**Tasks 1: Database Design:**

**1. Create the database named "HMBank"**

**A.** create database HMBank;

**2. Define the schema for the Customers, Accounts, and Transactions tables based on the provided schema.**

**A. 1. Creating Customers table:**

create table Customers (

customer\_id INT PRIMARY KEY,

customer\_name VARCHAR(50) NOT NULL,

address VARCHAR(100),

contact\_number VARCHAR(15)

);

**2. Creating Accounts table:**

create table Accounts (

account\_id INT PRIMARY KEY,

customer\_id INT,

account\_type VARCHAR(20) NOT NULL,

balance DECIMAL(10, 2) DEFAULT 0,

FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)

);

**3. Creating Transactions table:**

create table Transactions (

transaction\_id INT PRIMARY KEY,

account\_id INT,

transaction\_type VARCHAR(20) NOT NULL,

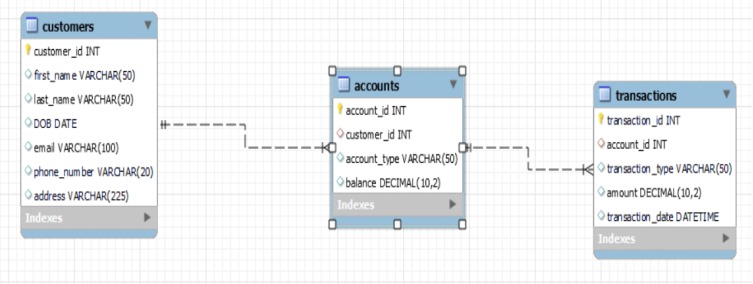
amount DECIMAL(10, 2) NOT NULL,

transaction\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

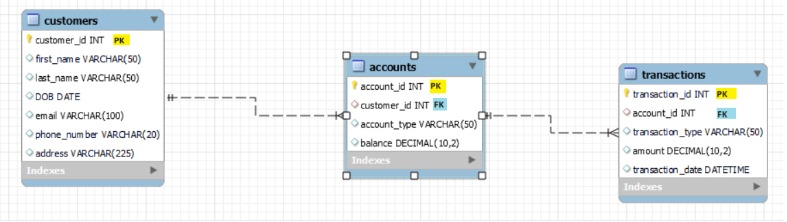
FOREIGN KEY (account\_id) REFERENCES Accounts(account\_id)

);

**3. Create an ERD (Entity Relationship Diagram) for the database.**

**A.**

**4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.**

**A.**

**5. Write SQL scripts to create the mentioned tables with appropriate data types, constraints,**

**and relationships.**

**• Customers**

**• Accounts**

**• Transactions**

**A.**

**1. Customers table:**

create table Customers (

customer\_id INT PRIMARY KEY,

customer\_name VARCHAR(50) NOT NULL,

address VARCHAR(100),

contact\_number VARCHAR(15)

);

**2. Accounts table:**

create table Accounts (

account\_id INT PRIMARY KEY,

customer\_id INT,

account\_type VARCHAR(20) NOT NULL,

balance DECIMAL(10, 2) DEFAULT 0,

FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)

);

**3. Transactions table:**

create table Transactions (

transaction\_id INT PRIMARY KEY,

account\_id INT,

transaction\_type VARCHAR(20) NOT NULL,

amount DECIMAL(10, 2) NOT NULL,

transaction\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (account\_id) REFERENCES Accounts(account\_id)

);

**Tasks 2: Select, Where, Between, AND, LIKE:**

**1. Insert at least 10 sample records into each of the following tables.**

**• Customers**

**• Accounts**

**• Transactions**

**A. 1. Inserting into Customers Table:**

insert into Customers (customer\_id, first\_name,last\_name,DOB, email, phone\_number,address) VALUES

(1, ‘Goutham’,’Kakarla’,’2001-10-31’,’gautham@gmail.com’,’99999999’,’Kovvur’),

(2, ‘Preetham’,’Kakarla’,’2003-11-23’,’preetham@gmail.com’,’999999998’,’Kovvur’),

(3, ‘Indhu’,’Jalem’,’1998-10-18’,’indhu@gmail.com’,’9999999997’,’Nidadavole’),

(4, ‘Chandu’,’Jalem’,’1997-07-16’,’chandhu@gmail.com’,’9999999996’,’Nidadavole’),

(5, ‘Chanti’,’Barthi’, ‘2000-09-12’,’chanti@gmail.com’,’9999999995’,’Chennai’),

(6, ‘Manju’,’Barthi’,’2002-09-04’,’manju@gmail.com’,’9999999994’, ‘Eluru’),

(7, ‘Dinku’,’Barthi’,’2004-11-05’,’dinku@gmail.com’,’9999999993’, ‘Hyderabad’),

(8, ‘Sonu’,’Bontha’,’2002-08-01’,’sonu@gmail.com’,’9999999992’, ‘Machilipatnam’),

(9, ‘Minnu’,’Barthi’ ,’2005-04-21’,’minnu@gmail.com’,’9999999991’, ‘Gopalapuram’),

(10, ‘Binnu’,’Barthi’ ,’2006-05-13’,’binnu@gmail.com’,’9999999990’, ‘Gopalapuram’)

;

**2. Inserting into Accounts Table:**

**A.** insert into Accounts (account\_id, customer\_id, account\_type, balance)VALUES

(101, 1, ‘Savings’, 5000.00),

(102, 2, ‘Current’, 1000.00),

(103, 3, ‘Savings’, 3000.00),

(104, 4, ‘Current’, 2000.00),

(105, 5, ‘Savings’, 7000.00),

(106, 6, ‘zero\_balance’, 0.00),

(107, 7, ‘Savings’, 4500.00),

(108, 8, ‘Current’, 3000.00),

(109, 9, ‘Savings’, 6000.00),

(110, 10, ‘zero\_balance’, 0.00),

**3. Inserting into Transactions Table:**

**A.**  insert into Transactions (transaction\_id, account\_id, transaction\_type, amount, transaction\_date)

VALUES

(1001, 101, ‘Deposit’, 1000.00, ‘2024-01-13 12:30:00’),

(1002, 101, ‘Withdrawal’, 500.00, ‘2024-01-14 10:45:00’),

(1003, 102, ‘Deposit’, 2000.00, ‘2024-01-15 15:20:00’),

(1004, 102, ‘Withdrawal’, 1000.00, ‘2024-01-16 09:10:00’),

(1005, 103, ‘Deposit’, 1500.00, ‘2024-01-17 14:45:00’),

(1006, 104, ‘Withdrawal’, 800.00, ‘2024-01-18 11:30:00’),

(1007, 105, ‘Deposit’, 3000.00, ‘2024-01-19 13:15:00’),

(1008, 106, ‘Withdrawal’, 1200.00, ‘2024-01-20 16:40:00’),

(1009, 107, ‘Deposit’, 2500.00, ‘2024-01-21 08:55:00’),

(1010, 108, ‘Withdrawal’, 900.00, ‘2024-01-22 10:25:00’);

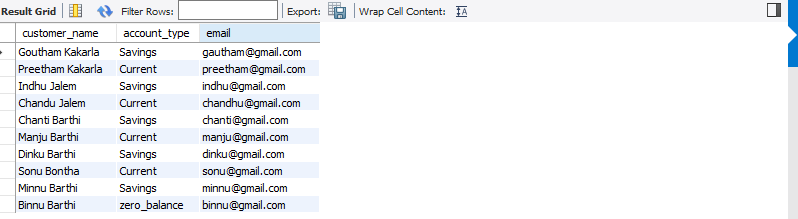
**2. Write SQL queries for the following tasks:**

**1. Write a SQL query to retrieve the name, account type and email of all customers.**

**A.** select concat(C.first\_name, ‘ ‘, C.last\_name) as customer\_name, A.account\_type, C.email

from Customers C

join Accounts A on C.customer\_id = A.customer\_id;



**2. Write a SQL query to list all transaction corresponding customer.**

**A.** SELECT

T.transaction\_id,

T.account\_id,

T.transaction\_type,

T.amount,

T.transaction\_date,

C.customer\_id,

CONCAT(C.first\_name, ‘ ‘, C.last\_name) AS customer\_name,

C.email

FROM

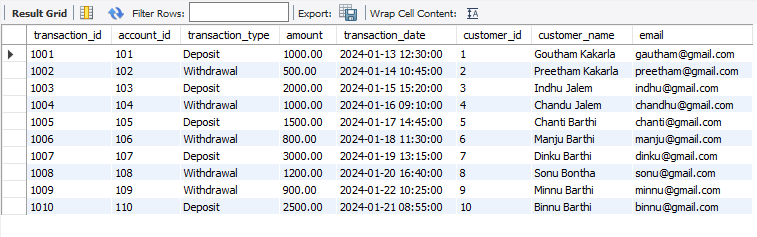
Transactions T

JOIN

Accounts A ON T.account\_id = A.account\_id

JOIN

Customers C ON A.customer\_id = C.customer\_id;



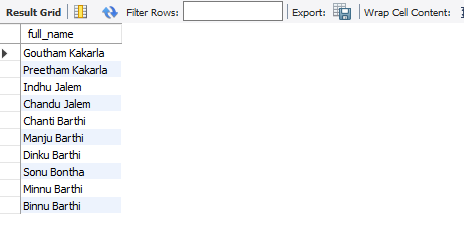
**3. Write a SQL query to increase the balance of a specific account by a certain amount.**

**A.** UPDATE Accounts

SET balance = balance + 10000.00

WHERE account\_id = 101;

**4. Write a SQL query to Combine first and last names of customers as a full\_name.**

**A.**  SELECT CONCAT(first\_name, ‘ ‘, last\_name) AS full\_name FROM Customers;

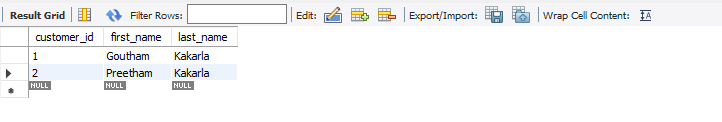
**5. Write a SQL query to remove accounts with a balance of zero where the account**

**type is savings.**

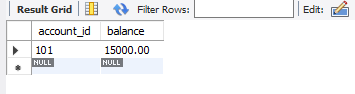
**A.**  DELETE FROM Accounts

WHERE balance = 0 AND account\_type = ‘Savings’;

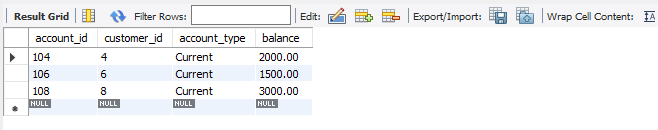
**6. Write a SQL query to Find customers living in a specific city.**

**A.**  SELECT customer\_id, first\_name,last\_name FROM Customers WHERE address = ‘Kovvur’;

**7. Write a SQL query to Get the account balance for a specific account.**

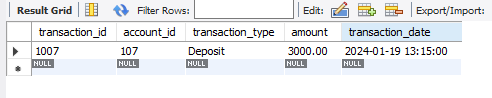
**A.**  SELECT account\_id, balance FROM Accounts WHERE account\_id = 101;

**8. Write a SQL query to List all current accounts with a balance greater than $1,000.**

**A.** SELECT account\_id, customer\_id, account\_type, balance FROM Accounts WHERE account\_type = ‘Current’ AND balance > 1000.00;

**9. Write a SQL query to Retrieve all transactions for a specific account.**

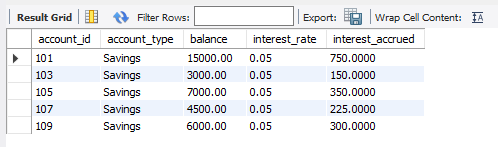
**A.**  SELECT transaction\_id, account\_id, transaction\_type, amount, transaction\_date

 FROM transactions WHERE account\_id = 107;

**10. Write a SQL query to Calculate the interest accrued on savings accounts based on a**

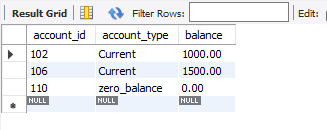
**given interest rate.**

A. SELECT account\_id, account\_type. balance, 0.05 AS interest\_rate, balance \* 0.05

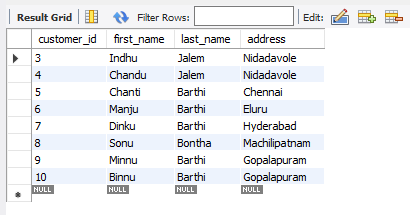
AS interest\_accrued FROM Accounts WHERE account\_type = ‘Savings’;

**11. Write a SQL query to Identify accounts where the balance is less than a specified**

**overdraft limit.**

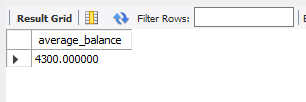
A. SELECT account\_id, account\_type, balance FROM Accounts WHERE balance < 2000;

**12. Write a SQL query to Find customers not living in a specific city.**

**A.**  SELECT customer\_id, first\_name, last\_name, address, FROM Customers WHERE city != ‘Kovvur’;

**Tasks 3: Aggregate functions, Having, Order By, GroupBy and Joins:**

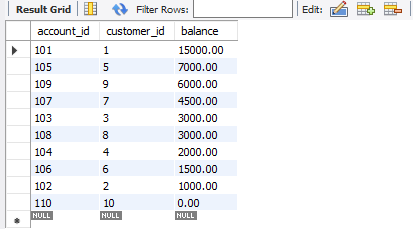
**1. Write a SQL query to Find the average account balance for all customers.**

**A.**  SELECT AVG(balance) AS average\_balance FROM Accounts;

**2. Write a SQL query to Retrieve the top 10 highest account balances.**

**A.**  SELECT account\_id, customer\_id,balance

FROM Accounts ORDER BY balance DESC LIMIT 10;

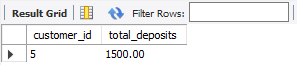


**3. Write a SQL query to Calculate Total Deposits for All Customers in specific date.**

**A.**  SELECT A.customer\_id, SUM(T.amount) AS total\_deposits FROM Transactions T

JOIN

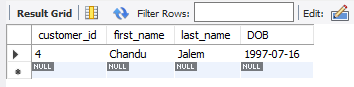
Accounts A ON T.account\_id = A.account\_id WHERE T.transaction\_type = ‘deposit’

 AND DATE(T.transaction\_date) = ‘2024-01-17’ GROUP BY A.customer\_id;

**4. Write a SQL query to Find the Oldest and Newest Customers.**

**A.** **Oldest Customer:**

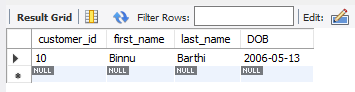
SELECT customer\_id, first\_name, last\_name,DOB FROM Customers

ORDER BY DOB ASC LIMIT 1;

**Newest Customer:**

SELECT customer\_id,first\_name, last\_name, DOB FROM Customers

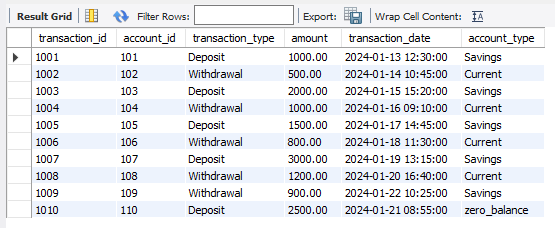
ORDER BY DOB DESC LIMIT 1;



**5. Write a SQL query to Retrieve transaction details along with the account type.**

**A.**  SELECT T.transaction\_id, T.account\_id, T.transaction\_type,T.amount, T.transaction\_date,

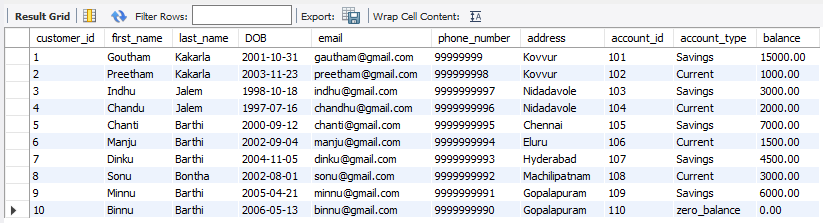
A.account\_type FROM Transactions T

 JOIN Accounts A ON T.account\_id = A.account\_id;

**6. Write a SQL query to Get a list of customers along with their account details.**

**A.** SELECT C.customer\_id, C.first\_name, C.last\_name, C.DOB, C.email, C.phone\_number, C.address,

A.account\_id, A.account\_type, A.balance FROM Customers C

 JOIN Accounts A ON C.customer\_id = A.customer\_id;

**7. Write a SQL query to Retrieve transaction details along with customer information for a**

**specific account.**

**A.** SELECT T.transaction\_id, T.account\_id, T.transaction\_type, T.amount, T.transaction\_date

C.customer\_id, C.first\_name, C.last\_name, C.DOB, C.email, C.phone\_number, C.address

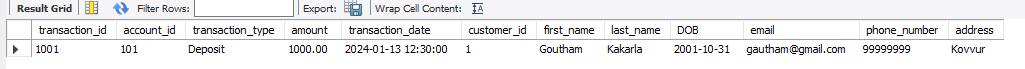
FROM Transactions T

JOIN

Accounts A ON T.account\_id = A.account\_id

JOIN

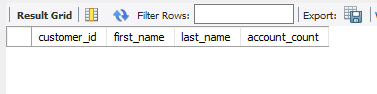
Customers C ON A.customer\_id = C.customer\_id

WHERE T.account\_id = 101;

**8. Write a SQL query to Identify customers who have more than one account.**

**A.** SELECT C.customer\_id, C.first\_name, C.last\_name, COUNT(A.account\_id) AS account\_count FROM

Customers C JOIN Accounts A ON C.customer\_id = A.customer\_id GROUP BY C.customer\_id

 HAVING COUNT(A.account\_id) > 1;

Since there are no multiple accounts for any customer, no data is returned.

**9. Write a SQL query to Calculate the difference in transaction amounts between deposits and**

**withdrawals.**

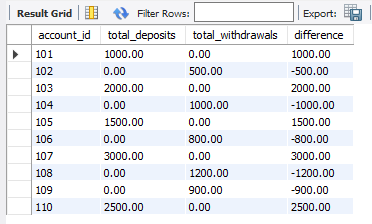
**A.** SELECT account\_id,

SUM(CASE WHEN transaction\_type = ‘deposit’ THEN amount ELSE 0 END) AS total\_deposits,

SUM(CASE WHEN transaction\_type = ‘withdrawal’ THEN amount ELSE 0 END) AS

total\_withdrawals,

SUM(CASE WHEN transaction\_type = ‘deposit’ THEN amount ELSE -amount END) AS difference

 FROM Transactions GROUP BY account\_id;

**10. Write a SQL query to Calculate the average daily balance for each account over a specified**

**period.**

**A.** SELECT

account\_id,

AVG(balance) AS average\_daily\_balance

FROM (

SELECT

A.account\_id,

T.transaction\_date,

SUM(CASE WHEN T.transaction\_type = ‘deposit’ THEN amount

WHEN T.transaction\_type = ‘withdrawal’ THEN -amount

ELSE 0 END) AS balance

FROM

Accounts A

JOIN

Transactions T ON A.account\_id = T.account\_id

WHERE

T.transaction\_date BETWEEN ‘2024-01-17’ AND ‘2024-01-22’

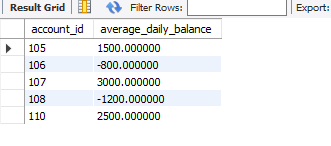
GROUP BY

A.account\_id,

T.transaction\_date

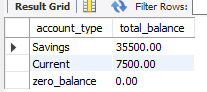
) AS DailyBalances

GROUP BY

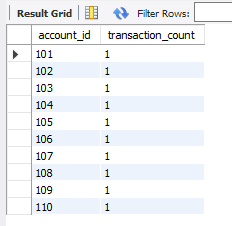
 account\_id;

**11. Calculate the total balance for each account type.**

**A.** SELECT account\_type, SUM(balance) AS total\_balance FROM Accounts

 GROUP BY account\_type;

**12. Identify accounts with the highest number of transactions order by descending order.**

**A.** SELECT account\_id, COUNT(transaction\_id) AS transaction\_count FROM Transactions GROUP BY account\_id ORDER BY transaction\_count DESC;

**13. List customers with high aggregate account balances, along with their account types.**

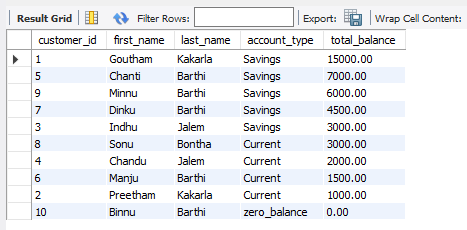
**A.** SELECT C.customer\_id, C.first\_name, C.last\_name, A.account\_type,

SUM(A.balance) AS total\_balance FROM Customers C

JOIN

Accounts A ON C.customer\_id = A.customer\_id GROUP BY

C.customer\_id, C.first\_name, C.last\_name, C.DOB, C.email, C.phone\_number, C.address,

 A.account\_type ORDER BY total\_balance DESC;

**14. Identify and list duplicate transactions based on transaction amount, date, and account.**

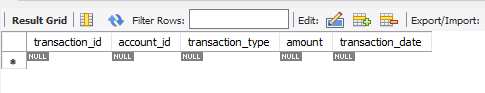
**A.**  SELECT transaction\_id, account\_id, transaction\_type, amount, transaction\_date FROM

Transactions WHERE (amount, transaction\_date, account\_id) IN (

SELECT amount, transaction\_date, account\_id FROM Transactions

GROUP BY amount, transaction\_date, account\_id HAVING COUNT(\*) > 1

)

ORDER BY amount, transaction\_date, account\_id;

There are no duplicate transactions, therefore no data is returned.

**Tasks 4: Subquery and its type:**

**1.Retrieve the customer(s) with the highest account balance.**

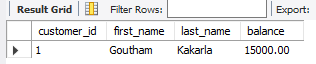
**A.** WITH RankedCustomers AS (

SELECT C.customer\_id, C.first\_name, C.last\_name, A.balance, RANK() OVER (ORDER BY

A.balance DESC) AS balance\_rank FROM Customers C

JOIN Accounts A ON C.customer\_id = A.customer\_id )

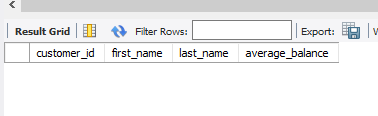
SELECT customer\_id, first\_name, last\_name, balance FROM RankedCustomers

 WHERE balance\_rank = 1;

**2. Calculate the average account balance for customers who have more than one account.**

**A**. SELECT C.customer\_id, C.first\_name, C.last\_name, AVG(A.balance) AS average\_balance FROM

Customers C JOIN Accounts A ON C.customer\_id = A.customer\_id GROUP BY C.customer\_id,

 C.first\_name, C.last\_name HAVING COUNT(A.account\_id) > 1;

**3. Retrieve accounts with transactions whose amounts exceed the average transaction amount.**

**A.** WITH TransactionAverages AS (

SELECT AVG(amount) AS average\_transaction\_amount FROM Transactions)

SELECT A.account\_id, A.customer\_id, A.account\_type, T.transaction\_id, T.transaction\_type,

T.amount, T.transaction\_date FROM Accounts A

JOIN

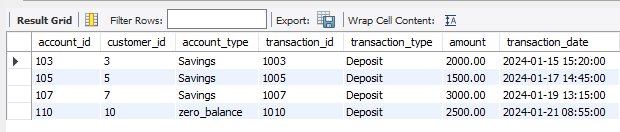
Transactions T ON A.account\_id = T.account\_id

CROSS JOIN

TransactionAverages

WHERE

T.amount > TransactionAverages.average\_transaction\_amount;



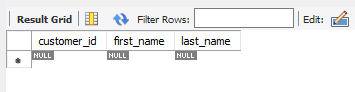
**4. Identify customers who have no recorded transactions.**

**A.**  SELECT C.customer\_id, C.first\_name, C.last\_name FROM Customers C WHERE

C.customer\_id NOT IN (

SELECT DISTINCT customer\_id

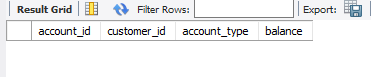
FROM Transactions

 );

**5. Calculate the total balance of accounts with no recorded transactions.**

**A.** SELECT A.account\_id, A.customer\_id, A.account\_type, A.balance FROM Accounts A

LEFT JOIN

 Transactions T ON A.account\_id = T.account\_id WHERE T.account\_id IS NULL;

**6. Retrieve transactions for accounts with the lowest balance.**

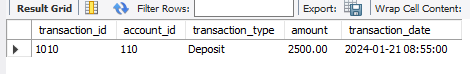
**A.** SELECT T.transaction\_id, T.account\_id, T.transaction\_type, T.amount,

T.transaction\_date FROM Transactions T

JOIN (

SELECT account\_id, RANK() OVER (ORDER BY balance) AS balance\_rank FROM

Accounts) AS RankedAccounts ON T.account\_id = RankedAccounts.account\_id

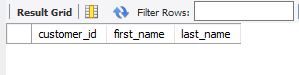
 WHERE balance\_rank = 1;

**7. Identify customers who have accounts of multiple types.**

**A.** SELECT C.customer\_id, C.first\_name, C.last\_name FROM Customers C

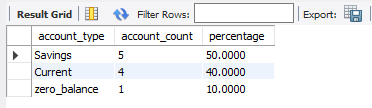
JOIN

Accounts A ON C.customer\_id = A.customer\_id GROUP BY C.customer\_id HAVING

 COUNT(DISTINCT A.account\_type) > 1;

**8. Calculate the percentage of each account type out of the total number of accounts.**

**A.** SELECT account\_type, COUNT(\*) AS account\_count, (COUNT(\*) / (SELECT COUNT(\*) FROM

 Accounts)) \* 100 AS percentage FROM Accounts GROUP BY account\_type;

**9. Retrieve all transactions for a customer with a given customer\_id.**

**A.**  SELECT T.transaction\_id, T.account\_id, T.transaction\_type, T.amount,

T.transaction\_date FROM Transactions T

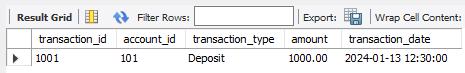
JOIN

Accounts A ON T.account\_id = A.account\_id

JOIN

Customers C ON A.customer\_id = C.customer\_id

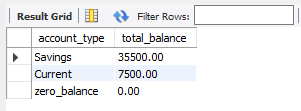
WHERE C.customer\_id = 1;



**10. Calculate the total balance for each account type, including a subquery within the SELECT clause.**

**A.** SELECT account\_type, (SELECT SUM(balance) FROM Accounts A WHERE

A.account\_type = Accounts.account\_type) AS total\_balance FROM Accounts GROUP BY

 account\_type;