**INDIVIDUAL ASSIGNMENT**

**BANKING SYSTEM**

**Task 1: Database Design:**

1. Create the database named "HMBank"

* CREATE DATABASE HMBank;

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

* Customers Table

customer\_id (Primary Key)

first\_name

last\_name

DOB

email

phone\_number

address

Accounts Table

account\_id (Primary Key)

customer\_id (Foreign Key)

account\_type

balance

Transactions Table

transaction\_id (Primary Key)

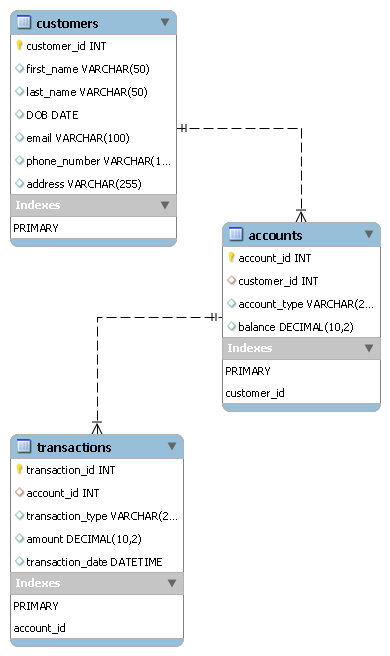
account\_id (Foreign Key)

transaction\_type

amount

transaction\_date

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



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

Customers

customer\_id is the Primary Key.

Accounts

account\_id is the Primary Key.

customer\_id is a Foreign Key referencing Customers(customer\_id).

Transactions

transaction\_id is the Primary Key.

account\_id is a Foreign Key referencing Accounts(account\_id).

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

• Customers

• Accounts

• Transactions

Use HMBank;

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 (100),

phone\_number VARCHAR (15) NOT NULL,

address VARCHAR (255)

);

CREATE TABLE Accounts (

account\_id INT PRIMARY KEY,

customer\_id INT NOT NULL,

account\_type VARCHAR (20) NOT NULL,

balance DECIMAL (10, 2) NOT NULL DEFAULT 0.00,

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

);

CREATE TABLE Transactions (

transaction\_id INT PRIMARY KEY,

account\_id INT NOT NULL,

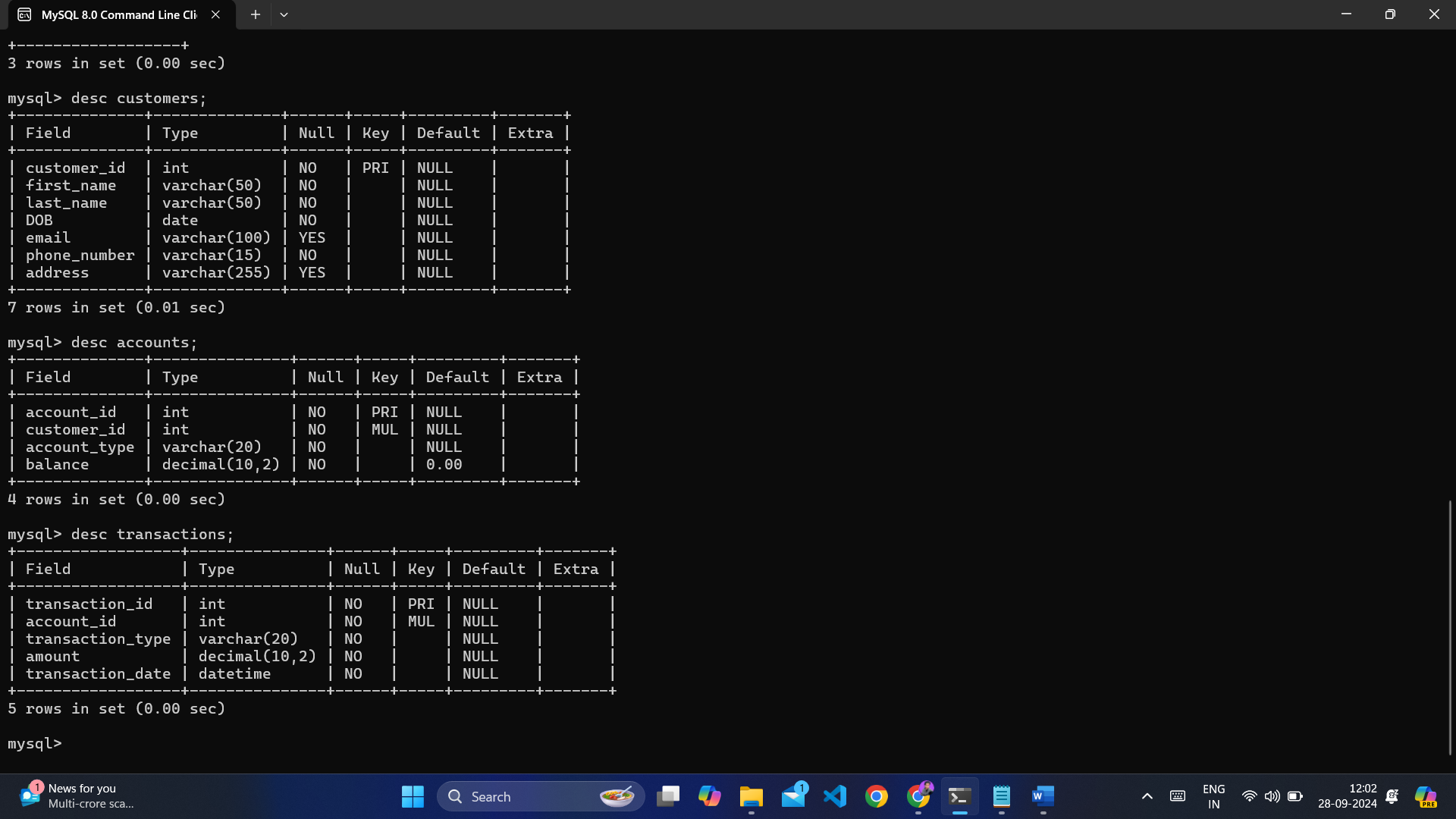
transaction\_type VARCHAR (20) NOT NULL,

amount DECIMAL (10, 2) NOT NULL,

transaction\_date DATETIME NOT NULL,

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

);



**Task 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, 'Alzeb', 'Khan', '2001-06-24', 'alzebkhan@gmail.com', '9815515165', 'Jhansi'),

(2, 'Kartikey', 'Rajpoot', '2000-08-20', 'kartikey@gmail.com', '7054885265', 'Mumbai'),

(3, 'Aditya', 'Singh', '1999-12-05', 'aditya.singh@gmail.com', '9776543210', 'Bangalore'),

(4, 'Anaya', 'Kaur', '1980-02-15', 'anaya.kaur@gmail.com', '9765432109', 'Chennai'),

(5, 'Diya', 'Patel', '1995-03-25', 'diya.patel@gmail.com', '9654321098', 'Hyderabad'),

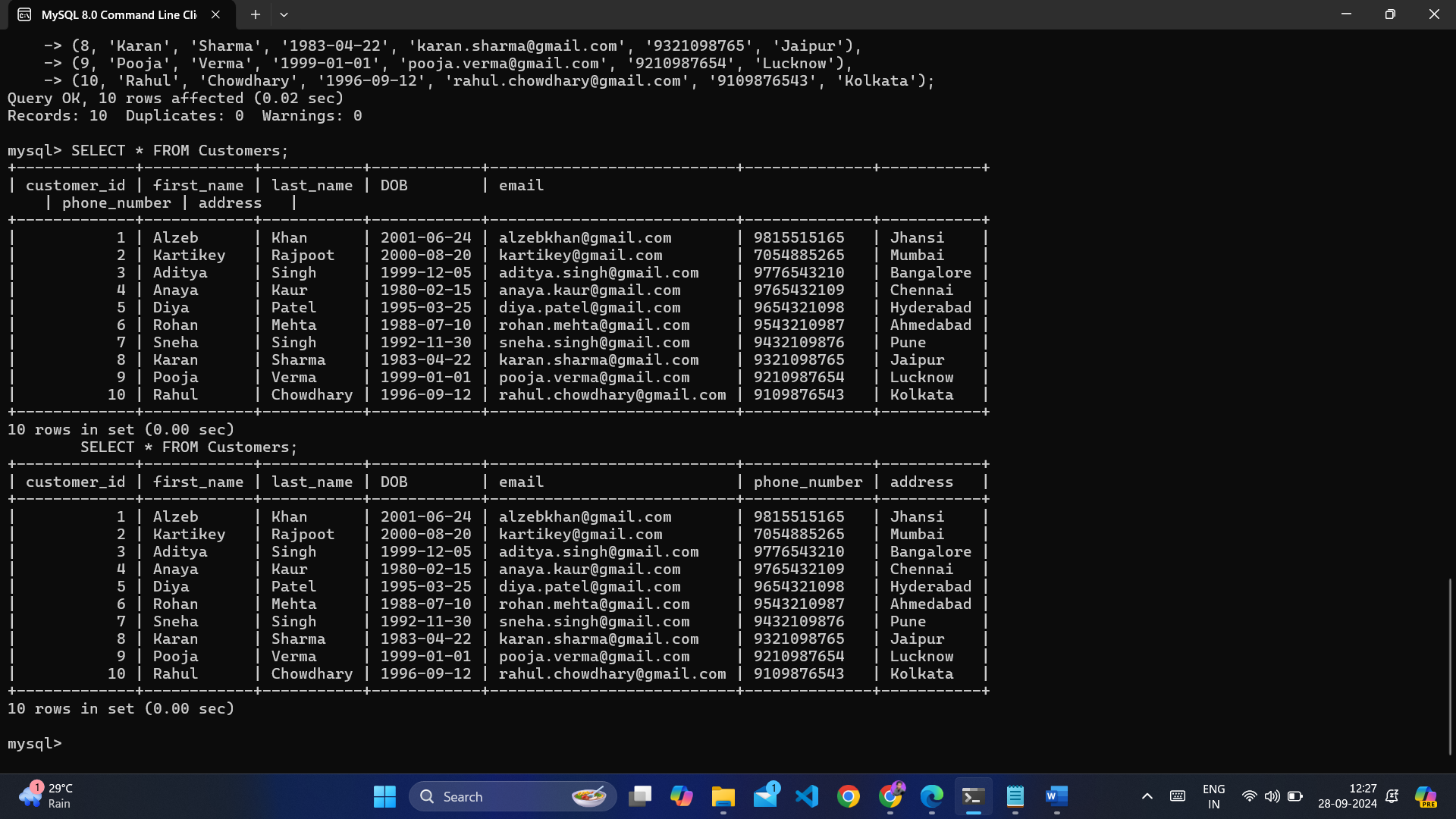
(6, 'Rohan', 'Mehta', '1988-07-10', 'rohan.mehta@gmail.com', '9543210987', 'Ahmedabad'),

(7, 'Sneha', 'Singh', '1992-11-30', 'sneha.singh@gmail.com', '9432109876', 'Pune'),

(8, 'Karan', 'Sharma', '1983-04-22', 'karan.sharma@gmail.com', '9321098765', 'Jaipur'),

(9, 'Pooja', 'Verma', '1999-01-01', 'pooja.verma@gmail.com', '9210987654', 'Lucknow'),

(10, 'Rahul', 'Chowdhary', '1996-09-12', 'rahul.chowdhary@gmail.com', '9109876543', 'Kolkata');



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

(101, 1, 'savings', 1500.00),

(102, 2, 'current', 2500.00),

(103, 3, 'savings', 0.00),

(104, 4, 'zero\_balance', 0.00),

(105, 5, 'savings', 300.00),

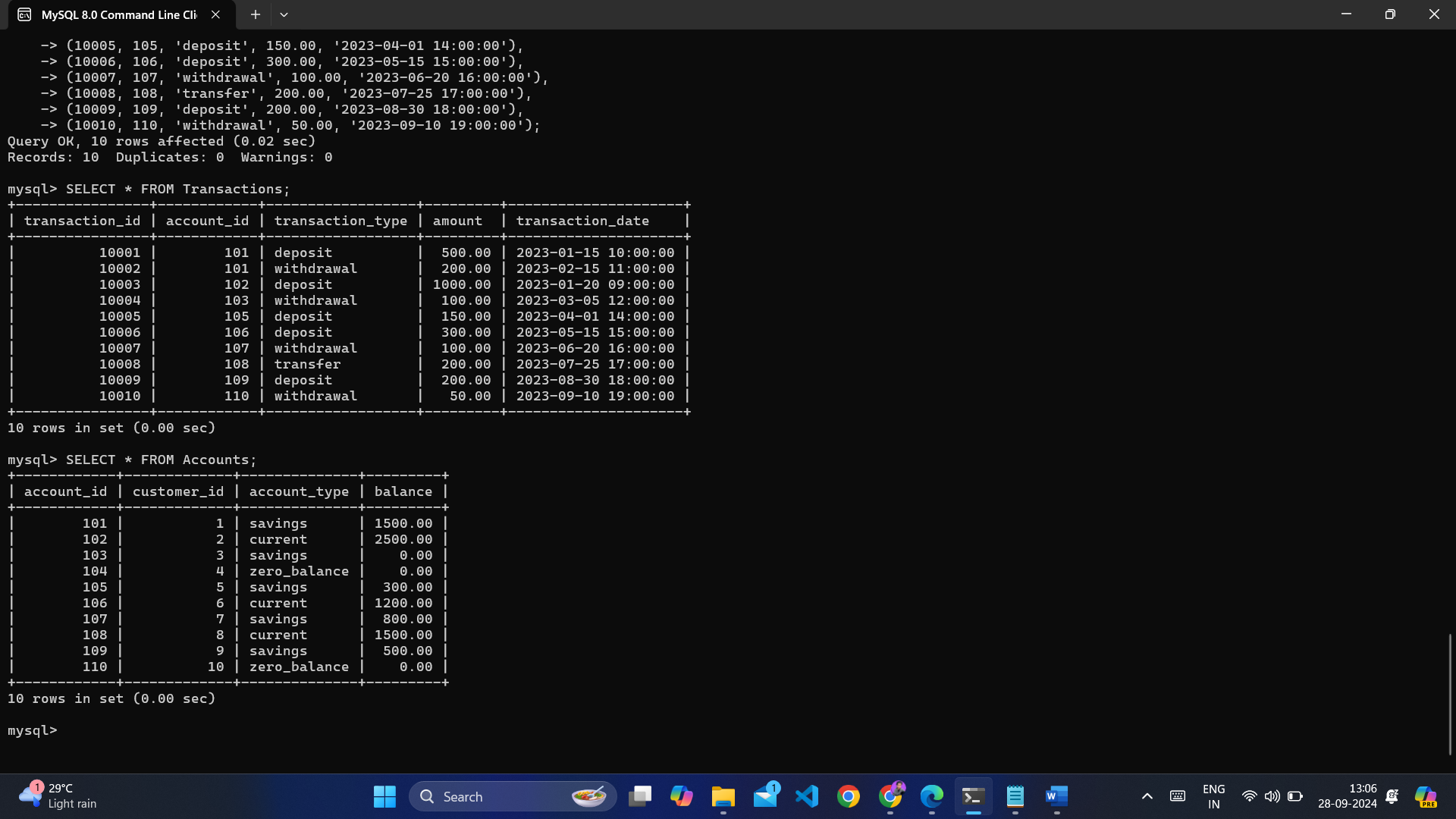
(106, 6, 'current', 1200.00),

(107, 7, 'savings', 800.00),

(108, 8, 'current', 1500.00),

(109, 9, 'savings', 500.00),

(110, 10, 'zero\_balance', 0.00);



.

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

(10001, 101, 'deposit', 500.00, '2023-01-15 10:00:00'),

(10002, 101, 'withdrawal', 200.00, '2023-02-15 11:00:00'),

(10003, 102, 'deposit', 1000.00, '2023-01-20 09:00:00'),

(10004, 103, 'withdrawal', 100.00, '2023-03-05 12:00:00'),

(10005, 105, 'deposit', 150.00, '2023-04-01 14:00:00'),

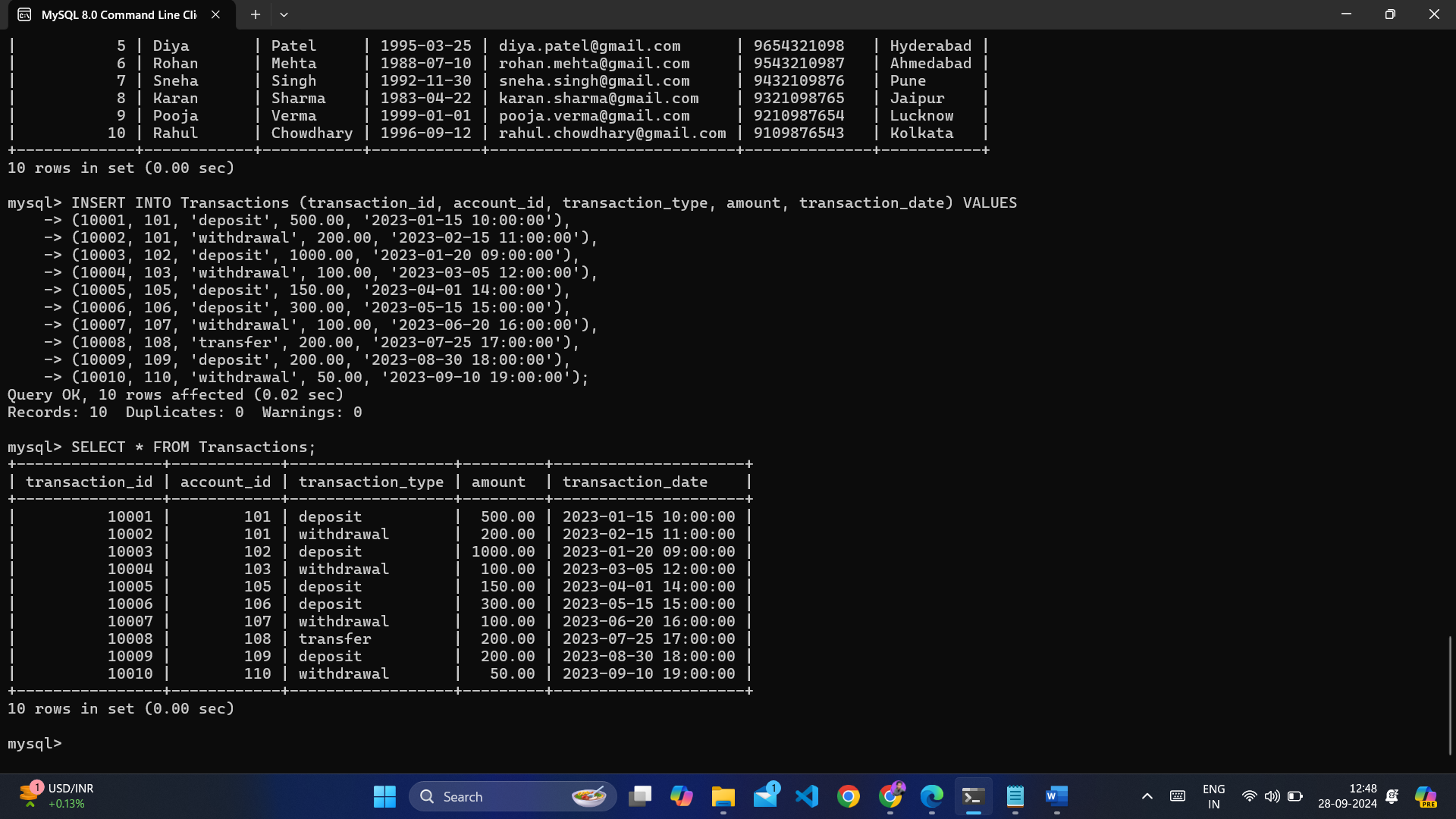
(10006, 106, 'deposit', 300.00, '2023-05-15 15:00:00'),

(10007, 107, 'withdrawal', 100.00, '2023-06-20 16:00:00'),

(10008, 108, 'transfer', 200.00, '2023-07-25 17:00:00'),

(10009, 109, 'deposit', 200.00, '2023-08-30 18:00:00'),

(10010, 110, 'withdrawal', 50.00, '2023-09-10 19:00:00');



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 first\_name, last\_name, account\_type, email

FROM Customers, Accounts

WHERE Customers.customer\_id = Accounts.customer\_id;

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

SELECT first\_name, last\_name, transaction\_type, amount, transaction\_date

FROM Customers, Accounts, Transactions

WHERE Customers.customer\_id = Accounts.customer\_id

AND Accounts.account\_id = Transactions.account\_id;

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

UPDATE Accounts

SET balance = balance + 100.00

WHERE account\_id = 101;

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 first\_name, last\_name, address FROM Customers

WHERE address LIKE '%Jhansi%';

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

SELECT account\_id, balance FROM Accounts

WHERE account\_id = 101;

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 = 101;

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

SELECT account\_id, (balance \* 0.03) 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 < -100.00;

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

SELECT \* FROM Customers

WHERE address NOT LIKE '%Jhansi%';

**Task 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 account\_id, balance FROM Accounts

ORDER BY balance DESC LIMIT 10;

1. 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) = '2023-01-15';

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

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

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

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

1. 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

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

FROM Customers c

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

GROUP BY c.customer\_id

HAVING COUNT(a.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 = 'withdrawal') AS difference;

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

?

1. 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 a.account\_id, COUNT(t.transaction\_id) AS transaction\_count

FROM Accounts a

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

GROUP BY a.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, 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 total\_balance > 1000;

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

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

FROM Transactions t

GROUP BY t.account\_id, t.amount, t.transaction\_date

HAVING transaction\_count > 1

ORDER BY t.account\_id, t.transaction\_date;

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

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

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

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

4. Identify customers who have no recorded transactions.

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

6. Retrieve transactions for accounts with the lowest balance.

7. Identify customers who have accounts of multiple types.

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

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

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