**Banking System**

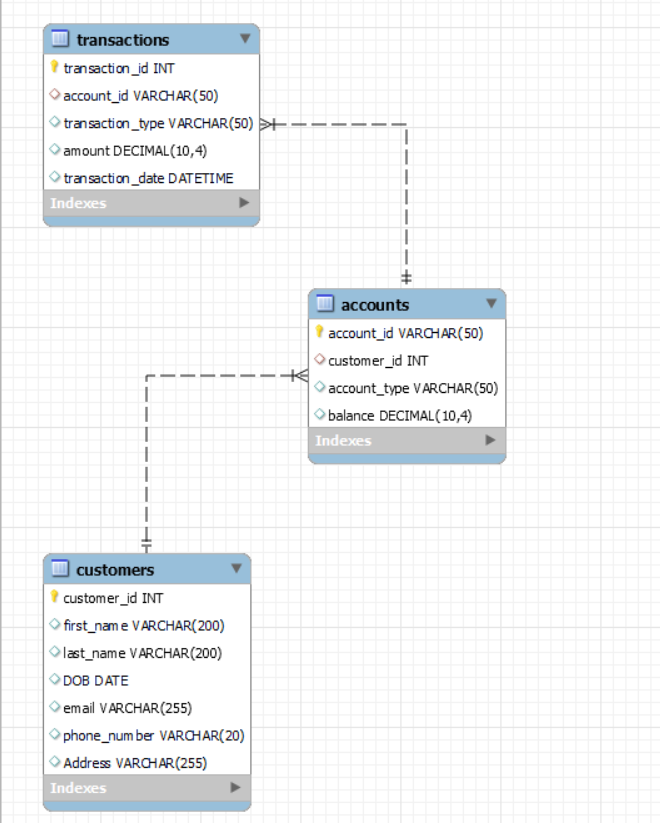
**Tasks 1: Database Design :**

1. Create the database named "HMBank"

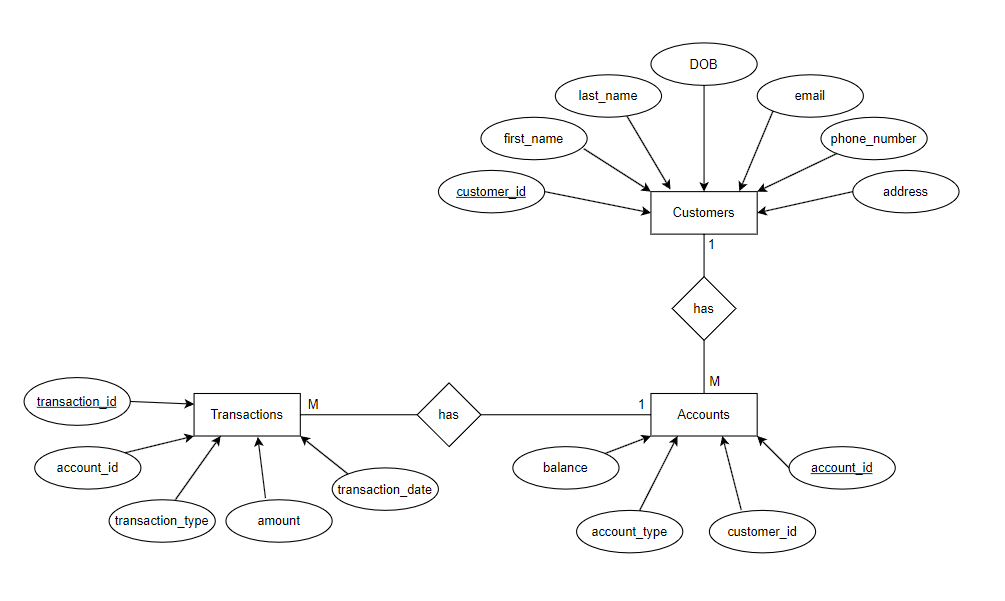
**CREATE DATABASE** HMBank;

**USE** HMBank;

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



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



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

-- Adding Primary Key constraint for Customers table

**ALTER TABLE** Customers

**ADD CONSTRAINT** PK\_Customers **PRIMARY KEY** (customer\_id);

-- Adding Primary Key constraint for Accounts table

**ALTER TABLE** Accounts

**ADD CONSTRAINT** PK\_Accounts **PRIMARY KEY** (account\_id);

-- Adding Primary Key constraint for Transactions table

**ALTER TABLE** Transactions

**ADD CONSTRAINT** PK\_Transactions **PRIMARY** **KEY** (transaction\_id);

-- Adding Foreign Key constraint for Accounts table

**ALTER TABLE** Accounts

**ADD CONSTRAINT** FK\_Accounts\_Customers

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

-- Adding Foreign Key constraint for Transactions table

**ALTER TABLE** Transactions

**ADD CONSTRAINT** FK\_Transactions\_Accounts

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

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

• Customers

• Accounts

• Transactions

**CREATE** **TABLE** Customers (

customer\_id **INT** **AUTO\_INCREMENT** **PRIMARY** **KEY**,

first\_name **VARCHAR**(200),

last\_name **VARCHAR**(200),

DOB **DATE**,

email **VARCHAR**(255),

phone\_number **VARCHAR**(20),

Address **VARCHAR**(255)

);

**CREATE** **TABLE** Accounts (

account\_id **varchar**(50) **PRIMARY** **KEY**,

customer\_id **INT**,

account\_type **VARCHAR**(50),

balance **DECIMAL**(10, 4),

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

);

**CREATE** **TABLE** Transactions (

transaction\_id **INT** **PRIMARY** **KEY**,

account\_id **varchar**(50),

transaction\_type **VARCHAR**(50),

amount **DECIMAL**(10, 4),

transaction\_date **DATETIME**,

**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 (first\_name, last\_name, DOB, email, phone\_number, Address)

**VALUES**

('Ram', 'Lakshman', '2000-01-15', '[Ram.lakshman@gmail.com](mailto:Ram.lakshman@gmail.com)', '1234567890', '123 Main St'),

('Raju', 'Smith', '2001-05-22', '[raju.smith@gmail.com](mailto:raju.smith@gmail.com)', '9876543210', '456 Oak St'),

('Rols', 'Johnson', '2005-08-10', '[rols.johnson@gmail.com](mailto:rols.johnson@gmail.com)', '5551234567', '789 Pine St'),

('John', 'Doe', '2001-01-15', '[john.doe@gmail.com](mailto:john.doe@gmail.com)', '1234567890', '123 Main St'),

('Bharat', 'RamChandra', '2005-05-22', '[bharat@gmail.com](mailto:bharat@gmail.com)', '9876543210', '456 Oak St'),

('Varshini', 'Uda', '1999-08-10', '[varshiniiuda@gmail.com](mailto:varshiniiuda@gmail.com)', '5551234567', '789 Pine St'),

('Santosh', 'Sai', '1999-04-05', '[saisantosh2102@gmail.com](mailto:saisantosh2102@gmail.com)', '3219876543', '987 Elm St'),

('Thar', 'Mahindra', '2003-08-10', '[Mahindra.thar@gmail.com](mailto:Mahindra.thar@gmail.com)', '5551234567', '789 Pine St'),

('Nehal', 'Praveen', '2003-04-05', '[Nehal@gmail.com](mailto:Nehal@gmail.com)', '3219876543', '987 Elm St'),

('Chetana', 'Sureddi', '2002-04-05', '[Chetana@gmail.com](mailto:Chetana@gmail.com)', '3219876543', '987 Elm St');

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

(8976456898, 1 , 'savings', 100000),

(8972145689, 2 , 'savings', 100000)

(7654234568, 3 , 'current', 30000),

(2348823938, 4 , 'current', 50000),

(4568945162, 5 , 'zer\_balance', 1000),

(1476718317, 6 , 'current', 9000),

(8173871381, 7 , 'savings', 10000),

(7634739740, 8 , 'savings', 40000),

(1734879820, 9 , 'current', 56000),

(6991824128, 10, 'zero\_balance', 800);

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

(2145678, 8976456898, 'transfer', 1000, **CURRENT\_TIMESTAMP**),

(4723423, 8972145689, 'withdrawal', 0.0, **CURRENT\_TIMESTAMP**),

(5431671, 7654234568, 'transfer', 68.9, **CURRENT\_TIMESTAMP**),

(7685749, 2348823938, 'deposit', 7458.8, **CURRENT\_TIMESTAMP**),

(3473849, 4568945162, 'withdrawal', 46.09, **CURRENT\_TIMESTAMP**),

(3437434, 1476718317, 'transfer', 6472, **CURRENT\_TIMESTAMP**),

(4252311, 8173871381, 'withdrawal', 454, **CURRENT\_TIMESTAMP**),

(6534849, 7634739740, 'deposit', 40000, **CURRENT\_TIMESTAMP**),

(4584594, 1734879820, 'transfer', 4545, **CURRENT\_TIMESTAMP**),

(4758459, 6991824128, 'withdrawal', 4211, **CURRENT\_TIMESTAMP**);

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(first\_name, last\_name) AS name , a.account\_type , c.email

**FROM** Customers c, Accounts a **WHERE** a.customer\_id = c.customer\_id;

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

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

**FROM** Customers, Accounts, Transactions

**WHERE** Customers.customer\_id = Accounts.customer\_id **AND** Accounts.account\_id = Transactions.account\_id;

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

**UPDATE** Accounts **SET** balance = balance + '2000' **WHERE** account\_id = '1476718317';

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 c;

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 LIKE '123%';

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

**SELECT** balance **FROM** Accounts **WHERE** account\_id = '1734879820';

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 > 1000.0;

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

**SELECT** \* **FROM** Transactions **WHERE** account\_id = '1734879820';

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

**SELECT** account\_id, balance \* 0.1 AS Interest **FROM** Accounts **WHERE** account\_type = 'savings';

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

**SELECT** \* **FROM** Accounts **WHERE** balance < 4000;

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

**SELECT** \* **FROM** Customers **WHERE** Address **NOT** **LIKE** '123%';

**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;

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;

3. 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';

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

**SELECT** **MIN**(DOB) **AS** oldest\_customer\_dob, **MAX**(DOB) **AS** newest\_customer\_dob **FROM** Customers;

5. 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;

6. 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;

7. 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 = '8976456898';

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

**SELECT** customer\_id **FROM** Accounts **GROUP** **BY** customer\_id **HAVING** **COUNT**(\*) > 1;

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

**SELECT** account\_id, **SUM**(**CASE** **WHEN** transaction\_type = 'Deposit' **THEN** amount **ELSE** 0 **END**) - **SUM**(**CASE** **WHEN** transaction\_type = 'Withdrawal' **THEN** amount **ELSE** 0 **END**) **AS** balance\_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.

**SELECT** account\_id, round(AVG(daily\_balance),1) **AS** average\_daily\_balance, transaction\_types

**FROM** (

**SELECT** a.account\_id, t.transaction\_date, GROUP\_CONCAT(t.transaction\_type SEPARATOR ', ') as transaction\_types, a.balance -

**COALESCE(SUM**(CASE WHEN t.transaction\_type <> 'deposit' **THEN** t.amount ELSE 0 END), 0) +

**COALESCE(SUM**(CASE WHEN t.transaction\_type = 'deposit' **THEN** t.amount ELSE 0 END), 0)

AS daily\_balance

**FROM** accounts a

**LEFT JOIN** transactions t **ON** a.account\_id = t.account\_id **AND** t.transaction\_date BETWEEN '2023-12-01' AND '2023-12-08'

**GROUP** **BY** a.account\_id, t.transaction\_date, a.balance

) **AS** DailyBalances

**GROUP** **BY** account\_id **ORDER** **BY** account\_id;

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

**SELECT** account\_id, **COUNT**(\*) **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.

**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;

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

**SELECT** t1.transaction\_id, t1.account\_id, t1.transaction\_type, t1.amount, t1.transaction\_date **FROM** Transactions t1 **JOIN** Transactions t2 **ON** t1.amount = t2.amount **AND** t1.transaction\_date = t2.transaction\_date **AND** t1.account\_id = t2.account\_id **AND** t1.transaction\_id <> t2.transaction\_id;

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

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

**SELECT** c.customer\_id, c.first\_name, c.last\_name **FROM** Customers c **JOIN** Accounts a **ON** c.customer\_id = a.customer\_id **WHERE** a.balance = (**SELECT** **MAX**(balance) **FROM** Accounts);

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

**SELECT** customer\_id, **AVG**(balance) **AS** average\_balance **FROM** Accounts **GROUP** **BY** customer\_id **HAVING** **COUNT**(\*) > 1;

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

**SELECT** a.account\_id, a.balance, 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);

4. Identify customers who have no recorded transactions.

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

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

**SELECT** **COALESCE**(**SUM**(balance), 0) **AS** total\_balance **FROM** Accounts a **LEFT** **JOIN** Transactions t **ON** a.account\_id = t.account\_id **WHERE** t.transaction\_id **IS** **NULL**;

6. Retrieve transactions for accounts with the lowest balance.

**SELECT** t.\* **FROM** Transactions t **JOIN** Accounts a **ON** t.account\_id = a.account\_id **WHERE** a.balance = (**SELECT** **MIN**(balance) **FROM** Accounts);

7. Identify customers who have accounts of multiple types.

**SELECT** customer\_id **FROM** Accounts **GROUP** **BY** customer\_id **HAVING** **COUNT**(**DISTINCT** account\_type) > 1;

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

**SELECT** account\_type, **COUNT**(\*) **AS** total\_accounts, (**COUNT**(\*) \* 100.0) / (**SELECT** **COUNT**(\*) **FROM** Accounts) **AS** percentage **FROM** Accounts **GROUP** **BY** account\_type;

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

**SELECT** \* **FROM** Transactions **WHERE** account\_id **IN** (**SELECT** account\_id **FROM** Accounts **WHERE** customer\_id = '1');

10. 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 = a.account\_type) **AS** total\_balance **FROM** Accounts a **GROUP** **BY** account\_type;