
1 v DECLARE
        CURSOR emp_cursor IS
          SELECT employee_id, first_name, last_name, job
3
            FROM employee
 4
           WHERE department_id = 10;
 7
        v_empno employee.employee_id%TYPE;
 8
        v_ename employee.first_name%TYPE;
 9
        v_lname employee.last_name%TYPE;
10
        v_job employee.job%TYPE;
11 v BEGIN
12
        OPEN emp_cursor;
13 <sub>v</sub>
14
            FETCH emp_cursor INTO v_empno, v_ename, v_lname, v_job;
15
            EXIT WHEN emp_cursor%NOTFOUND;
           DBMS_OUTPUT.PUT_LINE('Emp No: ' || v_empno || ', Name: ' || v_ename || ' ' || v_lname || ', Job: ' ||
16
        END LOOP;
17
18
        CLOSE emp_cursor;
19 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.

```
1 v DECLARE
2
         CURSOR top_paid_emp_cursor IS
3
            SELECT employee_id, first_name, last_name
            FROM employee
 4
 5
            ORDER BY salary DESC
 6
            FETCH FIRST 5 ROWS ONLY;
 7
 8
         v_empno employee.employee_id%TYPE;
 9
         v ename employee.first name%TYPE;
10
         v lname employee.last name%TYPE;
11 v BEGIN
12
         OPEN top_paid_emp_cursor;
13 <sub>v</sub>
             FETCH top_paid_emp_cursor INTO v_empno, v_ename, v_lname;
14
             EXIT WHEN top_paid_emp_cursor%NOTFOUND;
15
16
             DBMS_OUTPUT.PUT_LINE('Emp No: ' || v_empno || ', Name: ' || v_ename || ' ' || v_lname);
17
         END LOOP;
         CLOSE top_paid_emp_cursor;
18
19 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
     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
      RETURN num1 * num2;
   END multiply numbers:
   FUNCTION divide_numbers(num1 IN NUMBER, num2 IN NUMBER) RETURN NUMBER IS
      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);
      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.

```
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.

Statement processed. error trying to SELECT too many rows

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

Statement processed. dividing by zero please check the values again the value of a is 10 the value of b is 0 $^{\circ}$