# SQL ASSIGNMENT

# Banking System

**Tasks 1: Database Design:**

1.Create the database named "HMBank"

CREATE DATABASE HMBank;

USE HMBank;

**#OUTPUT**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 86 | 15:34:23 | CREATE DATABASE HMBank | 1 row(s) affected | 0.000 sec |
| 3 | 87 | 15:34:46 | USE HMBank | 0 row(s) affected | 1. c |

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

**Customers Table:**

customer\_id: Integer, Primary Key

first\_name: String (VARCHAR(50))

last\_name: String (VARCHAR(50))

DOB: Date

email: String (VARCHAR(100))

phone\_number: String (VARCHAR(15))

address: String (VARCHAR(255))

**Accounts Table:**

account\_id: Integer, Primary Key

customer\_id: Integer, Foreign Key referencing customer\_id in the Customers table

account\_type: String (VARCHAR(50))

balance: Decimal (DECIMAL(10, 2))

**Transactions Table:**

transaction\_id: Integer, Primary Key

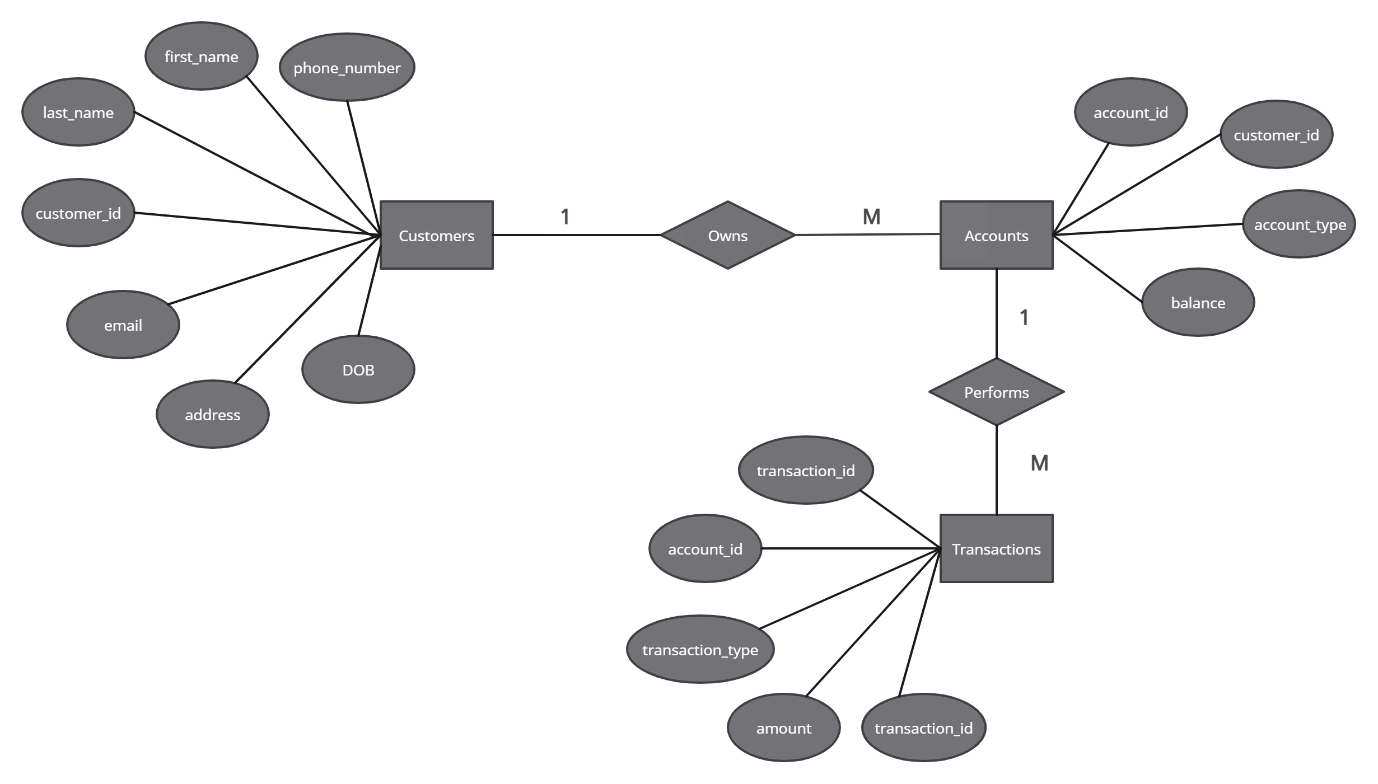
account\_id: Integer, Foreign Key referencing account\_id in the Accounts table

transaction\_type: String (VARCHAR(50))

amount: Decimal (DECIMAL(10, 2))

transaction\_date: Date

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



5.Create appropriate Primary Key and Foreign Key constraints for referential integrity.

**Customers Table:**

customer\_id is the Primary Key.

**Accounts Table:**

account\_id is the Primary Key.

customer\_id is a Foreign Key that references customer\_id in the Customers table.

**Transactions Table**:

transaction\_id is the Primary Key.

account\_id is a Foreign Key that references account\_id in the Accounts table.

6.Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.

• Customers

• Accounts

• Transactions

CREATE TABLE Customers (

customer\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

DOB DATE,

email VARCHAR(100),

phone\_number VARCHAR(15),

address VARCHAR(255)

);

CREATE TABLE Accounts (

account\_id INT PRIMARY KEY,

customer\_id INT,

account\_type VARCHAR(50),

balance DECIMAL(10, 2),

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

);

CREATE TABLE Transactions (

transaction\_id INT PRIMARY KEY,

account\_id INT,

transaction\_type VARCHAR(50),

amount DECIMAL(10, 2),

transaction\_date DATE,

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

INSERT INTO Customers (customer\_id, first\_name, last\_name, DOB, email, phone\_number, address)

VALUES

(1, 'Liji', 'Lawrence', '2001-01-01', 'lijilaw@gmail.com', '9134567890', '123 Main St'),

(2, 'Jany', 'Dim', '2002-02-02', 'janydim@gmail.com', '9887654321', '456 Maple Ave'),

(3, 'Agnel', 'James', '1990-03-03', 'agneljames@gmail.com', '9862223333', '789 Oak Dr'),

(4, 'Bergi', 'Johnson', '1995-04-04', 'bergi@gmail.com', '9823334444', '321 Pine Ln'),

(5, 'Reshma', 'Gigi', '2000-05-05', 'reshgigi@gmail.com', '9734445555', '654 Willow Rd'),

(6, 'David', 'Davis', '1999-06-06', 'david.davis@gmail.com', '9345556666', '987 Cedar St'),

(7, 'Evangeline', 'Evans', '2001-07-07', 'evanevans@gmail.com', '9056667777', '345 Birch Ave'),

(8, 'Frank', 'Franklin', '1998-08-08', 'frankfranklin@gmail.com', '9667778888', '678 Spruce Dr'),

(9, 'Grace', 'Griffin', '1997-09-09', 'gracegriffin@gmail.com', '9778889999', '912 Elm Ln'),

(10, 'Harsi', 'Harrison', '2002-10-10', 'harryharrison@gmail.com', '9889990000', '234 Poplar Rd');

INSERT INTO Accounts (account\_id, customer\_id, account\_type, balance)

VALUES

(11, 1, 'Current', 1200.00),

(21, 2, 'Savings', 5000.00),

(31, 3, 'Savings', 31000.00),

(41, 4, 'Savings', 21000.00),

(51, 5, 'Current', 50000.00),

(61, 6, 'Savings', 65000.00),

(71, 7, 'Savings', 6000.00),

(81, 8, 'Savings', 18000.00),

(91, 9, 'Current', 9000.00),

(12, 10, 'Savings', 70000.00);

INSERT INTO Transactions (transaction\_id, account\_id, transaction\_type, amount, transaction\_date)

VALUES

(101, 11, 'Deposit', 500.00, '2023-10-05'),

(201, 21, 'Deposit', 2000.00, '2023-06-02'),

(301, 31, 'Deposit', 3000.00, '2023-07-03'),

(401, 41, 'Withdrawal', 4000.00, '2023-07-11'),

(501, 51, 'Deposit', 10000.00, '2023-07-05'),

(601, 61, 'Deposit', 6000.00, '2023-08-02'),

(701, 71, 'Deposit', 5000.00, '2023-08-15'),

(801, 81, 'Withdrawal', 8000.00, '2023-09-20'),

(901, 91, 'Deposit', 5000.00, '2023-10-23'),

(111, 12, 'Withdrawal', 30000.00, '2023-11-14');

2.Write SQL queries for the following tasks:

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

SELECT first\_name, last\_name,

(SELECT account\_type FROM Accounts WHERE Customers.customer\_id = Accounts.customer\_id) AS account\_type, email FROM Customers;

**#OUTPUT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **first\_name** | **last\_name** | **account\_type** | **email** |
|  | Liji | Lawrence | Current | lijilaw@gmail.com |
|  | Jany | Dim | Savings | janydim@gmail.com |
|  | Agnel | James | Savings | agneljames@gmail.com |
|  | Bergi | Johnson | Savings | bergi@gmail.com |
|  | Reshma | Gigi | Current | reshgigi@gmail.com |
|  | David | Davis | Savings | david.davis@gmail.com |
|  | Evangeline | Evans | Savings | evanevans@gmail.com |
|  | Frank | Franklin | Savings | frankfranklin@gmail.com |
|  | Grace | Griffin | Current | gracegriffin@gmail.com |
|  | Harsi | Harrison | Savings | harryharrison@gmail.com |

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

SELECT transaction\_id, transaction\_type, amount, transaction\_date,

(SELECT first\_name FROM Customers WHERE customer\_id = (SELECT customer\_id FROM Accounts WHERE account\_id = Transactions.account\_id)) AS first\_name,

(SELECT last\_name FROM Customers WHERE customer\_id = (SELECT customer\_id FROM Accounts WHERE account\_id = Transactions.account\_id)) AS last\_name

FROM Transactions;

**#OUTPUT**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **transaction\_id** | **transaction\_type** | **amount** | **transaction\_date** | **first\_name** | **last\_name** |
|  | 101 | Deposit | 500.00 | 2023-10-05 | Liji | Lawrence |
|  | 111 | Withdrawal | 30000.00 | 2023-11-14 | Harsi | Harrison |
|  | 201 | Deposit | 2000.00 | 2023-06-02 | Jany | Dim |
|  | 301 | Deposit | 3000.00 | 2023-07-03 | Agnel | James |
|  | 401 | Withdrawal | 4000.00 | 2023-07-11 | Bergi | Johnson |
|  | 501 | Deposit | 10000.00 | 2023-07-05 | Reshma | Gigi |
|  | 601 | Deposit | 6000.00 | 2023-08-02 | David | Davis |
|  | 701 | Deposit | 5000.00 | 2023-08-15 | Evangeline | Evans |
|  | 801 | Withdrawal | 8000.00 | 2023-09-20 | Frank | Franklin |
|  | 901 | Deposit | 5000.00 | 2023-10-23 | Grace | Griffin |
|  |  |  |  |  |  |  |

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

UPDATE Accounts

SET balance = balance + 12000

WHERE account\_id = 31;

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

SELECT CONCAT(first\_name, ' ', last\_name) AS full\_name

FROM Customers;

**#OUTPUT**

|  |  |
| --- | --- |
|  | **full\_name** |
|  | Liji Lawrence |
|  | Jany Dim |
|  | Agnel James |
|  | Bergi Johnson |
|  | Reshma Gigi |
|  | David Davis |
|  | Evangeline Evans |
|  | Frank Franklin |
|  | Grace Griffin |
|  | Harsi Harrison |
|  |  |

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

DELETE FROM Accounts

WHERE balance = 0 AND account\_type = 'Savings';

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

SELECT \* FROM Customers WHERE address = '654 Willow Rd';

**#OUTPUT**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **customer\_id** | **first\_name** | **last\_name** | **DOB** | **email** | **phone\_number** | **address** |
|  | 5 | Reshma | Gigi | 2000-05-05 | reshgigi@gmail.com | 9734445555 | 654 Willow Rd |
|  |  |  |  |  |  |  |  |

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

SELECT account\_id,balance FROM Accounts WHERE account\_id = 51;

**#OUTPUT**

|  |  |  |
| --- | --- | --- |
|  | **account\_id** | **balance** |
|  | 51 | 50000.00 |
|  |  |  |

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

SELECT \* FROM Accounts WHERE account\_type = 'Current' AND balance > 1000;

**#OUTPUT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **account\_id** | **customer\_id** | **account\_type** | **balance** |
|  | 11 | 1 | Current | 1200.00 |
|  | 51 | 5 | Current | 50000.00 |
|  | 91 | 9 | Current | 9000.00 |
|  |  |  |  |  |

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

SELECT \* FROM Transactions WHERE account\_id = 91;

**#OUTPUT**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **transaction\_id** | **account\_id** | **transaction\_type** | **amount** | **transaction\_date** |
|  | 901 | 91 | Deposit | 5000.00 | 2023-10-23 |
|  |  |  |  |  |  |

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

SELECT account\_id, balance, balance \* 0.02 AS interest\_accrued FROM Accounts

WHERE account\_type = 'Savings';

*(Calculated for 2 % interest rate per year)*

**#OUTPUT**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **account\_id** | **balance** | **interest\_accrued** |
|  | 12 | 70000.00 | 1400.0000 |
|  | 21 | 5000.00 | 100.0000 |
|  | 31 | 43000.00 | 860.0000 |
|  | 41 | 21000.00 | 420.0000 |
|  | 61 | 65000.00 | 1300.0000 |
|  | 71 | 6000.00 | 120.0000 |
|  | 81 | 18000.00 | 360.0000 |

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

SELECT \* FROM Accounts

WHERE balance < 5000;

*(In case of current account , minimum balance is 5000.00)*

**#OUTPUT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **account\_id** | **customer\_id** | **account\_type** | **balance** |
|  | 11 | 1 | Current | 1200.00 |
|  |  |  |  |  |

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

SELECT \* FROM Customers

WHERE address NOT LIKE '987 Cedar St';

**#OUTPUT**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **customer\_id** | **first\_name** | **last\_name** | **DOB** | **email** | **phone\_number** | **address** |
|  | 1 | Liji | Lawrence | 2001-01-01 | lijilaw@gmail.com | 9134567890 | 123 Main St |
|  | 2 | Jany | Dim | 2002-02-02 | janydim@gmail.com | 9887654321 | 456 Maple Ave |
|  | 3 | Agnel | James | 1990-03-03 | agneljames@gmail.com | 9862223333 | 789 Oak Dr |
|  | 4 | Bergi | Johnson | 1995-04-04 | bergi@gmail.com | 9823334444 | 321 Pine Ln |
|  | 5 | Reshma | Gigi | 2000-05-05 | reshgigi@gmail.com | 9734445555 | 654 Willow Rd |
|  | 7 | Evangeline | Evans | 2001-07-07 | evanevans@gmail.com | 9056667777 | 345 Birch Ave |
|  | 8 | Frank | Franklin | 1998-08-08 | frankfranklin@gmail.com | 9667778888 | 678 Spruce Dr |
|  | 9 | Grace | Griffin | 1997-09-09 | gracegriffin@gmail.com | 9778889999 | 912 Elm Ln |
|  | 10 | Harsi | Harrison | 2002-10-10 | harryharrison@gmail.com | 9889990000 | 234 Poplar Rd |
|  |  |  |  |  |  |  |  |

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

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

SELECT AVG(balance) AS average\_balance

FROM Accounts;

**#OUTPUT**

|  |  |
| --- | --- |
|  | **average\_balance** |
|  | 28820.000000 |

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

SELECT account\_id, balance FROM Accounts

ORDER BY balance DESC LIMIT 10;

**#OUTPUT**

|  |  |  |
| --- | --- | --- |
|  | **account\_id** | **balance** |
|  | 12 | 70000.00 |
|  | 61 | 65000.00 |
|  | 51 | 50000.00 |
|  | 31 | 43000.00 |
|  | 41 | 21000.00 |
|  | 81 | 18000.00 |
|  | 91 | 9000.00 |
|  | 71 | 6000.00 |
|  | 21 | 5000.00 |
|  | 11 | 1200.00 |
|  |  |  |

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

SELECT SUM(amount) AS total\_deposits FROM Transactions

WHERE transaction\_type = 'Deposit' AND DATE(transaction\_date) = '2023-08-15';

**#OUTPUT**

|  |  |
| --- | --- |
|  | **total\_deposits** |
|  | 5000.00 |

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

*(Altering Customer table to store joining dates)*

ALTER TABLE Customers ADD COLUMN join\_date DATE;

UPDATE Customers SET join\_date = '2022-04-12'

WHERE customer\_id = 1;

UPDATE Customers SET join\_date = '2019-03-10'

WHERE customer\_id = 2;

UPDATE Customers SET join\_date = '2021-01-21'

WHERE customer\_id = 3;

UPDATE Customers SET join\_date = '2014-09-13'

WHERE customer\_id = 4;

UPDATE Customers SET join\_date = '2022-10-02'

WHERE customer\_id = 5;

UPDATE Customers SET join\_date = '2022-03-03'

WHERE customer\_id = 6;

UPDATE Customers SET join\_date = '2022-11-01'

WHERE customer\_id = 7;

UPDATE Customers SET join\_date = '2023-06-23'

WHERE customer\_id = 8;

UPDATE Customers SET join\_date = '2023-01-16'

WHERE customer\_id = 9;

UPDATE Customers SET join\_date = '2021-06-19'

WHERE customer\_id = 10;

**Query: Oldest Customer**

SELECT Customer\_id,first\_name,last\_name,join\_date FROM Customers

ORDER BY join\_date ASC

LIMIT 1;

**#OUTPUT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Customer\_id** | **first\_name** | **last\_name** | **join\_date** |
|  | 4 | Bergi | Johnson | 2014-09-13 |

**Query: Newest Customer**

SELECT Customer\_id, first\_name,last\_name,join\_date FROM Customers

ORDER BY join\_date DESC

LIMIT 1;

**#OUTPUT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Customer\_id** | **first\_name** | **last\_name** | **join\_date** |
|  | 8 | Frank | Franklin | 2023-06-23 |

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

SELECT Transactions.transaction\_id, Transactions.transaction\_type, Transactions.amount, Transactions.transaction\_date, Accounts.account\_type

FROM Transactions

INNER JOIN Accounts ON Transactions.account\_id = Accounts.account\_id;

**#OUTPUT**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **transaction\_id** | **transaction\_type** | **amount** | **transaction\_date** | **account\_type** |
|  | 101 | Deposit | 500.00 | 2023-10-05 | Current |
|  | 111 | Withdrawal | 30000.00 | 2023-11-14 | Savings |
|  | 201 | Deposit | 2000.00 | 2023-06-02 | Savings |
|  | 301 | Deposit | 3000.00 | 2023-07-03 | Savings |
|  | 401 | Withdrawal | 4000.00 | 2023-07-11 | Savings |
|  | 501 | Deposit | 10000.00 | 2023-07-05 | Current |
|  | 601 | Deposit | 6000.00 | 2023-08-02 | Savings |
|  | 701 | Deposit | 5000.00 | 2023-08-15 | Savings |
|  | 801 | Withdrawal | 8000.00 | 2023-09-20 | Savings |
|  | 901 | Deposit | 5000.00 | 2023-10-23 | Current |

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

SELECT Customers.customer\_id, Customers.first\_name, Customers.last\_name, Accounts.account\_id, Accounts.account\_type, Accounts.balance

FROM Customers

INNER JOIN Accounts ON Customers.customer\_id = Accounts.customer\_id;

**#OUTPUT**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **customer\_id** | **first\_name** | **last\_name** | **account\_id** | **account\_type** | **balance** |
|  | 1 | Liji | Lawrence | 11 | Current | 1200.00 |
|  | 2 | Jany | Dim | 21 | Savings | 5000.00 |
|  | 3 | Agnel | James | 31 | Savings | 43000.00 |
|  | 4 | Bergi | Johnson | 41 | Savings | 21000.00 |
|  | 5 | Reshma | Gigi | 51 | Current | 50000.00 |
|  | 6 | David | Davis | 61 | Savings | 65000.00 |
|  | 7 | Evangeline | Evans | 71 | Savings | 6000.00 |
|  | 8 | Frank | Franklin | 81 | Savings | 18000.00 |
|  | 9 | Grace | Griffin | 91 | Current | 9000.00 |
|  | 10 | Harsi | Harrison | 12 | Savings | 70000.00 |

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

SELECT Transactions.transaction\_id, Transactions.transaction\_type, Transactions.amount, Transactions.transaction\_date, Customers.customer\_id, Customers.first\_name, Customers.last\_name

FROM Transactions

INNER JOIN Accounts ON Transactions.account\_id = Accounts.account\_id

INNER JOIN Customers ON Accounts.customer\_id = Customers.customer\_id

WHERE Accounts.account\_id = 41;

**#OUTPUT**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **transaction\_id** | **transaction\_type** | **amount** | **transaction\_date** | **customer\_id** | **first\_name** | **last\_name** |
|  | 401 | Withdrawal | 4000.00 | 2023-07-11 | 4 | Bergi | Johnson |

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

*(Since there are not more than one account per customer, inserting two values)*

INSERT INTO Accounts (account\_id, customer\_id, account\_type, balance) VALUES

(32, 1, 'Savings', 6000.00),

(42, 1, 'Current', 2100.00);

**Query:**

SELECT Customers.customer\_id, Customers.first\_name, Customers.last\_name, COUNT(Accounts.account\_id) AS NoOfAccounts

FROM Customers

INNER JOIN Accounts ON Customers.customer\_id = Accounts.customer\_id

GROUP BY Customers.customer\_id, Customers.first\_name, Customers.last\_name

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

**#OUTPUT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **customer\_id** | **first\_name** | **last\_name** | **NoOfAccounts** |
|  | 1 | Liji | Lawrence | 3 |

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

SELECT 'Deposit' AS transaction\_type, SUM(amount) AS TotalAmount

FROM Transactions

WHERE transaction\_type = 'Deposit'

UNION ALL

SELECT 'Withdrawal' AS transaction\_type, SUM(amount) AS TotalAmount

FROM Transactions

WHERE transaction\_type = 'Withdrawal'

UNION ALL

SELECT 'Difference' AS transaction\_type,

((SELECT SUM(amount) FROM Transactions WHERE transaction\_type = 'Deposit') -

(SELECT SUM(amount) FROM Transactions WHERE transaction\_type = 'Withdrawal')) AS TotalAmount;

**#OUTPUT**

|  |  |  |
| --- | --- | --- |
|  | **transaction\_type** | **TotalAmount** |
|  | Deposit | 31500.00 |
|  | Withdrawal | 42000.00 |
|  | Difference | -10500.00 |

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

SELECT account\_id, transaction\_date, AVG(amount) AS AvgBal FROM Transactions

WHERE transaction\_date BETWEEN '2023-06-01' AND '2023-08-30'

GROUP BY account\_id, transaction\_date;

**#OUTPUT**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **account\_id** | **transaction\_date** | **AvgBal** |
|  | 21 | 2023-06-02 | 2000.000000 |
|  | 31 | 2023-07-03 | 3000.000000 |
|  | 41 | 2023-07-11 | 4000.000000 |
|  | 51 | 2023-07-05 | 10000.000000 |
|  | 61 | 2023-08-02 | 6000.000000 |
|  | 71 | 2023-08-15 | 5000.000000 |

11.Calculate the total balance for each account type.

SELECT account\_type, SUM(balance) AS TotalBalance FROM Accounts

GROUP BY account\_type;

**#OUTPUT**

|  |  |  |
| --- | --- | --- |
|  | **account\_type** | **TotalBalance** |
|  | Current | 62300.00 |
|  | Savings | 234000.00 |

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

SELECT account\_id, COUNT(\*) AS NoOfTransactions FROM Transactions

GROUP BY account\_id

ORDER BY NoOfTransactions DESC;

**#OUTPUT**

|  |  |  |
| --- | --- | --- |
|  | **account\_id** | **NoOfTransactions** |
|  | 11 | 1 |
|  | 12 | 1 |
|  | 21 | 1 |
|  | 31 | 1 |
|  | 41 | 1 |
|  | 51 | 1 |
|  | 61 | 1 |
|  | 71 | 1 |
|  | 81 | 1 |
|  | 91 | 1 |

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

SELECT Customers.customer\_id, Customers.first\_name, Customers.last\_name, Accounts.account\_type, SUM(Accounts.balance) AS TotalBalance

FROM Customers

INNER JOIN Accounts ON Customers.customer\_id = Accounts.customer\_id

GROUP BY Customers.customer\_id, Accounts.account\_type

ORDER BY TotalBalance DESC;

**#OUTPUT**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **customer\_id** | **first\_name** | **last\_name** | **account\_type** | **TotalBalance** |
|  | 10 | Harsi | Harrison | Savings | 70000.00 |
|  | 6 | David | Davis | Savings | 65000.00 |
|  | 5 | Reshma | Gigi | Current | 50000.00 |
|  | 3 | Agnel | James | Savings | 43000.00 |
|  | 4 | Bergi | Johnson | Savings | 21000.00 |
|  | 8 | Frank | Franklin | Savings | 18000.00 |
|  | 9 | Grace | Griffin | Current | 9000.00 |
|  | 1 | Liji | Lawrence | Savings | 6000.00 |
|  | 7 | Evangeline | Evans | Savings | 6000.00 |
|  | 2 | Jany | Dim | Savings | 5000.00 |
|  | 1 | Liji | Lawrence | Current | 3300.00 |

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

SELECT account\_id, transaction\_date, amount, COUNT(\*) AS NoOfDuplicates FROM Transactions

GROUP BY account\_id, transaction\_date, amount

HAVING COUNT(\*) > 1;

**#OUTPUT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **account\_type** | **Transaction\_date** | **amount** | **NoOfDuplicates** |

*(Duplicate transactions are not available)*

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

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

SELECT Customers.customer\_id, Customers.first\_name, Customers.last\_name, Accounts.balance AS HighestBalance FROM Customers

INNER JOIN Accounts ON Customers.customer\_id = Accounts.customer\_id

WHERE Accounts.balance = (SELECT MAX(balance) FROM Accounts);

**#OUTPUT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **customer\_id** | **first\_name** | **last\_name** | **HighestBalance** |
|  | 10 | Harsi | Harrison | 70000.00 |

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

SELECT customer\_id, COUNT(account\_id) AS NoOfAccounts, AVG(balance) AS AvgBal FROM Accounts

WHERE customer\_id IN (

SELECT customer\_id FROM Accounts

GROUP BY customer\_id

HAVING COUNT(account\_id) > 1

)

GROUP BY customer\_id;

**#OUTPUT**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **customer\_id** | **NoOfAccounts** | **AvgBal** |
|  | 1 | 3 | 3100.000000 |

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

SELECT account\_id FROM Transactions

WHERE amount > ( SELECT AVG(amount) FROM Transactions );

**#OUTPUT**

|  |  |
| --- | --- |
|  | **account\_id** |
|  | 12 |
|  | 51 |
|  | 81 |

4.Identify customers who have no recorded transactions.

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

LEFT JOIN Accounts ON Customers.customer\_id = Accounts.customer\_id

LEFT JOIN Transactions ON Accounts.account\_id = Transactions.account\_id

WHERE Transactions.transaction\_id IS NULL;

**#OUTPUT**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **customer\_id** | **first\_name** | **last\_name** |
|  | 1 | Liji | Lawrence |
|  | 1 | Liji | Lawrence |

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

SELECT Accounts.account\_id, Accounts.balance, TotalBalance.total FROM Accounts

LEFT JOIN Transactions ON Accounts.account\_id = Transactions.account\_id

CROSS JOIN (

SELECT SUM(balance) AS total

FROM Accounts

LEFT JOIN Transactions ON Accounts.account\_id = Transactions.account\_id

WHERE Transactions.transaction\_id IS NULL

) AS TotalBalance WHERE Transactions.transaction\_id IS NULL;

**#OUTPUT**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **account\_id** | **balance** | **total** |
|  | 32 | 6000.00 | 8100.00 |
|  | 42 | 2100.00 | 8100.00 |

6.Retrieve transactions for accounts with the lowest balance.

SELECT Transactions.\* FROM Transactions

WHERE account\_id IN (

SELECT account\_id FROM Accounts

WHERE balance = ( SELECT MIN(balance) FROM Accounts)

);

**#OUTPUT**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **transaction\_id** | **account\_id** | **transaction\_type** | **amount** | **transaction\_date** |
|  | 101 | 11 | Deposit | 500.00 | 2023-10-05 |
|  |  |  |  |  |  |

7.Identify customers who have accounts of multiple types.

SELECT account\_id ,customer\_id, account\_type FROM Accounts

GROUP BY customer\_id,account\_id

HAVING COUNT(DISTINCT account\_type) > 1;

**#OUTPUT**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **account\_id** | **customer\_id** | **account\_type** |
|  |  |  |  |

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

SELECT account\_type, COUNT(\*) / (SELECT COUNT(\*) FROM Accounts) \* 100 AS percentage FROM Accounts

GROUP BY account\_type;

**#OUTPUT**

|  |  |  |
| --- | --- | --- |
|  | **account\_type** | **percentage** |
|  | Current | 33.3333 |
|  | Savings | 66.6667 |

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

SELECT Transactions.\* FROM Transactions

JOIN Accounts ON Transactions.account\_id = Accounts.account\_id

WHERE Accounts.customer\_id = 3;

**#OUTPUT**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **transaction\_id** | **account\_id** | **transaction\_type** | **amount** | **transaction\_date** |
|  | 301 | 31 | Deposit | 3000.00 | 2023-07-03 |

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

SELECT account\_type,

(SELECT SUM(balance) FROM Accounts AS SubAccounts WHERE SubAccounts.account\_type = MainAccounts.account\_type) AS TotalBalance

FROM Accounts AS MainAccounts

GROUP BY account\_type;

**#OUTPUT**

|  |  |  |
| --- | --- | --- |
|  | **account\_type** | **TotalBalance** |
|  | Current | 62300.00 |
|  | Savings | 234000.00 |