**Tasks 1: Database Design:**

1. Create the database named "HMBank":

**CREATE DATABASE** HMBank**;**

**USE** HMBank**;**

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

A screenshot of a computer

Description automatically generated

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

A diagram of a customer account

Description automatically generated

5. Create appropriate Primary Key and Foreign Key constraints for referential integrity:  
 **Customers Table :  
 customer\_id INT PRIMARY KEY  
 Accounts Table:  
 account\_id INT PRIMARY KEY and  
 FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id) ON DELETE CASCADE  
 Transactions Table:  
 transaction\_id INT PRIMARY KEY and  
 FOREIGN KEY (account\_id) REFERENCES accounts(account\_id) ON DELETE CASCADE**

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

* Customers:

CREATE TABLE customers(

customer\_id INT PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

DOB DATE NOT NULL,

email VARCHAR(30),

phone\_number VARCHAR(15) UNIQUE NOT NULL,

address VARCHAR(250) NOT NULL

);

* Accounts:

CREATE TABLE accounts(

account\_id INT PRIMARY KEY,

customer\_id INT,

account\_type VARCHAR(50),

balance DECIMAL(15,2) NOT NULL CHECK(balance>=0),

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id) ON DELETE CASCADE

);

* Transactions:

CREATE TABLE transactions(

transaction\_id INT PRIMARY KEY,

account\_id INT,

transaction\_type VARCHAR(50) NOT NULL,

amount DECIMAL(15,2) NOT NULL CHECK(amount>=1),

transaction\_date DATE NOT NULL,

FOREIGN KEY (account\_id) REFERENCES accounts(account\_id) ON DELETE CASCADE

);

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

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

* Customers:

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

(1,'Usman','Patel','2002-08-08','usman@gmail.com','7058102180','Plot No 111, Solapur'),

(2,'Ankit','Chowdhary','1999-01-15','ankit@gmail.com','9856321470','Plot No 1, Mumbai'),

(3,'Anamika','Patil','2002-03-01','anamika@gmail.com','1234567890','Plot No 12, Pune'),

(4,'Roy','Kapoor','2000-12-10','roy@gmail.com','5412369780','Plot No 401, NCR'),

(5,'Anu','Rai','2001-9-22','anu@gmail.com','4123569870','Plot NO 42, Hyderabad'),

(6,'Arjun','Reddy','1995-10-01','arjunreddy@gmail.com','7412369850','Plot No 32, Secundarabad'),

(7,'Vijay','Singh','1990-11-29','vijay@gmail.com','8741239650','Plot No 1, Delhi'),

(8,'Varun','Mehta','1985-08-15','varun@gmail.com','6987412350','Plot No 17, Chennai'),

(9,'Rajni','Nellam','1975-07-10','rajni@gmail.com','2587410369','Plot No 2, Chennai'),

(10,'Praveen','Miishra','1962-05-10',null,'3216549870','Plot No 4 , Chandigarh');

* Accounts:

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

(101,1,'savings',10000),

(102,2,'savings',5000),

(103,3,'zero\_balance',0),

(104,4,'current','2000'),

(105,5,'current',3000),

(106,6,'fixed\_deposit',500000),

(107,7,'fixed\_deposit',10000000),

(108,8,'savings',15000),

(109,9,'savings',100000),

(110,10,'savings',25000);

* Transactions:

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

(1001,101,'deposit',10000,'2023-12-08'),

(1002,102,'deposit',5000,'2023-12-08'),

(1003,103,'withdraw',5000,'2023-12-04'),

(1004,104,'transfer',1000,'2023-12-05'),

(1005,105,'transfer',1500,'2023-12-06'),

(1006,106,'deposit',500000,'2023-12-08'),

(1007,107,'deposit',5000000,'2023-12-07'),

(1008,108,'deposit',15000,'2023-12-03'),

(1009,109,'withdraw',2000,'2023-12-06'),

(1010,110,'withdraw',2500,'2023-12-05');

1. **Write SQL queries for the following tasks:**
2. Write a SQL query to retrieve the name, account type and email of all customers:

SELECT CONCAT(c.first\_name,' ',c.last\_name) as Name,a.account\_type as Account\_Type,c.email as Email

FROM customers c,accounts a

WHERE c.customer\_id = a.customer\_id;

1. Write a SQL query to list all transaction corresponding customer:

SELECT CONCAT(c.first\_name,' ',c.last\_name) as Name, t.transaction\_id, t.transaction\_type, t.amount, t.transaction\_date

FROM customers c, accounts a, transactions t

WHERE c.customer\_id = a.customer\_id and t.account\_id = a.account\_id;

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

UPDATE accounts SET balance = balance + 500 WHERE account\_id = 101;

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

SELECT CONCAT(c.first\_name,' ',c.last\_name) as Full\_Name FROM customers c;

1. 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';

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

SELECT \*

FROM customers

WHERE address LIKE '%Mumbai%';

1. Write a SQL query to Get the account balance for a specific account:

SELECT balance AS Account\_Balance FROM accounts WHERE account\_id=105;

1. 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;

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

SELECT \* FROM transactions WHERE account\_id=105;

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

SELECT account\_id, balance \* 0.05 AS Interest

FROM accounts

WHERE account\_type = 'savings';

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

SELECT \* FROM accounts WHERE balance < 1000;

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

SELECT \* FROM customers

WHERE address NOT LIKE '%Chennai%';

**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;

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

SELECT \* FROM accounts

ORDER BY balance DESC LIMIT 10;

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

SELECT transaction\_date ,SUM(amount) AS Total\_Deposits

FROM transactions

WHERE transaction\_type = 'deposit' AND transaction\_date = '2023-12-08';

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

(SELECT \* ,'Oldest' as Age\_Status FROM customers ORDER BY DOB ASC LIMIT 1)

UNION ALL

(SELECT \* ,'Newest' as Age\_Status FROM customers ORDER BY DOB DESC LIMIT 1);

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

SELECT t.\*, a.account\_type

FROM transactions t

JOIN accounts a ON t.account\_id = a.account\_id;

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

SELECT c.\*, a.\*

FROM customers c

JOIN accounts a ON c.customer\_id = a.customer\_id;

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

SELECT t.\*, c.\*

FROM transactions t

JOIN accounts a ON t.account\_id = a.account\_id

JOIN customers c ON a.customer\_id = c.customer\_id

WHERE a.account\_id = 101;

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

SELECT customer\_id

FROM accounts

GROUP BY customer\_id

HAVING COUNT(account\_id) > 1;

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

SELECT

(SELECT SUM(amount) FROM transactions WHERE transaction\_type = 'deposit') -

(SELECT SUM(amount) FROM transactions WHERE transaction\_type <> 'deposit') AS Difference;

1. Write a SQL query to Calculate the average daily balance for each account over a specified period:
2. Calculate the total balance for each account type:

SELECT account\_type, SUM(balance) AS total\_balance

FROM accounts

GROUP BY account\_type;

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

SELECT account\_id, COUNT(transaction\_id) AS transaction\_count

FROM transactions

GROUP BY account\_id

ORDER BY transaction\_count DESC;

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

SELECT c.customer\_id, c.first\_name, c.last\_name,

GROUP\_CONCAT(a.account\_type SEPARATOR ', ') as account\_types,

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

ORDER BY total\_balance DESC;

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

SELECT transaction\_id, account\_id, amount, transaction\_date, COUNT(\*)

FROM transactions

GROUP BY account\_id, amount, transaction\_date

HAVING COUNT(\*) > 1;

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

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

SELECT c.\*, a.balance as High\_Account\_Balance

FROM customers c

JOIN accounts a ON c.customer\_id = a.customer\_id

WHERE a.balance = (SELECT MAX(balance) FROM accounts);

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

SELECT customer\_id,COUNT(account\_id), AVG(balance) AS average\_balance

FROM accounts

WHERE customer\_id IN (

SELECT customer\_id

FROM accounts

GROUP BY customer\_id

HAVING COUNT(account\_id) > 1

);

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

SELECT a.\*,t.transaction\_type,t.amount, t.transaction\_date

FROM accounts a

JOIN transactions t ON a.account\_id = t.account\_id

WHERE t.amount > (

SELECT AVG(amount)

FROM transactions

);

1. Identify customers who have no recorded transactions:

SELECT c.\*

FROM customers c

LEFT JOIN accounts a ON c.customer\_id = a.customer\_id

LEFT JOIN transactions t ON a.account\_id = t.account\_id

WHERE t.transaction\_id IS NULL;

1. Calculate the total balance of accounts with no recorded transactions:

SELECT account\_id, SUM(balance) AS total\_balance

FROM accounts

WHERE account\_id NOT IN (

SELECT DISTINCT account\_id

FROM transactions

);

1. Retrieve transactions for accounts with the lowest balance:

SELECT a.account\_id, t.\*,a.balance as Account\_Balance

FROM transactions t

JOIN accounts a ON t.account\_id = a.account\_id

WHERE a.balance = (SELECT MIN(balance) FROM accounts);

1. Identify customers who have accounts of multiple types:

SELECT c.\*

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;

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

SELECT account\_type, COUNT(\*) \* 100.0 / (SELECT COUNT(\*) FROM accounts) AS percentage

FROM accounts

GROUP BY account\_type;

1. Retrieve all transactions for a customer with a given customer\_id:

SELECT CONCAT(c.first\_name,' ',c.last\_name) as Name, t.\*

FROM transactions t

JOIN accounts a ON a.account\_id = t.account\_id

JOIN customers c ON c.customer\_id = a.customer\_id

WHERE a.customer\_id = 4;

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

SELECT account\_type, (SELECT SUM(balance) FROM accounts WHERE account\_type = a1.account\_type) AS total\_balance

FROM accounts a1

GROUP BY account\_type;