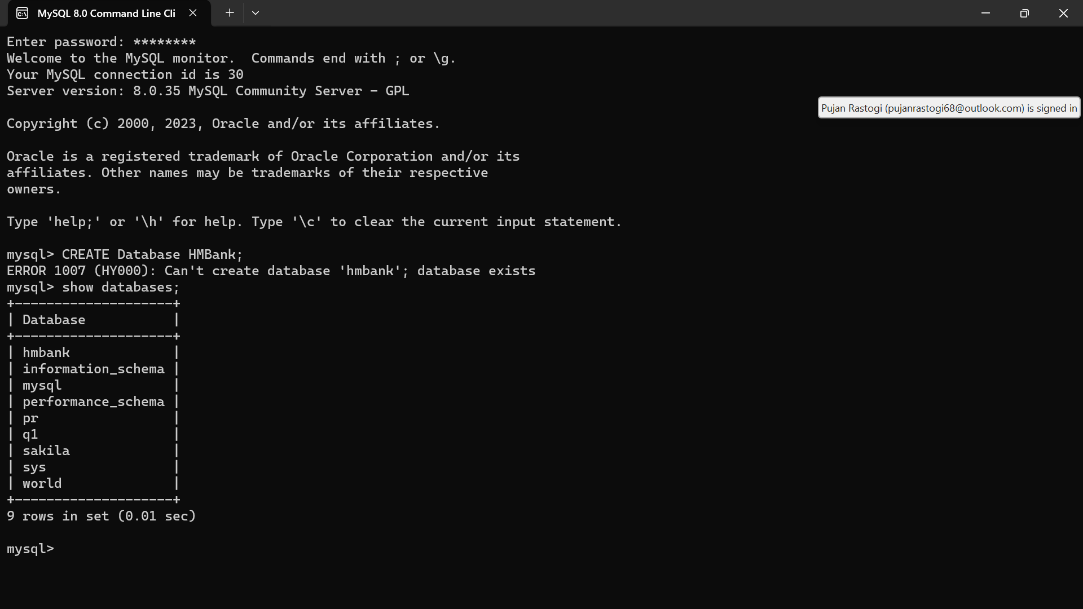
**Task 1** **Database Design:**

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.**

Schema for Customer:

CREATE table Customers(

customer\_id INT PRIMARY KEY,

first\_name VARCHAR(100),

last\_name VARCHAR(100),

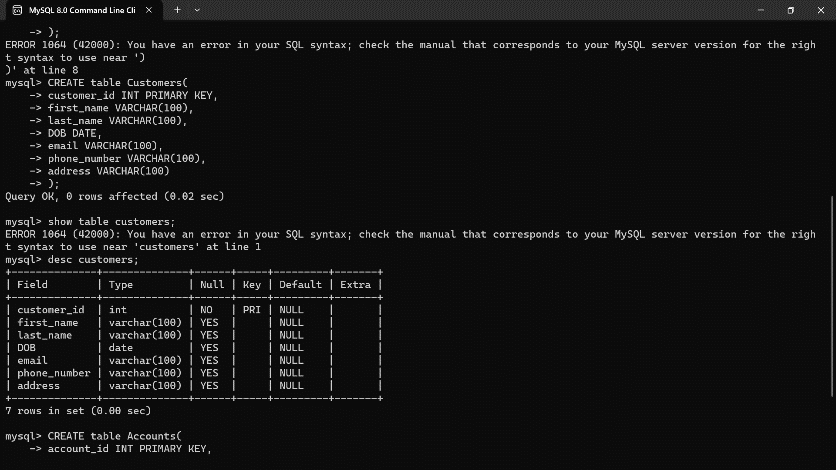
DOB DATE,

email VARCHAR(100),

phone\_number VARCHAR(100),

address VARCHAR(100)

);



Schema for Accounts:

CREATE table Accounts(

account\_id INT PRIMARY KEY,

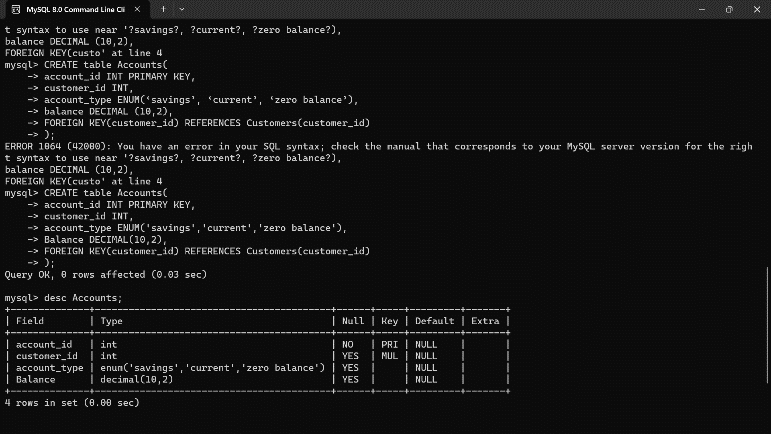
customer\_id INT,

account\_type ENUM(‘savings’, ‘current’, ‘zero balance’),

balance DECIMAL (10,2),

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

);



Schema for Transactions:

CREATE table Transactions(

transaction\_id INT PRIMARY KEY,

account\_id INT,

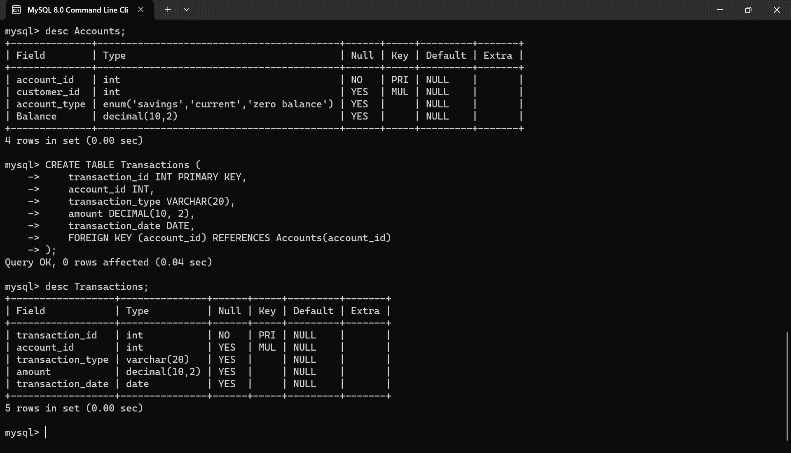
transaction\_type ENUM(‘deposit’, ‘withdrawal’, ‘transfer’),

amount DECIMAL(10,2),

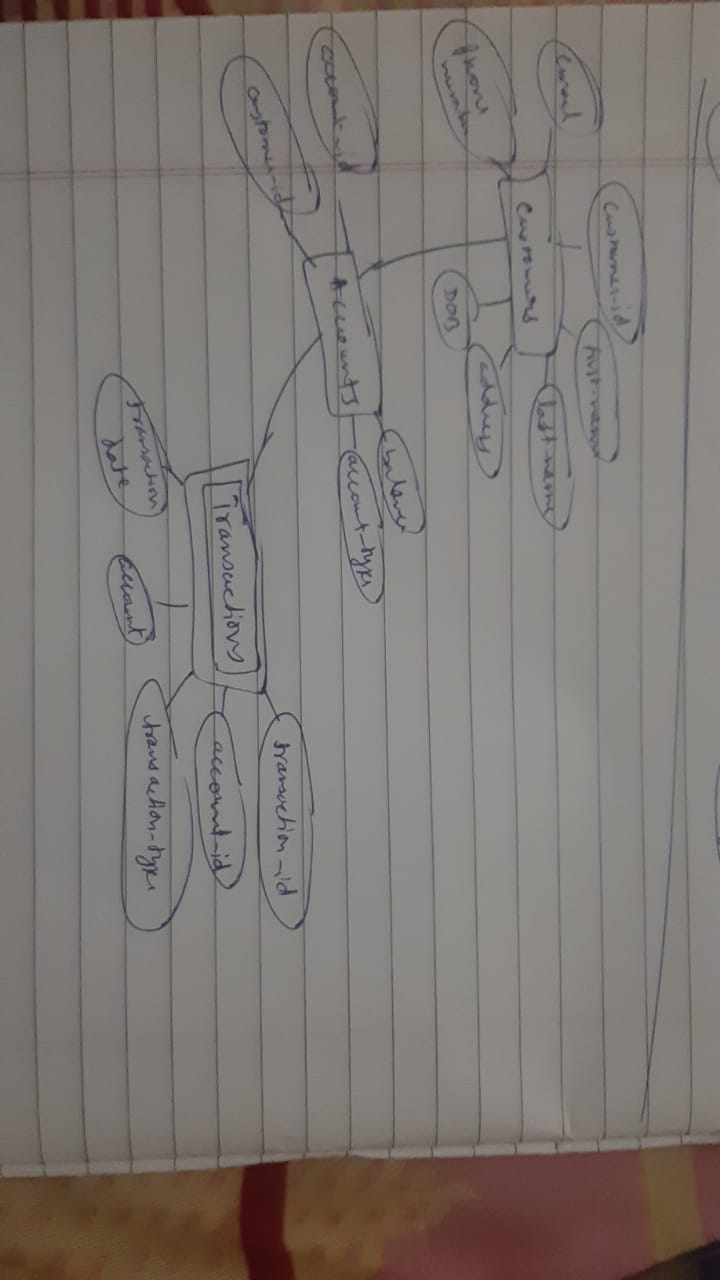
transaction\_date DATE,

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

);



**ER DIAGRAM**



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

For customer table:

Primary Key constraint for Customers table:

ALTER TABLE Customers

ADD CONSTRAINT P\_Customers PRIMARY KEY (customer\_id);

Unique constraint for email in Customers table:

ALTER TABLE Customers

ADD CONSTRAINT U\_Email UNIQUE (email);

For accounts table:

Primary Key constraint for Accounts table:

ALTER TABLE Accounts

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

Foreign Key constraint referencing Customers table:

ALTER TABLE Accounts

ADD CONSTRAINT FK\_Accounts\_Customers FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id);

For transactions table:

Primary Key constraint for Transactions table:

ALTER TABLE Transactions

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

Foreign Key constraint referencing Accounts table:

ALTER TABLE Transactions

ADD CONSTRAINT FK\_Transactions\_Accounts 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**

For Customer table:

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

VALUES

(1, 'Pujan', 'Rastogi', '2002-08-23', 'pujan1@gmail.com', '1234567890', '123 A Street'),

(2, 'Abhi', 'Ag', '2000-09-20', 'abhi@gmail.com', '9876543210', '456 B School'),

(3, 'Abhishek', 'Chaursaiya', '1999-12-10', 'abhich1@gmail.com', '9812347650', '789 C Pass'),

(4, 'Raghav', 'Aggarwal', '1993-01-11', 'raghav@gmail.com', '1237890456', '901 D Road'),

(5, 'Ishan', 'Sharma', '1995-07-17', 'ishan@gmail.com', '2345678901', '232 E Park'),

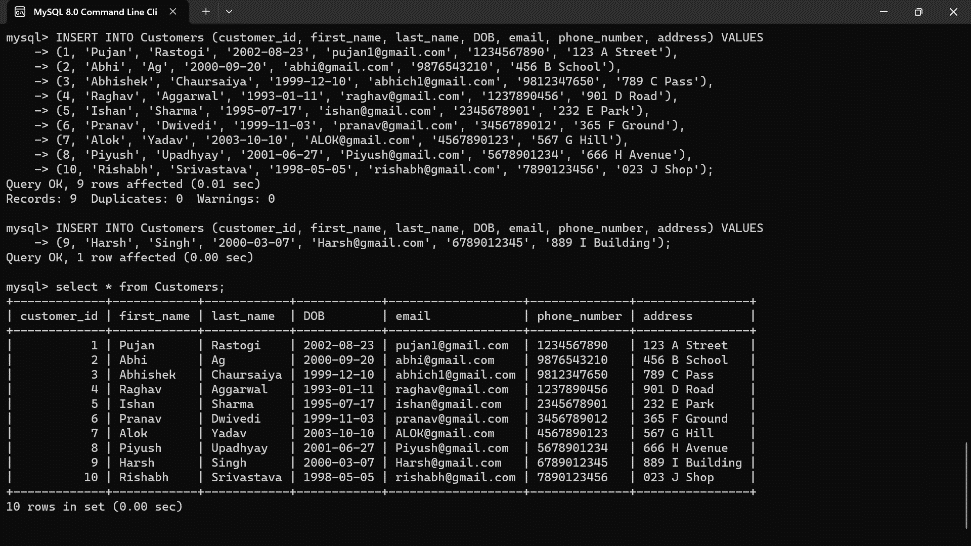
(6, 'Pranav', 'Dwivedi', '1999-11-03', 'pranav@gmail.com', '3456789012', '365 F Ground'),

(7, 'Alok', 'Yadav', '2003-10-10', 'ALOK@gmail.com', '4567890123', '567 G Hill'),

(8, 'Piyush', 'Upadhyay', '2001-06-27', 'Piyush@gmail.com', '5678901234', '666 H Avenue'),

(9, 'Harsh', 'Singh', '2000-03-07', 'Harsh@gmail.com', '6789012345', '889 I Building'),

(10, 'Rishabh', 'Srivastava', '1998-05-05', 'rishabh@gmail.com', '7890123456', '023 J Shop');



For Accounts table:

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

VALUES

(101, 'savings', 50000.00, 1),

(102, 'current', 25000.00, 2),

(103, 'zero balance', 0.00, 3),

(104, ‘savings’, 20000.00,4),

(105, 'current', 75000.00, 5),

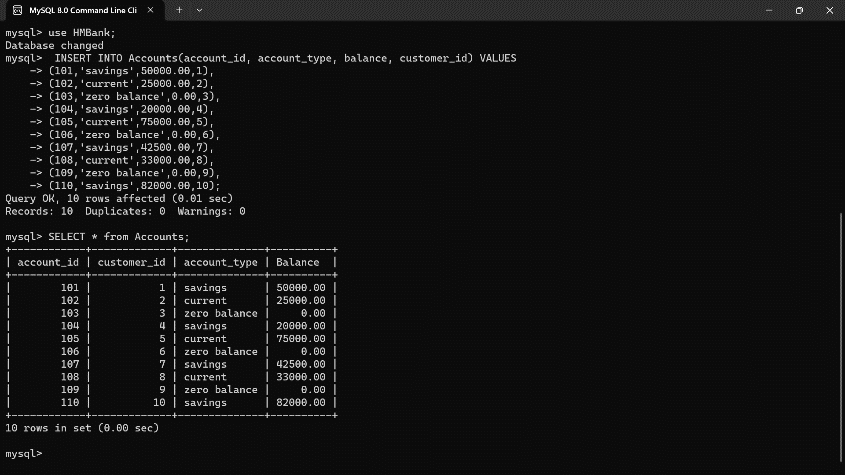
(106, 'zero balance', 0.00, 6),

(107, 'savings', 42500.00, 7),

(108, 'current', 33000.00, 8),

(109, 'zero balance', 0.00, 9),

(110, 'savings', 82000.00, 10);



For Transaction table:

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

VALUES

(1001, 'deposit', 1000.00, '2023-01-10', 101),

(1002, 'withdrawal', 500.00, '2023-02-05', 102),

(1003, 'deposit', 3000.00, '2023-03-15', 103),

(1004, 'withdrawal', 6000.00, '2023-04-25', 104),

(1005, 'deposit', 7000.00, '2023-05-20', 105),

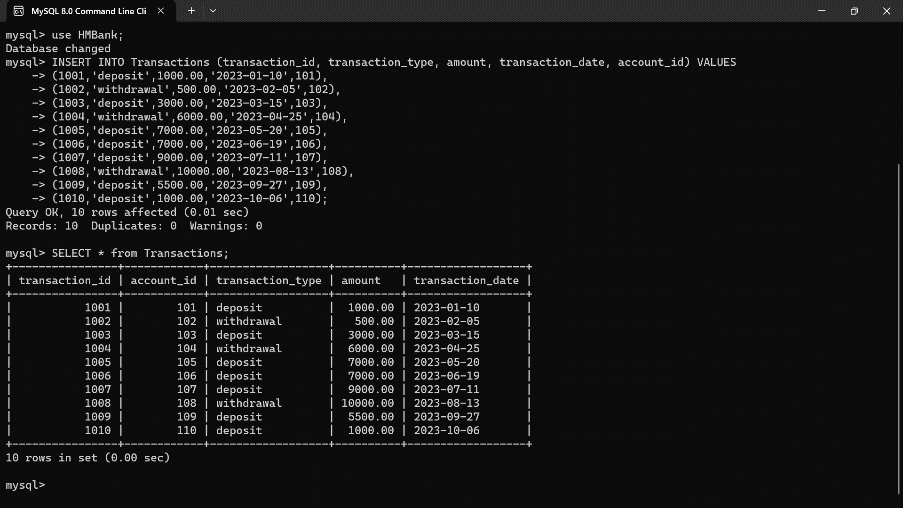
(1006, 'deposit', 7000.00, '2023-06-19', 106),

(1007, 'deposit', 9000.00, '2023-07-11', 107),

(1008, 'withdrawal', 10000.00, '2023-08-13', 108),

(1009, 'deposit', 5500.00, '2023-09-27', 109),

(1010, 'withdrawal', 1000.00, '2023-10-06', 110);

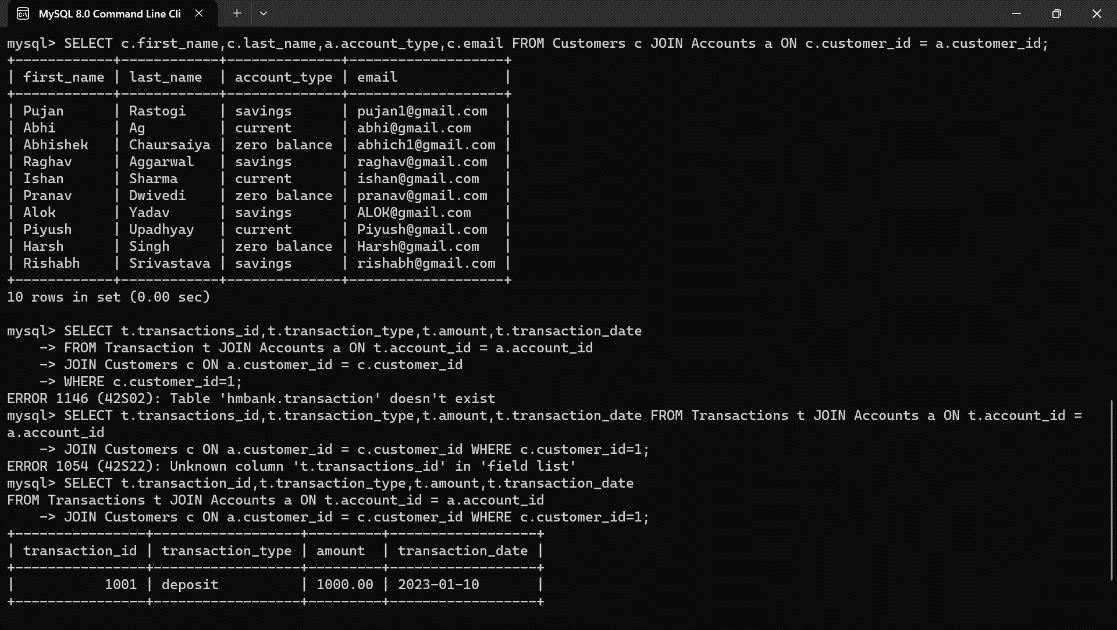


* 1. **Write a SQL query to retrieve the name, account type and email of all customers**

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

FROM Customers c

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



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

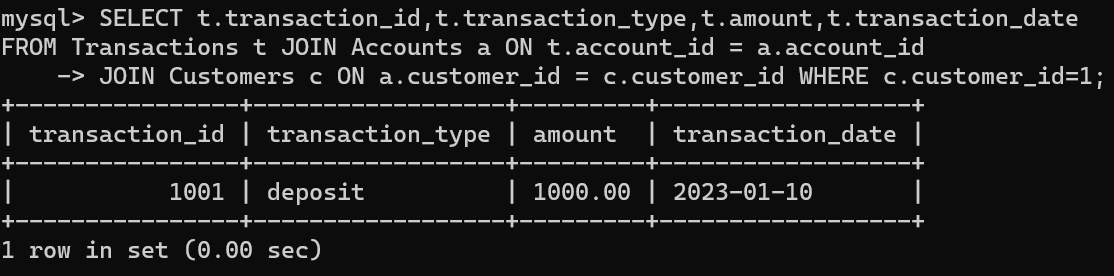
SELECT t.transaction\_id, t.transaction\_type, t.amount, t.transaction\_date

FROM Transactions t

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

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

WHERE c.customer\_id = 1;

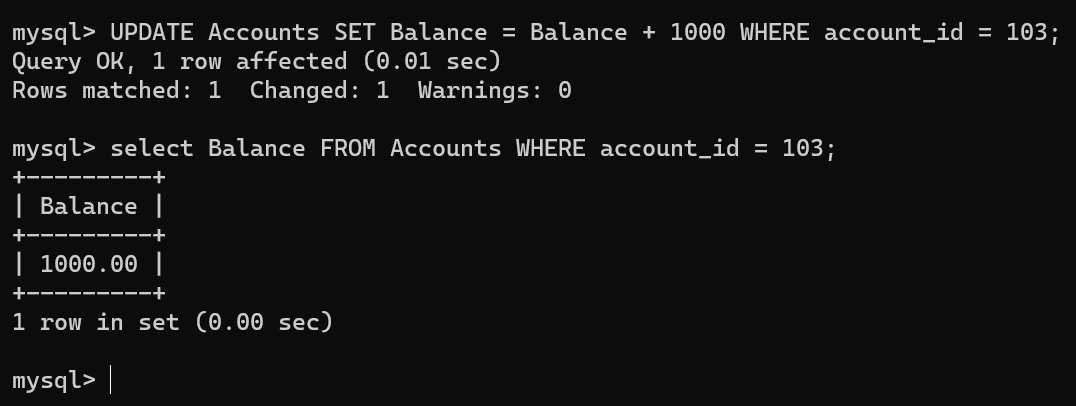


**2.3** **Write a SQL query to increase the balance of a specific account by a certain amount.**

UPDATE Accounts

SET balance = balance + 1000

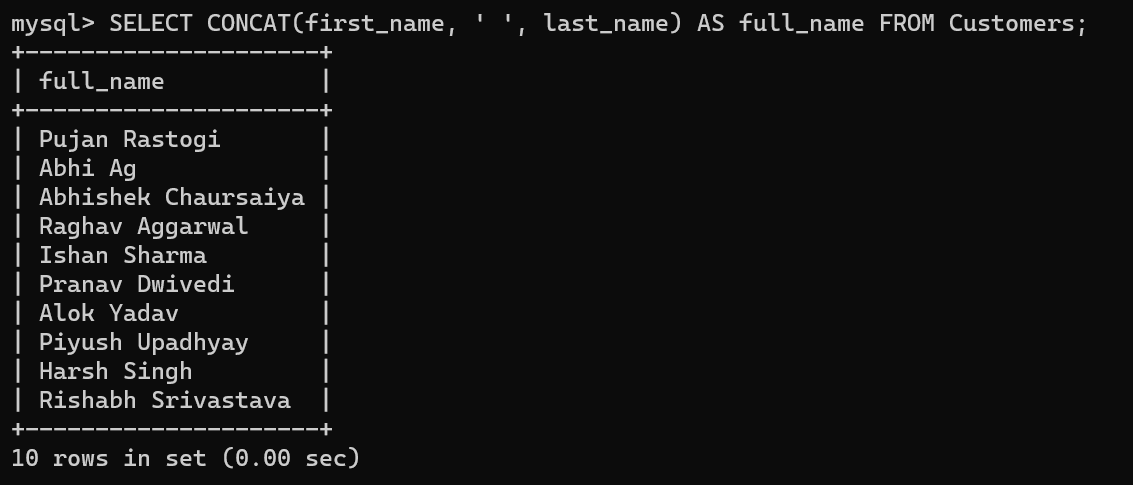
WHERE account\_id = 103;



**2.4 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

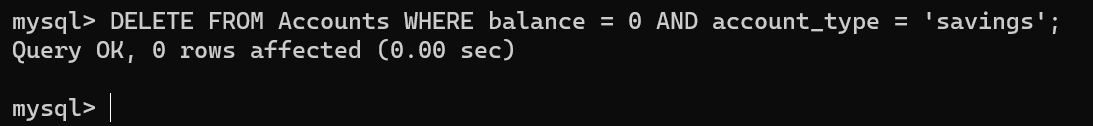
FROM Customers;



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

DELETE FROM Accounts

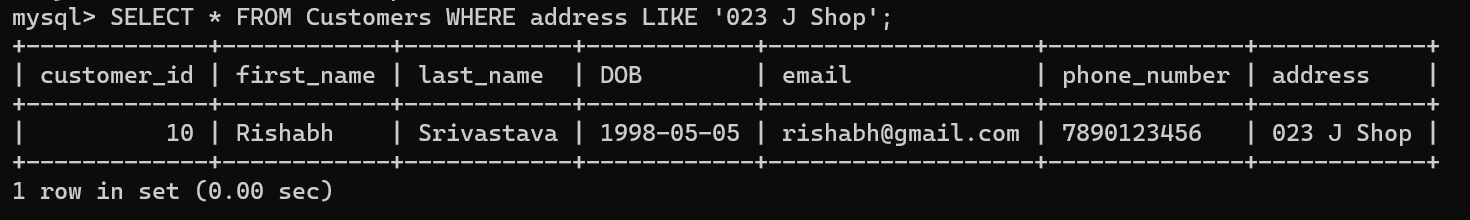
WHERE balance = 0 AND account\_type = 'savings';



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

SELECT \* FROM Customers

WHERE address LIKE ‘023 J Shop’;

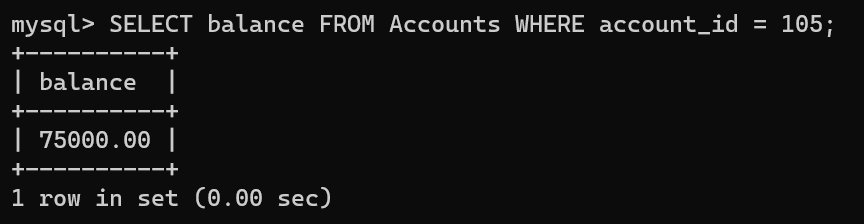


**2.7 Write a SQL query to Get the account balance for a specific account.**

SELECT Balance

FROM Accounts

WHERE account\_id = 105;

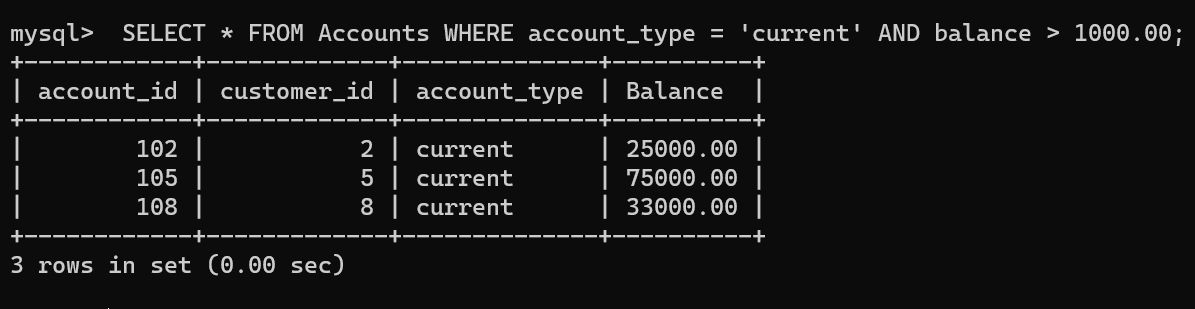


**2.8 Write a SQL query to List all current accounts with a balance greater than $1,000.**

SELECT \*

FROM Accounts

WHERE account\_type = 'current' AND balance > 1000.00;

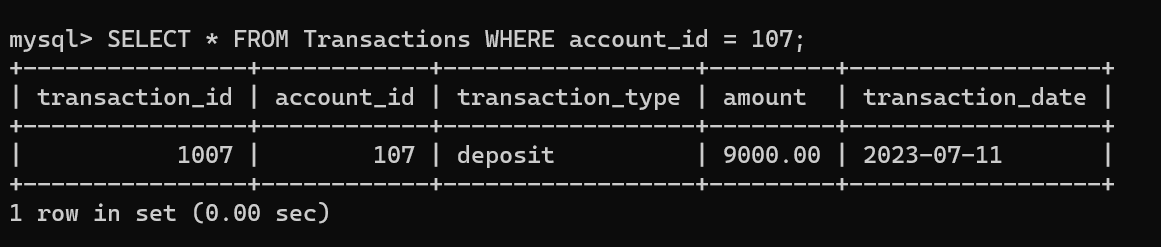


**2.9 Write a SQL query to Retrieve all transactions for a specific account.**

SELECT \*

FROM Transactions

WHERE account\_id = 107;

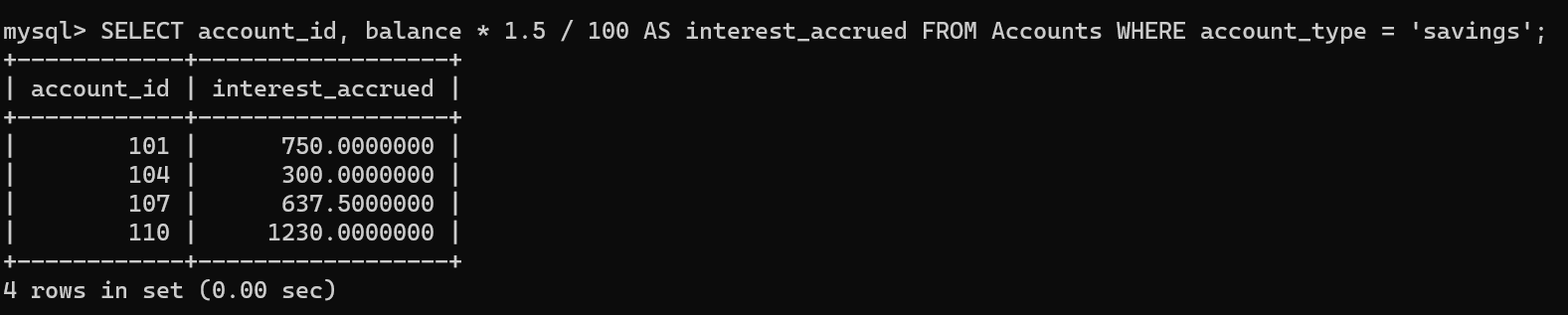


**2.10 Write a SQL query to Calculate the interest accrued on savings accounts based on a given interest rate.**

SELECT account\_id, balance \* 1.5 / 100 AS interest\_accrued

FROM Accounts

WHERE account\_type = 'savings';

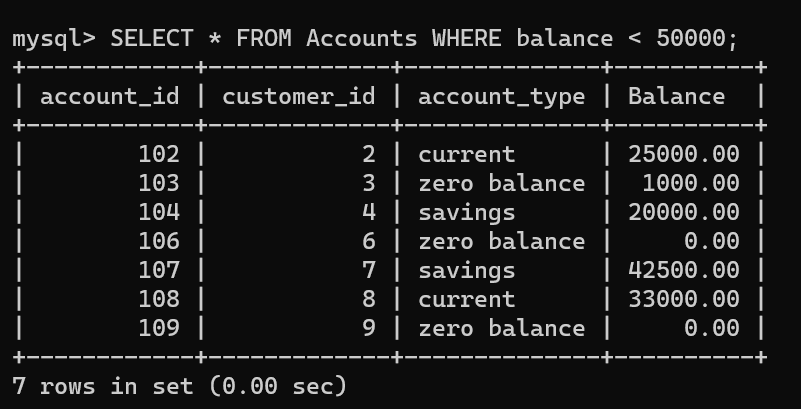


**2.11 . Write a SQL query to Identify accounts where the balance is less than a specified overdraft limit.**

SELECT \*

FROM Accounts

WHERE balance < 50000;

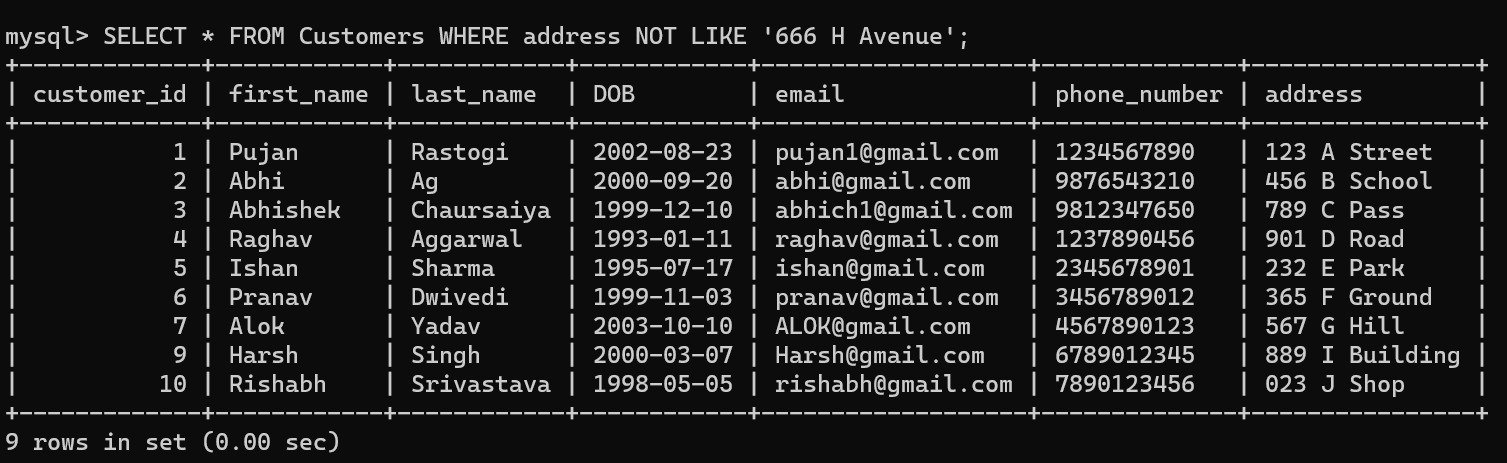


**2.12 Write a SQL query to Find customers not living in a specific city.**

SELECT \*

FROM Customers

WHERE address NOT LIKE 666 H Avenue;

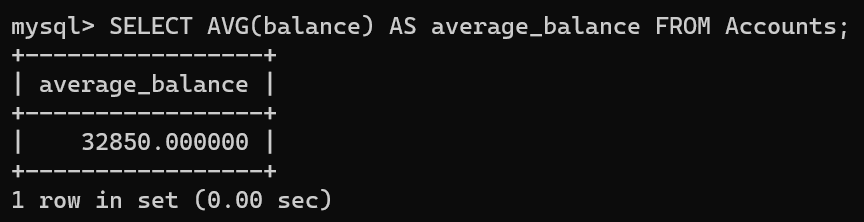


**TASK 3:** **Aggregate functions, Having, Order By, GroupBy and Joins:**

**3.1 Write a SQL query to Find the average account balance for all customers.**

SELECT AVG(balance) AS average\_balance

FROM Accounts;

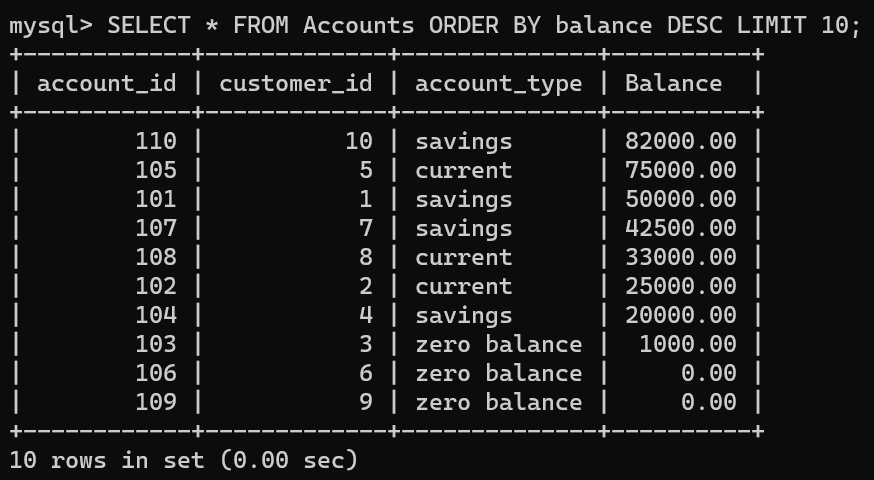


**3.2 Write a SQL query to Retrieve the top 10 highest account balances.**

SELECT \* FROM Accounts

ORDER BY balance DESC

LIMIT 10;

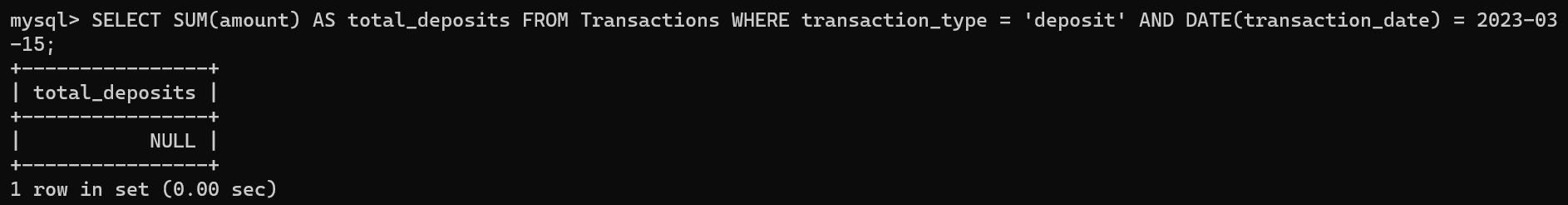


**3.3 Write a SQL query to Calculate Total Deposits for All Customers in specific date.**

SELECT SUM(amount) AS total\_deposits

FROM Transactions

WHERE transaction\_type = 'deposit' AND DATE(transaction\_date) = 2023-03-15;



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

SELECT \*

FROM Customers

ORDER BY DOB ASC /\* or DESC \*/

LIMIT 1; /\*For the oldest\*/

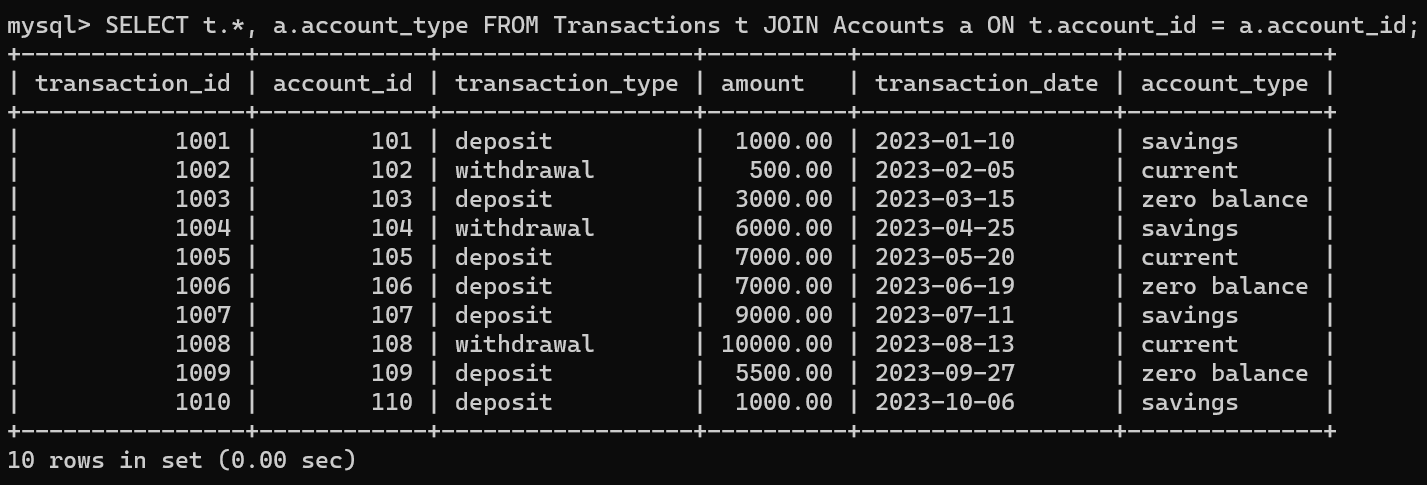


**3.5 Write a SQL query to Retrieve transaction details along with the account type.**

SELECT t.\*, a.account\_type

FROM Transactions t

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

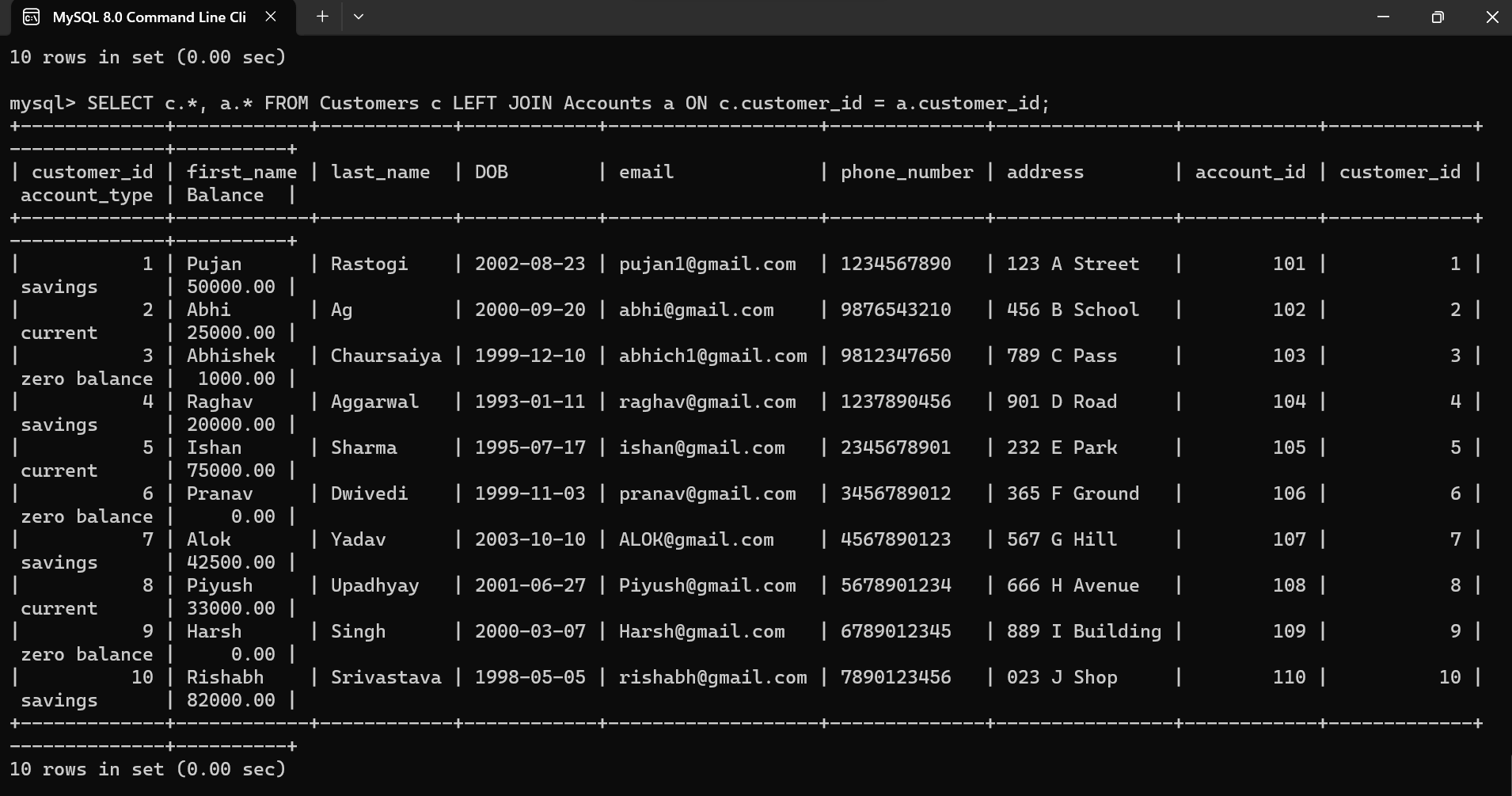


**3.6 Write a SQL query to Get a list of customers along with their account details.**

SELECT c.\*, a.\*

FROM Customers c

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



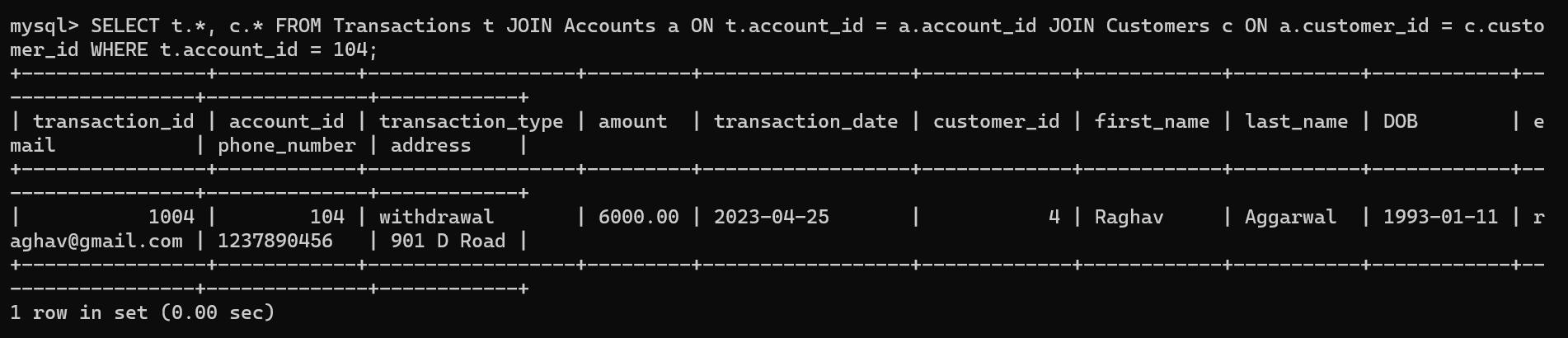
**3.7 Write a SQL query to Retrieve transaction details along with customer information for a specific account.**

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 t.account\_id = 104;



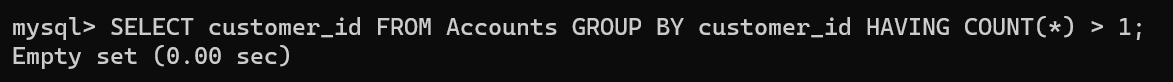
**3.8 Write a SQL query to Identify customers who have more than one account.**

SELECT customer\_id

FROM Accounts

GROUP BY customer\_id

HAVING COUNT(\*) > 1;



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

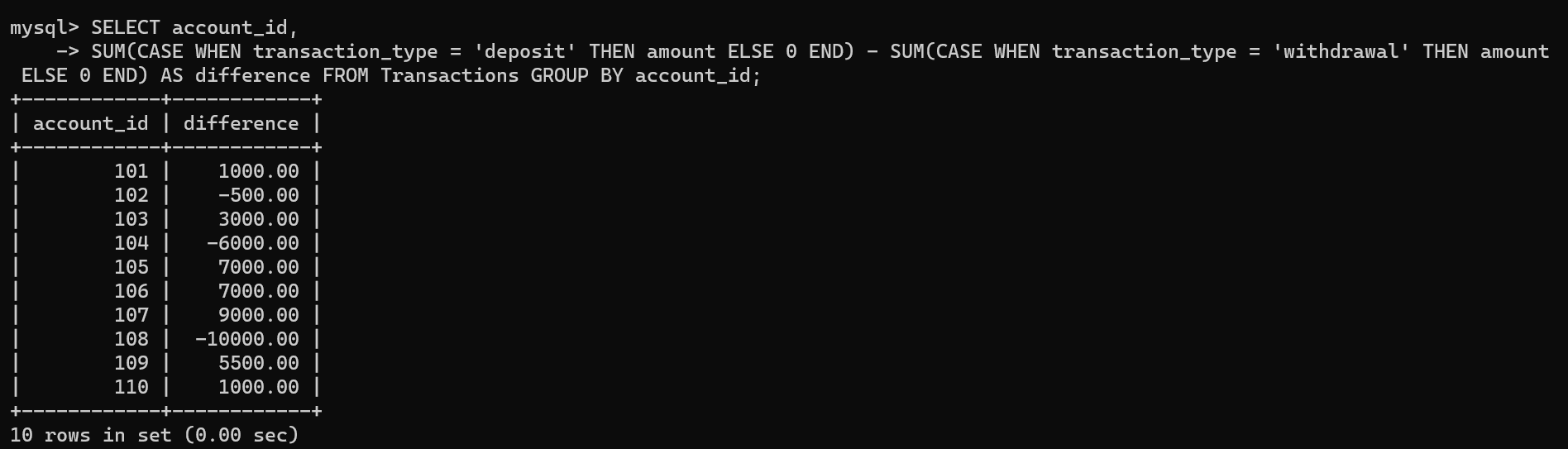
SELECT account\_id,

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

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

FROM Transactions

GROUP BY account\_id;



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

SELECT transaction\_id,

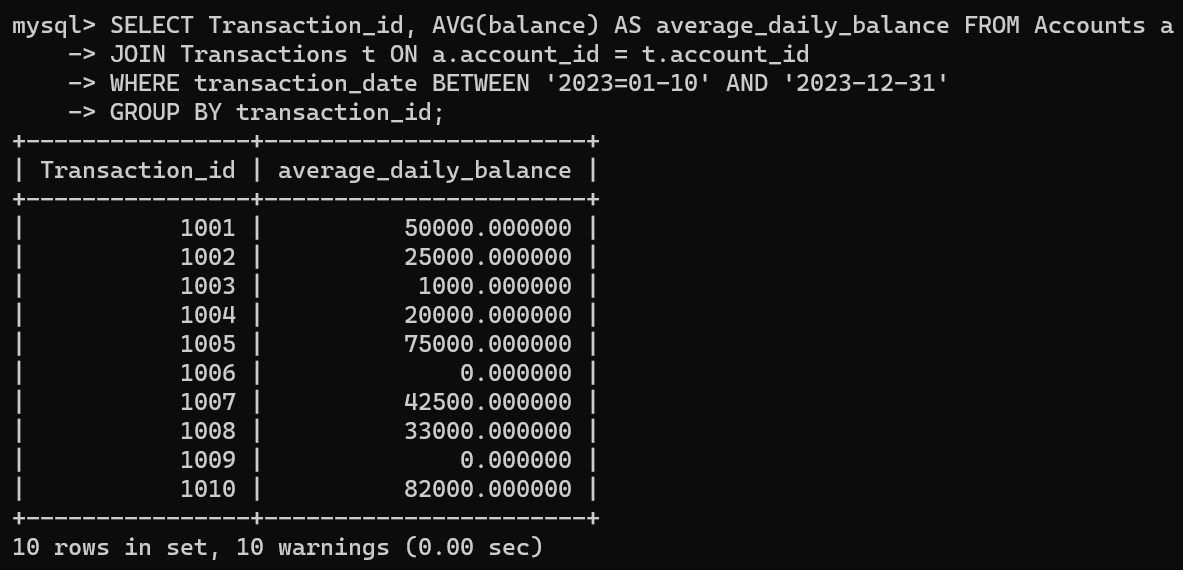
AVG(balance) AS average\_daily\_balance

FROM Accounts a

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

WHERE transaction\_date BETWEEN ‘2023=01-10’ AND ‘2023-12-31’

GROUP BY transaction\_id;

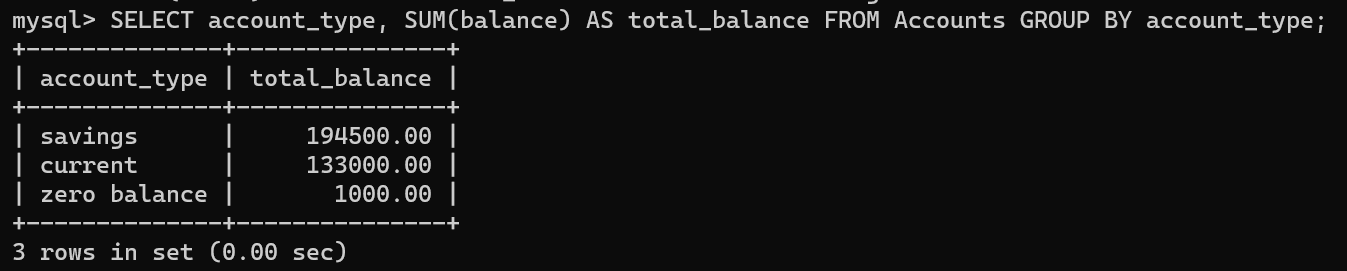


**3.11 Calculate the total balance for each account type.**

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

FROM Accounts

GROUP BY account\_type;



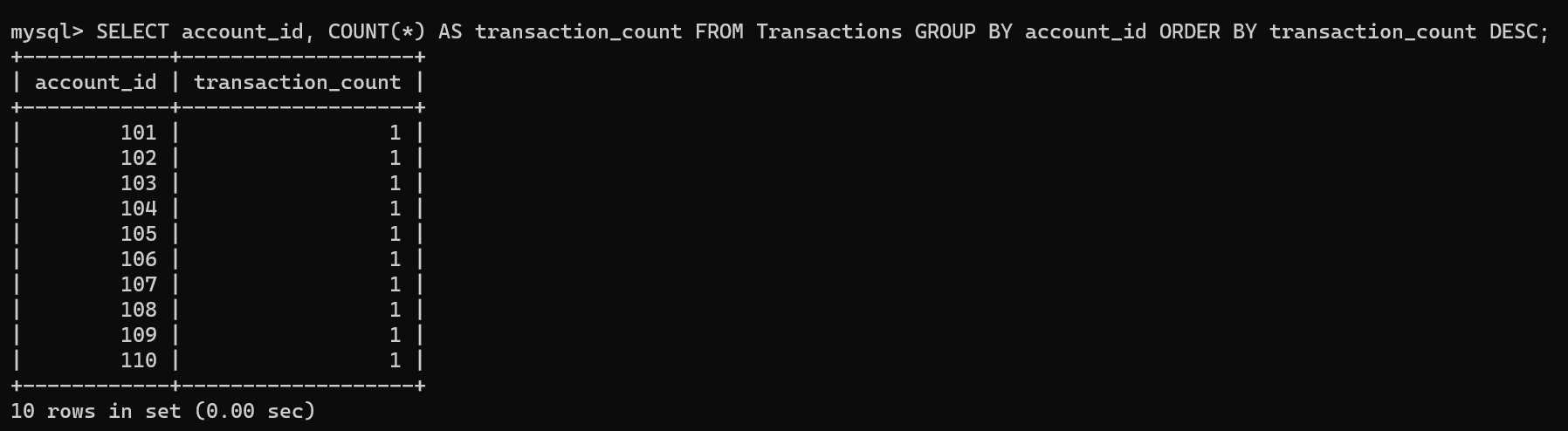
**3.12 Identify accounts with the highest number of transactions order by descending order.**

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

FROM Transactions

GROUP BY account\_id

ORDER BY transaction\_count DESC;



**3.13 List customers with high aggregate account balances, along with their account types.**

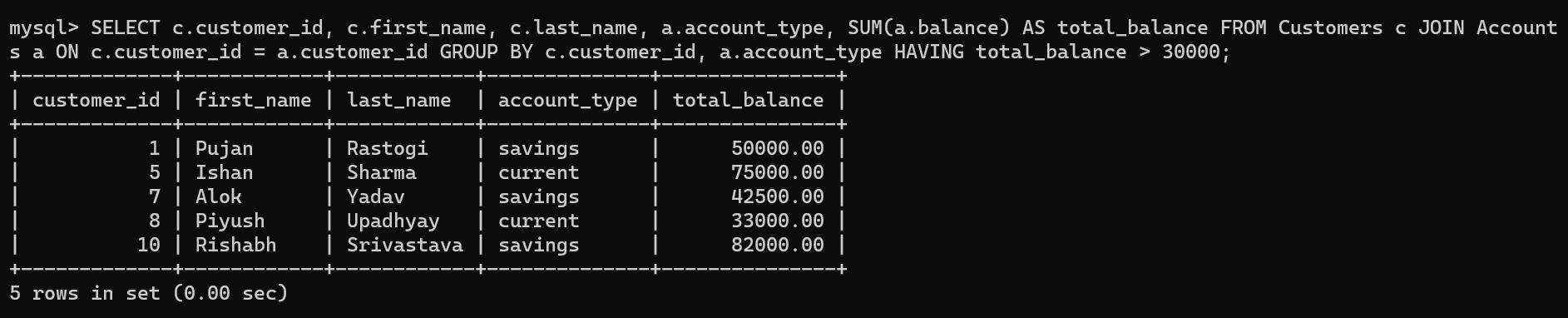
SELECT c.customer\_id, c.first\_name, c.last\_name, a.account\_type, SUM(a.balance) AS total\_balance

FROM Customers c

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

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

HAVING total\_balance > 30000;



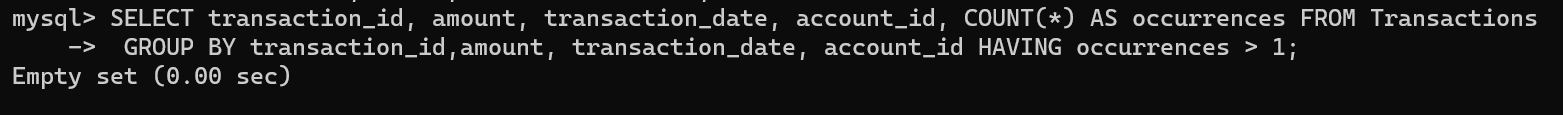
**3.14 Identify and list duplicate transactions based on transaction amount, date, and account.**

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

FROM Transactions

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

HAVING occurrences > 1;



**TASK 4: Subquery and its type:**

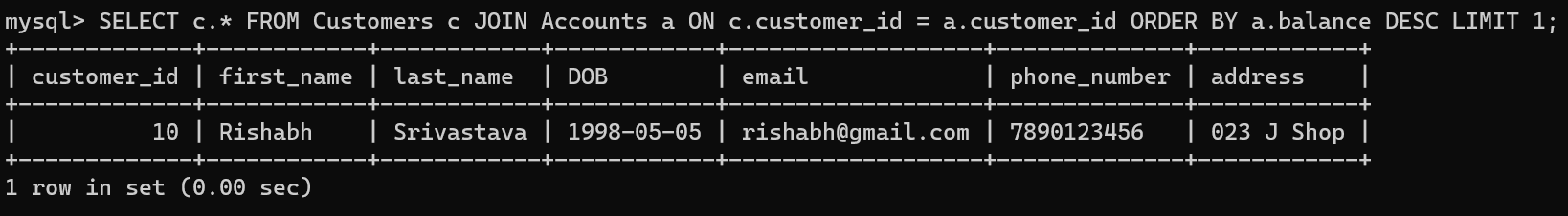
**4.1 Retrieve the customer(s) with the highest account balance.**

SELECT c.\* FROM Customers c

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

ORDER BY a.balance DESC

LIMIT 1;

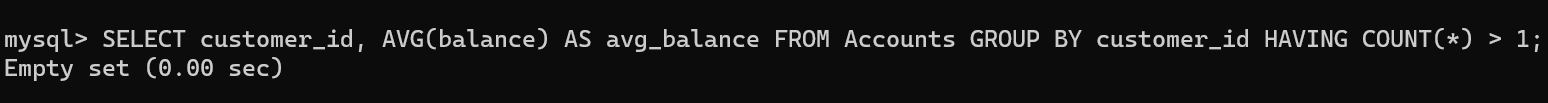


**4.2 Calculate the average account balance for customers who have more than one account.**

SELECT customer\_id, AVG(balance) AS avg\_balance FROM Accounts

GROUP BY customer\_id

HAVING COUNT(\*) > 1;

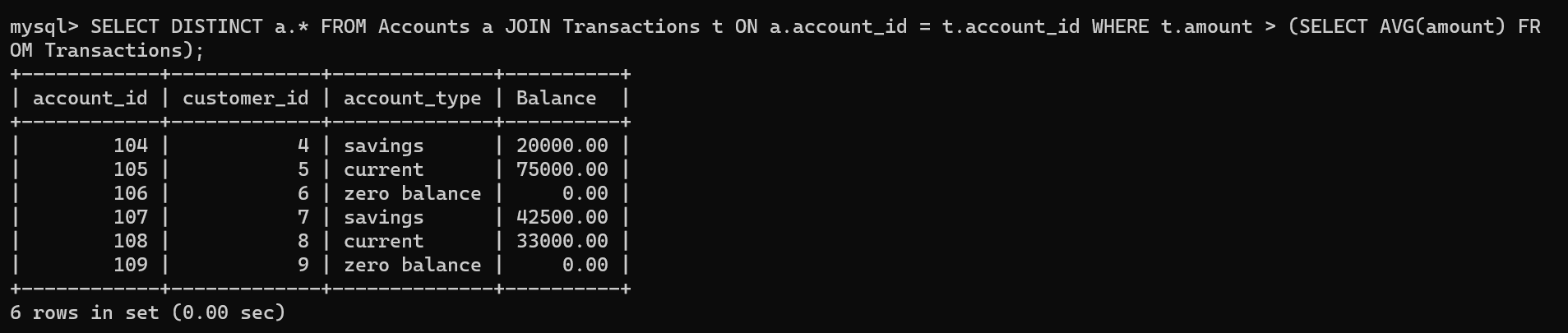


4.3 Retrieve accounts with transactions whose amounts exceed the average transaction amount.

SELECT DISTINCT a.\* FROM Accounts a

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

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



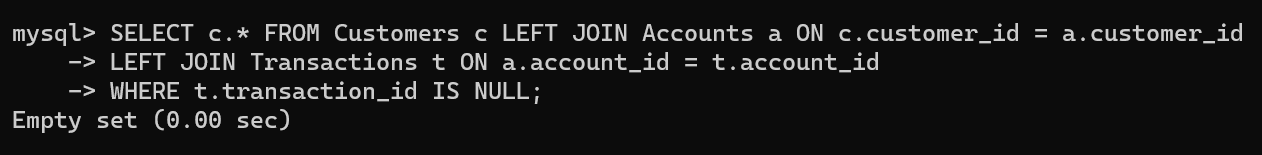
**4.4 Identify customers who have no recorded transactions.**

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

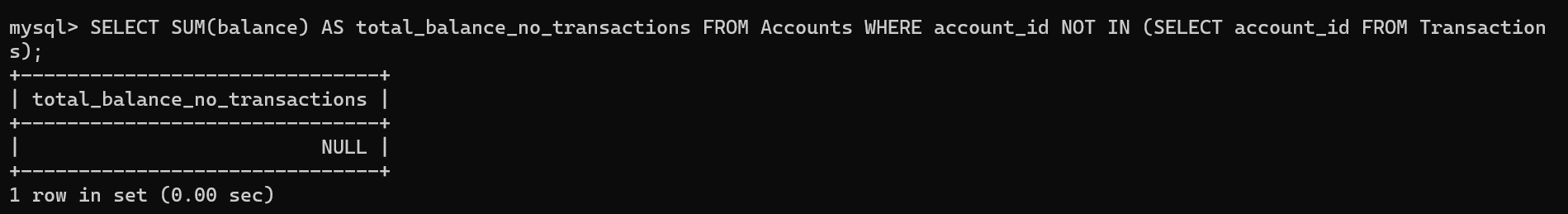
WHERE t.transaction\_id IS NULL;



**4.5 Calculate the total balance of accounts with no recorded transactions.**

SELECT SUM(balance) AS total\_balance\_no\_transactions FROM Accounts

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

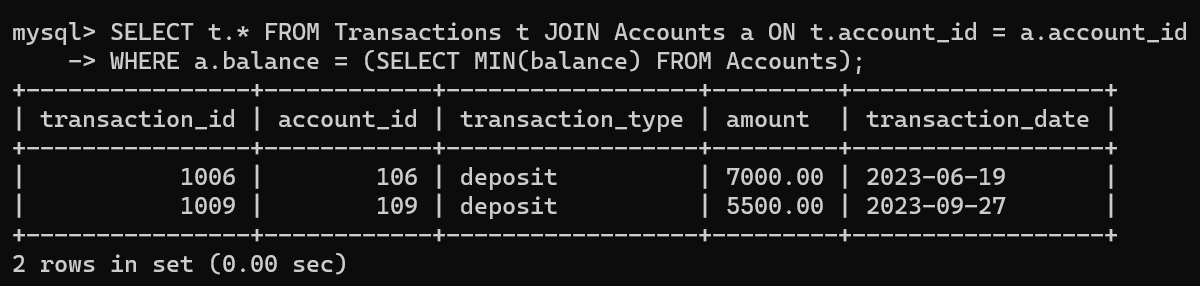


**4.6 Retrieve transactions for accounts with the lowest balance.**

SELECT t.\* FROM Transactions t

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

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

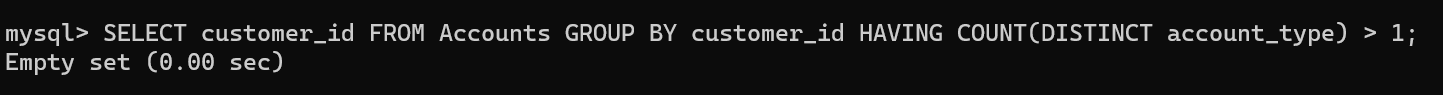


**4.7 Identify customers who have accounts of multiple types.**

SELECT customer\_id FROM Accounts

GROUP BY customer\_id

HAVING COUNT(DISTINCT account\_type) > 1;

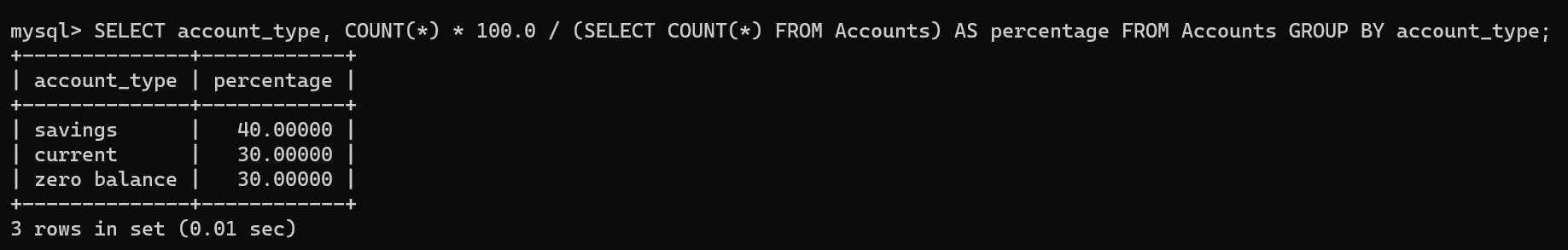


**4.8 Calculate the percentage of each account type out of the total number of accounts.**

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

FROM Accounts

GROUP BY account\_type;

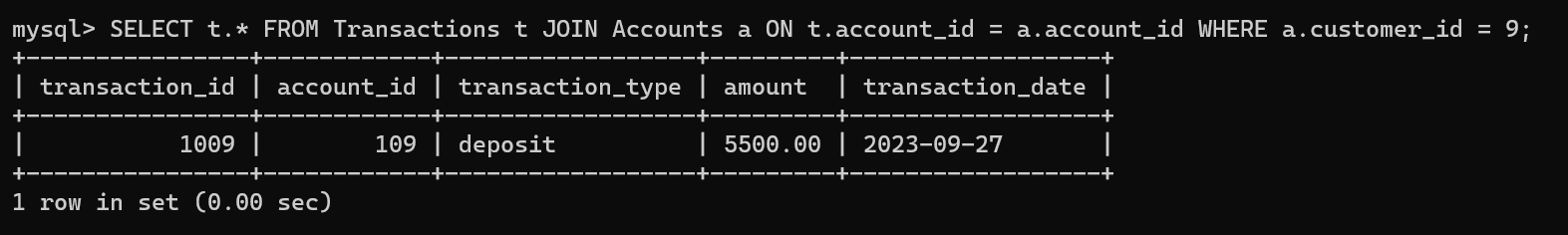


**4.9 Retrieve all transactions for a customer with a given customer\_id.**

SELECT t.\* FROM Transactions t

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

WHERE a.customer\_id = 9;



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

SELECT account\_type, (SELECT SUM(balance) FROM Accounts a WHERE a.account\_type = acc.account\_type) AS total\_balance

FROM (SELECT DISTINCT account\_type FROM Accounts) AS acc;

