Banking System

# Tasks 1: Database Design:

1. Create the database named "HMBank"

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

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

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

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

• Customers  
• Accounts  
• Transactions

1. create database hmbank;  
 use hmbank;  
  
2. create table customers (  
 customer\_id int primary key,  
 first\_name varchar(100),  
 last\_name varchar(100),  
 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)  
);

### 3.ERD



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

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

• Customers  
• Accounts  
• Transactions

## Insert Statements

Insert into Customers  
insert into customers values   
(1, 'ravi', 'kumar', '1990-05-12', 'ravi@gmail.com', '9876543210', 'mumbai'),   
(2, 'anita', 'singh', '1988-07-22', 'anita@gmail.com', '8765432109', 'delhi'),  
(3, 'arjun', 'verma', '1995-01-15', 'arjun@gmail.com', '7654321098', 'chennai'),  
(4, 'meena', 'das', '1992-03-05', 'meena@gmail.com', '6543210987', 'kolkata'),  
(5, 'vijay', 'rao', '1998-11-30', 'vijay@gmail.com', '5432109876', 'bangalore'),  
(6, 'sita', 'iyer', '1993-08-15', 'sita@gmail.com', '9123456780', 'chennai'),  
(7, 'rahul', 'sharma', '1985-12-01', 'rahul@gmail.com', '9345678123', 'pune'),  
(8, 'neha', 'jain', '1997-06-20', 'neha@gmail.com', '9988776655', 'hyderabad'),  
(9, 'akash', 'mehra', '1991-09-11', 'akash@gmail.com', '9887766554', 'ahmedabad'),  
(10, 'pooja', 'reddy', '1994-04-05', 'pooja@gmail.com', '9776655443', 'kochi');  
  
 Insert into Accounts  
insert into accounts values   
(101, 1, 'savings', 50000.00),   
(102, 2, 'current', 120000.00),   
(103, 3, 'zero\_balance', 1000.00),   
(104, 4, 'savings', 80000.00),   
(105, 5, 'current', 150000.00),  
(106, 6, 'savings', 30000.00),  
(107, 7, 'current', 110000.00),  
(108, 8, 'zero\_balance', 500.00),  
(109, 9, 'savings', 60000.00),  
(110, 10, 'current', 95000.00);  
  
Insert into Transactions  
insert into transactions values   
(201, 101, 'deposit', 10000.00, '2024-06-10'),   
(202, 102, 'withdrawal', 5000.00, '2024-06-11'),   
(203, 103, 'deposit', 2000.00, '2024-06-12'),   
(204, 104, 'withdrawal', 3000.00, '2024-06-13'),   
(205, 105, 'deposit', 25000.00, '2024-06-14'),  
(206, 106, 'deposit', 15000.00, '2024-06-15'),  
(207, 107, 'withdrawal', 20000.00, '2024-06-16'),  
(208, 108, 'deposit', 1000.00, '2024-06-17'),  
(209, 109, 'deposit', 25000.00, '2024-06-18'),  
(210, 110, 'withdrawal', 5000.00, '2024-06-19');

## 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 c.first\_name, c.last\_name, c.email, a.account\_type

FROM customers c, accounts a

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

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

SELECT c.first\_name, c.last\_name, t.transaction\_type, t.amount, t.transaction\_date

FROM customers c, accounts a, transactions t

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

AND a.account\_id = t.account\_id;

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

UPDATE accounts SET balance = balance + 5000 WHERE account\_id = 101;

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

SELECT 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 = 'chennai';

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

SELECT balance FROM accounts a WHERE account\_id = 102;

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

SELECT \* FROM accounts a WHERE account\_type = 'current' AND balance > 1000;

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

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

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

SELECT account\_id, balance, balance \* 0.04 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 < 2000;

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

SELECT \* FROM customers WHERE address != 'mumbai';

# 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 TOP 10 account\_id, balance

FROM accounts

ORDER BY balance DESC;

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

SELECT SUM(amount) AS total\_deposits FROM transactions t

WHERE transaction\_type = 'deposit'

AND transaction\_date = '2024-06-14';

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

SELECT MIN(dob), MAX(dob) FROM customers;

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

SELECT t.transaction\_id, t.account\_id, a.account\_type,

t.transaction\_type, t.amount, t.transaction\_date

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

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.

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

c.first\_name, c.last\_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

WHERE t.account\_id = 105;

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

SELECT customer\_id, COUNT(account\_id) FROM accounts

GROUP BY customer\_id HAVING COUNT(account\_id) > 1;

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

SELECT SUM(CASE WHEN transaction\_type = 'deposit' THEN amount ELSE 0 END) -

SUM(CASE WHEN transaction\_type = 'withdrawal' THEN amount ELSE 0 END)

AS difference FROM transactions;

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

SELECT account\_id, AVG(balance) AS avg\_daily\_balance

FROM accounts GROUP BY account\_id;

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

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

FROM accounts GROUP BY account\_type;

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

SELECT account\_id, COUNT(\*) AS total\_txns FROM transactions

GROUP BY account\_id ORDER BY total\_txns DESC;

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

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, a.account\_type

HAVING SUM(a.balance) > 100000;

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

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

FROM transactions

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

# Tasks 4: Subquery and its type:

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

SELECT \* FROM customers c WHERE c.customer\_id IN (SELECT customer\_id FROM accounts WHERE balance = (SELECT MAX(balance) FROM accounts));

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

SELECT AVG(balance) FROM accounts WHERE customer\_id IN (SELECT customer\_id FROM accounts GROUP BY customer\_id HAVING COUNT(\*) > 1);

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

SELECT \* FROM transactions WHERE amount > (SELECT AVG(amount) FROM transactions);

### Identify customers who have no recorded transactions.

SELECT \* FROM customers c WHERE c.customer\_id NOT IN (SELECT a.customer\_id FROM accounts a JOIN transactions t ON a.account\_id = t.account\_id);

### Calculate the total balance of accounts with no recorded transactions

SELECT SUM(balance) FROM accounts a WHERE a.account\_id NOT IN (SELECT account\_id FROM transactions);

### Retrieve transactions for accounts with the lowest balance.

SELECT \* FROM transactions WHERE account\_id IN (SELECT account\_id FROM accounts WHERE balance = (SELECT MIN(balance) FROM accounts));

### Identify customers who have accounts of multiple types.

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

### 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 = 3);

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

SELECT account\_type, (SELECT SUM(balance) FROM accounts a2 WHERE a2.account\_type = a1.account\_type) AS total\_balance FROM accounts a1 GROUP BY account\_type;