**Name- Satyam Hans(J521)**

**Topic- Banking System**

**Tasks 1**: **Database Design**:

create database Bank;

CREATE TABLE Customers (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(100) NOT NULL,

last\_name VARCHAR(100) NOT NULL,

DOB DATE,

email VARCHAR(255) UNIQUE,

phone\_number VARCHAR(15),

address VARCHAR(255)

);

create table Accounts (

account\_id int auto\_increment primary key,

customer\_id int,

account\_type enum('savings','current','zero\_balance'),

balance decimal(10,2) not null,

foreign key(customer\_id) references Customers(customer\_id) on delete cascade on update cascade

);

create table Transactions (

transaction\_id int auto\_increment primary key,

account\_id int,

transaction\_type enum('deposit', 'withdrawal', 'transfer'),

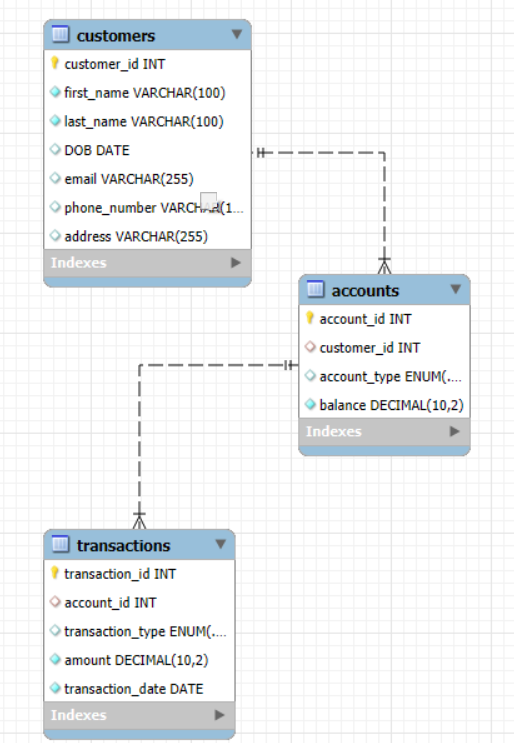
amount decimal(10,2) not null,

transaction\_date date not null,

foreign key Accounts(account\_id) references Accounts(account\_id) on delete cascade on update cascade

);

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



**Task 2**  
1) 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

('John', 'Doe', '1985-03-15', 'john.doe@example.com', '1234567890', '123 Elm St'),

('Jane', 'Smith', '1990-06-25', 'jane.smith@example.com', '0987654321', '456 Oak St'),

('Sam', 'Johnson', '1975-11-05', 'sam.johnson@example.com', '9876543210', '789 Pine St'),

('Sara', 'Williams', '1983-02-10', 'sara.williams@example.com', '3216549870', '321 Maple St'),

('David', 'Brown', '1995-09-20', 'david.brown@example.com', '6541239876', '654 Cedar St'),

('Emily', 'Davis', '1998-12-12', 'emily.davis@example.com', '7412589630', '963 Birch St'),

('Michael', 'Miller', '1987-08-01', 'michael.miller@example.com', '8523697412', '852 Ash St'),

('Laura', 'Wilson', '1993-07-07', 'laura.wilson@example.com', '9638527410', '741 Beech St'),

('Tom', 'Taylor', '1980-01-25', 'tom.taylor@example.com', '1597534862', '258 Willow St'),

('Anna', 'Anderson', '1992-03-14', 'anna.anderson@example.com', '2589631470', '147 Elmwood St');

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

VALUES

(1, 'savings', 15000.00),

(2, 'current', 25000.00),

(3, 'savings', 32000.00),

(4, 'zero\_balance', 0.00),

(5, 'current', 50000.00),

(6, 'savings', 22000.50),

(7, 'current', 30000.00),

(8, 'savings', 12000.75),

(9, 'zero\_balance', 0.00),

(10, 'current', 42000.00);

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

VALUES

(1, 'deposit', 5000.00, '2024-09-01'),

(1, 'withdrawal', 2000.00, '2024-09-05'),

(2, 'deposit', 10000.00, '2024-09-02'),

(3, 'withdrawal', 3000.00, '2024-09-03'),

(4, 'deposit', 4000.00, '2024-09-04'),

(5, 'withdrawal', 15000.00, '2024-09-05'),

(6, 'deposit', 2500.50, '2024-09-06'),

(7, 'deposit', 5000.00, '2024-09-07'),

(8, 'withdrawal', 1200.75, '2024-09-08'),

(9, 'deposit', 2000.00, '2024-09-09'),

(10, 'withdrawal', 4000.00, '2024-09-10');

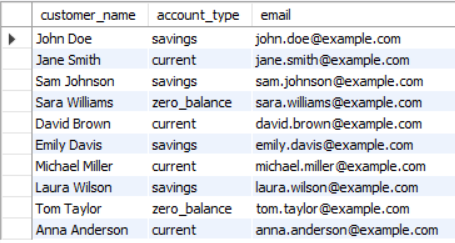
2) Write SQL queries for the following tasks:

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

select 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;



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

select concat(c.first\_name, ' ', c.last\_name) as customer\_name,

t.transaction\_id,

t.transaction\_type,

t.amount,

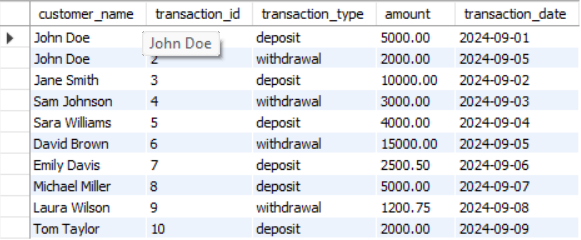
t.transaction\_date

from customers c join accounts a

on c.customer\_id=a.customer\_id

join transactions t

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



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

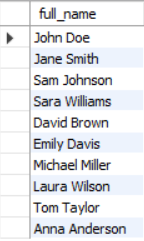
update accounts

set balance=balance + 10000

where account\_id=6;

1. **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;

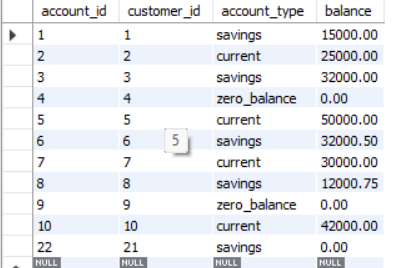


1. **Write a SQL query to remove accounts with a balance of zero where the account type is savings.**

DELETE FROM Accounts

WHERE balance = 0

AND account\_type = 'savings';

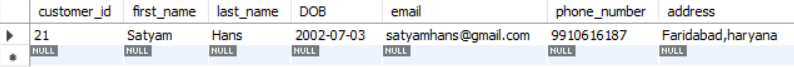
****

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

SELECT \*

FROM Customers

WHERE address LIKE '%Faridabad%';



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

select balance from accounts

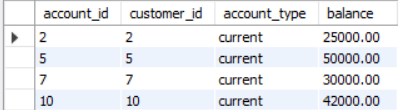
where customer\_id=3;



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

select \* from accounts

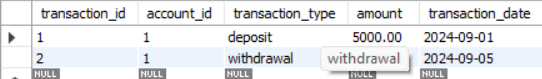
where account\_type='current' and balance > '10000';



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

select \* from transactions

where account\_id=1;



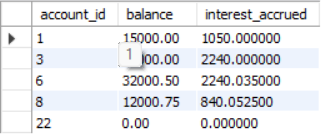
**10. Write a SQL query to Calculate the interest accrued on savings accounts based on a given interest rate.**

SELECT account\_id, balance,

(balance \* 7 / 100) AS interest\_accrued

FROM Accounts

WHERE account\_type = 'savings';

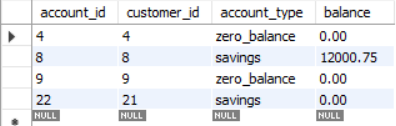


**11. Write a SQL query to Identify accounts where the balance is less than a specified overdraft limit.**

SELECT account\_id, customer\_id, account\_type, balance

FROM Accounts

WHERE balance < 15000;

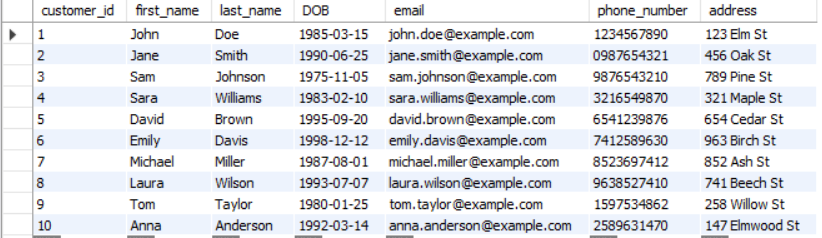


**12. Write a SQL query to Find customers not living in a specific city**

SELECT \*

FROM Customers

WHERE address NOT LIKE '%Faridabad%';



**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;



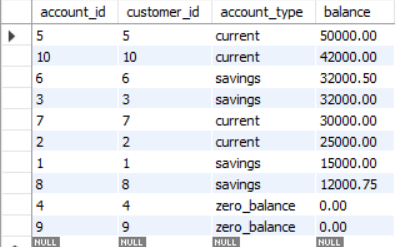
**2. Write a SQL query to Retrieve the top 10 highest account balances.**

SELECT \*

FROM Accounts

ORDER BY balance DESC

LIMIT 10;



**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 transaction\_date = '2024-09-01';



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

SELECT first\_name, last\_name, DOB —--------(oldest)

FROM Customers

ORDER BY DOB ASC

LIMIT 1;



SELECT first\_name, last\_name, DOB—---------(newest)

FROM Customers

ORDER BY DOB DESC

LIMIT 1;

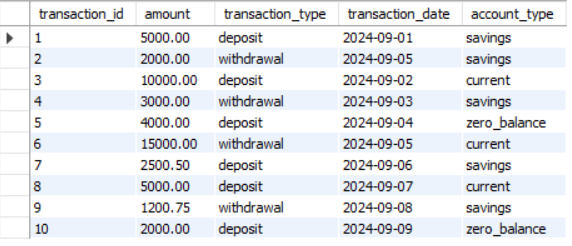


**5. Write a SQL query to Retrieve transaction details along with the account type**.

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

FROM Transactions t

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

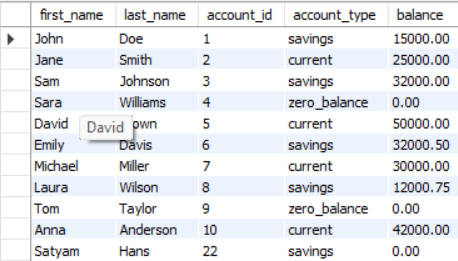


**6. Write a SQL query to Get a list of customers along with their account details.**

SELECT 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.\*, 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 = 5;



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

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

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.

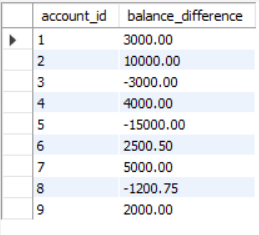
SELECT account\_id,

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, AVG(balance) AS average\_daily\_balance

FROM Transactions

WHERE transaction\_date BETWEEN 'YYYY-MM-DD' AND 'YYYY-MM-DD'

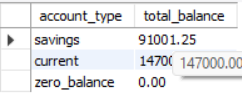
GROUP BY account\_id;

11. Calculate the total balance for each account type.

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

FROM Accounts

GROUP BY account\_type;



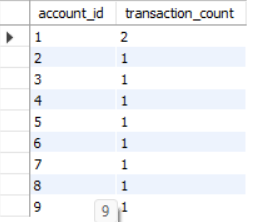
12. Identify accounts with the highest number of transactions order by descending order.

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.

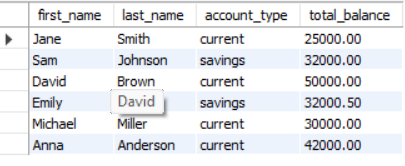
SELECT c.first\_name, c.last\_name, a.account\_type, SUM(a.balance) AS total\_balance

FROM Customers c

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

GROUP BY c.customer\_id, a.account\_type

HAVING SUM(a.balance) > 20000;



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

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

FROM Transactions

GROUP BY account\_id, transaction\_date, amount

HAVING COUNT(\*) > 1;

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

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

SELECT first\_name, last\_name, balance

FROM Customers c

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

WHERE 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

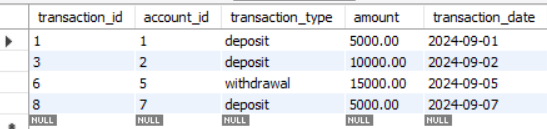
HAVING COUNT(account\_id) > 1;



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

FROM Transactions t

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



4. Identify customers who have no recorded transactions.

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

FROM Customers

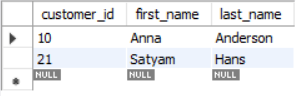
WHERE customer\_id NOT IN (

SELECT DISTINCT customer\_id

FROM Accounts

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

);



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

SELECT SUM(balance) AS total\_balance

FROM Accounts

WHERE account\_id NOT IN (SELECT DISTINCT account\_id FROM Transactions);

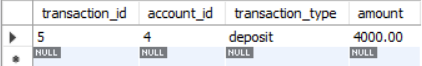


6. Retrieve transactions for accounts with the lowest balance.

SELECT transaction\_id, account\_id, transaction\_type, amount

FROM Transactions

WHERE account\_id = (SELECT account\_id FROM Accounts ORDER BY balance ASC LIMIT 1);



7. Identify customers who have accounts of multiple types.

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

FROM Customers

WHERE customer\_id IN (

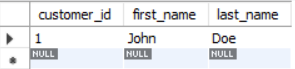
SELECT customer\_id

FROM Accounts

GROUP BY customer\_id

HAVING COUNT(DISTINCT account\_type) > 1

);



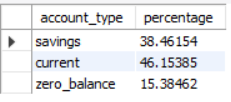
8. Calculate the percentage of each account type out of the total number of accounts.

SELECT account\_type,

(COUNT(\*) \* 100.0) / (SELECT COUNT(\*) FROM Accounts) AS percentage

FROM Accounts

GROUP BY account\_type;



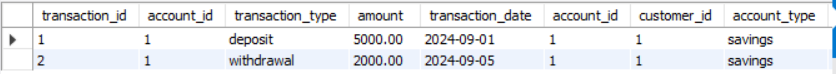
9. Retrieve all transactions for a customer with a given customer\_id.

SELECT \*

FROM Transactions

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

WHERE Accounts.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;

