

# 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:** Yashwant Verma

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

1. Create the following tables

### Customer

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

### Orders

<u>Column_name</u>	<u>Data type</u>	<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

SQL Worksheet

Clear

Find

Actions

Save

Run

```
1 CREATE TABLE Customer (  
2     SID VARCHAR2(4) PRIMARY KEY,  
3     First_Name CHAR(20),  
4     Last_Name CHAR(20)  
5 );  
6  
7 CREATE TABLE Orders (  
8     Order_ID VARCHAR2(4) PRIMARY KEY,  
9     Order_date CHAR(20),  
10    Customer_SID VARCHAR2(4),  
11    Amount NUMBER CHECK (Amount > 20000),  
12    CONSTRAINT FK_Customer_SID FOREIGN KEY (Customer_SID) REFERENCES Customer(SID)  
13 );  
14
```

Table created.

Table created.

2. Insert five records for each table.

```
SQL Worksheet

1 INSERT INTO Customer (SID, First_Name, Last_Name)
2 VALUES ('C001', 'Alice', 'Johnson');
3
4 INSERT INTO Customer (SID, First_Name, Last_Name)
5 VALUES ('C002', 'Bob', 'Smith');
6
7 INSERT INTO Customer (SID, First_Name, Last_Name)
8 VALUES ('C003', 'Charlie', 'Brown');
9
10 INSERT INTO Customer (SID, First_Name, Last_Name)
11 VALUES ('C004', 'Diana', 'Prince');
12
13 INSERT INTO Customer (SID, First_Name, Last_Name)
14 VALUES ('C005', 'Edward', 'Norton');
15
16 INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount)
17 VALUES ('O101', '2024-11-01', 'C001', 25000);
18
19 INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount)
20 VALUES ('O102', '2024-11-02', 'C002', 30000);
21
22 INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount)
23 VALUES ('O103', '2024-11-03', 'C003', 22000);
24
25 INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount)
26 VALUES ('O104', '2024-11-04', 'C004', 28000);
27
28 INSERT INTO Orders (Order_ID, Order_date, Customer_SID, Amount)
29 VALUES ('O105', '2024-11-05', 'C005', 35000);
```

[illegible]

3. Customer\_SID column in the ORDERS table is a foreign key pointing to the SID column in the CUSTOMER table.

```
SQL Worksheet
Clear Find Actions Save Run
1 CREATE TABLE Customer (
2   SID VARCHAR2(4) PRIMARY KEY,
3   First_Name CHAR(20),
4   Last_Name CHAR(20)
5 );
6
7 CREATE TABLE Orders (
8   Order_ID VARCHAR2(4) PRIMARY KEY,
9   Order_date CHAR(20),
10  Customer_SID VARCHAR2(4),
11  Amount NUMBER CHECK (Amount > 20000),
12  CONSTRAINT FK_Customer_SID FOREIGN KEY (Customer_SID) REFERENCES Customer(SID)
13 );
14
```

Table created.

Table created.

4. Insert five records for both tables

```
SQL Worksheet
Clear Find Actions Save Run
1 select * from customer;
```

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”.

SQL Worksheet

Clear Find Actions Save Run

```
1 SELECT First_Name, Last_Name
2 FROM Customer
3 WHERE First_Name LIKE '%s' OR Last_Name LIKE '%s';
```

no data found

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

SQL Worksheet

Clear Find Actions Save Run

```
1 SELECT *
2 FROM Orders
3 WHERE Amount 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”.

SQL Worksheet

Clear Find Actions Save Run

```
1 SELECT *, Amount + 500 AS "New Amount"
2 FROM Orders;
```

9. Display the order\_id and total amount of orders

SQL Worksheet

Clear Find Actions Save Run

```
1 SELECT Order_ID, Amount AS Total_Amount
2 FROM Orders;
```

ORDER_ID	TOTAL_AMOUNT
0101	25000
0102	30000
0103	22000
0104	28000
0105	35000

Download CSV

5 rows selected.

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

SQL Worksheet

Clear

Find

Actions

Save

Run

```
1 SELECT SUM(Amount) AS Total_Amount
2 FROM Orders
3 WHERE Amount > 15000;
```

TOTAL\_AMOUNT

140000

Download CSV

12. Create the following tables

**Student**

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

**Student1**

<u>Column_name</u>	<u>Data type</u>	<u>Size</u>	<u>Constraint</u>
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.

SQL Worksheet

Clear Find Actions Save Run

```
1 SELECT RollNo, Name, Class, Marks FROM Student
2 UNION
3 SELECT R_No, Name, Class, Marks FROM Student1;
```

SQL Worksheet

Clear Find Actions Save Run

ROLLNO	NAME	CLASS	MARKS
R1	Frank	11	91.35
R2	Grace	10	87.5
R3	Henry	11	93.1
R4	Iris	10	89.85
R5	Jack	11	92.2
S1	Alice	10	95.5
S2	Bob	11	88.25
S3	Charlie	10	92.75
S4	David	11	85
S5	Eve	10	90.2

Download CSV

10 rows selected.

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

SQL Worksheet

Clear Find Actions Save Run

```
1 SELECT RollNo, Name, Class, Marks FROM Student
2 INTERSECT
3 SELECT R_No, Name, Class, Marks FROM Student1;
```

no data found

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

SQL Worksheet

Clear Find Actions Save Run

```
1 SELECT s.Name AS Student_Name, s1.Name AS Student1_Name
2 FROM Student s
3 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.

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

Table created.

```

1 v INSERT INTO employee (employee_id, first_name, last_name, job, salary, department_id)
2   VALUES (110, 'Isha', 'Kumar', 'IT_PROG', 62000, 10);
3
4 v INSERT INTO employee (employee_id, first_name, last_name, job, salary, department_id)
5   VALUES (111, 'Ravi', 'Sharma', 'HR_REP', 56000, 20);
6
7 v INSERT INTO employee (employee_id, first_name, last_name, job, salary, department_id)
8   VALUES (112, 'Priya', 'Singh', 'FIN_ANALYST', 71000, 30);
9
10 v INSERT INTO employee (employee_id, first_name, last_name, job, salary, department_id)
11   VALUES (113, 'Amit', 'Patel', 'SALES_REP', 46000, 40);
12
13 v INSERT INTO employee (employee_id, first_name, last_name, job, salary, department_id)
14   VALUES (114, 'Neha', 'Verma', 'MARKETING', 51000, 50);
15
16 v 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 v BEGIN
4   SELECT salary
5   INTO v_total_salary
6   FROM employees
7   WHERE employee_id = 100;
8
9   DBMS_OUTPUT.PUT_LINE('Total Salary of Employee 100: ' || v_total_salary);
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
    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.

```
1 v DECLARE
2     CURSOR emp_cursor IS
3         SELECT employee_id, first_name, last_name, job
4         FROM employee
5         WHERE department_id = 10;
6
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 v LOOP
14         FETCH emp_cursor INTO v_empno, v_ename, v_lname, v_job;
15         EXIT WHEN emp_cursor%NOTFOUND;
16         DBMS_OUTPUT.PUT_LINE('Emp No: ' || v_empno || ', Name: ' || v_ename || ' ' || v_lname || ', Job: ' ||
17     END LOOP;
18     CLOSE emp_cursor;
19 END;
20
```

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
4         FROM employee
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 v LOOP
14     FETCH top_paid_emp_cursor INTO v_empno, v_ename, v_lname;
15     EXIT WHEN top_paid_emp_cursor%NOTFOUND;
16     DBMS_OUTPUT.PUT_LINE('Emp No: ' || v_empno || ', Name: ' || v_ename || ' ' || v_lname);
17 END LOOP;
18 CLOSE top_paid_emp_cursor;
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
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;

```

Package 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 ||
```

---

```

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

END math_functions;

```

Package Body created.

---

```

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

```

Statement processed.

Addition: 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

