DBMS LAB : 4.5CA153C01

MANAV RACHNA INTERNATIONAL INSTITUTE OF RESEARCH AND STUDIES

DEPARTMENT OF COMPUTER SCIENCE

SUBMITTED BY	
Student name	Taranpreet Kaur
Roll No.	24/SCA/BCA(AI&ML)/068
Programme	BCA(AI&ML)
Semester	1
Section\ Group	B / GROUP-2
Department	SCA
Session\ Batch	2024-2027
SUBMITTED TO	
Faculty name	Mrs. Aastha budhiraja

Q1: Create the following tables

STUDENT

INPUT:

```
CREATE TABLE Student (
   StudentId NUMBER(4) PRIMARY KEY,
   StudentName VARCHAR2(40) NOT NULL,
   Address1 VARCHAR2(300),
   Gender VARCHAR2(15),
   Course VARCHAR2(8)
);
```

OUTPUT:

```
sql

Table created.
```

COURSE

INPUT:

```
CREATE TABLE Course (
   DeptNo NUMBER(2) PRIMARY KEY,
   Dname VARCHAR2(20),
   Location VARCHAR2(10)
);
```

OUTPUT:

```
sql

Table created.
```

1. Insert five records for each table.

STUDENT

INPUT:

```
INSERT INTO Student (StudentId, StudentName, Address1, Gender, Course)
VALUES (1, 'John Doe', '123 Main St', 'Male', 'MCA');

INSERT INTO Student (StudentId, StudentName, Address1, Gender, Course)
VALUES (2, 'Jane Smith', '456 Oak St', 'Female', 'BCA');

INSERT INTO Student (StudentId, StudentName, Address1, Gender, Course)
VALUES (3, 'Alice Johnson', '789 Pine St', 'Female', 'MCA');

INSERT INTO Student (StudentId, StudentName, Address1, Gender, Course)
VALUES (4, 'Bob Brown', '101 Maple St', 'Male', 'BCA');

INSERT INTO Student (StudentId, StudentName, Address1, Gender, Course)
VALUES (5, 'Charlie Davis', '202 Birch St', 'Male', 'MCA');
```

OUTPUT:

```sql
5 rows inserted.
```

COURSE

INPUT:

```
INSERT INTO Course (DeptNo, Dname, Location)
VALUES (1, 'Computer Science', 'New York');

INSERT INTO Course (DeptNo, Dname, Location)
VALUES (2, 'Electrical Engineering', 'Los Angeles');

INSERT INTO Course (DeptNo, Dname, Location)
VALUES (3, 'Mechanical Engineering', 'Chicago');

INSERT INTO Course (DeptNo, Dname, Location)
VALUES (4, 'Civil Engineering', 'Dallas');

INSERT INTO Course (DeptNo, Dname, Location)
VALUES (5, 'Business Administration', 'Miami');
```

OUTPUT:

```
sql

5 rows inserted.
```

2. List all information about all students from student table
   INPUT:
   SELECT * FROM Student;
   OUTPUT:

Output

StudentId	StudentName	Address1	Gender	Course
1	John Doe	123 Main St	Male	MCA
2	Jane Smith	456 Oak St	Female	BCA
3	Alice Johnson	789 Pine St	Female	MCA
4	Bob Brown	101 Maple St	Male	BCA
5	Charlie Davis	202 Birch St	Male	MCA

3. List all student numbers along with their Courses.
   INPUT:
   SELECT StudentId, Course FROM Student;
   OUTPUT:

Output

StudentId	Course
1	MCA
2	BCA
3	MCA
4	BCA
5	MCA

4. List Course names and locations from the Course table
   INPUT:
   SELECT Dname, Location FROM Course;
   OUTPUT:

Output

Dname	Location
Computer Science	New York
Electrical Engineering	Los Angeles
Mechanical Engineering	Chicago
Civil Engineering	Dallas
Business Administration	Miami

5. List the details of the Students in MCA Course.

INPUT:

```
SELECT * FROM Student WHERE Course = 'MCA';
```

OUTPUT:

Output

StudentId	StudentName	Address1	Gender	Course
1	John Doe	123 Main St	Male	MCA
3	Alice Johnson	789 Pine St	Female	MCA
5	Charlie Davis	202 Birch St	Male	MCA

EMPLOYEE TABLE

INPUT:

```
CREATE TABLE Employee (
   EmployeeNo NUMBER(4) PRIMARY KEY,
   EmployeeName VARCHAR2(40) NOT NULL,
   DepartmentNo NUMBER(2),
   Salary NUMBER(8, 2),
   Commission NUMBER(8, 2)
);

INSERT INTO Employee (EmployeeNo, EmployeeName, DepartmentNo, Salary, Commission)
VALUES (7369, 'John Smith', 10, 5000, 500);
```

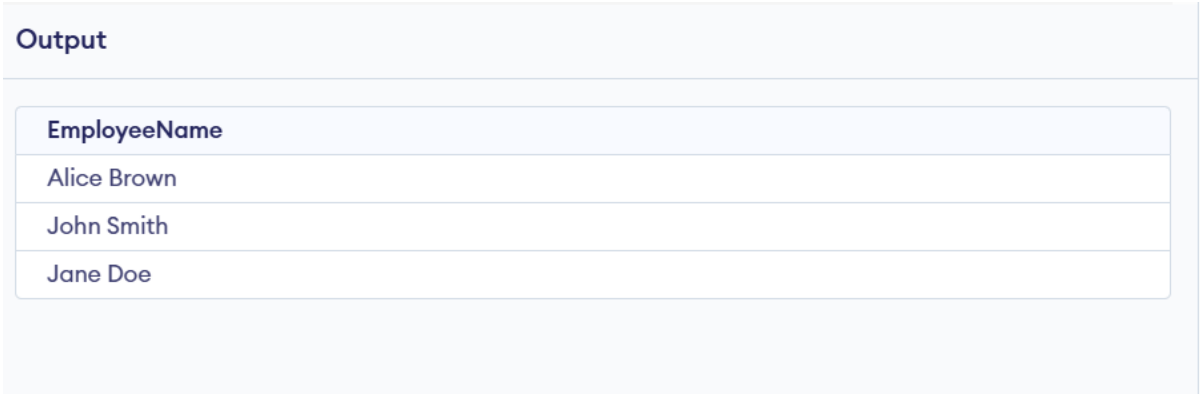
INSERT INTO Employee (EmployeeNo, EmployeeName, DepartmentNo, Salary, Commission)
VALUES (7777, 'Jane Doe', 20, 6000, 600);

INSERT INTO Employee (EmployeeNo, EmployeeName, DepartmentNo, Salary, Commission)
VALUES (2233, 'Alice Brown', 30, 5500, 550);

INSERT INTO Employee (EmployeeNo, EmployeeName, DepartmentNo, Salary, Commission)
VALUES (1111, 'Bob White', 40, 4500, 450);

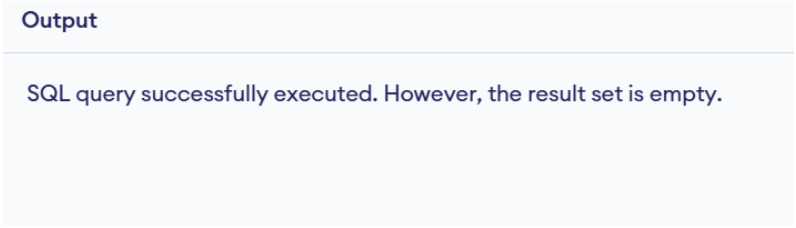
INSERT INTO Employee (EmployeeNo, EmployeeName, DepartmentNo, Salary, Commission)
VALUES (1001, 'Charlie Davis', 50, 4700, 470);

6. List the names of the employees whose employees numbers are 7369, 7777, 2233
   INPUT:
   SELECT EmployeeName FROM Employee WHERE EmployeeNo IN (7369, 7777, 2233);
   OUTPUT:

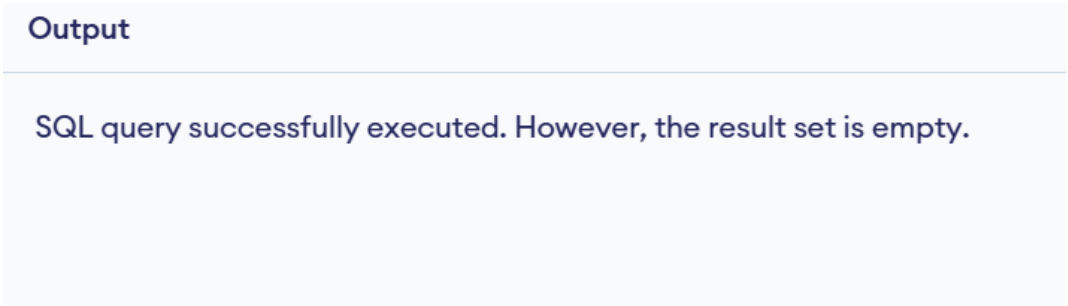

Output

EmployeeName
Alice Brown
John Smith
Jane Doe

7. List the employee names not belonging to the department 10, 40
   INPUT:
   SELECT EmployeeName FROM Employee WHERE DeptNo NOT IN (10, 40);
8. List the employee names who are not eligible for commission.
   INPUT:
   SELECT EmployeeName
   FROM Employee
   WHERE Commission IS NULL;
9. List the employees whose names start with “S” not s.
   INPUT:
   SELECT EmployeeName
   FROM Employee
   WHERE EmployeeName LIKE 'S%';
   OUTPUT:



10. List the employees ending with name “s”.
    INPUT:
    SELECT EmployeeName
    FROM Employee
    WHERE EmployeeName LIKE '%s';
    OUTPUT:



11. Display all the Arithmetic functions used in SQL.
    INPUT:
    SELECT
      5 + 3 AS Add_Operation,
      5 - 3 AS Subtract_Operation,
      5 * 3 AS Multiply_Operation,
      5 / 3 AS Divide_Operation,
     (5% 3) AS Modulus_Operation;
    OUTPUT:

12. List the names, salary and PF amount of all the employees (PF is calculated as 10% of salary)
    INPUT:
    SELECT EmployeeName, Salary, (Salary * 0.10) AS PF_Amount
    FROM Employee;
13. List the employee names having “k” as the second character.
    INPUT:
    SELECT EmployeeName
    FROM Employee

WHERE EmployeeName LIKE '_k%';

14. List the students not assigned to any department.
    INPUT:
    SELECT * FROM Student WHERE Course IS NULL;
    OUTPUT:

Output

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

15. List the students details in ascending order of course
    INPUT:
    SELECT * FROM Student ORDER BY Course ASC;
    OUTPUT:

Output

StudentId	StudentName	Address1	Gender	Course
2	Jane Smith	456 Oak St	Female	BCA
4	Bob Brown	101 Maple St	Male	BCA
1	John Doe	123 Main St	Male	MCA
3	Alice Johnson	789 Pine St	Female	MCA
5	Charlie Davis	202 Birch St	Male	MCA

16. List the number of Students in BCA course.
    INPUT:
    SELECT COUNT(*) FROM Student WHERE Course = 'BCA';
    OUTPUT:

Output

COUNT(*)
2

17. List the number of students available in student table.

INPUT:

SELECT COUNT(*) FROM Student;

OUTPUT:

Output

COUNT(*)
5

18. Create a table with a primary key constraint.

```
CREATE TABLE Employee (
   EmployeeNo NUMBER(4) PRIMARY KEY,
   EmployeeName VARCHAR2(40) NOT NULL,
   DepartmentNo NUMBER(2),
   Salary NUMBER(8, 2),
   Commission NUMBER(8, 2)
);
```

19. Create a table with all column having not null constraints

```
CREATE TABLE Student (
   StudentId NUMBER(4) PRIMARY KEY,
   StudentName VARCHAR2(40) NOT NULL,
   Address1 VARCHAR2(300) NOT NULL,
   Gender VARCHAR2(15) NOT NULL,
   Course VARCHAR2(8) NOT NULL
);
```

20. Create a foreign key constraint in a table

```
 CREATE TABLE Department (
   DeptNo NUMBER(2) PRIMARY KEY,
   DeptName VARCHAR2(20)
);

CREATE TABLE Employee (
   EmployeeNo NUMBER(4) PRIMARY KEY,
   EmployeeName VARCHAR2(40) NOT NULL,
   DeptNo NUMBER(2),
   FOREIGN KEY (DeptNo) REFERENCES Department(DeptNo)
);
```

21. Create a Table with a unique key constraint

```
CREATE TABLE Employee (
   EmployeeNo NUMBER(4) PRIMARY KEY,
   EmployeeName VARCHAR2(40) NOT NULL,
```

```
    EmployeeEmail VARCHAR2(100) UNIQUE,
    DepartmentNo NUMBER(2)
);
```

22. Display the different students in department 1 and 2.

INPUT:

```
SELECT * FROM Student
WHERE Course IN ('1', '2');
```

OUTPUT:

```
(Empty Set, depends on data)
```

23. Display list of student ordered by course

INPUT:

```
 SELECT * FROM Student
ORDER BY Course;
```

OUTPUT:

Output

StudentId	StudentName	Address1	Gender	Course
2	Jane Smith	456 Oak St	Female	BCA
4	Bob Brown	101 Maple St	Male	BCA
1	John Doe	123 Main St	Male	MCA
3	Alice Johnson	789 Pine St	Female	MCA
5	Charlie Davis	202 Birch St	Male	MCA

24. Display alphabetically sorted list of students

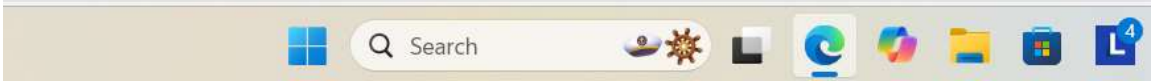
INPUT:

```
SELECT *
 FROM Student
ORDER BY StudentName;
```

OUTPUT:

Output

StudentId	StudentName	Address1	Gender	Course
3	Alice Johnson	789 Pine St	Female	MCA
4	Bob Brown	101 Maple St	Male	BCA
5	Charlie Davis	202 Birch St	Male	MCA
2	Jane Smith	456 Oak St	Female	BCA
1	John Doe	123 Main St	Male	MCA



Q2: Create the tables Customer and Orders as per the following:

INPUT:

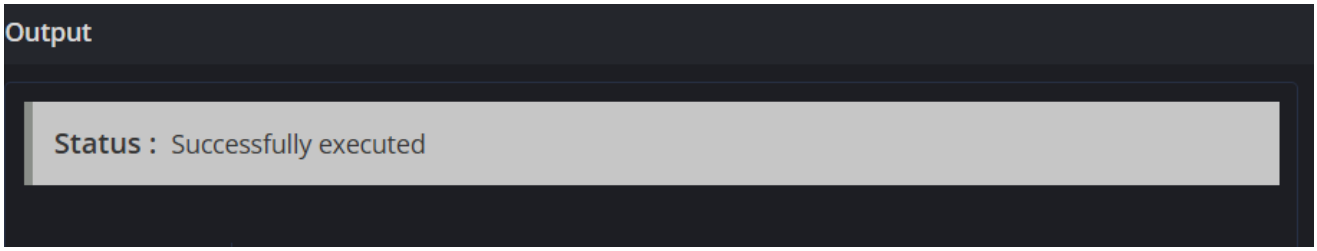
CUSTOMER table

```
CREATE TABLE CUSTOMER (
   SID NUMBER(4) PRIMARY KEY,
   Last_Name VARCHAR2(40),
   First_Name VARCHAR2(40)
);
```

ORDERS table

```
CREATE TABLE ORDERS (
   Order_ID NUMBER(4) PRIMARY KEY,
   Order_Date DATE,
   Customer_SID NUMBER(4),
   Amount NUMBER(10,2),
   CONSTRAINT fk_customer FOREIGN KEY (Customer_SID) REFERENCES CUSTOMER(SID),
   CONSTRAINT check_amount CHECK (Amount > 20000)
);
```

OUTPUT:



1. Insert five records for each table

INPUT:

CUSTOMER table

```
INSERT INTO CUSTOMER (SID, Last_Name, First_Name) VALUES (1, 'Smith', 'John');
INSERT INTO CUSTOMER (SID, Last_Name, First_Name) VALUES (2, 'Doe', 'Jane');
INSERT INTO CUSTOMER (SID, Last_Name, First_Name) VALUES (3, 'Brown', 'Charlie');
INSERT INTO CUSTOMER (SID, Last_Name, First_Name) VALUES (4, 'Johnson', 'Alice');
INSERT INTO CUSTOMER (SID, Last_Name, First_Name) VALUES (5, 'Williams', 'David');
```

ORDERS table

```
INSERT INTO ORDERS (Order_ID, Order_Date, Customer_SID, Amount) VALUES (101,
'2024-01-01', 1, 25000);
INSERT INTO ORDERS (Order_ID, Order_Date, Customer_SID, Amount) VALUES (102,
'2024-02-15', 2, 30000);
INSERT INTO ORDERS (Order_ID, Order_Date, Customer_SID, Amount) VALUES (103,
'2024-03-10', 3, 22000);
INSERT INTO ORDERS (Order_ID, Order_Date, Customer_SID, Amount) VALUES (104,
'2024-04-05', 4, 25000);
INSERT INTO ORDERS (Order_ID, Order_Date, Customer_SID, Amount) VALUES (105,
'2024-05-20',5, 21000);
```

OUTPUT:

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

INPUT:

CUSTOMER table

```
CREATE TABLE CUSTOMER (
```

```
    SID NUMBER(4) PRIMARY KEY,
    Last_Name VARCHAR2(40),
    First_Name VARCHAR2(40)
);

 ORDERS table with the foreign key constraint
CREATE TABLE ORDERS (
    Order_ID NUMBER(4) PRIMARY KEY,
    Order_Date DATE,
    Customer_SID NUMBER(4),
    Amount NUMBER(10,2),
    CONSTRAINT fk_customer FOREIGN KEY (Customer_SID) REFERENCES CUSTOMER(SID),
    CONSTRAINT check_amount CHECK (Amount > 20000)
);
```

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

INPUT:

```
SELECT c.SID, c.Last_Name, c.First_Name, o.Amount
FROM CUSTOMER c
JOIN ORDERS o ON c.SID = o.Customer_SID;
```

OUTPUT:

Output Generated Files

```
1|Smith|John|25000
2|Doe|Jane|30000
3|Brown|Charlie|22000
4|Johnson|Alice|25000
5|Williams|David|21000
```

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

INPUT:

```
SELECT *
FROM CUSTOMER
WHERE Last_Name LIKE '%s';
```

OUTPUT:

Output Generated Files

```
5|Williams|David
```

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

INPUT:

```
SELECT *
FROM ORDERS
WHERE Amount BETWEEN 21000 AND 30000;
```

OUTPUT:

Output Generated Files

```
101|2024-01-01|1|25000
102|2024-02-15|2|30000
103|2024-03-10|3|22000
104|2024-04-05|4|25000
105|2024-05-20|5|21000
```

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

INPUT:

```
SELECT Order_ID, Amount + 500 AS "new amount"
FROM ORDERS;
```

OUTPUT:

Output Generated Files

```
101|25500
102|30500
103|22500
104|25500
105|21500
```

7. Display the order_id and total amount of orders

INPUT:

```
SELECT Order_ID, Amount
FROM ORDERS;
```

OUTPUT:

Output Generated Files

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

INPUT:

```
SELECT SUM(Amount) AS Total_Amount
FROM ORDERS
WHERE Amount > 15000;
```

OUTPUT:

Output Generated Files

9. Display all the contents of s4 and s5 using union clause.

INPUT:

```
SELECT * FROM S4
UNION
SELECT * FROM S5;
```

10. Find out the intersection of s4 and s5 tables.

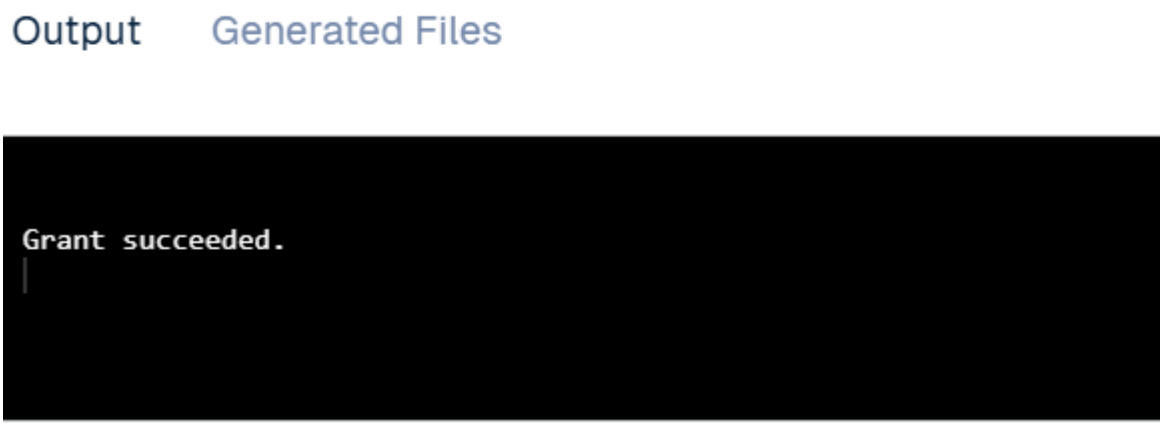
```
SELECT * FROM S4
INTERSECT
SELECT * FROM S5;
```

11. Display the names of s4 and s5 tables using left, right, inner and full join.

```
-- Left Join
SELECT *
FROM S4
LEFT JOIN S5 ON S4.id = S5.id;

-- Right Join
SELECT *
```

```
FROM S4
RIGHT JOIN S5 ON S4.id = S5.id;

-- Inner Join
SELECT *
FROM S4
INNER JOIN S5 ON S4.id = S5.id;

-- Full Join
SELECT *
FROM S4
FULL JOIN S5 ON S4.id = S5.id;
```

12. Display the first name of employee and their managers using self-join.
    INPUT:
    ```
    SELECT e.First_Name AS Employee_Name, m.First_Name AS Manager_Name
    FROM Employee e
    LEFT JOIN Employee m ON e.Manager_ID = m.EmployeeNo:
    ```
13. Find out the names of s4 which are distinct
    ```
    SELECT DISTINCT * FROM S4;
    ```
14. Write a query to Grant access and modification rights to customer table to user
    INPUT:
    ```
    GRANT SELECT, INSERT, UPDATE, DELETE ON CUSTOMER TO user_name;
    ```
    OUTPUT:



15. Write a query to revoke access rights to customer table to user
    CUSTOMER table
    ```
    REVOKE SELECT, INSERT, UPDATE, DELETE ON CUSTOMER FROM username;
    ```
16. Write a query to take backup of a database
    ```
    exp username/password@database full=y file=full_db_backup.dmp log=backup_log.txt
    ```
    USING RMAN (PHYSICAL BACKUP)
    ```
    rman target /
    BACKUP DATABASE;
    ```
17. Write a query to restore a database
    ```
    rman target /
    RESTORE DATABASE;
    ```