1. BASIC OF DATABASE

Write an SQL query to find the top 5 customers with the highest total order amounts.

Column	Type
customer_id	varchar
customer_name	char
region	char
order	int

```
SELECT TOP 5 Customer_ID,Customer_name
FROM Customer_Table
ORDER BY order DESC
```

2. AGGREGATE FUNCTION

Write an SQL query to find the second highest salary in the "Salary" table without using the LIMIT clause.

Column	Type
emp_id	varchar
emp_name	char
department	char
salary	int

Table: employee_table

```
SELECT Emp_ID,Emp_name,MAX( Salary) AS Salary
FROM Employee_table
WHERE Salary <> (SELECT MAX(Salary) FROM Employee_table)
```

3. NO DUPLICITY

Query the list of CITY names from the given table that do not end with vowels. Your result should not contain duplicate values.

Column	Type
id	int
city	char
state	char

Table: city_table

```
SELECT City
FROM City_Table
```

```
WHERE City NOT LIKE '%a' AND
    City NOT LIKE '%e' AND
        City NOT LIKE '%i' AND
        City NOT LIKE '%o' AND
        City NOT LIKE '%u'
```

4. DATA RETRIVAL & FILTERING

Write an SQL query to retrieve the names of employees who have worked in at least three different departments.

Column	Type
emp_id	int
emp_name	char
department	char
salary	int

Table: employee_table

```
SELECT Emp_ID, Emp_name,Department
FROM Emp_Table
GROUP BY Emp_name DESC
HAVING COUNT(Dapartment)>=3
```

5. DATA RETRIVAL & FILTERING

Write an SQL query to find the employees who have been with the company for more than 5 years and have not received a promotion.

Column	Type
emp_id	int
emp_name	char
department	char
duration	int

Table: employee_table

Assuming promotion table exists with hiring date and promotion status

```
SELECT *
FROM employees e, Emp_promotion ep
WHERE DATEDIFF(CURDATE(), ep.hire_date) > 1825 -- 5 years in days
  AND  ep.promotion_status = 'Not Promoted'
```

6. WINDOWS FUNCTION

Write an SQL query using Windows function to identify the employees who have a salary higher than the average salary of their respective departments.

Column	Type
emp_id	int
emp_name	char
department	char
salary	int

Table: employee_table

```
SELECT Emp_ID, Emp_Name, Department, Salary,
    AVG(Salary) OVER (PARTITION BY Department) AS AvgSalary
FROM Employee
WHERE (Salary > AVG(Salary) OVER (order by salary))
```

7. FOREIGN KEY

Given the table schemas below, write a query to print the ***company_code***, ***founder name***, ***total number of lead managers***, ***total number of senior managers***, ***total number of managers***, and ***total number of employees***. Order your output by ascending ***company_code***.

Column	Type
comany_code	varchar
founder	char

Table: company

Column	Type
lead_manager_code	varchar
founder	char

Table: lead_manager

Column	Type
senior_manager_code	varchar
lead_manager_code	varchar
company_code	varchar

Table: senior_manager

Column	Type
manager_code	varchar
senior_manager_code	varchar
lead_manager_code	varchar
company_code	varchar

Table: manager

Column	Type
employee_code	varchar
manager_code	varchar
senior_manager_code	varchar
lead_manager_code	varchar
company_code	varchar

Table: employee

```
SELECT
    C.company_code,
    C.founder,
    COUNT(DISTINCT SM.senior_manager_code) AS total_senior_managers,
    COUNT(DISTINCT M.manager_code) AS total_managers,
    COUNT(DISTINCT E.employee_code) AS total_employees
FROM
```

```
    company AS C
LEFT JOIN
    senior_manager AS SM ON C.company_code = SM.company_code
LEFT JOIN
    manager AS M ON C.company_code = M.company_code
LEFT JOIN
    employee AS E ON C.company_code = E.company_code
GROUP BY C.company_code ASC
```

8. JOIN OPERATION

Write an SQL query to perform following join operation
Left Join
Right Join
Inner Join
Full Outer Join
Self Join
Cartesian Join

Column	Type
emp_id	int
Name	char
Salary	int

Table: emp_table1

Column	Type
emp_id	int
department	char

Table: emp_table 2

Left Join

```
SELECT *
FROM left_table
LEFT JOIN right_table ON left_table.column_name = right_table.column_name;
```

Right Join

```
SELECT *
FROM left_table
RIGHT JOIN right_table ON left_table.column_name = right_table.column_name;
```

Inner Join

```
SELECT *
FROM table1
INNER JOIN table2 ON table1.column_name = table2.column_name;
```

Outer Join

```
SELECT *
FROM table1
FULL OUTER JOIN table2 ON table1.column_name = table2.column_name;
```

Self Join

```
SELECT *
FROM table AS t1
JOIN table AS t2 ON t1.column_name = t2.column_name;
```

Cross/Cartesian Join

```
SELECT *
FROM table1
CROSS JOIN table2;
```

9. TEMPORARY TABLE

Write an SQL query to perform following operations:
Create a Local Temporary Table and insert values in it.
Create a Global Temporary Table and insert values in it.
Delete Temporary Table

Local Temporary Table

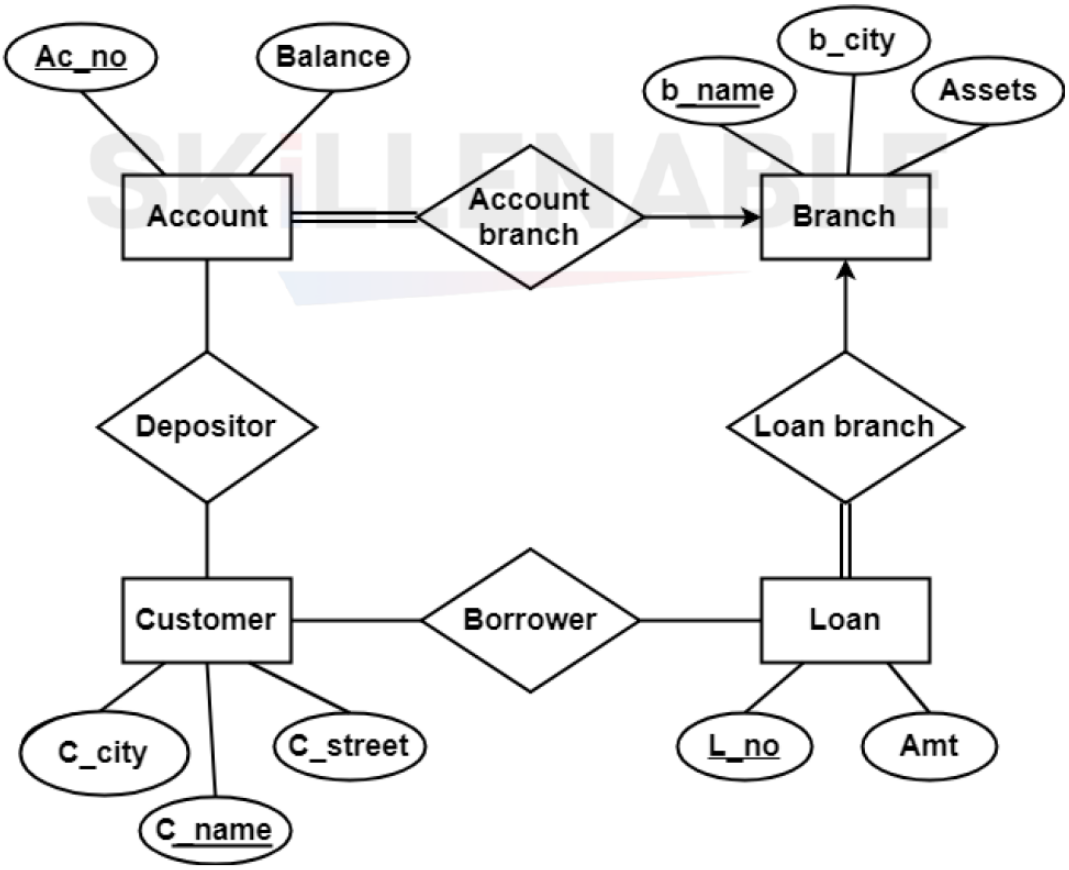
```
CREATE TABLE #Temp_table
  (
        ID  INT,
        Name VARCHAR(20),
        City VARCHAR(20),
        )
        INSERT INTO #Temp_table (ID, Name, City)
        VALUES(1, 'ABC', 'Hyderabad'),
                (2, 'XYZ', 'Mumbai'),
                (3, 'STU', 'Bhopal'),
                (4, 'MNO', 'Chennai')
        DROP TABLE #Temp_table
```

Global Temporary Table

```
CREATE TABLE ##Global_Temp_Table
(
ID  INT,
Name VARCHAR(20),
City VARCHAR(20),
)
INSERT INTO ##Global_Temp_Table (ID, Name, City)
VALUES(1, 'ABC', 'Hyderabad'),
        (2, 'XYZ', 'Mumbai'),
        (3, 'STU', 'Bhopal'),
        (4, 'MNO', 'Chennai')
DROP TABLE ##Global_Temp_Table
```

10. TEMPORARY TABLE

Convert ER diagram into table



Account Table

ACC_No int	Balance float	B_name varchar

Branch Table

B_name varchar	B_city varchar	Assests varchar

Customer Table

C_name varchar	C_city varchar	C_Street varchar

Loan Table

L_no int	L_amt float

Depositor Table

Acc_No int	C_Name varchar

Borrower Table

L_No int	C_Name varchar	B_Name varchar