# Pre-Requsites

# Run all the below comments

CREATE TABLE BANK\_CUSTOMER

( customer\_id INT PRIMARY KEY,

customer\_name VARCHAR(20),

Address VARCHAR(20),

state\_code VARCHAR(3) ,

Telephone VARCHAR(10));

INSERT INTO BANK\_CUSTOMER VALUES (123001,"Oliver", "225-5, Emeryville", "CA" , "1897614500");

INSERT INTO BANK\_CUSTOMER VALUES (123002,"George", "194-6,New brighton","MN" , "1897617000");

INSERT INTO BANK\_CUSTOMER VALUES (123003,"Harry", "2909-5,walnut creek","CA" , "1897617866");

INSERT INTO BANK\_CUSTOMER VALUES (123004,"Jack", "229-5, Concord",CA , "1897627999");

INSERT INTO BANK\_CUSTOMER VALUES (123005,"Jacob", "325-7, Mission Dist","SFO", "1897637000");

INSERT INTO BANK\_CUSTOMER VALUES (123006,"Noah", "275-9, saint-paul" , "MN" , "1897613200");

INSERT INTO BANK\_CUSTOMER VALUES (123007,"Charlie","125-1,Richfield", "MN" , "1897617666");

INSERT INTO BANK\_CUSTOMER VALUES (123008,"Robin","3005-1,Heathrow",NY , "1897614000");

CREATE TABLE Bank\_Account\_Details

(Customer\_id INT,

Account\_Number VARCHAR(19) PRIMARY KEY,

Account\_type VARCHAR(25) ,

Balance\_amount INT,

Account\_status VARCHAR(10),

Relationship\_type varchar(1)) ;

ALTER TABLE Bank\_Account\_Details ADD FOREIGN KEY (Customer\_id) REFERENCES bank\_customer(Customer\_id);

INSERT INTO Bank\_Account\_Details VALUES (123001, "4000-1956-3456", "SAVINGS" , 200000 ,"ACTIVE","P");

INSERT INTO Bank\_Account\_Details VALUES (123001, "5000-1700-3456", "RECURRING DEPOSITS" ,9400000 ,"ACTIVE","S");

INSERT INTO Bank\_Account\_Details VALUES (123002, "4000-1956-2001", "SAVINGS", 400000 ,"ACTIVE","P");

INSERT INTO Bank\_Account\_Details VALUES (123002, "5000-1700-5001", "RECURRING DEPOSITS" ,7500000 ,"ACTIVE","S");

INSERT INTO Bank\_Account\_Details VALUES (123003, "4000-1956-2900", "SAVINGS" ,750000,"INACTIVE","P");

INSERT INTO Bank\_Account\_Details VALUES (123004, "5000-1700-6091", "RECURRING DEPOSITS" ,7500000 ,"ACTIVE","S");

INSERT INTO Bank\_Account\_Details VALUES (123004, "4000-1956-3401", "SAVINGS" , 655000 ,"ACTIVE","P");

INSERT INTO Bank\_Account\_Details VALUES (123005, "4000-1956-5102", "SAVINGS" , 300000 ,"ACTIVE","P");

INSERT INTO Bank\_Account\_Details VALUES (123006, "4000-1956-5698", "SAVINGS" , 455000 ,"ACTIVE" ,"P");

INSERT INTO Bank\_Account\_Details VALUES (123007, "5000-1700-9800", "SAVINGS" , 355000 ,"ACTIVE" ,"P");

INSERT INTO Bank\_Account\_Details VALUES (123007, "4000-1956-9977", "RECURRING DEPOSITS" , 7025000,"ACTIVE" ,"S");

INSERT INTO Bank\_Account\_Details VALUES (123007, "9000-1700-7777-4321", "Credit Card",0 ,"INACTIVE", "P");

INSERT INTO Bank\_Account\_Details VALUES (123007, '5900-1900-9877-5543', "Add-on Credit Card" , 0 ,"ACTIVE", "S");

INSERT INTO Bank\_Account\_Details VALUES (123006, '5800-1700-9800-7755', "Credit Card" ,0,"ACTIVE", "P");

INSERT INTO Bank\_Account\_Details VALUES (123006, '5890-1970-7706-8912', "Add-on Credit Card" ,0,"ACTIVE", "S");

INSERT INTO Bank\_Account\_Details VALUES (123004,'5000-1700-7791','RECURRING DEPOSITS',40000,'ACTIVE','S');

CREATE TABLE Bank\_Account\_Relationship\_Details

( Customer\_id INT ,

Account\_Number VARCHAR(19) PRIMARY KEY ,

Account\_type VARCHAR(25),

Linking\_Account\_Number VARCHAR(19) );

ALTER TABLE Bank\_Account\_Relationship\_Details ADD FOREIGN KEY (Customer\_id) REFERENCES bank\_customer(Customer\_id);

ALTER TABLE Bank\_Account\_Relationship\_Details ADD FOREIGN KEY (Linking\_Account\_Number) REFERENCES bank\_account\_details(Account\_Number);

INSERT INTO Bank\_Account\_Relationship\_Details VALUES (123001, "4000-1956-3456", "SAVINGS" , NULL);

INSERT INTO Bank\_Account\_Relationship\_Details VALUES (123001, "5000-1700-3456", "RECURRING DEPOSITS" , "4000-1956-3456");

INSERT INTO Bank\_Account\_Relationship\_Details VALUES (123002, "4000-1956-2001", "SAVINGS" , NULL );

INSERT INTO Bank\_Account\_Relationship\_Details VALUES (123002, "5000-1700-5001", "RECURRING DEPOSITS" , "4000-1956-2001" );

INSERT INTO Bank\_Account\_Relationship\_Details VALUES (123003, "4000-1956-2900", "SAVINGS" , NULL );

INSERT INTO Bank\_Account\_Relationship\_Details VALUES (123004, "5000-1700-6091", "RECURRING DEPOSITS" , "4000-1956-2900" );

INSERT INTO Bank\_Account\_Relationship\_Details VALUES (123004, "5000-1700-7791", "RECURRING DEPOSITS" , "4000-1956-2900" );

INSERT INTO Bank\_Account\_Relationship\_Details VALUES (123007, "5000-1700-9800", "SAVINGS" , NULL);

INSERT INTO Bank\_Account\_Relationship\_Details VALUES (123007, "4000-1956-9977", "RECURRING DEPOSITS" , "5000-1700-9800" );

===================================================================================

Question 1

calculate the number of customer has a primary account type as "SAVINGS"

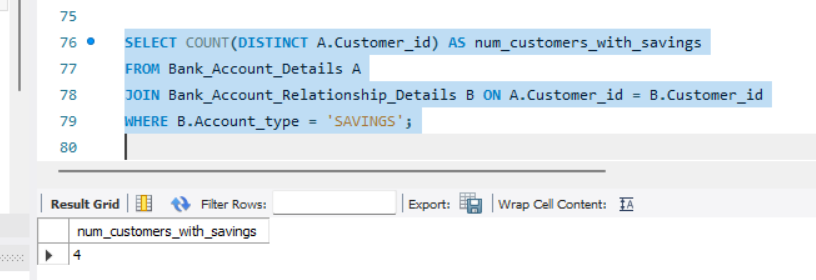
**CODE :**

**SELECT COUNT(DISTINCT A.Customer\_id) AS num\_customers\_with\_savings**

**FROM Bank\_Account\_Details A**

**JOIN Bank\_Account\_Relationship\_Details B ON A.Customer\_id = B.Customer\_id**

**WHERE B.Account\_type = 'SAVINGS';**

**OUTPUT :** 

Question 2

calculate the number of account owned by the each customer

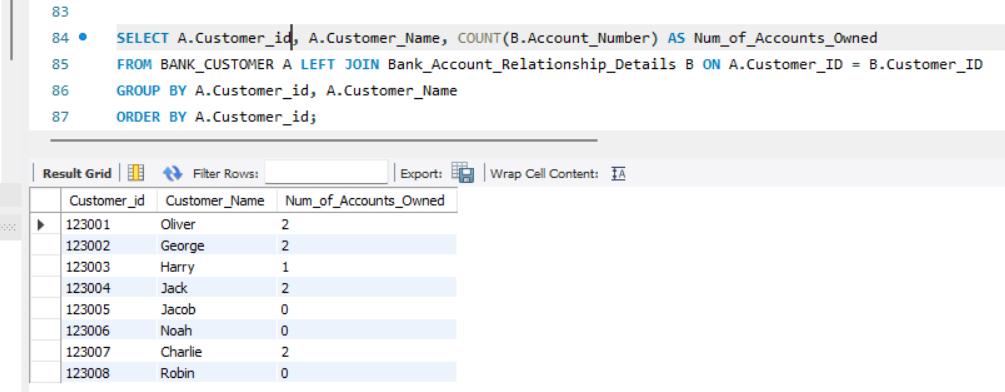
**CODE :**

**SELECT** A.Customer\_id, A.Customer\_Name, **COUNT**(B.Account\_Number) AS Num\_of\_Accounts\_Ownbed

**FROM** BANK\_CUSTOMER A **LEFT JOIN** Bank\_Account\_Relationship\_Details B **ON** A.Customer\_ID = B.Customer\_ID

**GROUP BY** A.Customer\_id, A.Customer\_Name

**ORDER BY** A.Customer\_id;

**OUTPUT :** 

2) calculate the number of account owned by the each customer having atleast two account

**CODE :**

**SELECT** A.Customer\_id, A.Customer\_Name, **COUNT**(B.Account\_Number**) AS** Num\_of\_Accounts\_Owned

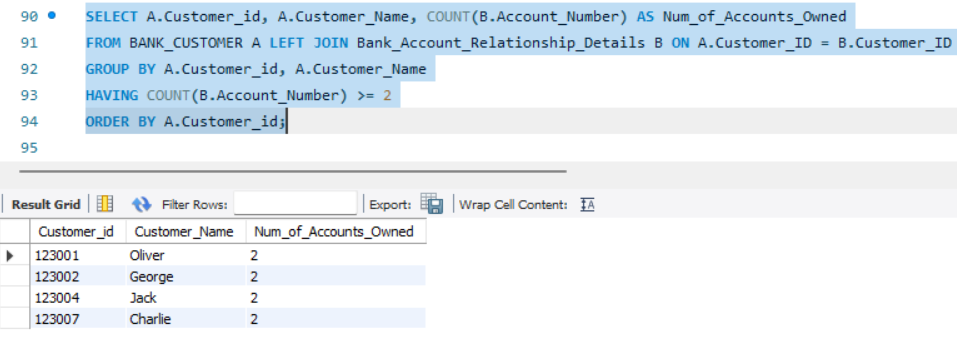
**FROM** BANK\_CUSTOMER A LEFT JOIN Bank\_Account\_Relationship\_Details B **ON** A.Customer\_ID = B.Customer\_ID

**GROUP BY** A.Customer\_id, A.Customer\_Name

**HAVING** COUNT(B.Account\_Number) >= 2

**ORDER BY** A.Customer\_id;

**OUTPUT :**



Question 3

1. list the customer name whose primary account type is "Credit Card"

**CODE :**

**SELECT**  A.Customer\_Name, **COUNT**(C.Account\_type) **AS**Customer\_with\_Credit\_card

**FROM**  BANK\_CUSTOMER **A**

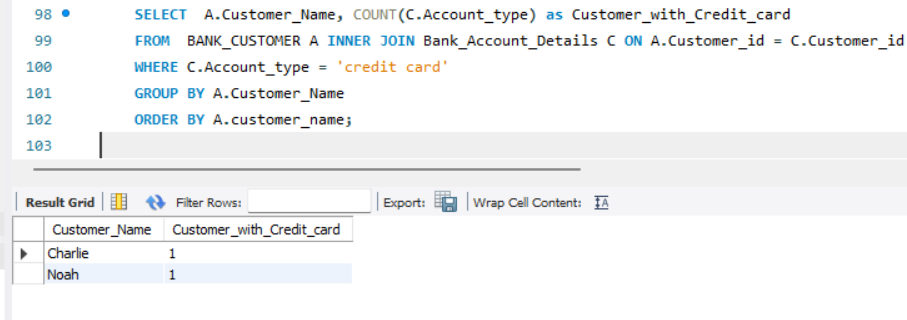
**INNER JOIN** Bank\_Account\_Details **C** ON A.Customer\_id = C.Customer\_id

**WHERE** C.Account\_type = 'credit card'

**GROUP BY** A.Customer\_Name

**ORDER BY** A.customer\_name;

**OUTPUT :**



1. list the customer name whose primary account type is "Credit Card" and order the result in descending manner

**CODE :**

**SELECT**  A.Customer\_Name, **COUNT**(C.Account\_type) **AS**Customer\_with\_Credit\_card

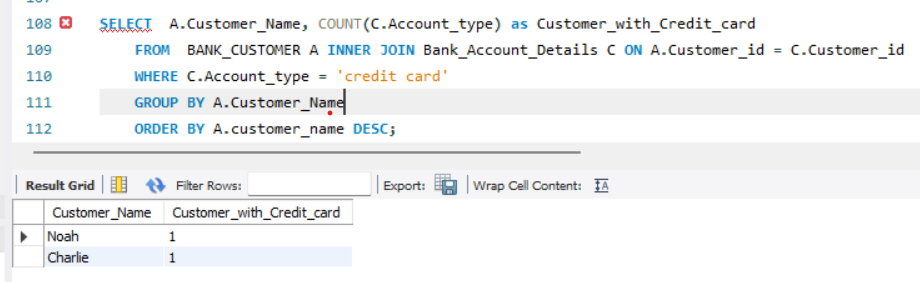
**FROM**  BANK\_CUSTOMER **A**

**INNER JOIN** Bank\_Account\_Details **C** ON A.Customer\_id = C.Customer\_id

**WHERE** C.Account\_type = 'credit card'

**GROUP BY** A.Customer\_Name

**ORDER BY** A.customer\_name **DESC;**

**OUTPUT :** 

Question 4

1). Query to report the details of customer(customer id, customer\_name, account number) having linking account number.

**CODE :**

**SELECT**

c.customer\_id, c.customer\_name,

r.Account\_Number AS linking\_account\_number

**FROM**

BANK\_CUSTOMER c

**JOIN**

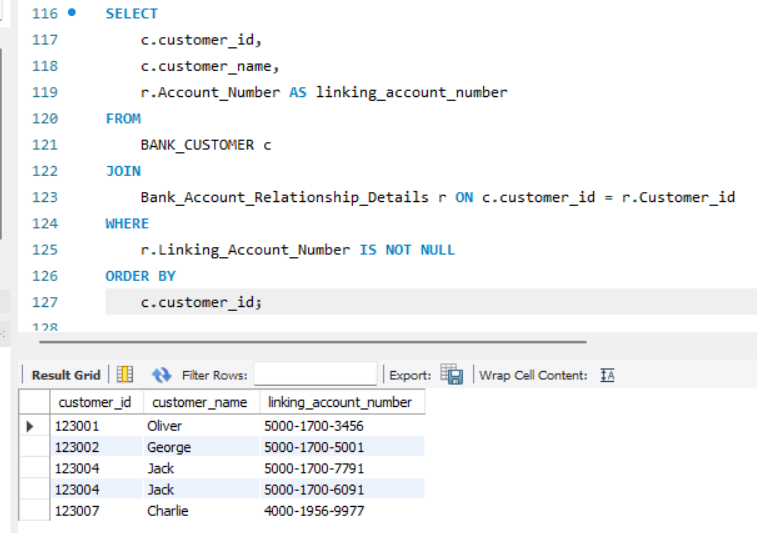
Bank\_Account\_Relationship\_Details r **ON** c.customer\_id = r.Customer\_id

**WHERE**

r.Linking\_Account\_Number IS NOT NULL

**ORDER BY**  c.customer\_id;

**OUTPUT :**



2). Query to report the details of all customer having linking account number or not.

**CODE :**

**SELECT**

c.customer\_id, c.customer\_name,

**CASE**

**WHEN** r.Linking\_Account\_Number IS NOT NULL **THE**N 'Yes'

**ELSE** 'No'

**END AS** having\_linking\_account

**FROM**  BANK\_CUSTOMER c

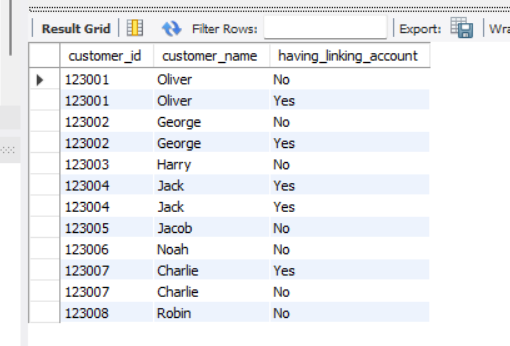
**LEFT JOIN**

Bank\_Account\_Relationship\_Details r **ON** c.customer\_id = r.Customer\_id

**ORDER BY**

c.customer\_id;

**OUTPUT :**



Question 5

Query to report the customer\_name, Address,Account\_Number, Account\_type,Balance\_amount,Linking\_Account\_Number from the above tables.

**CODE :**

**SELECT**

c.customer\_name,

c.Address,

d.Account\_Number,

d.Account\_type,

d.Balance\_amount,

r.Linking\_Account\_Number

**FROM**

BANK\_CUSTOMER c

**JOIN**

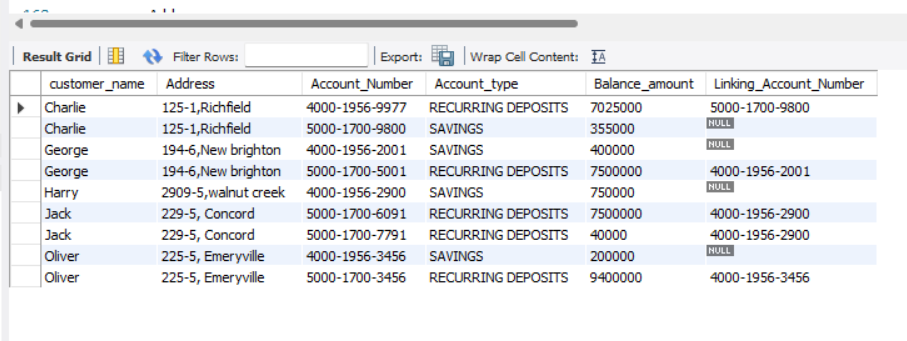
Bank\_Account\_Relationship\_Details r **ON** c.customer\_id = r.Customer\_id

**JOIN**

Bank\_Account\_Details d **ON** r.Account\_Number = d.Account\_Number

**ORDER BY**

c.customer\_name, d.Account\_Number;

**OUTPUT :** 

Question 6

Query to report all the Bank Customer and their respective account number,Account type,Balance amount,Account status and Relationship type.

**CODE :**

**SELECT**

c.customer\_name,

c.customer\_id,

d.Account\_Number,

d.Account\_type,

r.Balance\_amount,

r.Account\_status,

r.Relationship\_type

**FROM**

BANK\_CUSTOMER c

**LEFT JOIN**

Bank\_Account\_Relationship\_Details d **ON** c.customer\_id = d.Customer\_id

**LEFT JOIN**

Bank\_Account\_Details r ON d.customer\_id = r.Customer\_id

**ORDER BY**

c.customer\_name, d.Account\_Number;

Question 7

1) create a table called employees with following below data

EmpID Name Manager ID

1 Mark 3

2 Natasha 4

3 Chris 2

4 Robert Null

5 Steve 2

**CODE :**

|  |
| --- |
| **CREATE TABLE** Employees ( EMP\_ID int, EMP\_NAME VARCHAR(50), Manager\_ID int); |
| **INSERT into** Employees **VALUES** |
| (1,'Mark',3), |
| (2,'Natasha', 4), |
| (3 , 'Chris', 2), |
| (4 , 'Robert' , Null),  (5, 'Steve' , 2); |

2) query to get the all the employee id, employee name and their manager name and manager id

**CODE :**

SELECT

e1.Emp\_ID AS Employee\_ID,

e1.EMP\_NAME AS Employee\_Name,

e2.Emp\_ID AS Manager\_ID,

e2.EMP\_NAME AS Manager\_Name

FROM

Employees e1

LEFT JOIN

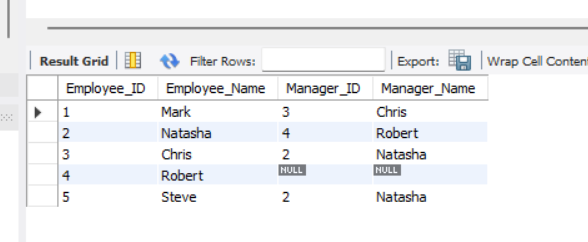
Employees e2 ON e1.Manager\_ID = e2.Emp\_ID;

## Another code using INNER JOIN

**SELECT** e1.EMP\_ID ,e1.EMP\_NAME,e1.Manager\_ID , e2.EMP\_NAME

**FROM** Employees e1 INNER join Employees e2 **ON** e1.Manager\_ID = e2.EMP\_ID;

**OUTPUT :**



**Question 8**

Write a difference between the inner join, left join, right join, self join and cross join

1. **Inner Join:**
   * An inner join returns only the rows where there is a match between the columns being joined from both tables.
   * Rows from either table that do not have a corresponding match in the other table are excluded from the result.
   * Inner joins are used to retrieve only the matching records between two tables.
2. **Left Join (Left Outer Join):**
   * A left join returns all the rows from the left (first) table and the matching rows from the right (second) table.
   * If there are no matches in the right table, NULL values are returned for columns from the right table.
   * Left joins are used to retrieve all records from the left table and the matching records from the right table.
3. **Right Join (Right Outer Join):**
   * A right join returns all the rows from the right (second) table and the matching rows from the left (first) table.
   * If there are no matches in the left table, NULL values are returned for columns from the left table.
   * Right joins are less common and are essentially the reverse of left joins.
4. **Self Join:**
   * A self join is a type of inner or outer join where a table is joined with itself.
   * It is useful when you have a table with a hierarchical or recursive structure, such as an employee table with a manager-to-employee relationship.
   * In a self join, different aliases are used to reference the same table in the join condition.
5. **Cross Join (Cartesian Join):**
   * A cross join returns the Cartesian product of two tables, meaning it combines each row from the first table with every row from the second table.
   * It doesn't require a specific condition for the join; it simply combines all possible combinations.
   * Cross joins can result in a large number of rows and are used sparingly.

Question 9

Explain full outer join and how to achieve in MySQL

A full outer join returns all the rows in both the left and right joins, including unmatched rows from both tables. Rows from either table that do not have a corresponding match in the other table are included in the result with NULL values for columns from the table that lacks a match.

In MySQL, there is no direct SQL syntax for a full outer join & we can achieve a full outer join effect by using a combination of **LEFT JOIN** and **RIGHT JOIN**, and then combining the results using the **UNION** operator.

**SELECT** \*

**FROM** table1

**LEFT JOIN** table2 ON table1.common\_column = table2.common\_column

**UNION**

SELECT \*

**FROM** table1

**RIGHT JOIN** table2 ON table1.common\_column = table2.common\_column;

Question 10

Query to report the customer\_name, Address,Account\_Number, Account\_type,Balance\_amount,Linking\_Account\_Number from the above tables whose customer name is Oliver,Harry.

**CODE :**

**SELECT DISTINCT** A.customer\_name,

A.Address,

B.Account\_Number,

B.Account\_type,

B.Linking\_Account\_Number,

C.Balance\_amount

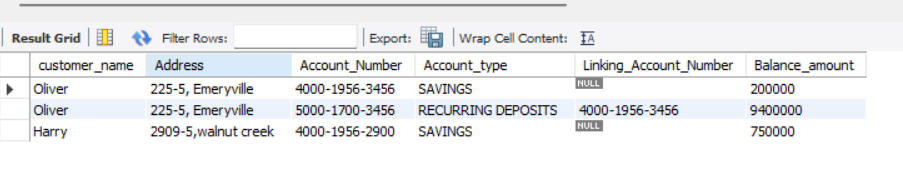
**FROM** BANK\_CUSTOMER A

**JOIN** Bank\_Account\_Relationship\_Details B **ON** A.customer\_id = B.Customer\_Id

**JOIN** Bank\_Account\_Details C **ON** B.Account\_Number = C.Account\_Number

**WHERE** A.customer\_name **IN** ('Oliver' , 'Harry');

**OUTPUT :**



Question 11

Query to report the average balance amount of the each customer in their account.

**CODE :**

**SELECT**

C.customer\_name,

C.customer\_id,

AVG(D.Balance\_amount) **AS** Average\_balance

**FROM**

BANK\_CUSTOMER C

**LEFT JOIN**

Bank\_Account\_Details D **ON** C.customer\_ID = D.Customer\_ID

**GROUP BY**

C.customer\_name, C.customer\_id

**ORDER BY**

C.customer\_name;

**OUTPUT :**

