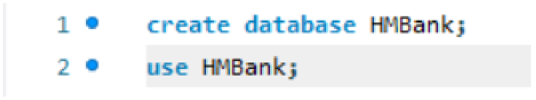
# ASSIGNMENT – 3
# Submitted by : SANKAR ROY

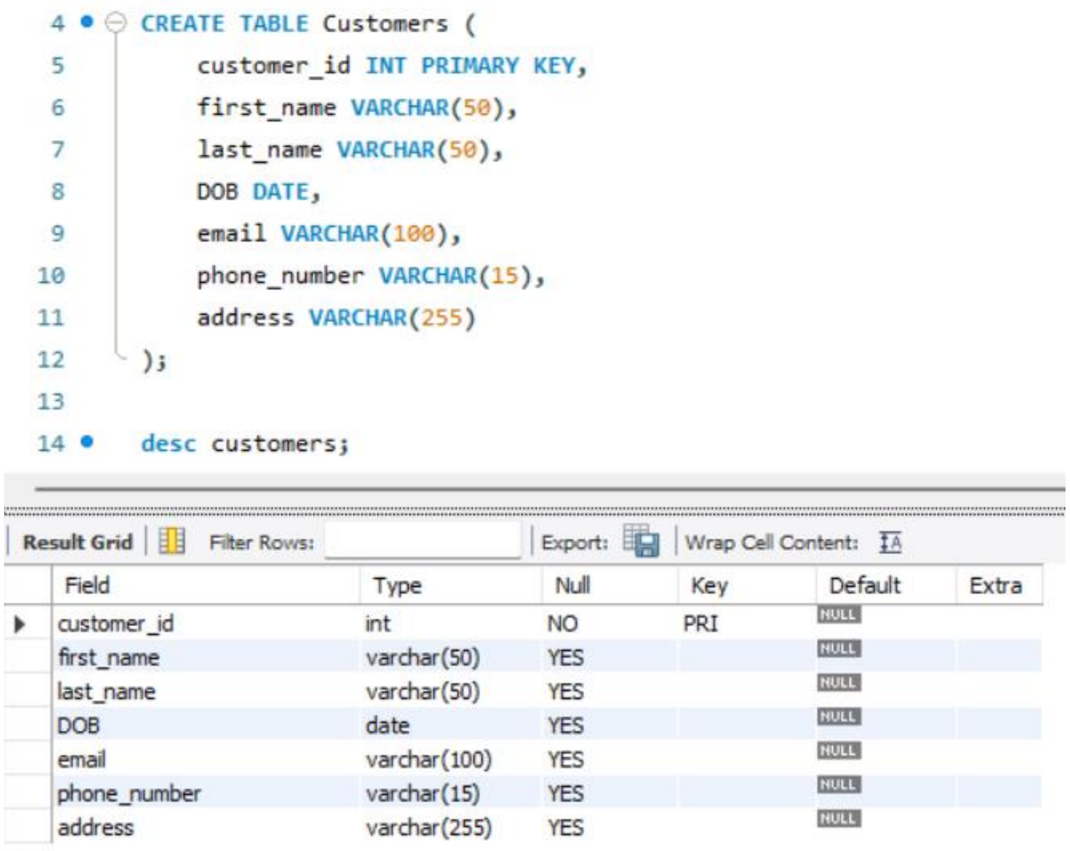
**Tasks 1: Database Design:**

1. Create the database named "HMBank"



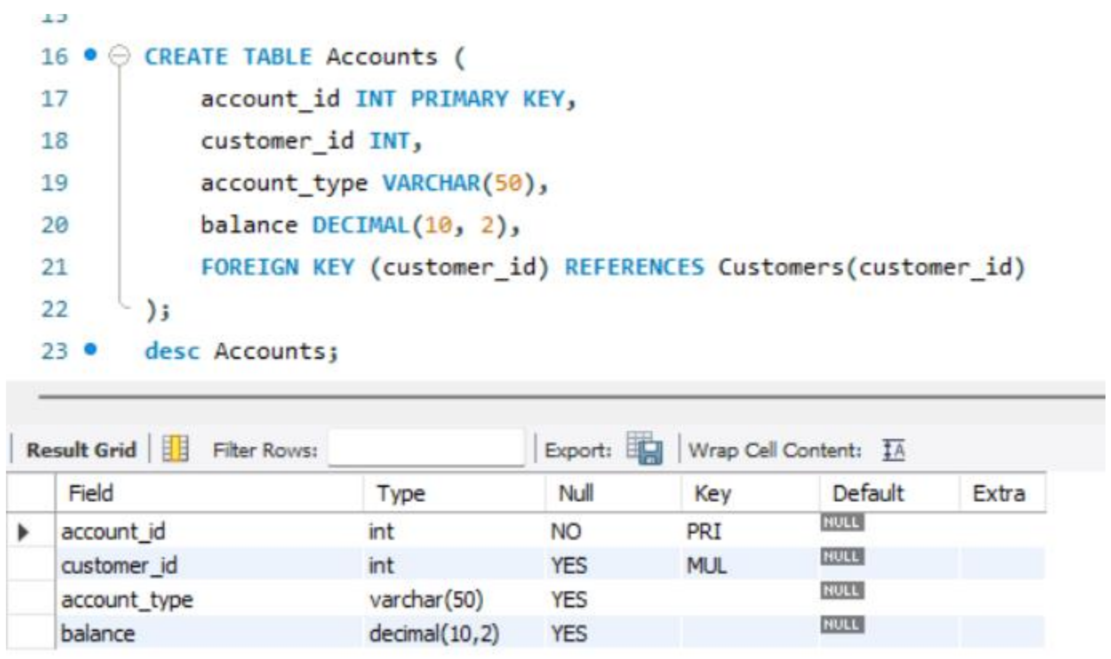
```sql
create database HMBank;
use HMBank;
```

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



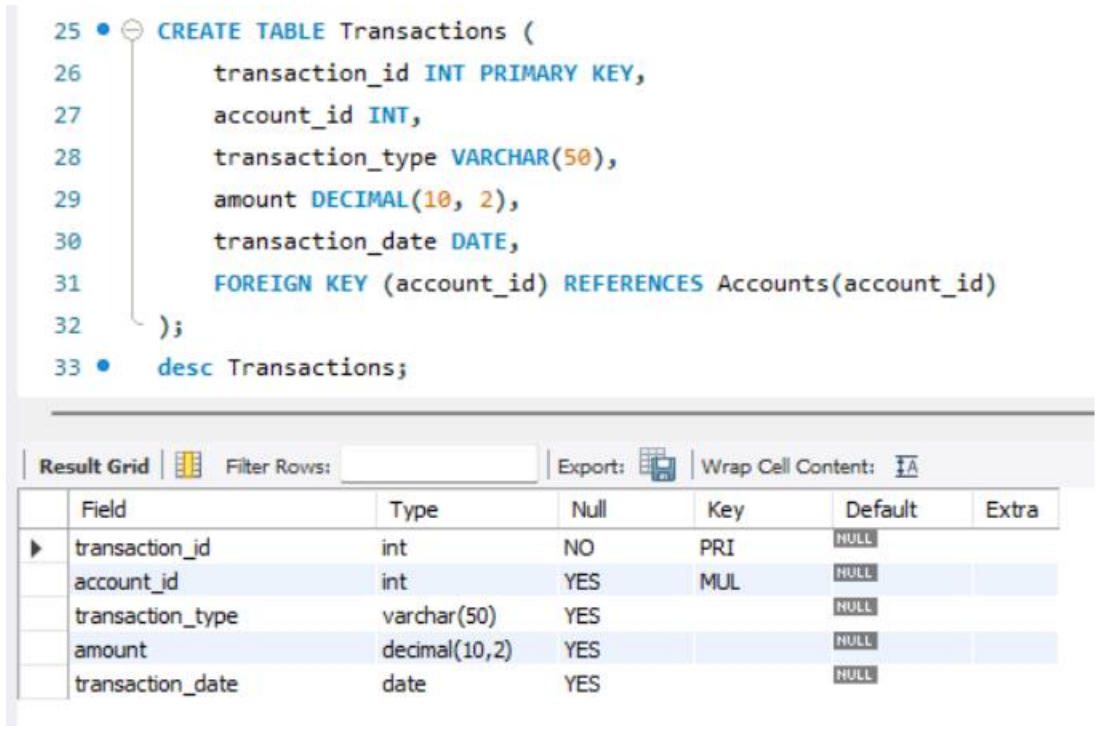
```sql
CREATE TABLE Customers (
    customer_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    DOB DATE,
    email VARCHAR(100),
    phone_number VARCHAR(15),
    address VARCHAR(255)
);

desc customers;
```

| Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|
| customer_id | int | NO | PRI | NULL | |
| first_name | varchar(50) | YES | | NULL | |
| last_name | varchar(50) | YES | | NULL | |
| DOB | date | YES | | NULL | |
| email | varchar(100) | YES | | NULL | |
| phone_number | varchar(15) | YES | | NULL | |
| address | varchar(255) | YES | | NULL | |



```sql
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)
);
desc Accounts;
```

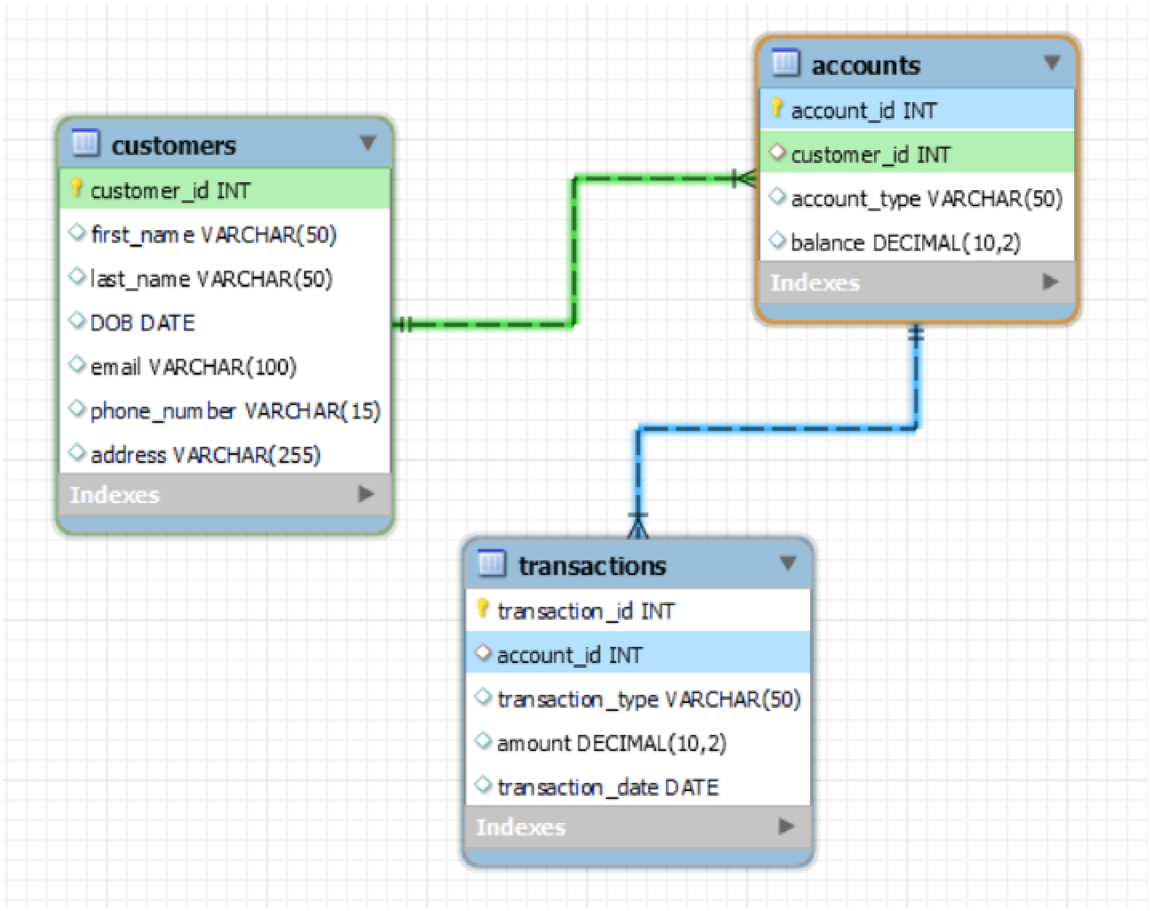
| Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|
| account_id | int | NO | PRI | NULL | |
| customer_id | int | YES | MUL | NULL | |
| account_type | varchar(50) | YES | | NULL | |
| balance | decimal(10,2) | YES | | NULL | |



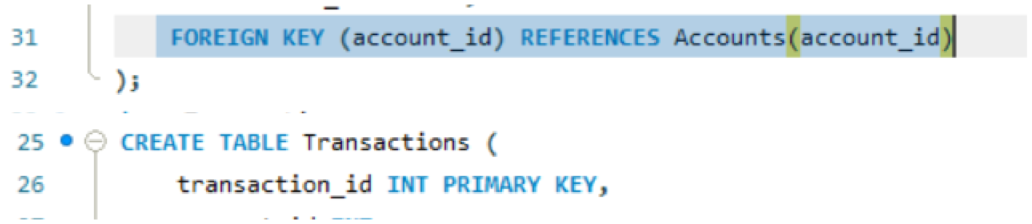
```sql
CREATE TABLE Transactions (
    transaction_id INT PRIMARY KEY,
    account_id INT,
    transaction_type VARCHAR(50),
    amount DECIMAL(10, 2),
    transaction_date DATE,
    FOREIGN KEY (account_id) REFERENCES Accounts(account_id)
);
desc Transactions;
```

| Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|
| transaction_id | int | NO | PRI | NULL | |
| account_id | int | YES | MUL | NULL | |
| transaction_type | varchar(50) | YES | | NULL | |
| amount | decimal(10,2) | YES | | NULL | |
| transaction_date | date | YES | | NULL | |

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



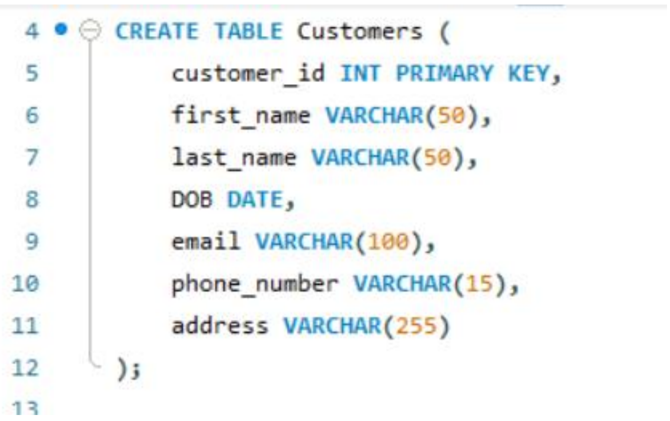
5. Create appropriate Primary Key and Foreign Key constraints for referential integrity.



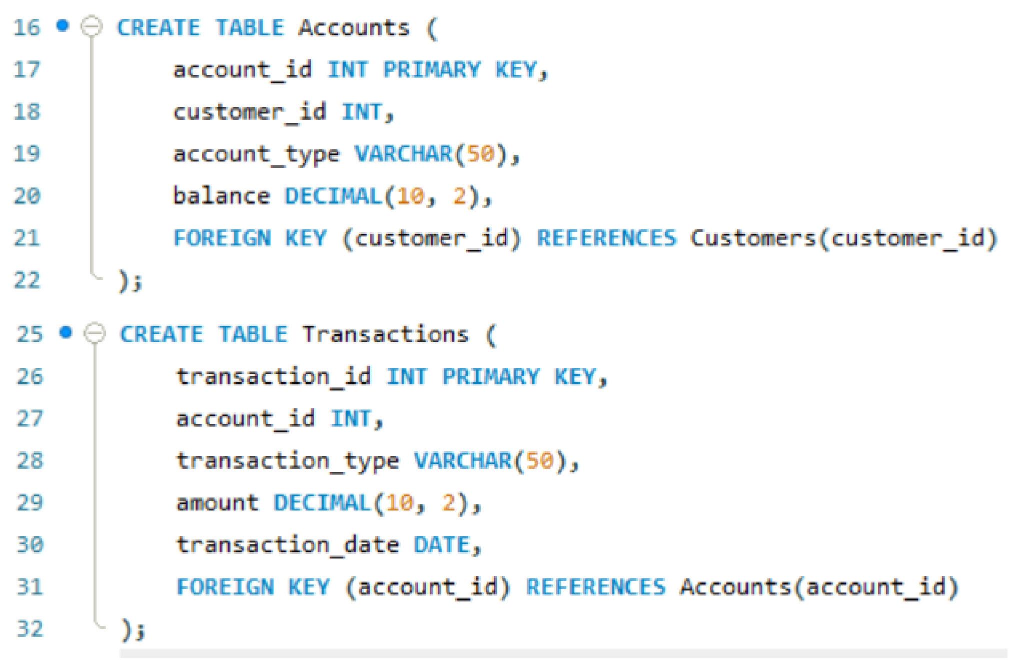
```
    FOREIGN KEY (account_id) REFERENCES Accounts(account_id)
);
```

```
CREATE TABLE Transactions (
    transaction_id INT PRIMARY KEY,
```

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



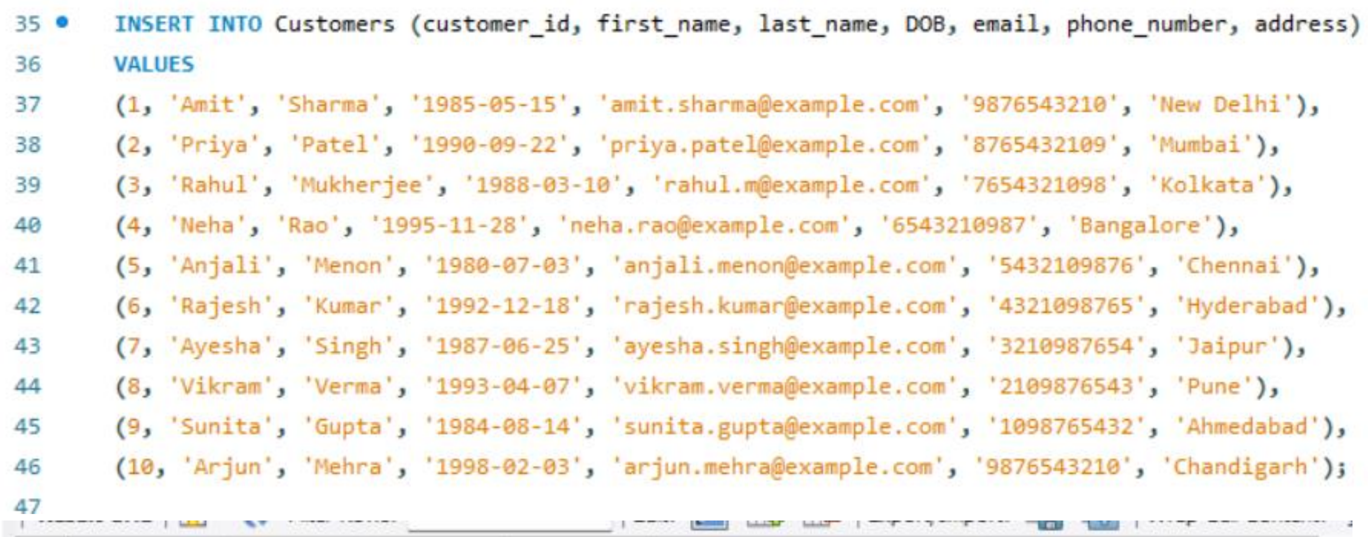
```
CREATE TABLE Customers (
    customer_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    DOB DATE,
    email VARCHAR(100),
    phone_number VARCHAR(15),
    address VARCHAR(255)
);
```



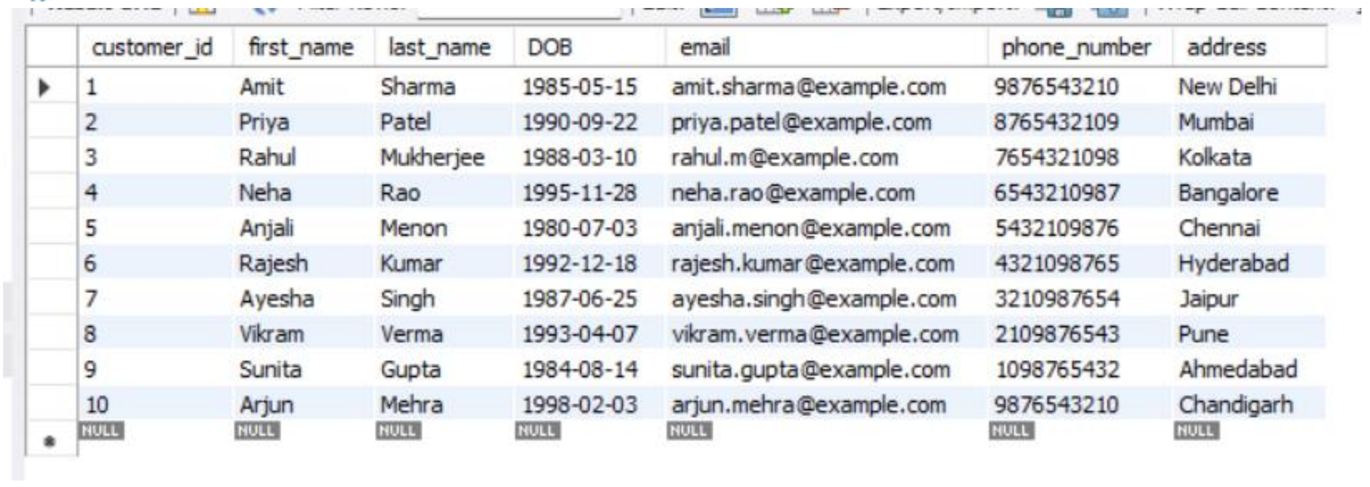
```sql
CREATE TABLE Accounts (
    account_id INT PRIMARY KEY,
    customer_id INT,
    account_type VARCHAR(50),
    balance DECIMAL(10, 2),
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);
CREATE TABLE Transactions (
    transaction_id INT PRIMARY KEY,
    account_id INT,
    transaction_type VARCHAR(50),
    amount DECIMAL(10, 2),
    transaction_date DATE,
    FOREIGN KEY (account_id) REFERENCES Accounts(account_id)
);
```

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

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

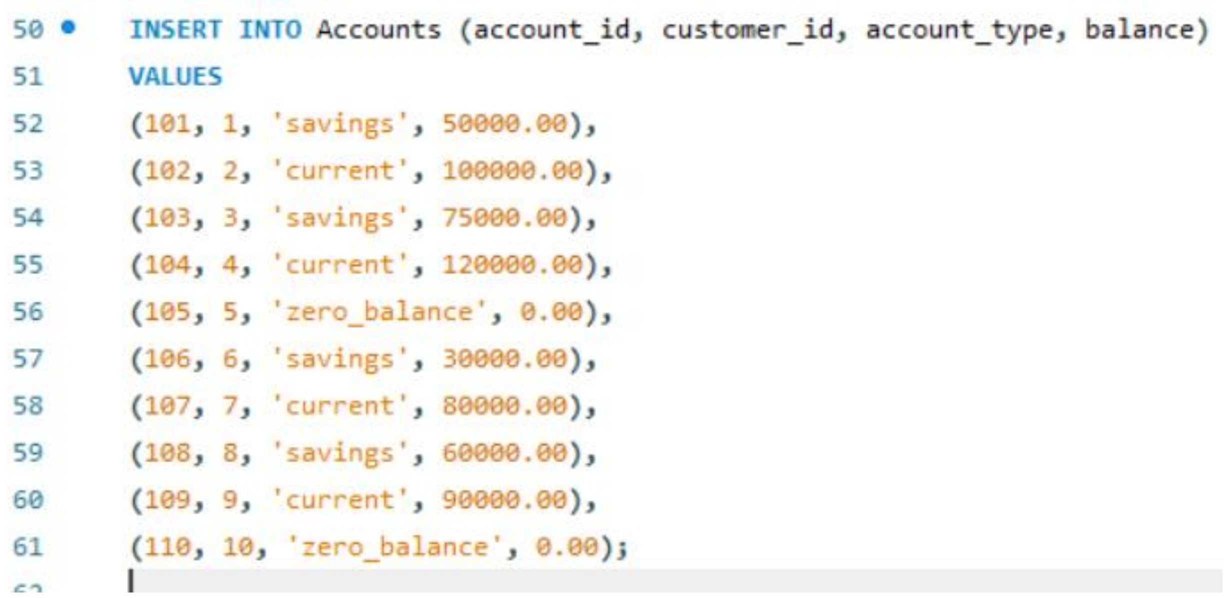


```sql
INSERT INTO Customers (customer_id, first_name, last_name, DOB, email, phone_number, address)
VALUES
(1, 'Amit', 'Sharma', '1985-05-15', 'amit.sharma@example.com', '9876543210', 'New Delhi'),
(2, 'Priya', 'Patel', '1990-09-22', 'priya.patel@example.com', '8765432109', 'Mumbai'),
(3, 'Rahul', 'Mukherjee', '1988-03-10', 'rahul.m@example.com', '7654321098', 'Kolkata'),
(4, 'Neha', 'Rao', '1995-11-28', 'neha.rao@example.com', '6543210987', 'Bangalore'),
(5, 'Anjali', 'Menon', '1980-07-03', 'anjali.menon@example.com', '5432109876', 'Chennai'),
(6, 'Rajesh', 'Kumar', '1992-12-18', 'rajesh.kumar@example.com', '4321098765', 'Hyderabad'),
(7, 'Ayesha', 'Singh', '1987-06-25', 'ayesha.singh@example.com', '3210987654', 'Jaipur'),
(8, 'Vikram', 'Verma', '1993-04-07', 'vikram.verma@example.com', '2109876543', 'Pune'),
(9, 'Sunita', 'Gupta', '1984-08-14', 'sunita.gupta@example.com', '1098765432', 'Ahmedabad'),
(10, 'Arjun', 'Mehra', '1998-02-03', 'arjun.mehra@example.com', '9876543210', 'Chandigarh');
```

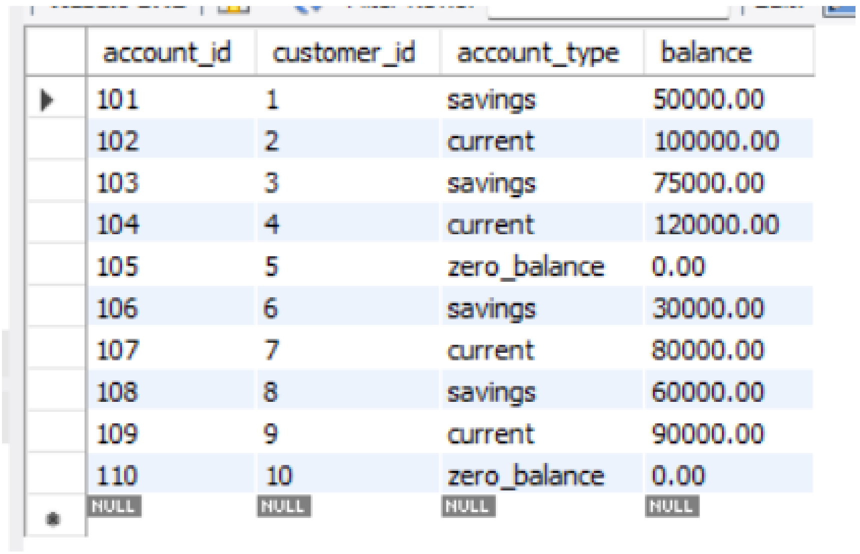


| customer_id | first_name | last_name | DOB | email | phone_number | address |
|---|---|---|---|---|---|---|
| 1 | Amit | Sharma | 1985-05-15 | amit.sharma@example.com | 9876543210 | New Delhi |
| 2 | Priya | Patel | 1990-09-22 | priya.patel@example.com | 8765432109 | Mumbai |
| 3 | Rahul | Mukherjee | 1988-03-10 | rahul.m@example.com | 7654321098 | Kolkata |
| 4 | Neha | Rao | 1995-11-28 | neha.rao@example.com | 6543210987 | Bangalore |
| 5 | Anjali | Menon | 1980-07-03 | anjali.menon@example.com | 5432109876 | Chennai |
| 6 | Rajesh | Kumar | 1992-12-18 | rajesh.kumar@example.com | 4321098765 | Hyderabad |
| 7 | Ayesha | Singh | 1987-06-25 | ayesha.singh@example.com | 3210987654 | Jaipur |
| 8 | Vikram | Verma | 1993-04-07 | vikram.verma@example.com | 2109876543 | Pune |
| 9 | Sunita | Gupta | 1984-08-14 | sunita.gupta@example.com | 1098765432 | Ahmedabad |
| 10 | Arjun | Mehra | 1998-02-03 | arjun.mehra@example.com | 9876543210 | Chandigarh |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |

- Accounts

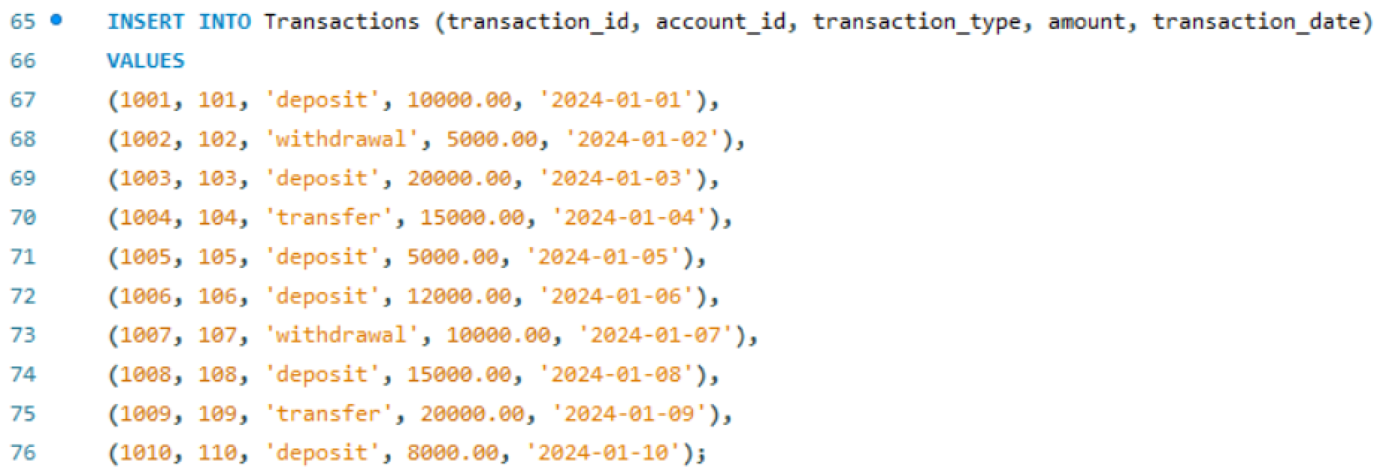


```sql
INSERT INTO Accounts (account_id, customer_id, account_type, balance)
VALUES
(101, 1, 'savings', 50000.00),
(102, 2, 'current', 100000.00),
(103, 3, 'savings', 75000.00),
(104, 4, 'current', 120000.00),
(105, 5, 'zero_balance', 0.00),
(106, 6, 'savings', 30000.00),
(107, 7, 'current', 80000.00),
(108, 8, 'savings', 60000.00),
(109, 9, 'current', 90000.00),
(110, 10, 'zero_balance', 0.00);
```

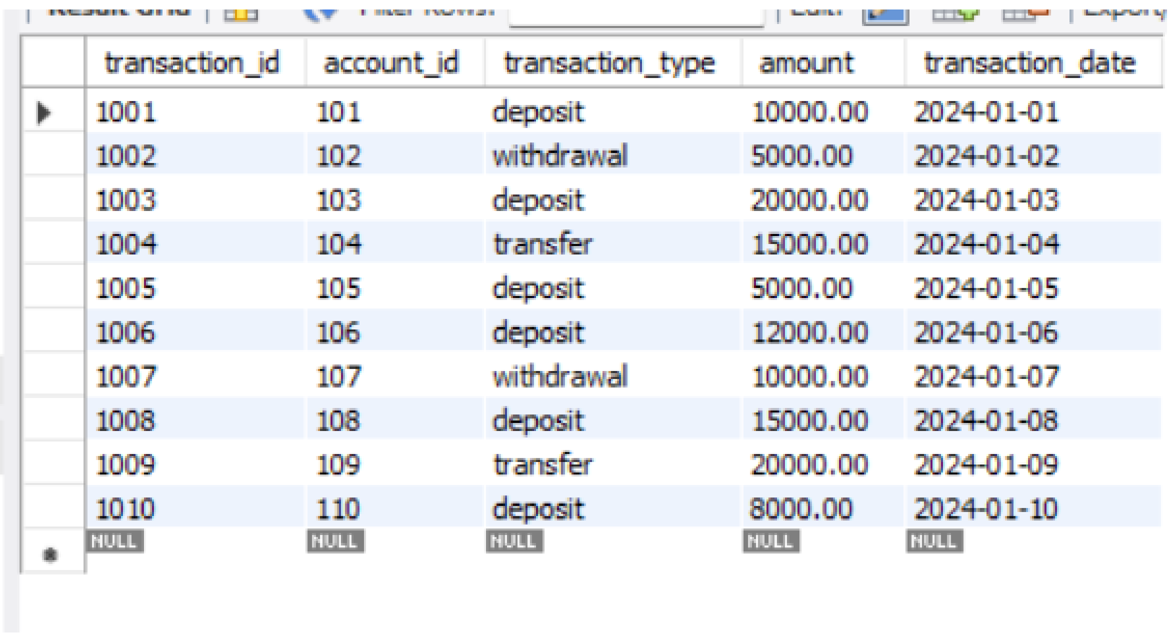


| account_id | customer_id | account_type | balance |
|---|---|---|---|
| 101 | 1 | savings | 50000.00 |
| 102 | 2 | current | 100000.00 |
| 103 | 3 | savings | 75000.00 |
| 104 | 4 | current | 120000.00 |
| 105 | 5 | zero_balance | 0.00 |
| 106 | 6 | savings | 30000.00 |
| 107 | 7 | current | 80000.00 |
| 108 | 8 | savings | 60000.00 |
| 109 | 9 | current | 90000.00 |
| 110 | 10 | zero_balance | 0.00 |
| NULL | NULL | NULL | NULL |

- Transactions

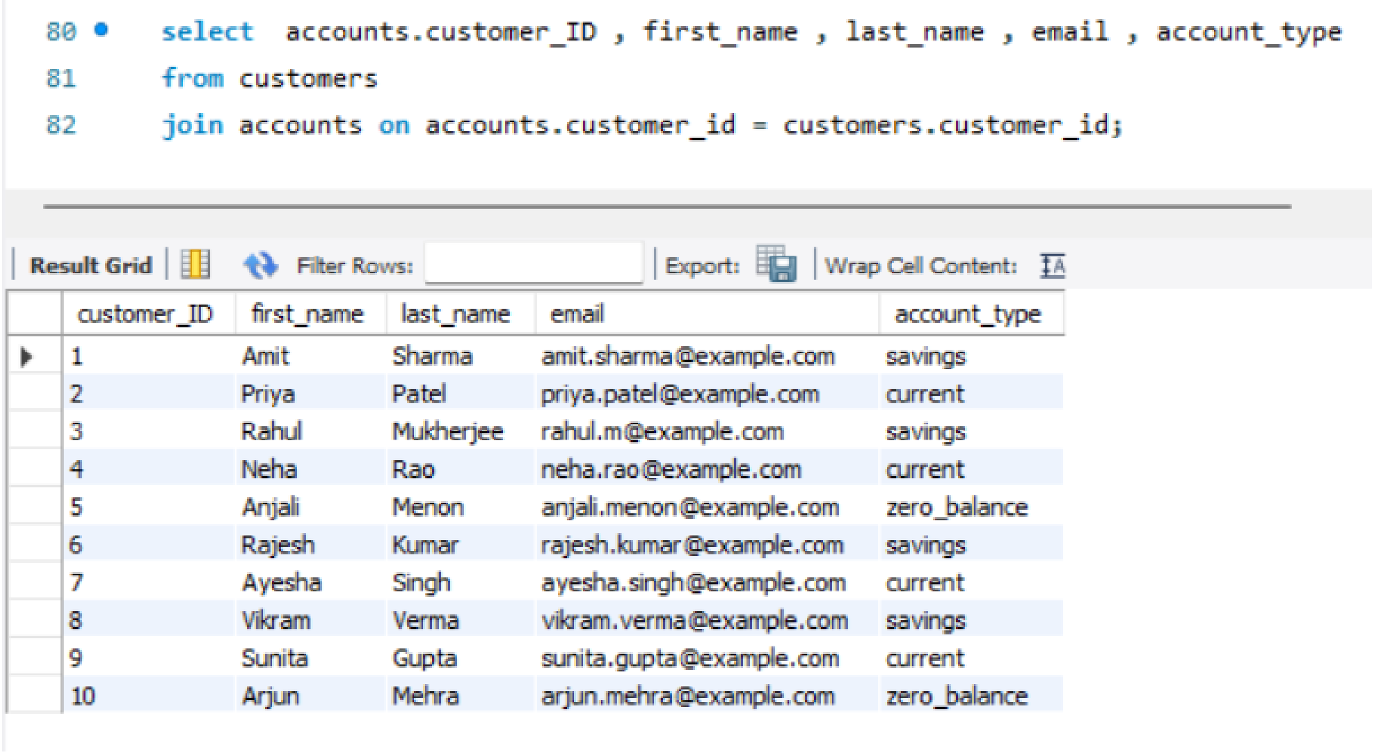


```sql
INSERT INTO Transactions (transaction_id, account_id, transaction_type, amount, transaction_date)
VALUES
(1001, 101, 'deposit', 10000.00, '2024-01-01'),
(1002, 102, 'withdrawal', 5000.00, '2024-01-02'),
(1003, 103, 'deposit', 20000.00, '2024-01-03'),
(1004, 104, 'transfer', 15000.00, '2024-01-04'),
(1005, 105, 'deposit', 5000.00, '2024-01-05'),
(1006, 106, 'deposit', 12000.00, '2024-01-06'),
(1007, 107, 'withdrawal', 10000.00, '2024-01-07'),
(1008, 108, 'deposit', 15000.00, '2024-01-08'),
(1009, 109, 'transfer', 20000.00, '2024-01-09'),
(1010, 110, 'deposit', 8000.00, '2024-01-10');
```



| | transaction_id | account_id | transaction_type | amount | transaction_date |
|---|---|---|---|---|---|
| ▶ | 1001 | 101 | deposit | 10000.00 | 2024-01-01 |
| | 1002 | 102 | withdrawal | 5000.00 | 2024-01-02 |
| | 1003 | 103 | deposit | 20000.00 | 2024-01-03 |
| | 1004 | 104 | transfer | 15000.00 | 2024-01-04 |
| | 1005 | 105 | deposit | 5000.00 | 2024-01-05 |
| | 1006 | 106 | deposit | 12000.00 | 2024-01-06 |
| | 1007 | 107 | withdrawal | 10000.00 | 2024-01-07 |
| | 1008 | 108 | deposit | 15000.00 | 2024-01-08 |
| | 1009 | 109 | transfer | 20000.00 | 2024-01-09 |
| | 1010 | 110 | deposit | 8000.00 | 2024-01-10 |
| * | NULL | NULL | NULL | NULL | NULL |

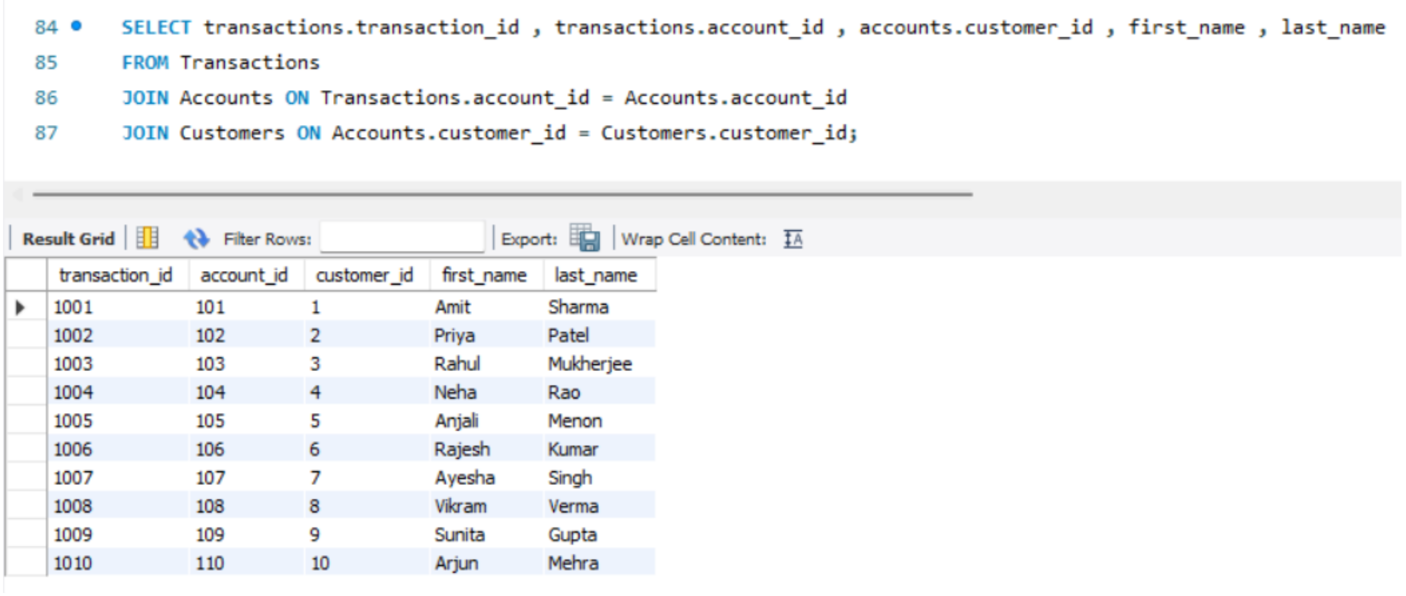
2. Write SQL queries for the following tasks:
    1. Write a SQL query to retrieve the name, account type and email of all customers.



```sql
select  accounts.customer_ID , first_name , last_name , email , account_type
from customers
join accounts on accounts.customer_id = customers.customer_id;
```

| | customer_ID | first_name | last_name | email | account_type |
|---|---|---|---|---|---|
| ▶ | 1 | Amit | Sharma | amit.sharma@example.com | savings |
| | 2 | Priya | Patel | priya.patel@example.com | current |
| | 3 | Rahul | Mukherjee | rahul.m@example.com | savings |
| | 4 | Neha | Rao | neha.rao@example.com | current |
| | 5 | Anjali | Menon | anjali.menon@example.com | zero_balance |
| | 6 | Rajesh | Kumar | rajesh.kumar@example.com | savings |
| | 7 | Ayesha | Singh | ayesha.singh@example.com | current |
| | 8 | Vikram | Verma | vikram.verma@example.com | savings |
| | 9 | Sunita | Gupta | sunita.gupta@example.com | current |
| | 10 | Arjun | Mehra | arjun.mehra@example.com | zero_balance |

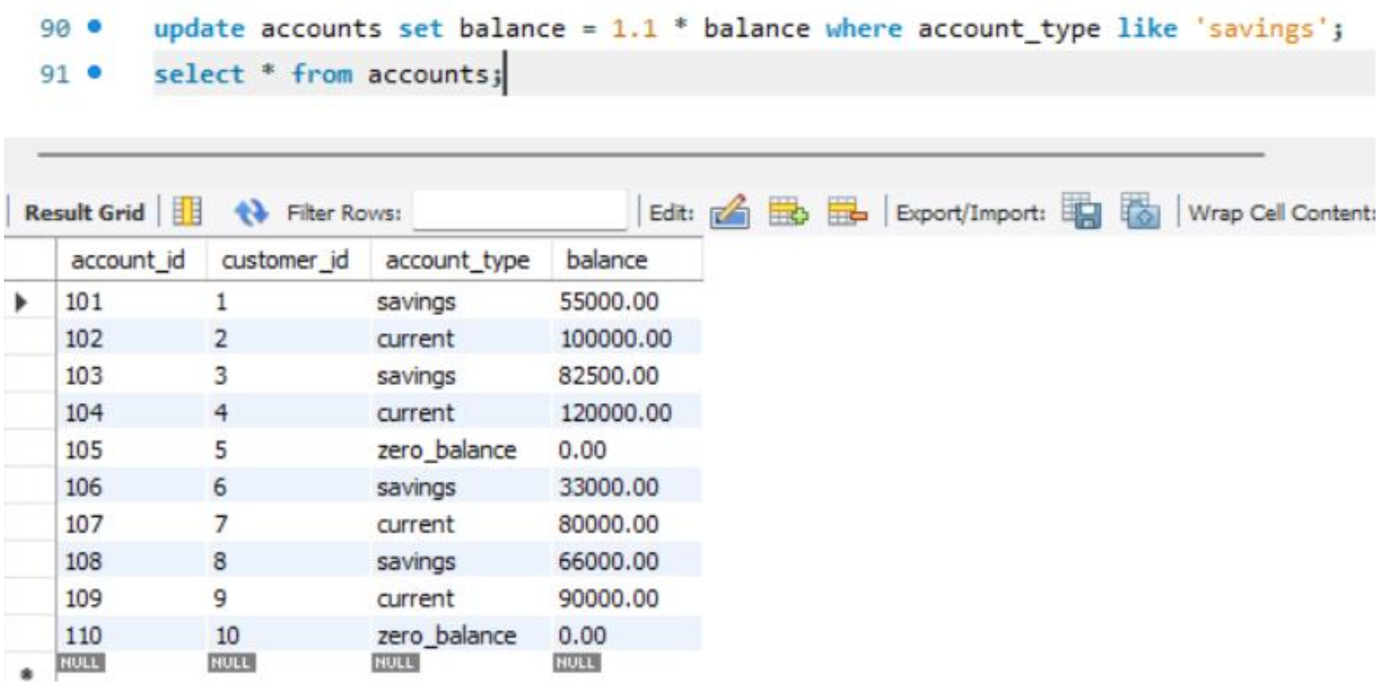
    2. Write a SQL query to list all transaction corresponding customer.



```sql
SELECT transactions.transaction_id , transactions.account_id , accounts.customer_id , first_name , last_name
FROM Transactions
JOIN Accounts ON Transactions.account_id = Accounts.account_id
JOIN Customers ON Accounts.customer_id = Customers.customer_id;
```

| transaction_id | account_id | customer_id | first_name | last_name |
|---|---|---|---|---|
| 1001 | 101 | 1 | Amit | Sharma |
| 1002 | 102 | 2 | Priya | Patel |
| 1003 | 103 | 3 | Rahul | Mukherjee |
| 1004 | 104 | 4 | Neha | Rao |
| 1005 | 105 | 5 | Anjali | Menon |
| 1006 | 106 | 6 | Rajesh | Kumar |
| 1007 | 107 | 7 | Ayesha | Singh |
| 1008 | 108 | 8 | Vikram | Verma |
| 1009 | 109 | 9 | Sunita | Gupta |
| 1010 | 110 | 10 | Arjun | Mehra |

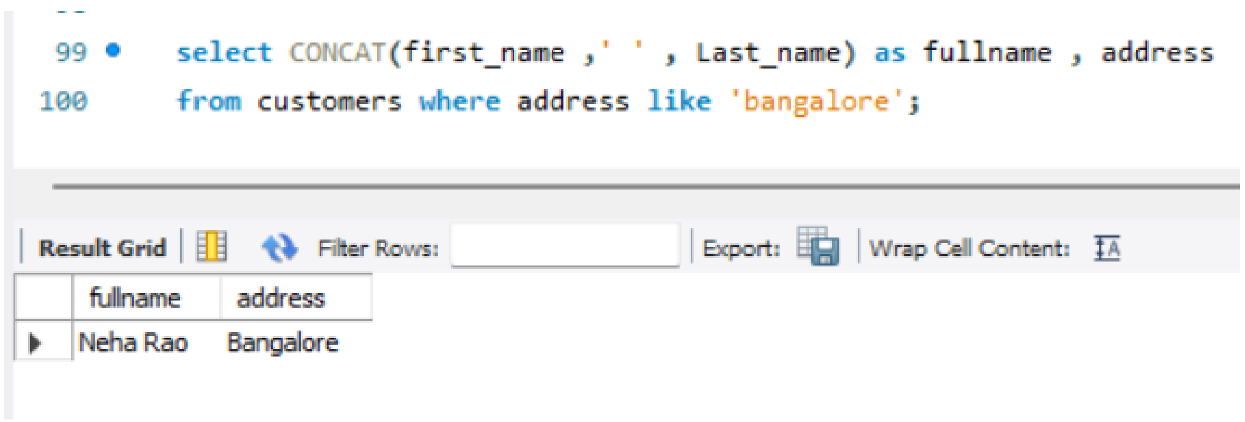
3. Write a SQL query to increase the balance of a specific account by a certain amount.



```sql
update accounts set balance = 1.1 * balance where account_type like 'savings';
select * from accounts;
```

| account_id | customer_id | account_type | balance |
|---|---|---|---|
| 101 | 1 | savings | 55000.00 |
| 102 | 2 | current | 100000.00 |
| 103 | 3 | savings | 82500.00 |
| 104 | 4 | current | 120000.00 |
| 105 | 5 | zero_balance | 0.00 |
| 106 | 6 | savings | 33000.00 |
| 107 | 7 | current | 80000.00 |
| 108 | 8 | savings | 66000.00 |
| 109 | 9 | current | 90000.00 |
| 110 | 10 | zero_balance | 0.00 |
| NULL | NULL | NULL | NULL |

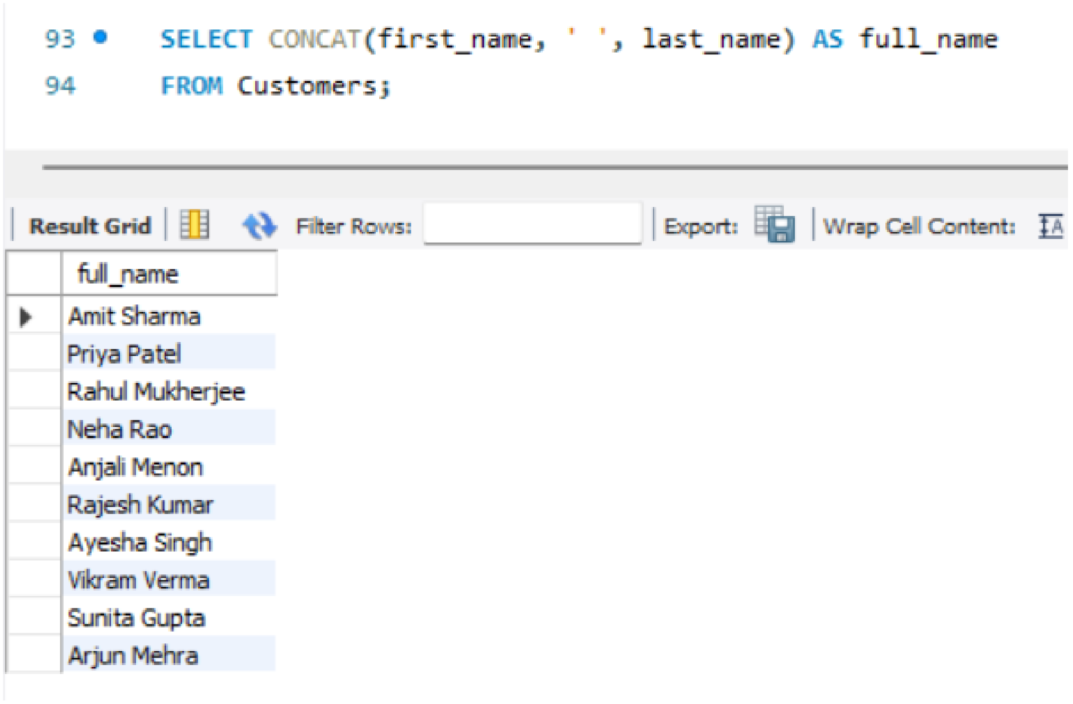
4. Write a SQL query to Combine first and last names of customers as a full_name.



```sql
select CONCAT(first_name ,' ' , Last_name) as fullname , address
from customers where address like 'bangalore';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| fullname | address |
|---|---|
| Neha Rao | Bangalore |

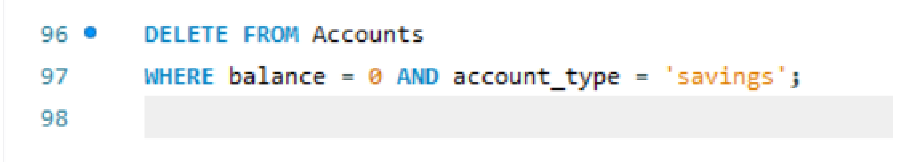


```sql
SELECT CONCAT(first_name, ' ', last_name) AS full_name
FROM Customers;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| full_name |
|---|
| Amit Sharma |
| Priya Patel |
| Rahul Mukherjee |
| Neha Rao |
| Anjali Menon |
| Rajesh Kumar |
| Ayesha Singh |
| Vikram Verma |
| Sunita Gupta |
| Arjun Mehra |

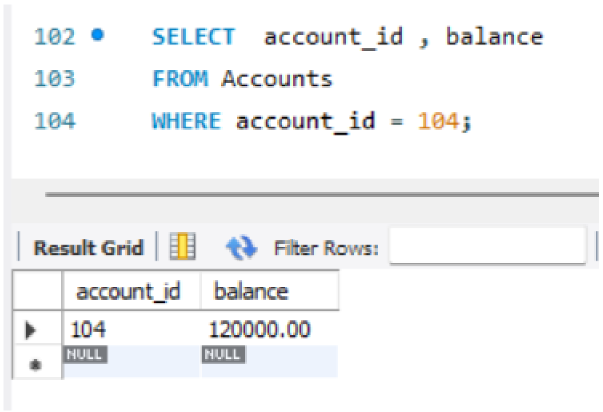
5. Write a SQL query to remove accounts with a balance of zero where the account type is savings.



```sql
DELETE FROM Accounts
WHERE balance = 0 AND account_type = 'savings';
```

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

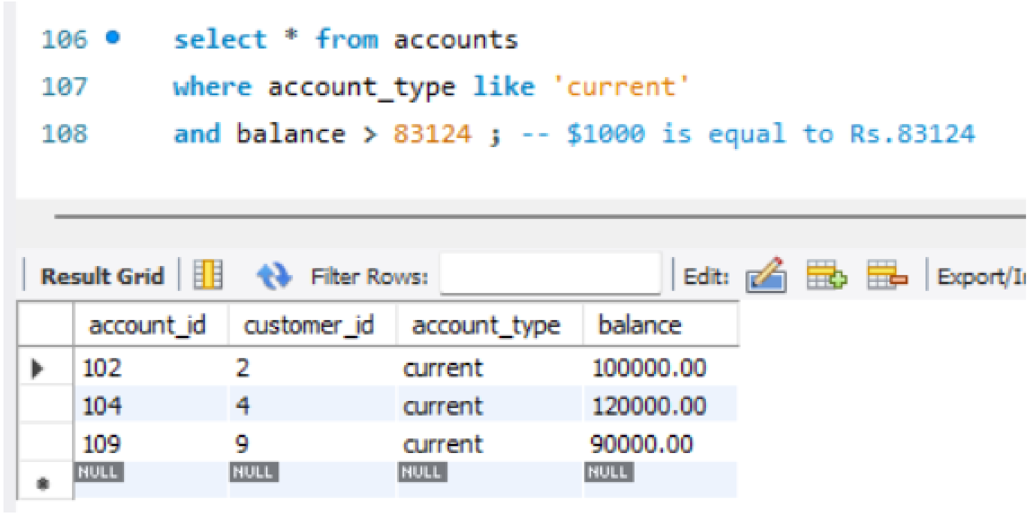
7. Write a SQL query to Get the account balance for a specific account.



```
SELECT  account_id , balance
FROM Accounts
WHERE account_id = 104;
```

| | account_id | balance |
|---|---|---|
| ▶ | 104 | 120000.00 |
| * | NULL | NULL |

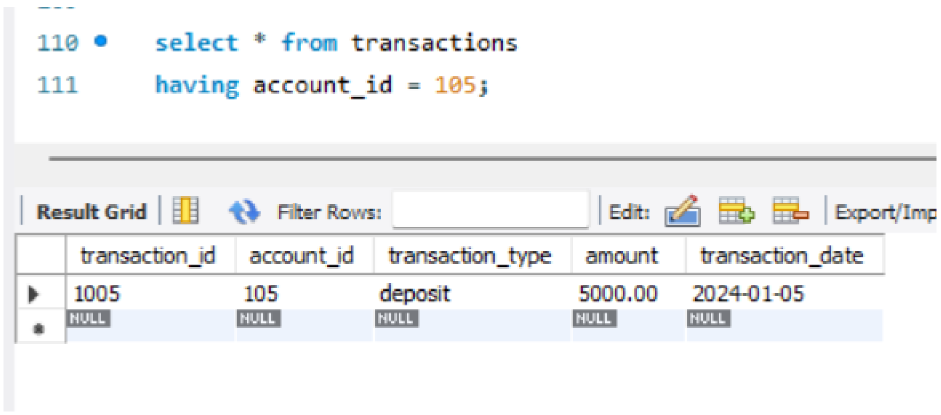
8. Write a SQL query to List all current accounts with a balance greater than $1,000.



```
select * from accounts
where account_type like 'current'
and balance > 83124 ; -- $1000 is equal to Rs.83124
```

| | account_id | customer_id | account_type | balance |
|---|---|---|---|---|
| ▶ | 102 | 2 | current | 100000.00 |
| | 104 | 4 | current | 120000.00 |
| | 109 | 9 | current | 90000.00 |
| * | NULL | NULL | NULL | NULL |

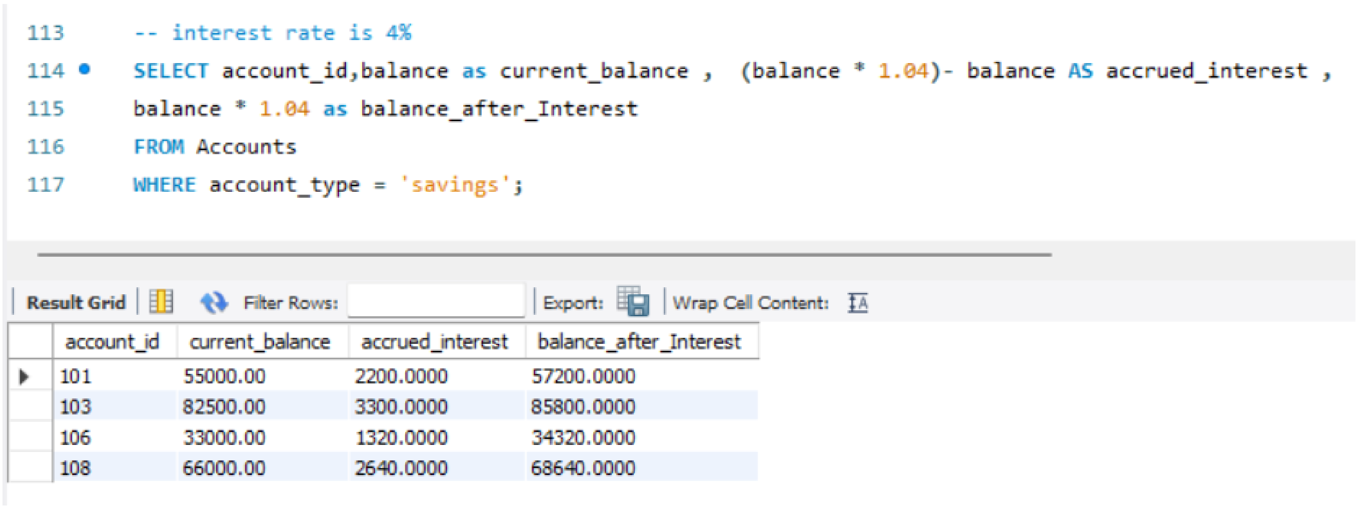
9. Write a SQL query to Retrieve all transactions for a specific account.



```
select * from transactions
having account_id = 105;
```

| | transaction_id | account_id | transaction_type | amount | transaction_date |
|---|---|---|---|---|---|
| ▶ | 1005 | 105 | deposit | 5000.00 | 2024-01-05 |
| * | NULL | NULL | NULL | NULL | NULL |

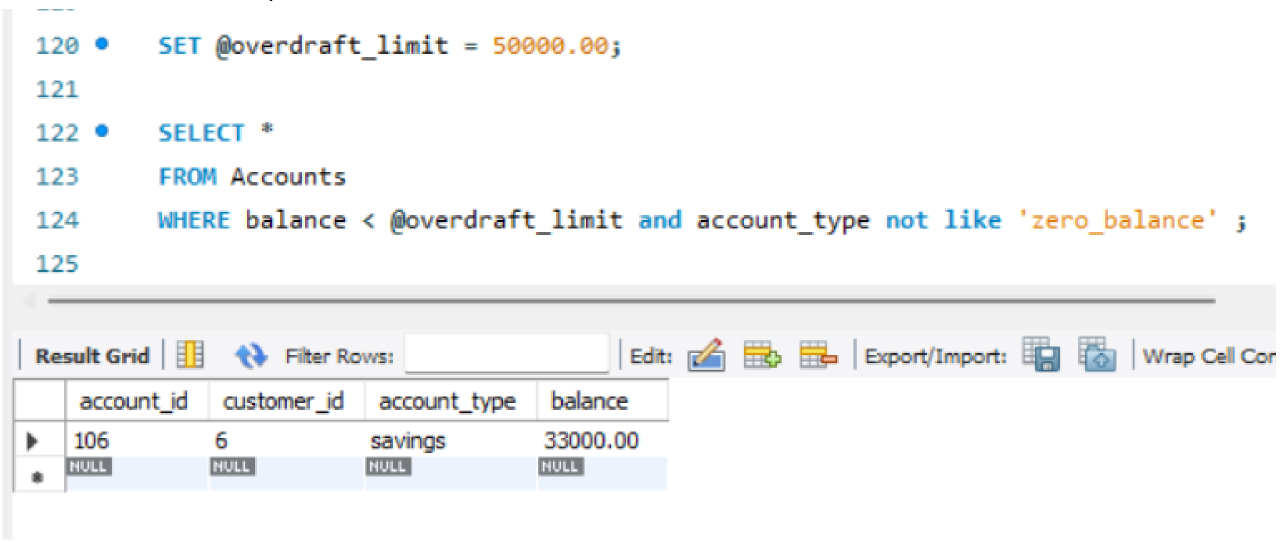
10. Write a SQL query to Calculate the interest accrued on savings accounts based on agiven interest rate.



```sql
-- interest rate is 4%
SELECT account_id,balance as current_balance ,  (balance * 1.04)- balance AS accrued_interest ,
balance * 1.04 as balance_after_Interest
FROM Accounts
WHERE account_type = 'savings';
```

| account_id | current_balance | accrued_interest | balance_after_Interest |
|---|---|---|---|
| 101 | 55000.00 | 2200.0000 | 57200.0000 |
| 103 | 82500.00 | 3300.0000 | 85800.0000 |
| 106 | 33000.00 | 1320.0000 | 34320.0000 |
| 108 | 66000.00 | 2640.0000 | 68640.0000 |

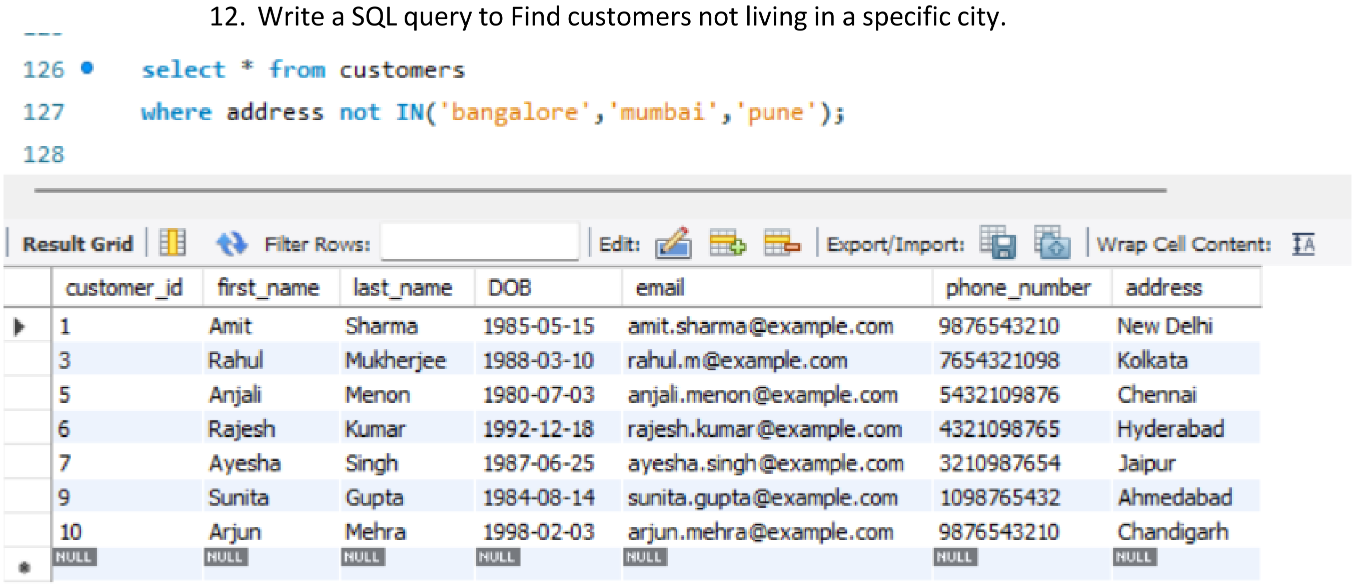
11. Write a SQL query to Identify accounts where the balance is less than a specifiedoverdraft limit.



```sql
SET @overdraft_limit = 50000.00;

SELECT *
FROM Accounts
WHERE balance < @overdraft_limit and account_type not like 'zero_balance' ;

```

| account_id | customer_id | account_type | balance |
|---|---|---|---|
| 106 | 6 | savings | 33000.00 |
| NULL | NULL | NULL | NULL |

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

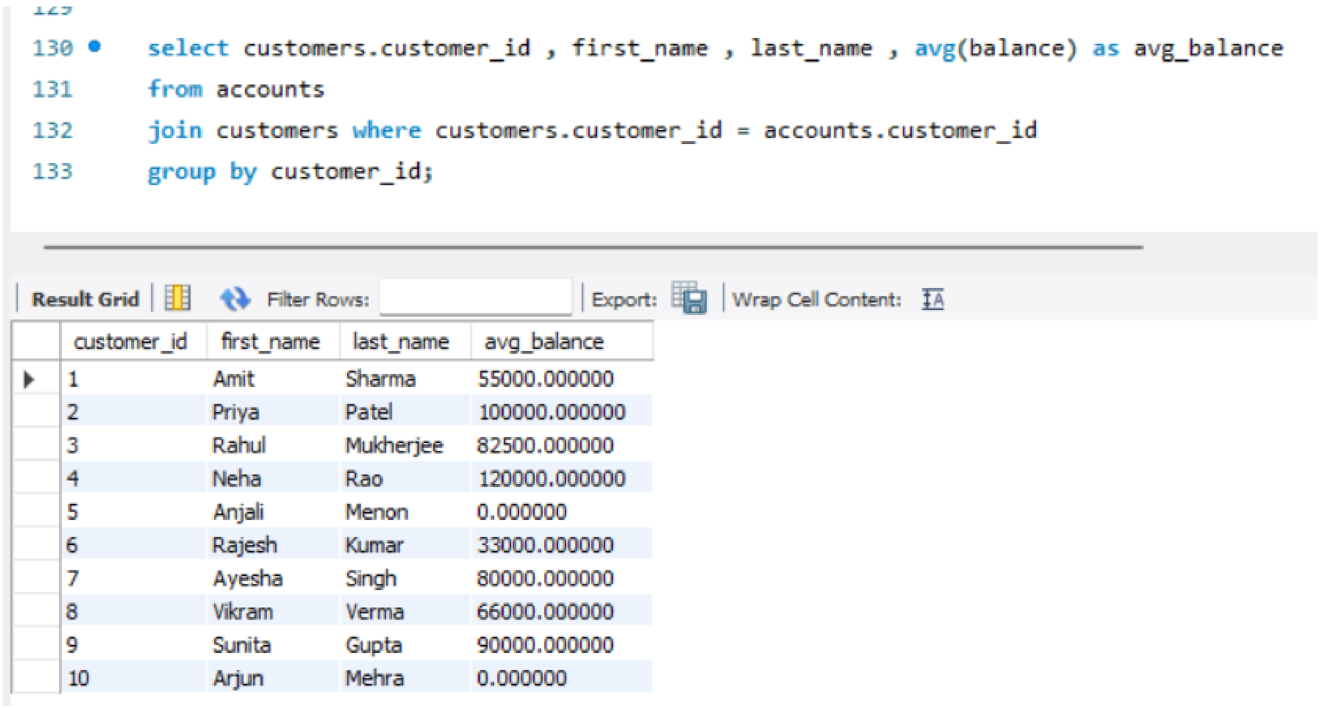


```sql
select * from customers
where address not IN('bangalore','mumbai','pune');

```

| customer_id | first_name | last_name | DOB | email | phone_number | address |
|---|---|---|---|---|---|---|
| 1 | Amit | Sharma | 1985-05-15 | amit.sharma@example.com | 9876543210 | New Delhi |
| 3 | Rahul | Mukherjee | 1988-03-10 | rahul.m@example.com | 7654321098 | Kolkata |
| 5 | Anjali | Menon | 1980-07-03 | anjali.menon@example.com | 5432109876 | Chennai |
| 6 | Rajesh | Kumar | 1992-12-18 | rajesh.kumar@example.com | 4321098765 | Hyderabad |
| 7 | Ayesha | Singh | 1987-06-25 | ayesha.singh@example.com | 3210987654 | Jaipur |
| 9 | Sunita | Gupta | 1984-08-14 | sunita.gupta@example.com | 1098765432 | Ahmedabad |
| 10 | Arjun | Mehra | 1998-02-03 | arjun.mehra@example.com | 9876543210 | Chandigarh |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |

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

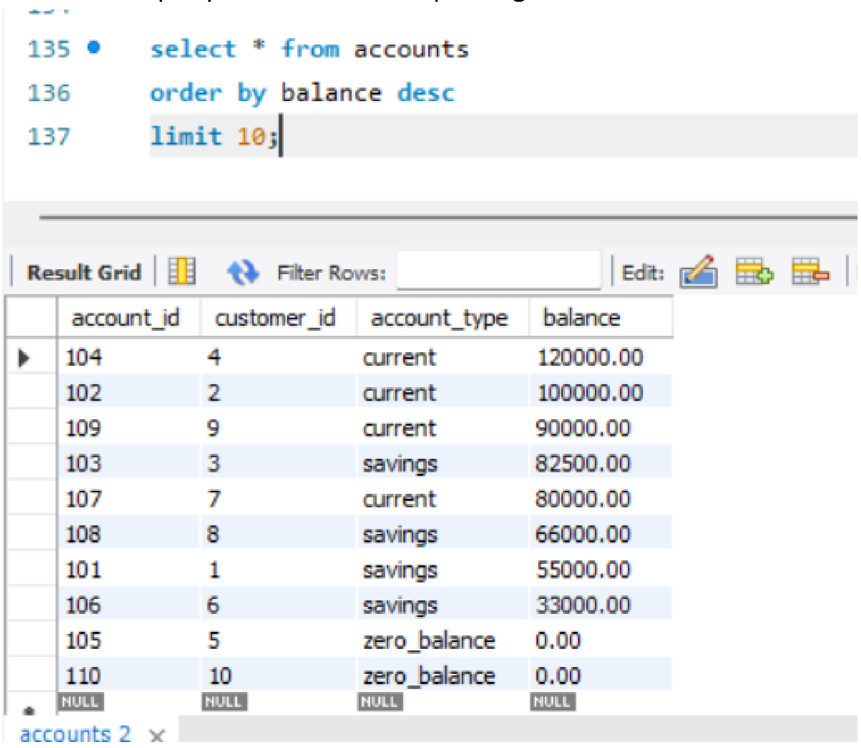
1. Write a SQL query to Find the average account balance for all customers.



```
select customers.customer_id , first_name , last_name , avg(balance) as avg_balance
from accounts
join customers where customers.customer_id = accounts.customer_id
group by customer_id;
```

| customer_id | first_name | last_name | avg_balance |
|---|---|---|---|
| 1 | Amit | Sharma | 55000.000000 |
| 2 | Priya | Patel | 100000.000000 |
| 3 | Rahul | Mukherjee | 82500.000000 |
| 4 | Neha | Rao | 120000.000000 |
| 5 | Anjali | Menon | 0.000000 |
| 6 | Rajesh | Kumar | 33000.000000 |
| 7 | Ayesha | Singh | 80000.000000 |
| 8 | Vikram | Verma | 66000.000000 |
| 9 | Sunita | Gupta | 90000.000000 |
| 10 | Arjun | Mehra | 0.000000 |

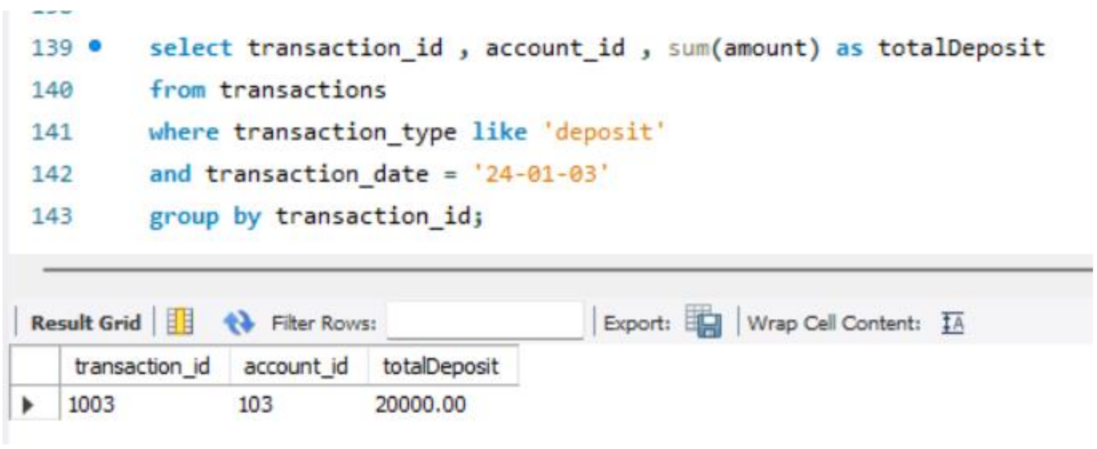
2. Write a SQL query to Retrieve the top 10 highest account balances.



```
select * from accounts
order by balance desc
limit 10;
```

| account_id | customer_id | account_type | balance |
|---|---|---|---|
| 104 | 4 | current | 120000.00 |
| 102 | 2 | current | 100000.00 |
| 109 | 9 | current | 90000.00 |
| 103 | 3 | savings | 82500.00 |
| 107 | 7 | current | 80000.00 |
| 108 | 8 | savings | 66000.00 |
| 101 | 1 | savings | 55000.00 |
| 106 | 6 | savings | 33000.00 |
| 105 | 5 | zero_balance | 0.00 |
| 110 | 10 | zero_balance | 0.00 |
| NULL | NULL | NULL | NULL |

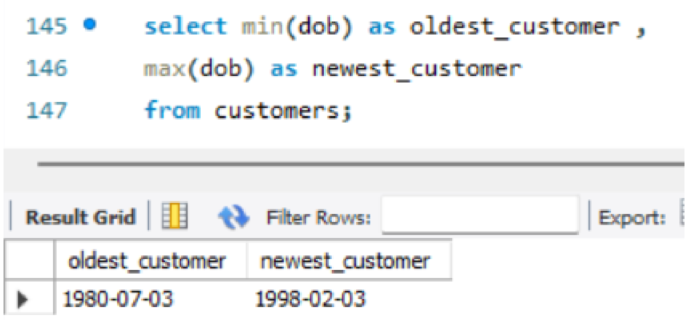
3. Write a SQL query to Calculate Total Deposits for All Customers in specific date.



```sql
select transaction_id , account_id , sum(amount) as totalDeposit
from transactions
where transaction_type like 'deposit'
and transaction_date = '24-01-03'
group by transaction_id;
```

| transaction_id | account_id | totalDeposit |
|---|---|---|
| 1003 | 103 | 20000.00 |

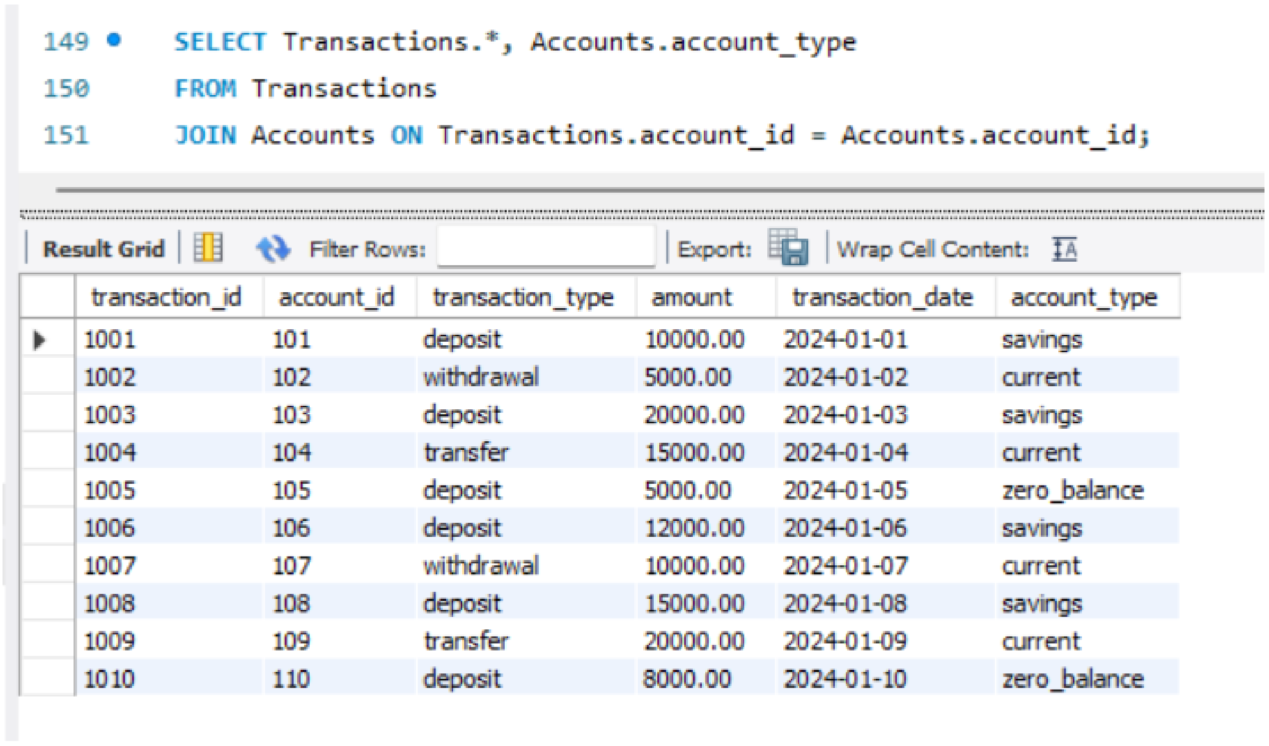
4. Write a SQL query to Find the Oldest and Newest Customers.



```sql
select min(dob) as oldest_customer ,
max(dob) as newest_customer
from customers;
```

| oldest_customer | newest_customer |
|---|---|
| 1980-07-03 | 1998-02-03 |

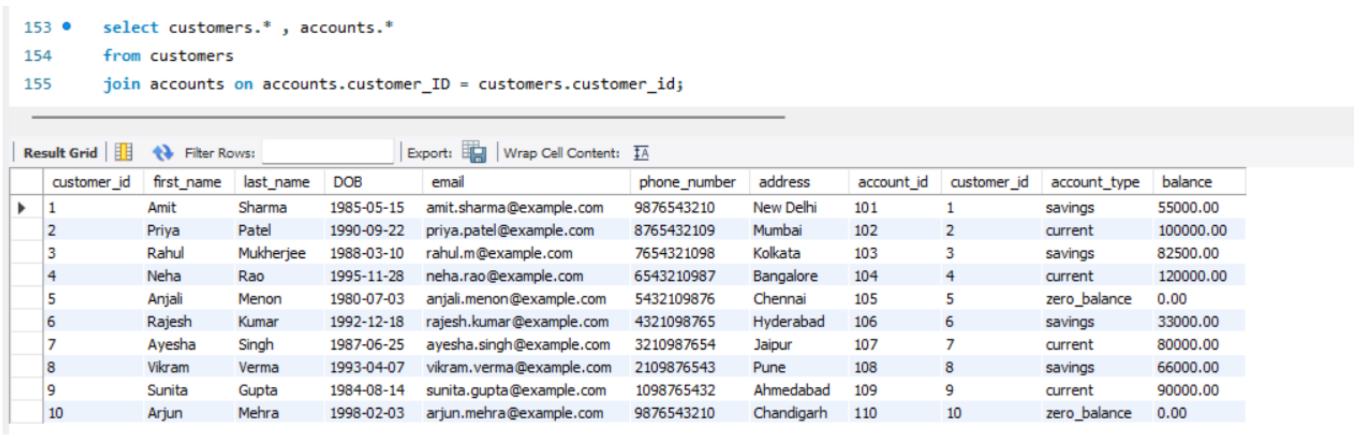
5. Write a SQL query to Retrieve transaction details along with the account type.



```sql
SELECT Transactions.*, Accounts.account_type
FROM Transactions
JOIN Accounts ON Transactions.account_id = Accounts.account_id;
```

| transaction_id | account_id | transaction_type | amount | transaction_date | account_type |
|---|---|---|---|---|---|
| 1001 | 101 | deposit | 10000.00 | 2024-01-01 | savings |
| 1002 | 102 | withdrawal | 5000.00 | 2024-01-02 | current |
| 1003 | 103 | deposit | 20000.00 | 2024-01-03 | savings |
| 1004 | 104 | transfer | 15000.00 | 2024-01-04 | current |
| 1005 | 105 | deposit | 5000.00 | 2024-01-05 | zero_balance |
| 1006 | 106 | deposit | 12000.00 | 2024-01-06 | savings |
| 1007 | 107 | withdrawal | 10000.00 | 2024-01-07 | current |
| 1008 | 108 | deposit | 15000.00 | 2024-01-08 | savings |
| 1009 | 109 | transfer | 20000.00 | 2024-01-09 | current |
| 1010 | 110 | deposit | 8000.00 | 2024-01-10 | zero_balance |

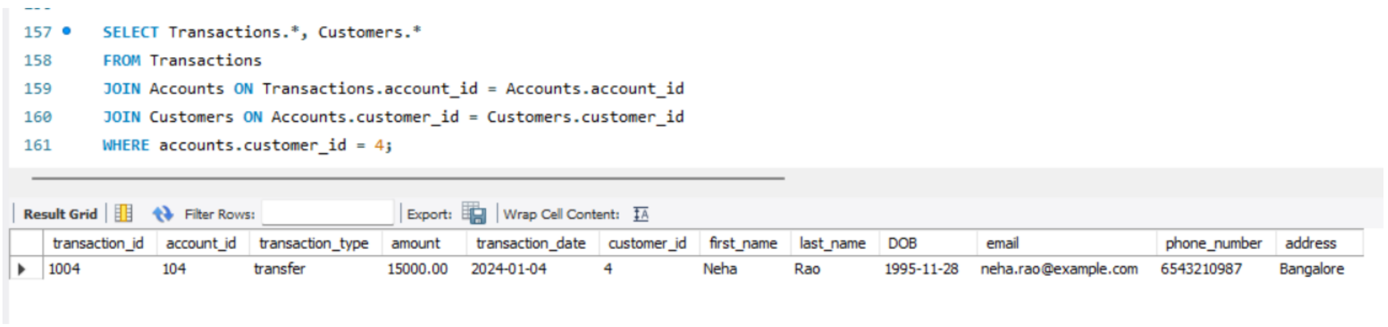
6. Write a SQL query to Get a list of customers along with their account details.



```sql
select customers.* , accounts.*
from customers
join accounts on accounts.customer_ID = customers.customer_id;
```

| customer_id | first_name | last_name | DOB | email | phone_number | address | account_id | customer_id | account_type | balance |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Amit | Sharma | 1985-05-15 | amit.sharma@example.com | 9876543210 | New Delhi | 101 | 1 | savings | 55000.00 |
| 2 | Priya | Patel | 1990-09-22 | priya.patel@example.com | 8765432109 | Mumbai | 102 | 2 | current | 100000.00 |
| 3 | Rahul | Mukherjee | 1988-03-10 | rahul.m@example.com | 7654321098 | Kolkata | 103 | 3 | savings | 82500.00 |
| 4 | Neha | Rao | 1995-11-28 | neha.rao@example.com | 6543210987 | Bangalore | 104 | 4 | current | 120000.00 |
| 5 | Anjali | Menon | 1980-07-03 | anjali.menon@example.com | 5432109876 | Chennai | 105 | 5 | zero_balance | 0.00 |
| 6 | Rajesh | Kumar | 1992-12-18 | rajesh.kumar@example.com | 4321098765 | Hyderabad | 106 | 6 | savings | 33000.00 |
| 7 | Ayesha | Singh | 1987-06-25 | ayesha.singh@example.com | 3210987654 | Jaipur | 107 | 7 | current | 80000.00 |
| 8 | Vikram | Verma | 1993-04-07 | vikram.verma@example.com | 2109876543 | Pune | 108 | 8 | savings | 66000.00 |
| 9 | Sunita | Gupta | 1984-08-14 | sunita.gupta@example.com | 1098765432 | Ahmedabad | 109 | 9 | current | 90000.00 |
| 10 | Arjun | Mehra | 1998-02-03 | arjun.mehra@example.com | 9876543210 | Chandigarh | 110 | 10 | zero_balance | 0.00 |

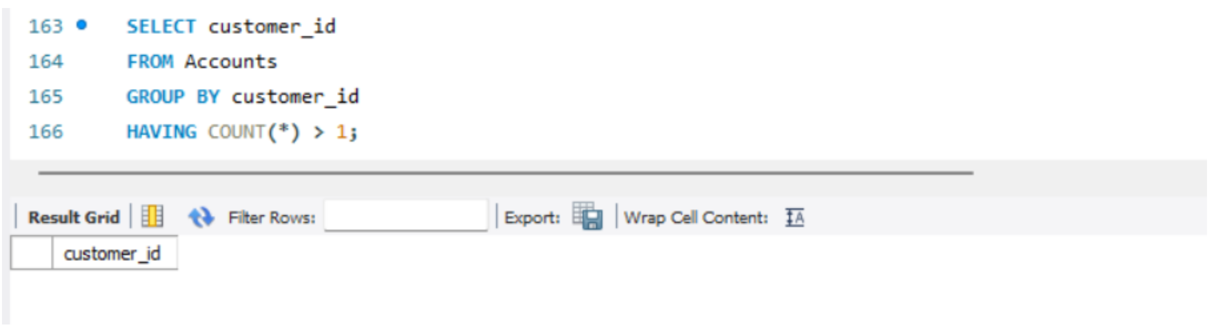
7. Write a SQL query to Retrieve transaction details along with customer information for aspecific account.



```sql
SELECT Transactions.*, Customers.*
FROM Transactions
JOIN Accounts ON Transactions.account_id = Accounts.account_id
JOIN Customers ON Accounts.customer_id = Customers.customer_id
WHERE accounts.customer_id = 4;
```

| transaction_id | account_id | transaction_type | amount | transaction_date | customer_id | first_name | last_name | DOB | email | phone_number | address |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1004 | 104 | transfer | 15000.00 | 2024-01-04 | 4 | Neha | Rao | 1995-11-28 | neha.rao@example.com | 6543210987 | Bangalore |

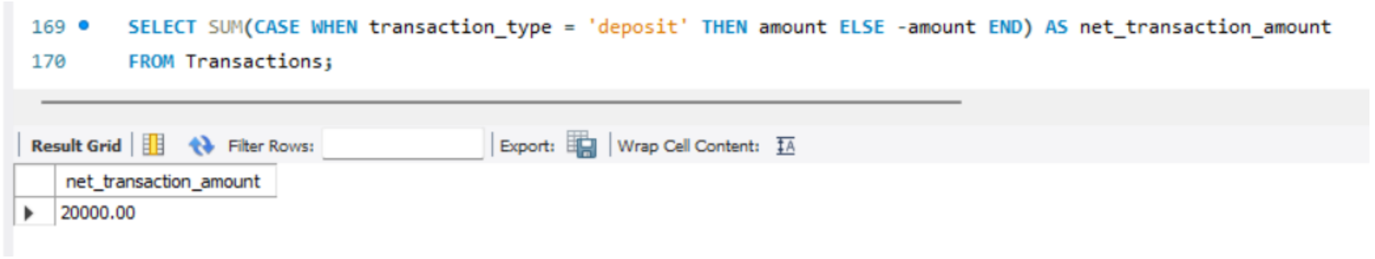
8. Write a SQL query to Identify customers who have more than one account.



```sql
SELECT customer_id
FROM Accounts
GROUP BY customer_id
HAVING COUNT(*) > 1;
```

| customer_id |
|---|

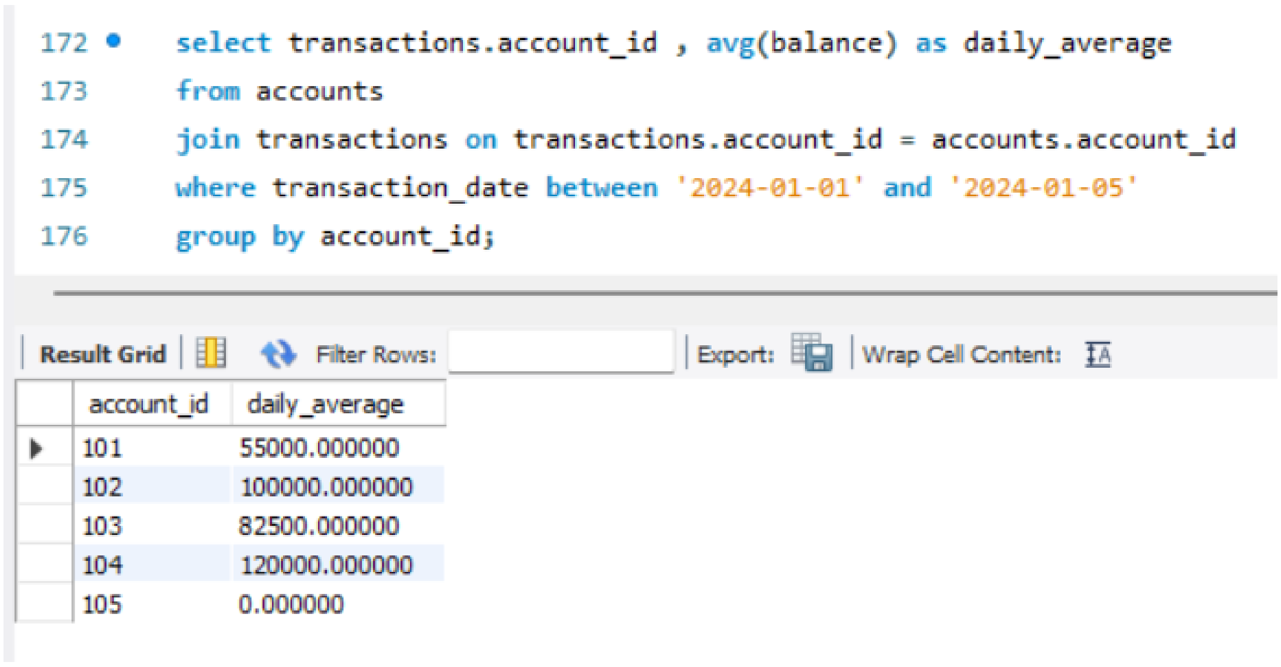
9. Write a SQL query to Calculate the difference in transaction amounts between deposits and withdrawals.



```sql
SELECT SUM(CASE WHEN transaction_type = 'deposit' THEN amount ELSE -amount END) AS net_transaction_amount
FROM Transactions;
```

| net_transaction_amount |
|---|
| 20000.00 |

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

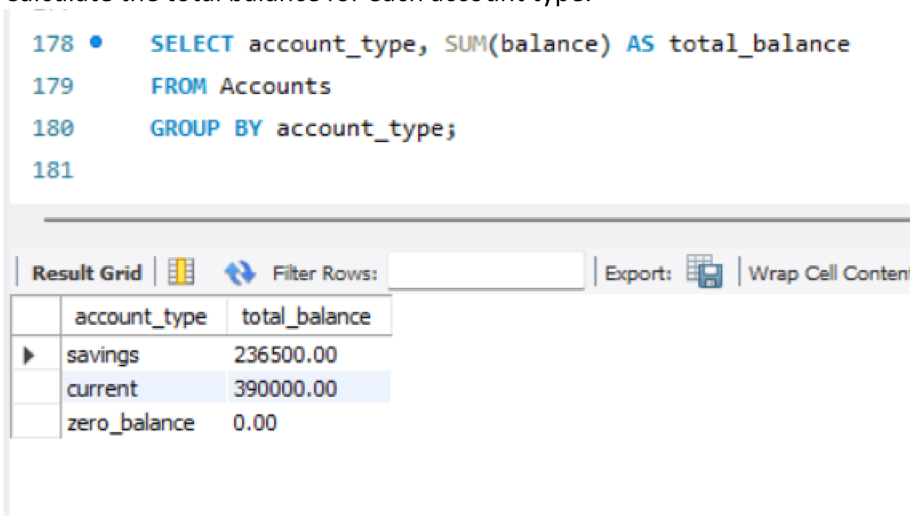


```sql
select transactions.account_id , avg(balance) as daily_average
from accounts
join transactions on transactions.account_id = accounts.account_id
where transaction_date between '2024-01-01' and '2024-01-05'
group by account_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| account_id | daily_average |
|---|---|
| 101 | 55000.000000 |
| 102 | 100000.000000 |
| 103 | 82500.000000 |
| 104 | 120000.000000 |
| 105 | 0.000000 |

11. Calculate the total balance for each account type.

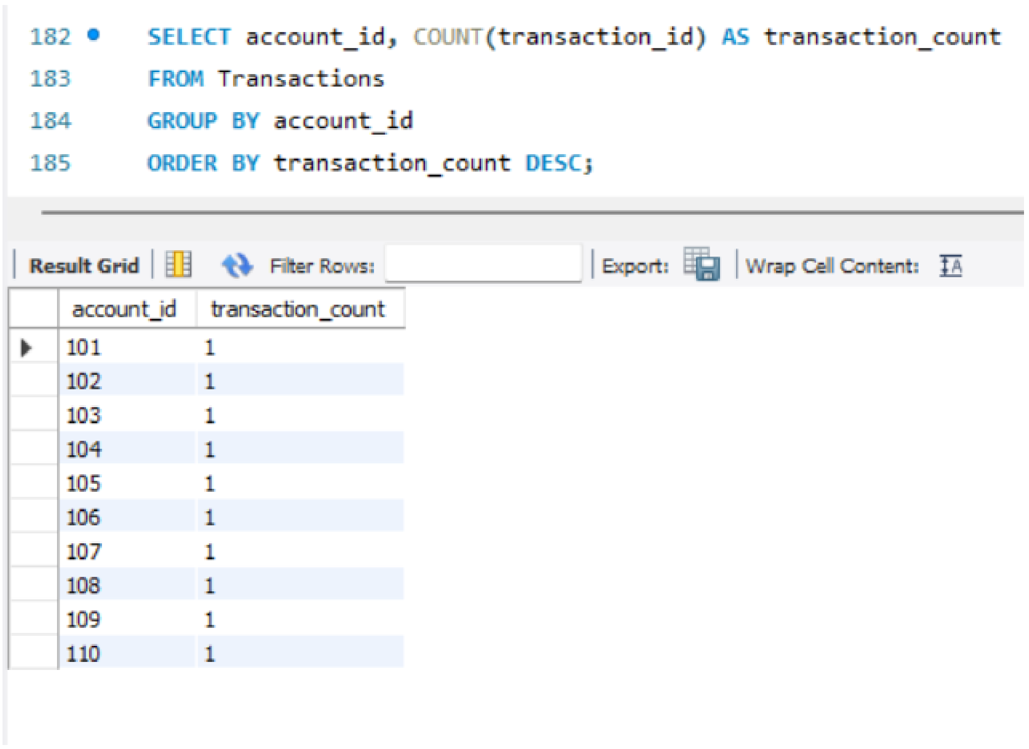


```sql
SELECT account_type, SUM(balance) AS total_balance
FROM Accounts
GROUP BY account_type;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content

| account_type | total_balance |
|---|---|
| savings | 236500.00 |
| current | 390000.00 |
| zero_balance | 0.00 |

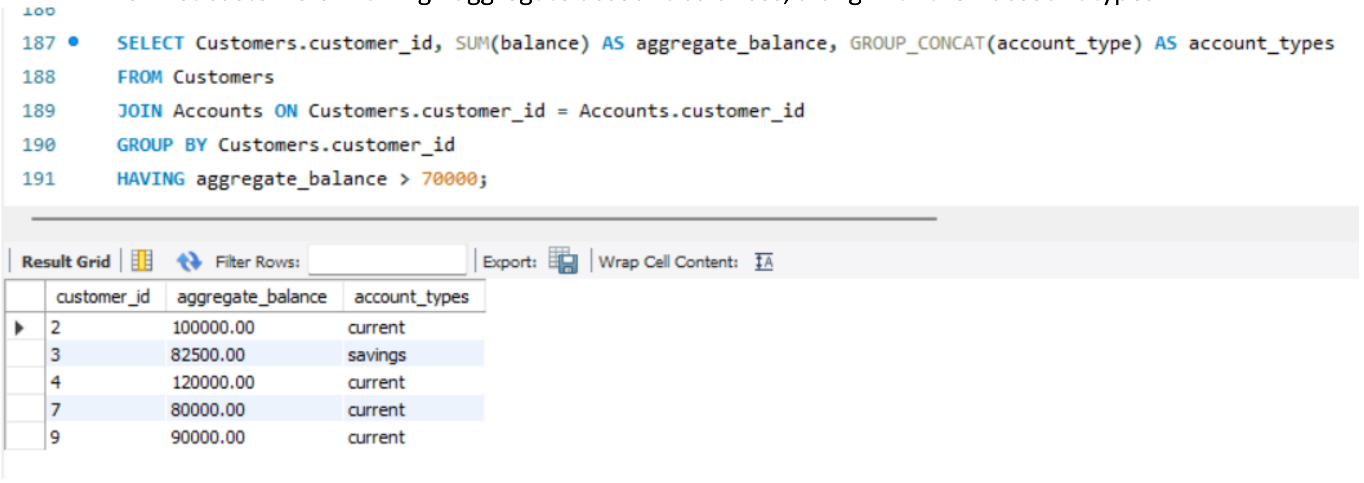
12. Identify accounts with the highest number of transactions order by descending order.



```sql
SELECT account_id, COUNT(transaction_id) AS transaction_count
FROM Transactions
GROUP BY account_id
ORDER BY transaction_count DESC;
```

| account_id | transaction_count |
|---|---|
| 101 | 1 |
| 102 | 1 |
| 103 | 1 |
| 104 | 1 |
| 105 | 1 |
| 106 | 1 |
| 107 | 1 |
| 108 | 1 |
| 109 | 1 |
| 110 | 1 |

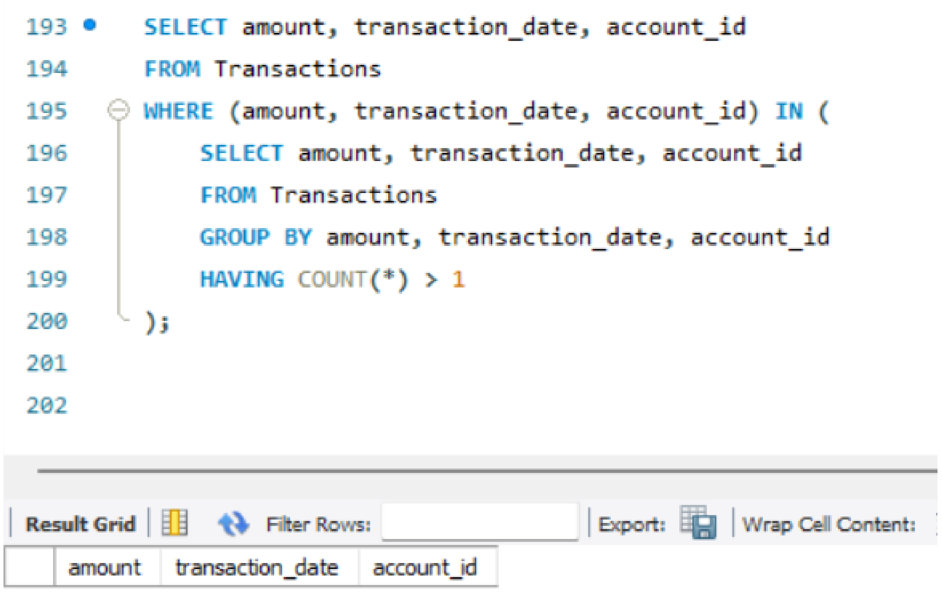
13. List customers with high aggregate account balances, along with their account types.



```sql
SELECT Customers.customer_id, SUM(balance) AS aggregate_balance, GROUP_CONCAT(account_type) AS account_types
FROM Customers
JOIN Accounts ON Customers.customer_id = Accounts.customer_id
GROUP BY Customers.customer_id
HAVING aggregate_balance > 70000;
```

| customer_id | aggregate_balance | account_types |
|---|---|---|
| 2 | 100000.00 | current |
| 3 | 82500.00 | savings |
| 4 | 120000.00 | current |
| 7 | 80000.00 | current |
| 9 | 90000.00 | current |

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

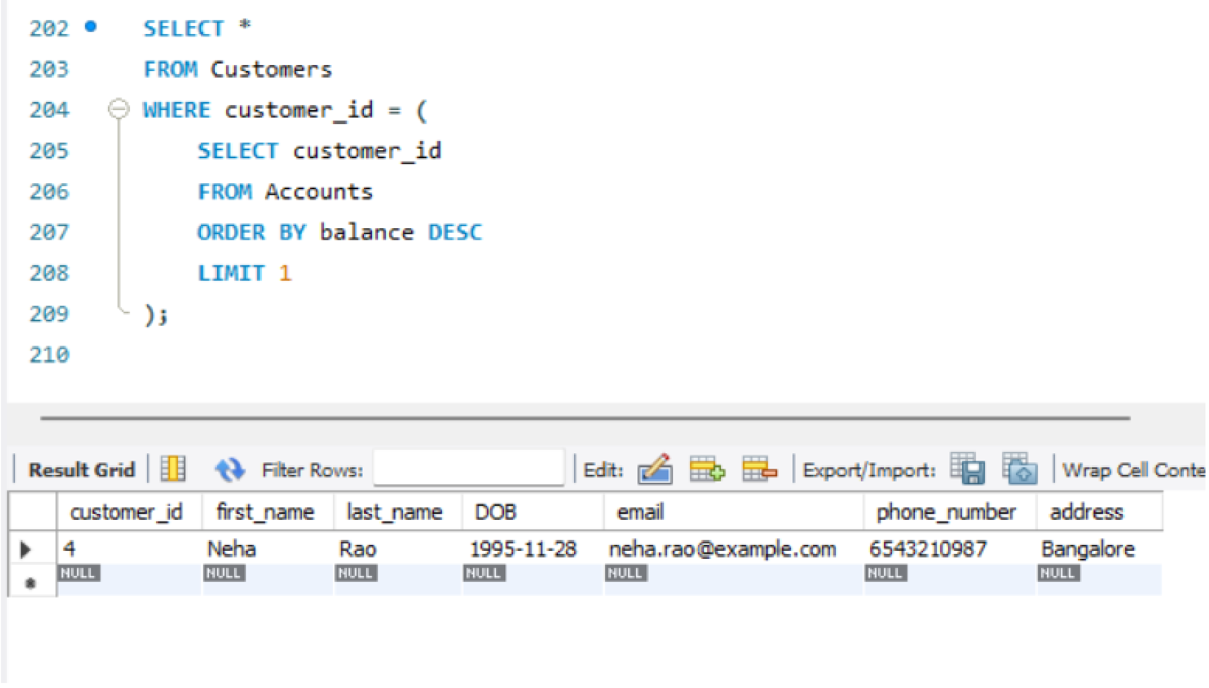


```sql
SELECT amount, transaction_date, account_id
FROM Transactions
WHERE (amount, transaction_date, account_id) IN (
    SELECT amount, transaction_date, account_id
    FROM Transactions
    GROUP BY amount, transaction_date, account_id
    HAVING COUNT(*) > 1
);
```

| amount | transaction_date | account_id |
|---|---|---|

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

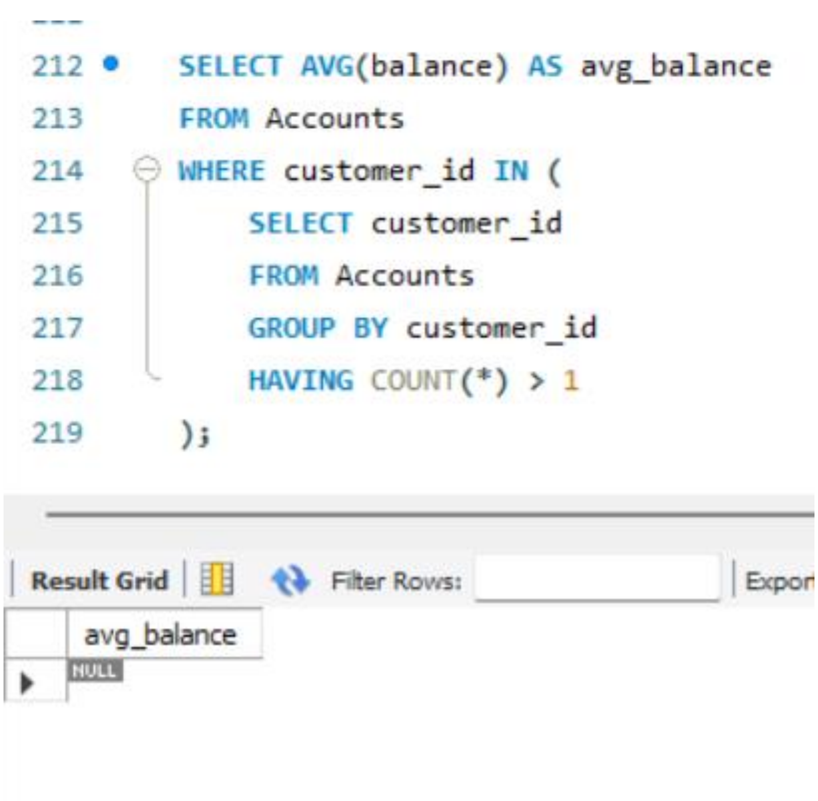
1. Retrieve the customer(s) with the highest account balance.



```sql
SELECT *
FROM Customers
WHERE customer_id = (
    SELECT customer_id
    FROM Accounts
    ORDER BY balance DESC
    LIMIT 1
);
```

| customer_id | first_name | last_name | DOB | email | phone_number | address |
|---|---|---|---|---|---|---|
| 4 | Neha | Rao | 1995-11-28 | neha.rao@example.com | 6543210987 | Bangalore |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |

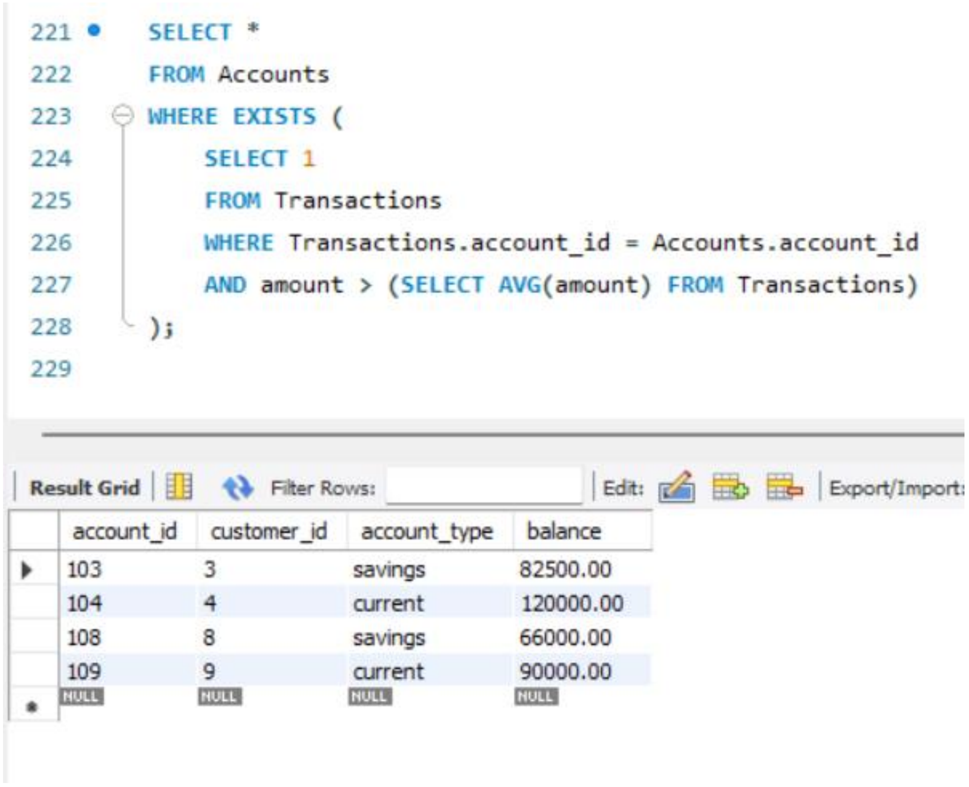
2. Calculate the average account balance for customers who have more than one account.



```
SELECT AVG(balance) AS avg_balance
FROM Accounts
WHERE customer_id IN (
    SELECT customer_id
    FROM Accounts
    GROUP BY customer_id
    HAVING COUNT(*) > 1
);
```

```
avg_balance
NULL
```

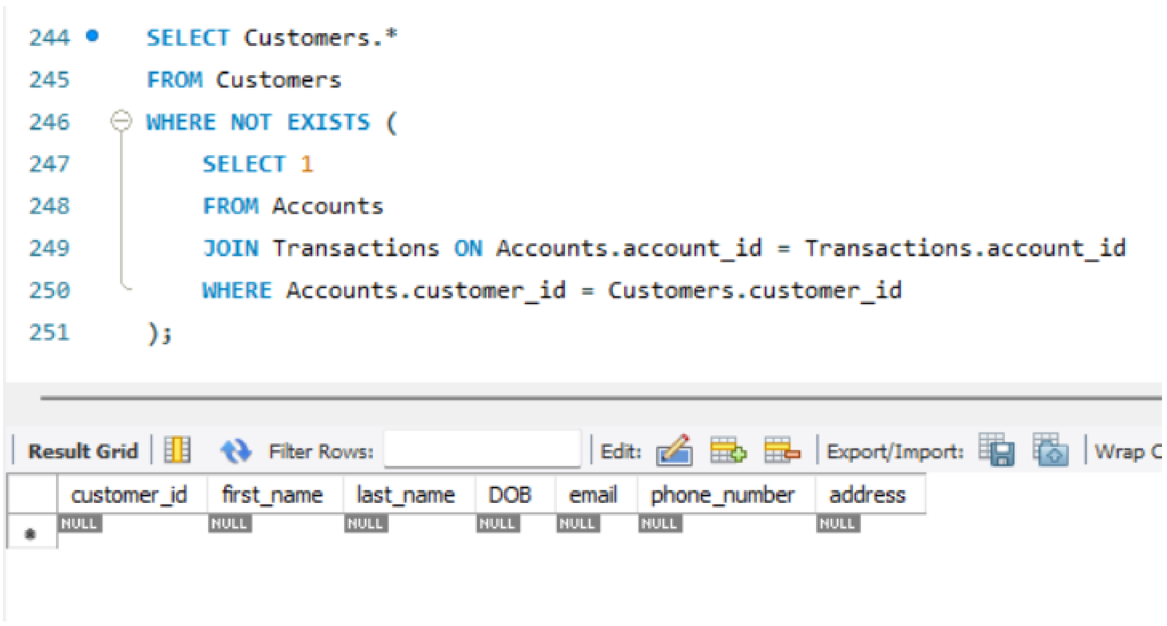
3. Retrieve accounts with transactions whose amounts exceed the average transaction amount.



```
SELECT *
FROM Accounts
WHERE EXISTS (
    SELECT 1
    FROM Transactions
    WHERE Transactions.account_id = Accounts.account_id
    AND amount > (SELECT AVG(amount) FROM Transactions)
);
```

```
account_id  customer_id  account_type  balance
103         3            savings       82500.00
104         4            current       120000.00
108         8            savings       66000.00
109         9            current       90000.00
NULL        NULL         NULL          NULL
```

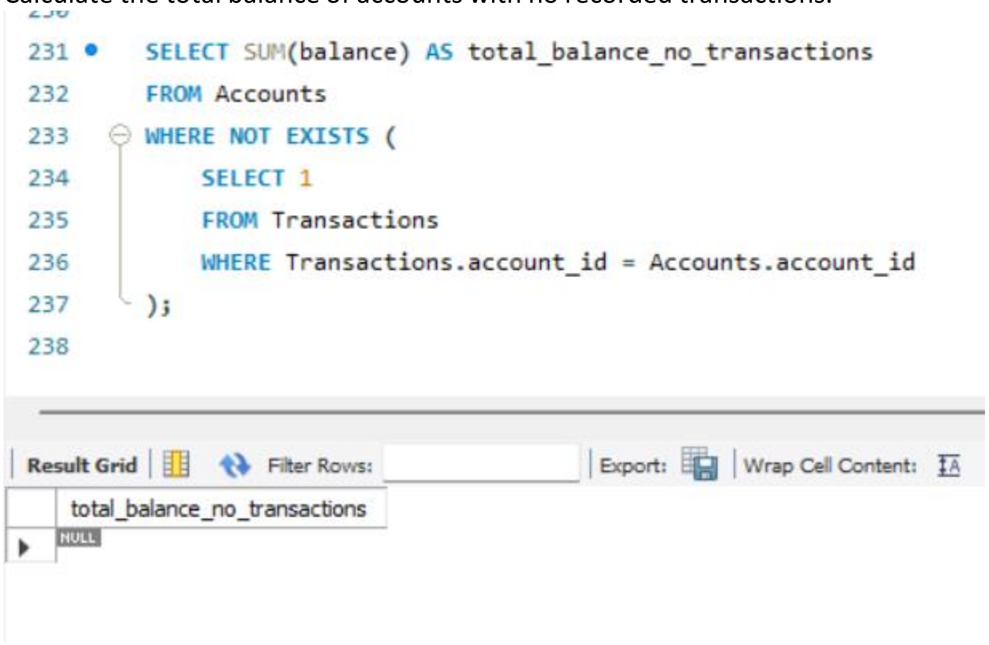
4. Identify customers who have no recorded transactions.



```sql
SELECT Customers.*
FROM Customers
WHERE NOT EXISTS (
    SELECT 1
    FROM Accounts
    JOIN Transactions ON Accounts.account_id = Transactions.account_id
    WHERE Accounts.customer_id = Customers.customer_id
);
```

| customer_id | first_name | last_name | DOB | email | phone_number | address |
|---|---|---|---|---|---|---|
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |

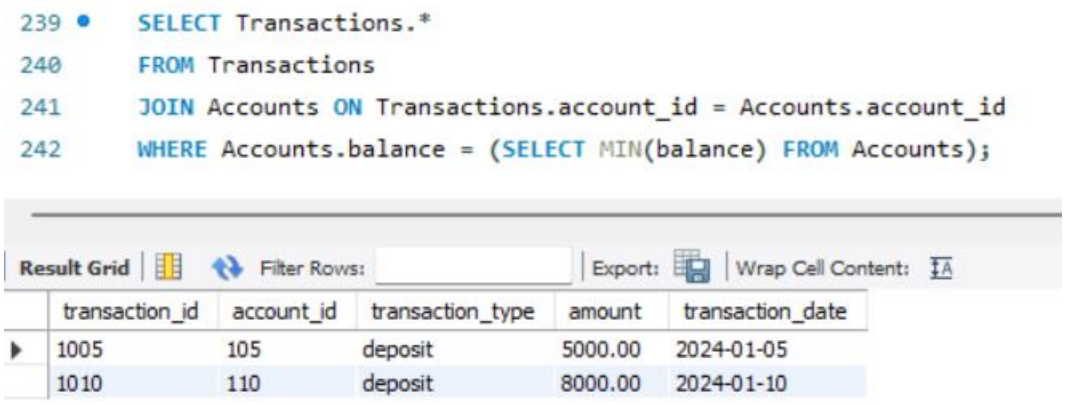
5. Calculate the total balance of accounts with no recorded transactions.



```sql
SELECT SUM(balance) AS total_balance_no_transactions
FROM Accounts
WHERE NOT EXISTS (
    SELECT 1
    FROM Transactions
    WHERE Transactions.account_id = Accounts.account_id
);
```

| total_balance_no_transactions |
|---|
| NULL |

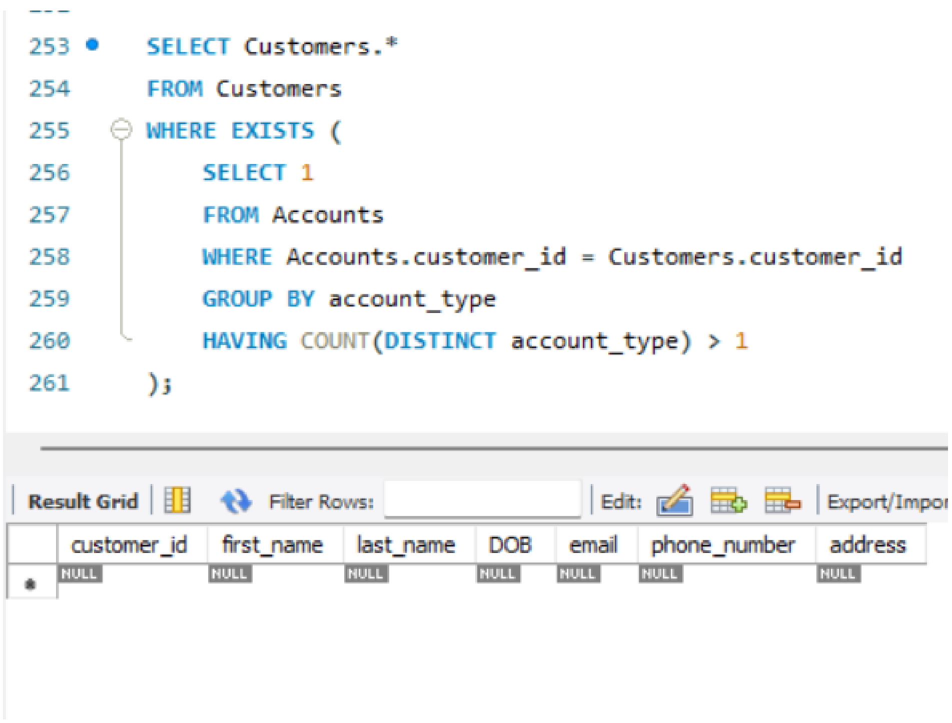
6. Retrieve transactions for accounts with the lowest balance.



```sql
SELECT Transactions.*
FROM Transactions
JOIN Accounts ON Transactions.account_id = Accounts.account_id
WHERE Accounts.balance = (SELECT MIN(balance) FROM Accounts);
```

| transaction_id | account_id | transaction_type | amount | transaction_date |
|---|---|---|---|---|
| 1005 | 105 | deposit | 5000.00 | 2024-01-05 |
| 1010 | 110 | deposit | 8000.00 | 2024-01-10 |

7. Identify customers who have accounts of multiple types.

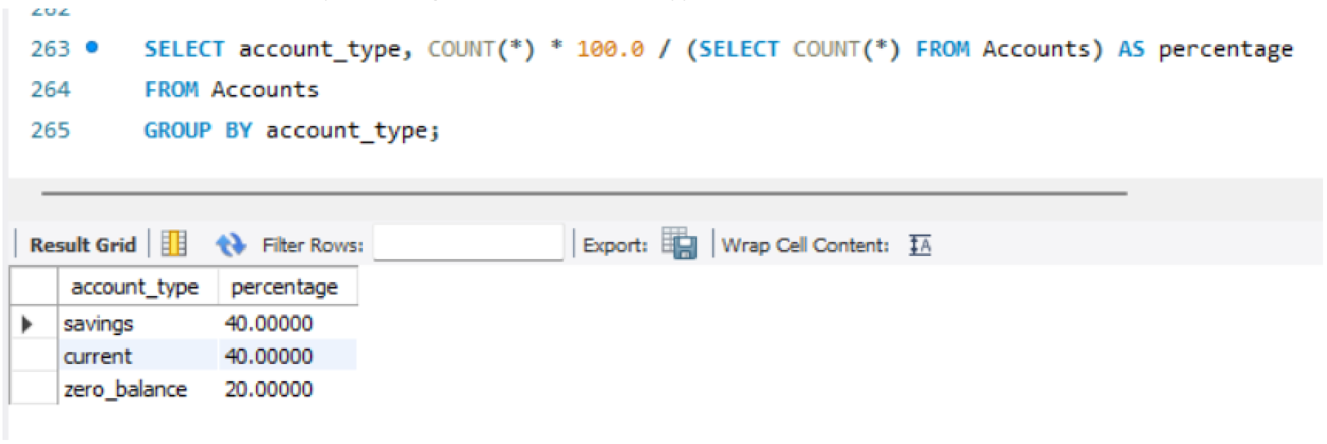


```sql
SELECT Customers.*
FROM Customers
WHERE EXISTS (
    SELECT 1
    FROM Accounts
    WHERE Accounts.customer_id = Customers.customer_id
    GROUP BY account_type
    HAVING COUNT(DISTINCT account_type) > 1
);
```

Result Grid | Filter Rows: | Edit: | Export/Impo

| | customer_id | first_name | last_name | DOB | email | phone_number | address |
|---|---|---|---|---|---|---|---|
| * | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

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

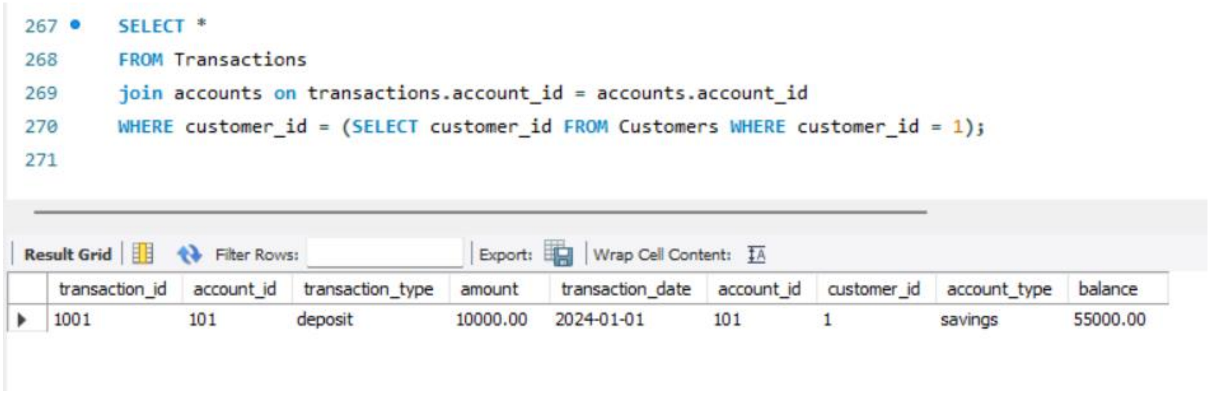


```sql
SELECT account_type, COUNT(*) * 100.0 / (SELECT COUNT(*) FROM Accounts) AS percentage
FROM Accounts
GROUP BY account_type;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| account_type | percentage |
|---|---|
| savings | 40.00000 |
| current | 40.00000 |
| zero_balance | 20.00000 |

9. Retrieve all transactions for a customer with a given customer_id.

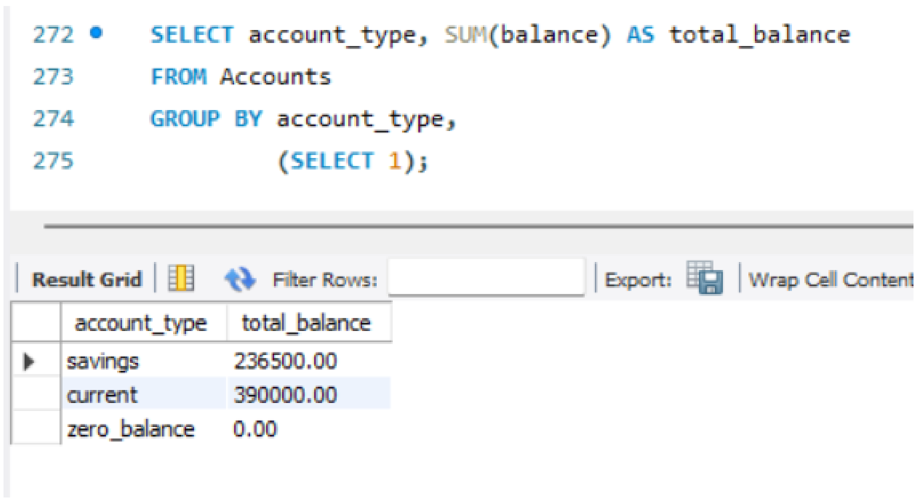


```sql
SELECT *
FROM Transactions
join accounts on transactions.account_id = accounts.account_id
WHERE customer_id = (SELECT customer_id FROM Customers WHERE customer_id = 1);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| transaction_id | account_id | transaction_type | amount | transaction_date | account_id | customer_id | account_type | balance |
|---|---|---|---|---|---|---|---|---|
| 1001 | 101 | deposit | 10000.00 | 2024-01-01 | 101 | 1 | savings | 55000.00 |

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


```sql
SELECT account_type, SUM(balance) AS total_balance
FROM Accounts
GROUP BY account_type,
         (SELECT 1);
```

| account_type | total_balance |
|---|---|
| savings | 236500.00 |
| current | 390000.00 |
| zero_balance | 0.00 |