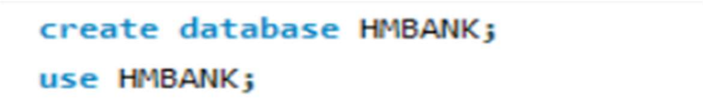
ASSIGNMENT

SUBMITTED BY: Fahimunnisha A

TOPIC: Banking System

Tasks 1: Database Design:

1.Create the database named "HMBank"



```
create database HMBANK;
use HMBANK;
```

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

Customers Table:

Column Name	Data Type	Constraints
CustomerID	INT	PRIMARY KEY, AUTO_INCREMENT
FirstName	VARCHAR(50)	NOT NULL
LastName	VARCHAR(50)	NOT NULL
DateOfBirth	DATE	NOT NULL
Email	VARCHAR(100)	UNIQUE
Phone	VARCHAR(15)	
Address	VARCHAR(255)	

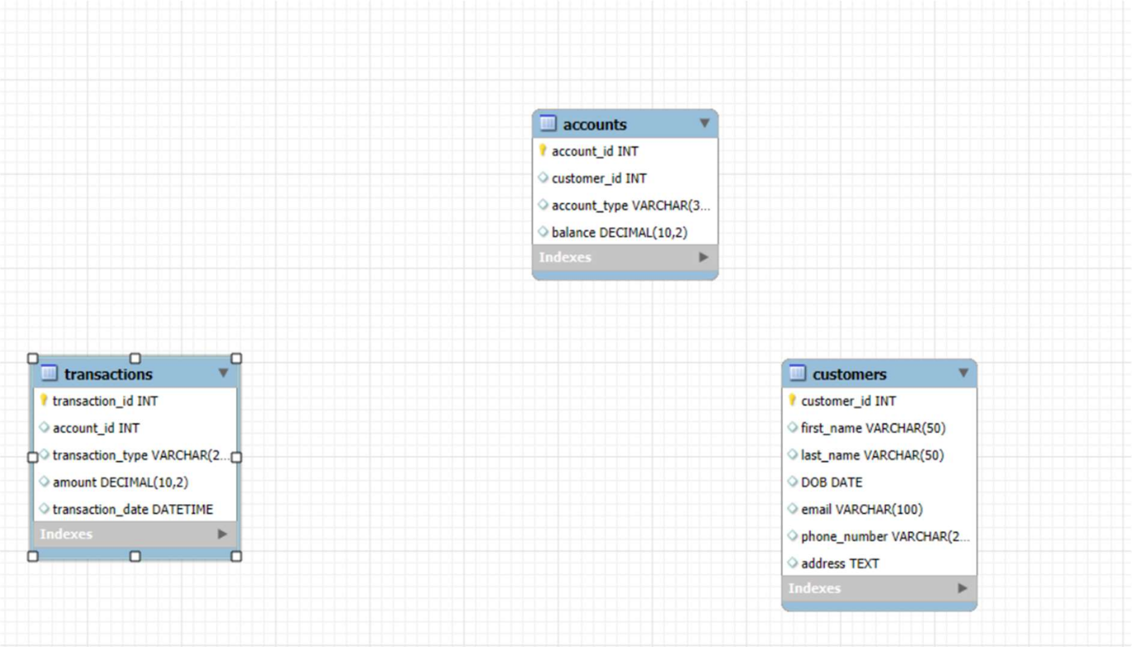
Accounts Table:

Column Name	Data type	Constraints
AccountID	INT	PRIMARY KEY, AUTO_INCREMENT
CustomerID	INT	FOREIGN KEY → Customers(CustomerID)
AccountType	VARCHAR(20)	NOT NULL
Balance	DECIMAL(12,2)	DEFAULT 0.00
OpenDate	DATE	NOT NULL

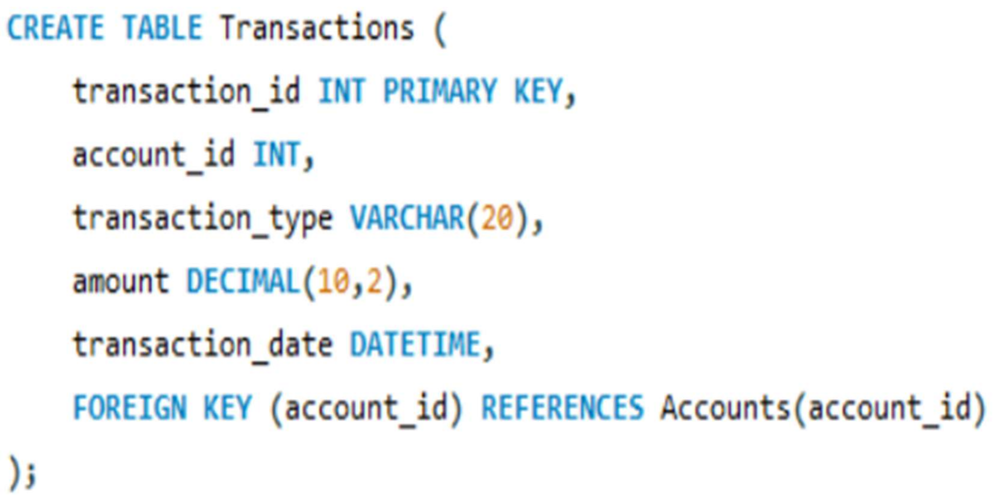
Transactions table:

Column Name	Data type	Constraints
TransactionID	INT	PRIMARY KEY, AUTO_INCREMENT
AccountID	INT	FOREIGN KEY → Accounts(AccountID)
TransactionType	VARCHAR(20)	NOT NULL (Deposit, Withdraw, etc.)
Amount	DECIMAL(12, 2)	NOT NULL
TransactionDate	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

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



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



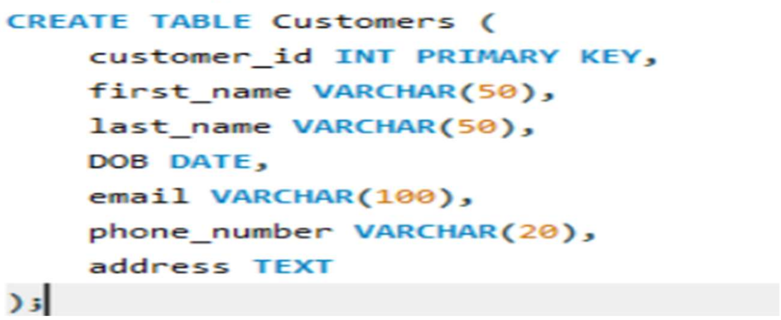
```
CREATE TABLE Transactions (
    transaction_id INT PRIMARY KEY,
    account_id INT,
    transaction_type VARCHAR(20),
    amount DECIMAL(10,2),
    transaction_date DATETIME,
    FOREIGN KEY (account_id) REFERENCES Accounts(account_id)
);
```

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

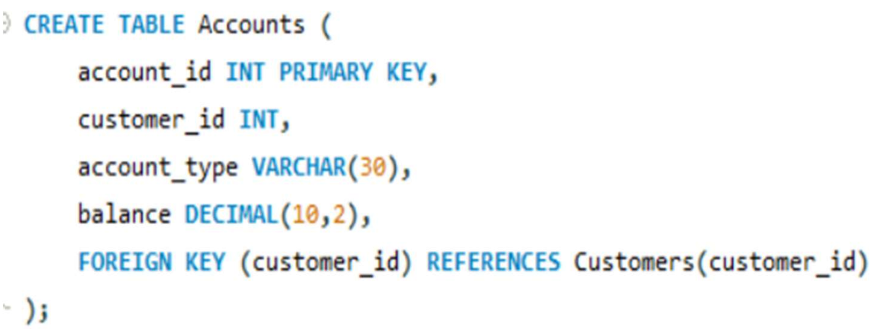
□Customers

□Accounts

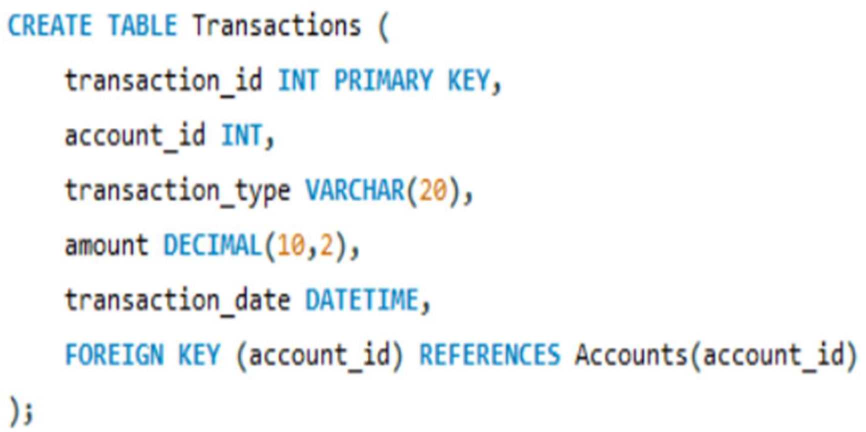
□Transactions



```
CREATE TABLE Customers (
    customer_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    DOB DATE,
    email VARCHAR(100),
    phone_number VARCHAR(20),
    address TEXT
);
```



```
CREATE TABLE Accounts (
    account_id INT PRIMARY KEY,
    customer_id INT,
    account_type VARCHAR(30),
    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(20),
    amount DECIMAL(10,2),
    transaction_date DATETIME,
    FOREIGN KEY (account_id) REFERENCES Accounts(account_id)
);
```

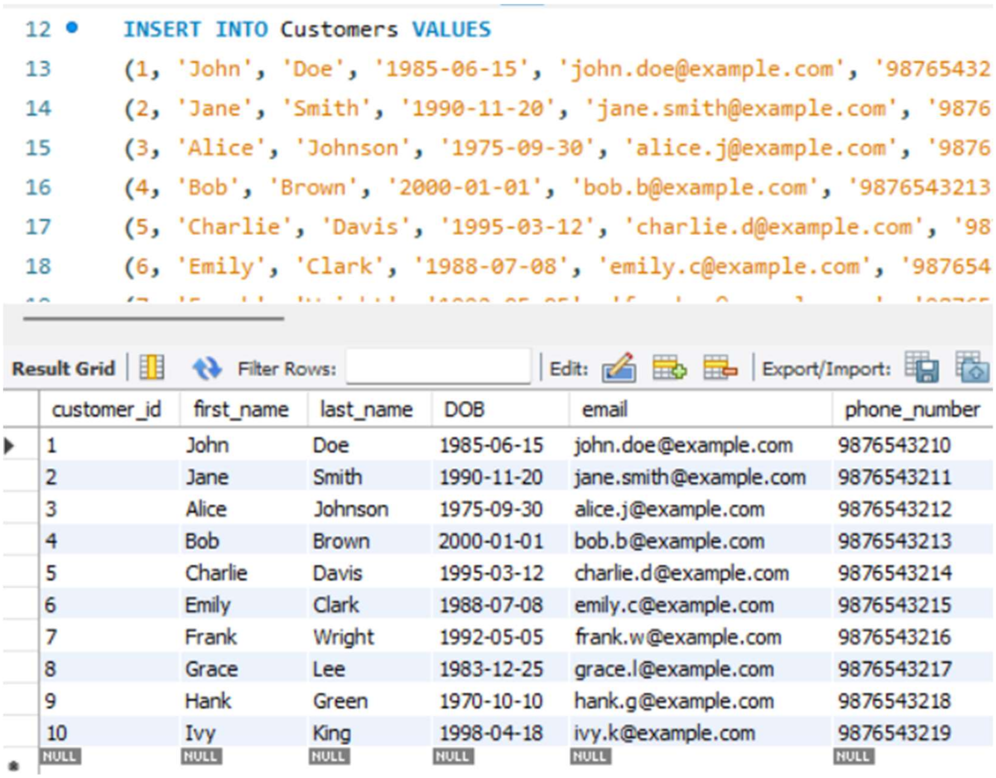
Tasks 2: Select, Where, Between, AND, LIKE:

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

□Customers

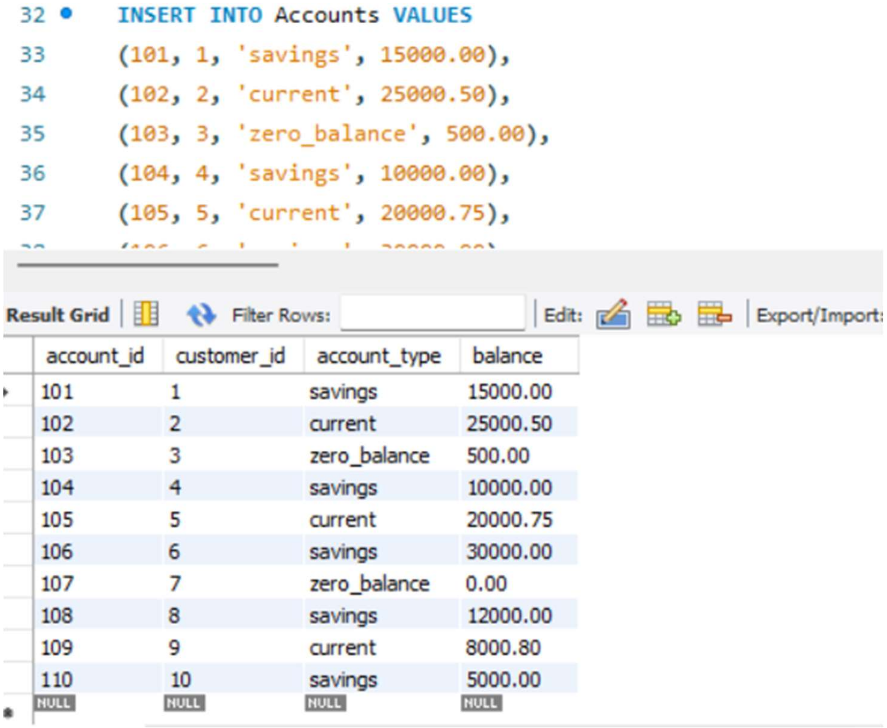
□Accounts

□Transactions



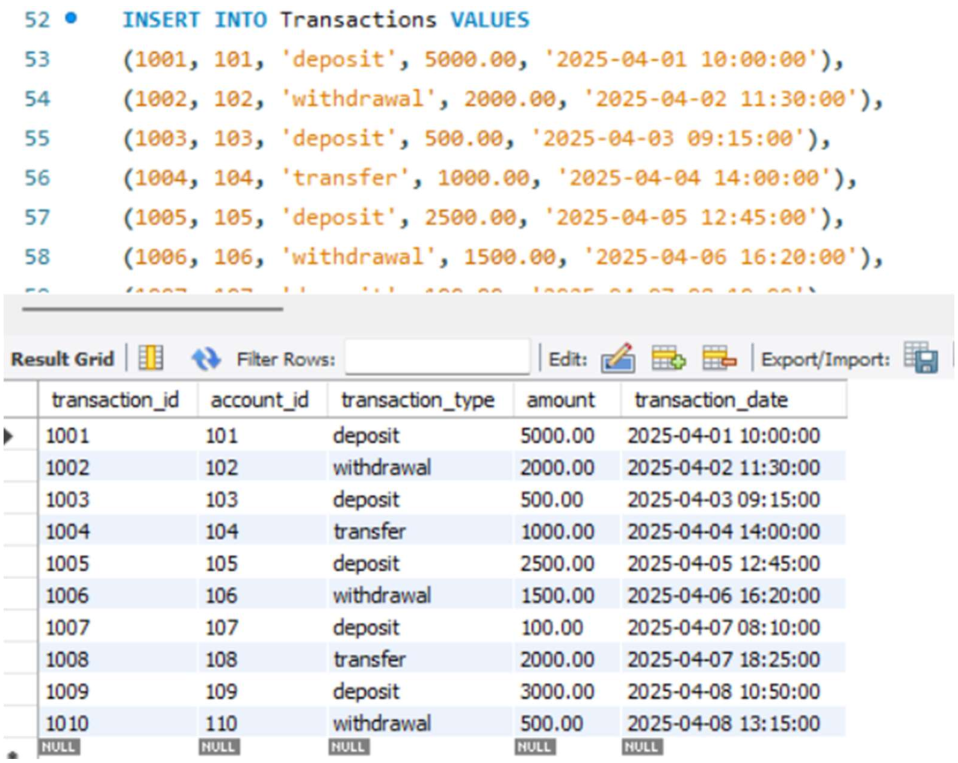
```
INSERT INTO Customers VALUES
(1, 'John', 'Doe', '1985-06-15', 'john.doe@example.com', '98765432
(2, 'Jane', 'Smith', '1990-11-20', 'jane.smith@example.com', '9876
(3, 'Alice', 'Johnson', '1975-09-30', 'alice.j@example.com', '9876
(4, 'Bob', 'Brown', '2000-01-01', 'bob.b@example.com', '9876543213
(5, 'Charlie', 'Davis', '1995-03-12', 'charlie.d@example.com', '98
(6, 'Emily', 'Clark', '1988-07-08', 'emily.c@example.com', '987654
```

customer_id	first_name	last_name	DOB	email	phone_number
1	John	Doe	1985-06-15	john.doe@example.com	9876543210
2	Jane	Smith	1990-11-20	jane.smith@example.com	9876543211
3	Alice	Johnson	1975-09-30	alice.j@example.com	9876543212
4	Bob	Brown	2000-01-01	bob.b@example.com	9876543213
5	Charlie	Davis	1995-03-12	charlie.d@example.com	9876543214
6	Emily	Clark	1988-07-08	emily.c@example.com	9876543215
7	Frank	Wright	1992-05-05	frank.w@example.com	9876543216
8	Grace	Lee	1983-12-25	grace.l@example.com	9876543217
9	Hank	Green	1970-10-10	hank.g@example.com	9876543218
10	Ivy	King	1998-04-18	ivy.k@example.com	9876543219
NULL	NULL	NULL	NULL	NULL	NULL



```
INSERT INTO Accounts VALUES
(101, 1, 'savings', 15000.00),
(102, 2, 'current', 25000.50),
(103, 3, 'zero_balance', 500.00),
(104, 4, 'savings', 10000.00),
(105, 5, 'current', 20000.75),
```

account_id	customer_id	account_type	balance
101	1	savings	15000.00
102	2	current	25000.50
103	3	zero_balance	500.00
104	4	savings	10000.00
105	5	current	20000.75
106	6	savings	30000.00
107	7	zero_balance	0.00
108	8	savings	12000.00
109	9	current	8000.80
110	10	savings	5000.00
NULL	NULL	NULL	NULL

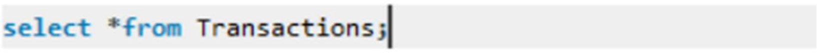


```
INSERT INTO Transactions VALUES
(1001, 101, 'deposit', 5000.00, '2025-04-01 10:00:00'),
(1002, 102, 'withdrawal', 2000.00, '2025-04-02 11:30:00'),
(1003, 103, 'deposit', 500.00, '2025-04-03 09:15:00'),
(1004, 104, 'transfer', 1000.00, '2025-04-04 14:00:00'),
(1005, 105, 'deposit', 2500.00, '2025-04-05 12:45:00'),
(1006, 106, 'withdrawal', 1500.00, '2025-04-06 16:20:00'),
```

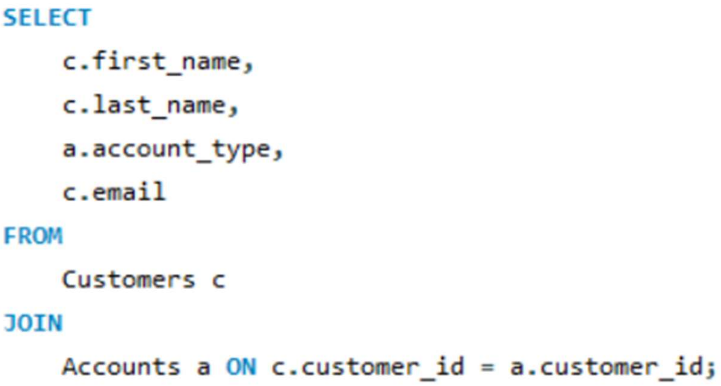
transaction_id	account_id	transaction_type	amount	transaction_date
1001	101	deposit	5000.00	2025-04-01 10:00:00
1002	102	withdrawal	2000.00	2025-04-02 11:30:00
1003	103	deposit	500.00	2025-04-03 09:15:00
1004	104	transfer	1000.00	2025-04-04 14:00:00
1005	105	deposit	2500.00	2025-04-05 12:45:00
1006	106	withdrawal	1500.00	2025-04-06 16:20:00
1007	107	deposit	100.00	2025-04-07 08:10:00
1008	108	transfer	2000.00	2025-04-07 18:25:00
1009	109	deposit	3000.00	2025-04-08 10:50:00
1010	110	withdrawal	500.00	2025-04-08 13:15:00
NULL	NULL	NULL	NULL	NULL

2. Write SQL queries for the following tasks:

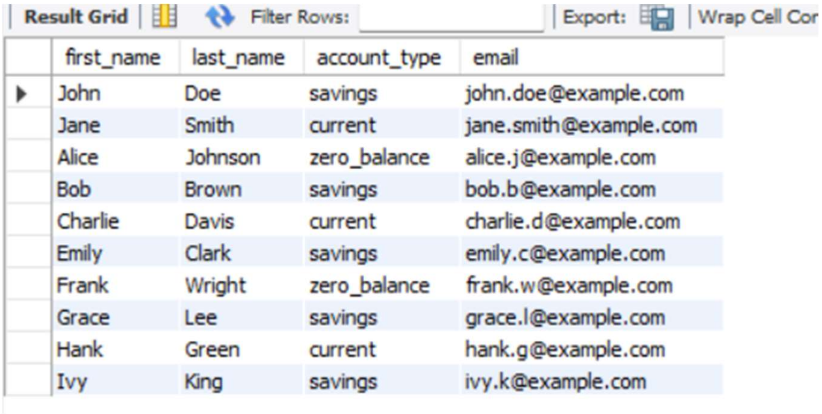
1. Write a SQL query to retrieve the name, account type and email of all customers.



```
select *from Transactions;
```

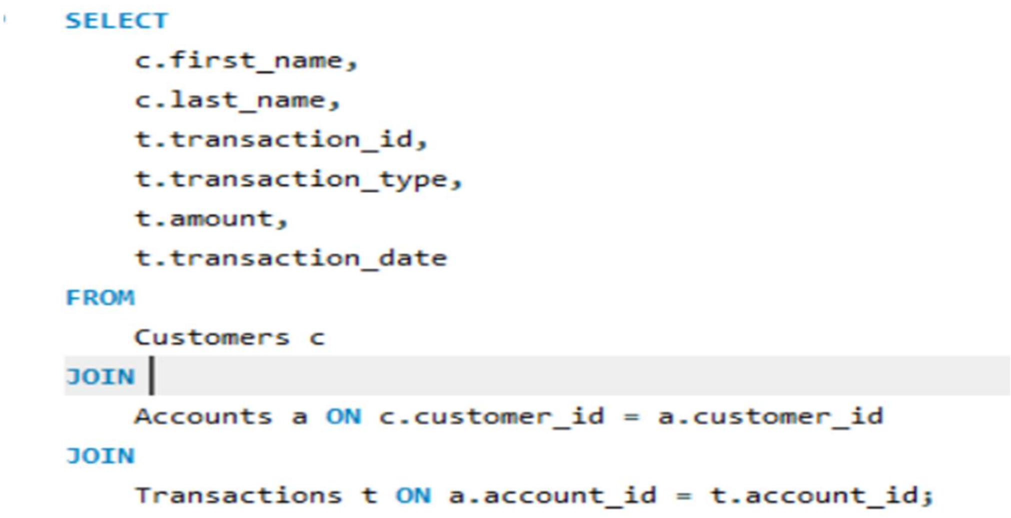


```
SELECT
    c.first_name,
    c.last_name,
    a.account_type,
    c.email
FROM
    Customers c
JOIN
    Accounts a ON c.customer_id = a.customer_id;
```

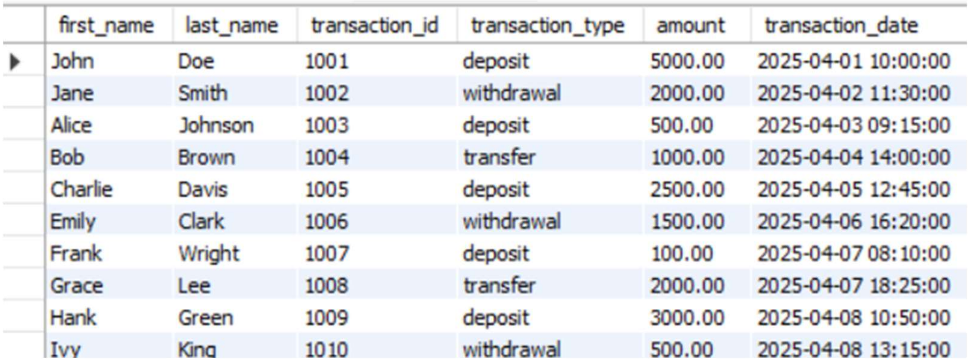


first_name	last_name	account_type	email
John	Doe	savings	john.doe@example.com
Jane	Smith	current	jane.smith@example.com
Alice	Johnson	zero_balance	alice.j@example.com
Bob	Brown	savings	bob.b@example.com
Charlie	Davis	current	charlie.d@example.com
Emily	Clark	savings	emily.c@example.com
Frank	Wright	zero_balance	frank.w@example.com
Grace	Lee	savings	grace.l@example.com
Hank	Green	current	hank.g@example.com
Ivy	King	savings	ivy.k@example.com

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

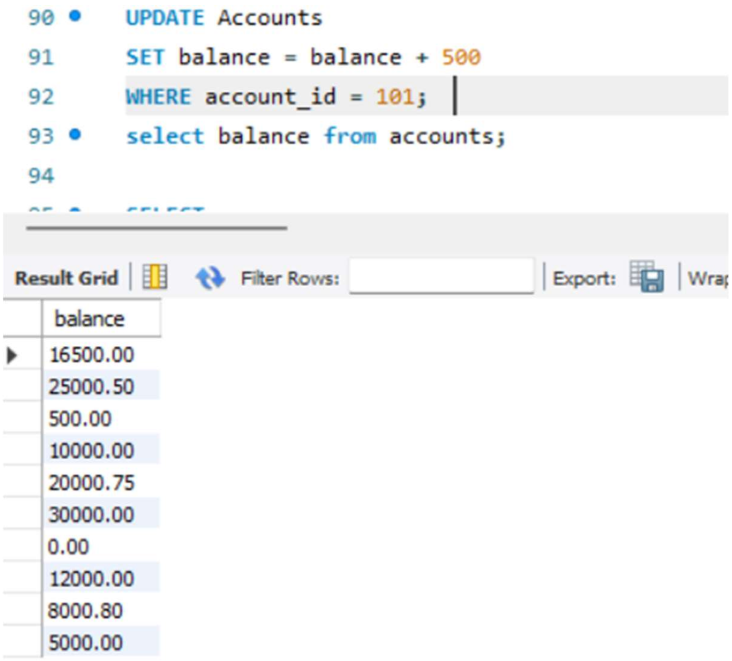


```
SELECT
    c.first_name,
    c.last_name,
    t.transaction_id,
    t.transaction_type,
    t.amount,
    t.transaction_date
FROM
    Customers c
JOIN
    Accounts a ON c.customer_id = a.customer_id
JOIN
    Transactions t ON a.account_id = t.account_id;
```



first_name	last_name	transaction_id	transaction_type	amount	transaction_date
John	Doe	1001	deposit	5000.00	2025-04-01 10:00:00
Jane	Smith	1002	withdrawal	2000.00	2025-04-02 11:30:00
Alice	Johnson	1003	deposit	500.00	2025-04-03 09:15:00
Bob	Brown	1004	transfer	1000.00	2025-04-04 14:00:00
Charlie	Davis	1005	deposit	2500.00	2025-04-05 12:45:00
Emily	Clark	1006	withdrawal	1500.00	2025-04-06 16:20:00
Frank	Wright	1007	deposit	100.00	2025-04-07 08:10:00
Grace	Lee	1008	transfer	2000.00	2025-04-07 18:25:00
Hank	Green	1009	deposit	3000.00	2025-04-08 10:50:00
Ivy	King	1010	withdrawal	500.00	2025-04-08 13:15:00

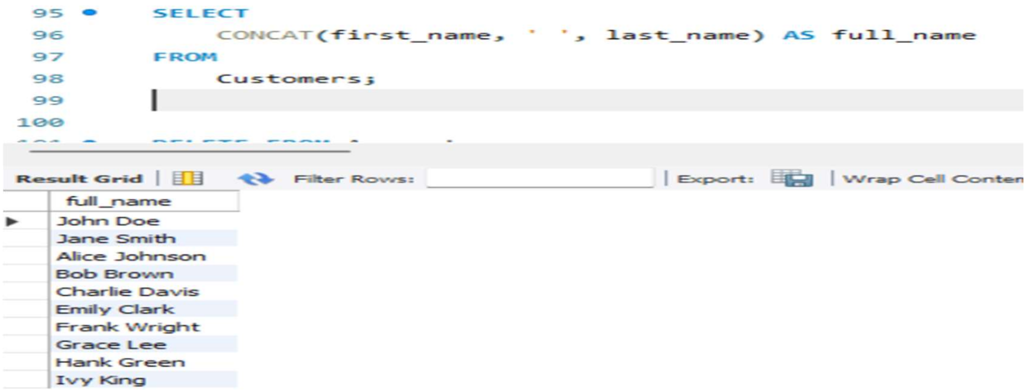
3. Write a SQL query to increase the balance of a specific account by a certain amount.



```sql
UPDATE Accounts
SET balance = balance + 500
WHERE account_id = 101;
select balance from accounts;
```

```
balance
16500.00
25000.50
500.00
10000.00
20000.75
30000.00
0.00
12000.00
8000.80
5000.00
```

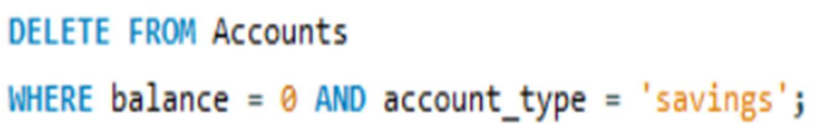
4. Write a SQL query to Combine first and last names of customers as a full_name.



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

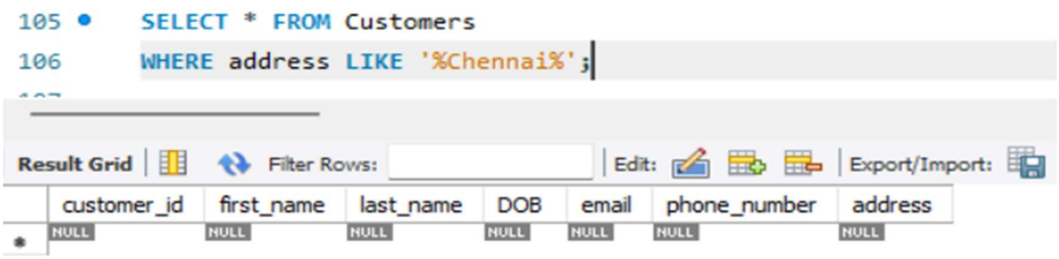
```
full_name
John Doe
Jane Smith
Alice Johnson
Bob Brown
Charlie Davis
Emily Clark
Frank Wright
Grace Lee
Hank Green
Ivy King
```

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';
```

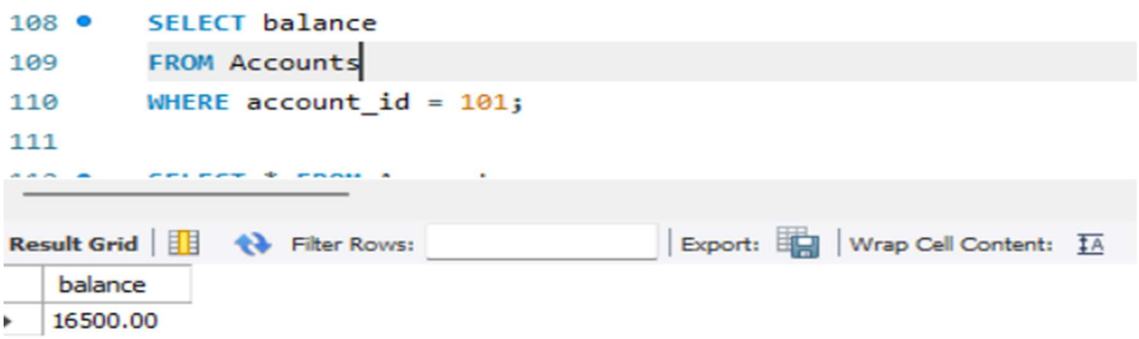
6. Write a SQL query to Find customers living in a specific city.



```
SELECT * FROM Customers
WHERE address LIKE '%Chennai%';
```

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

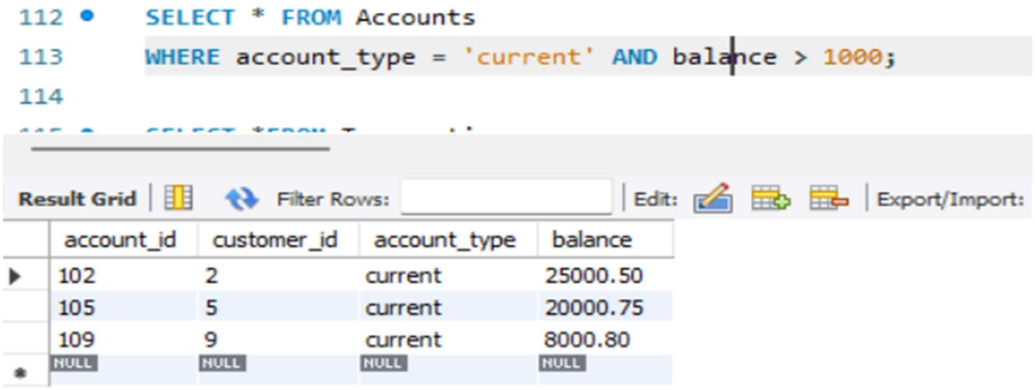
7. Write a SQL query to Get the account balance for a specific account.



```
SELECT balance
FROM Accounts
WHERE account_id = 101;
```

balance
16500.00

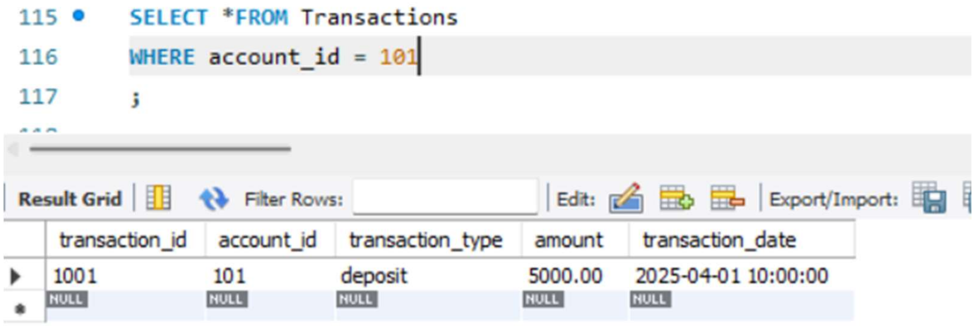
8. Write a SQL query to List all current accounts with a balance greater than $1,000.



```sql
SELECT * FROM Accounts
WHERE account_type = 'current' AND balance > 1000;
```

account_id	customer_id	account_type	balance
102	2	current	25000.50
105	5	current	20000.75
109	9	current	8000.80
NULL	NULL	NULL	NULL

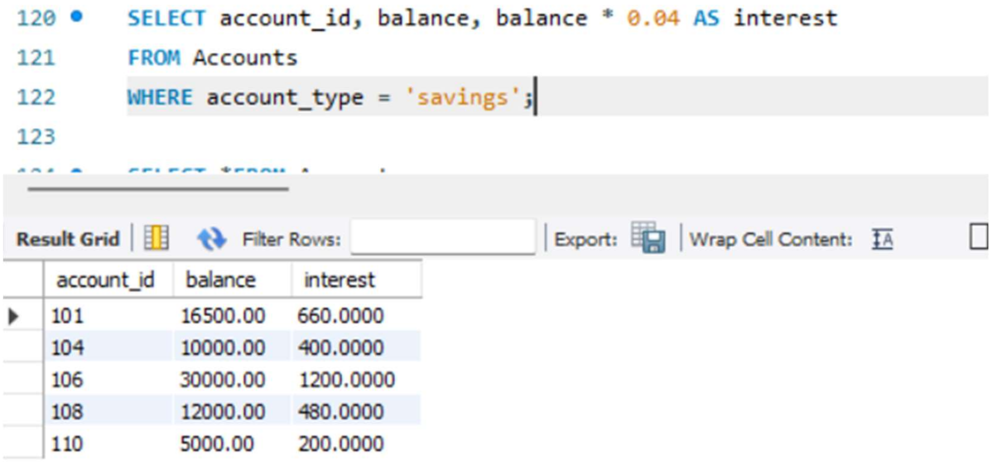
9. Write a SQL query to Retrieve all transactions for a specific account.



```sql
SELECT *FROM Transactions
WHERE account_id = 101
;
```

transaction_id	account_id	transaction_type	amount	transaction_date
1001	101	deposit	5000.00	2025-04-01 10:00:00
NULL	NULL	NULL	NULL	NULL

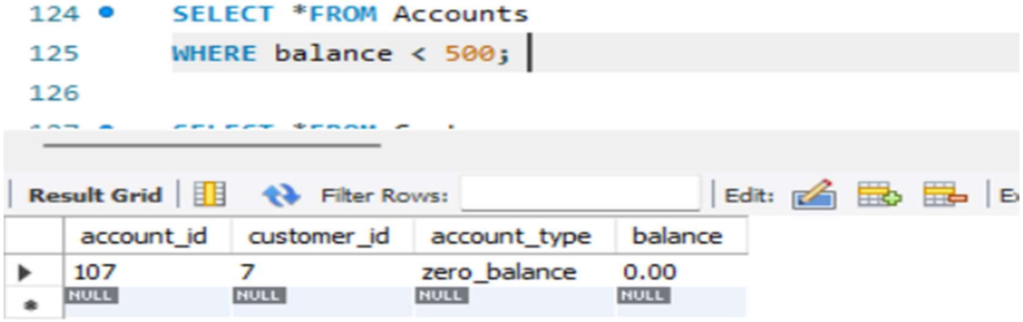
10. Write a SQL query to Calculate the interest accrued on savings accounts based on a given interest rate.



```sql
SELECT account_id, balance, balance * 0.04 AS interest
FROM Accounts
WHERE account_type = 'savings';
```

account_id	balance	interest
101	16500.00	660.0000
104	10000.00	400.0000
106	30000.00	1200.0000
108	12000.00	480.0000
110	5000.00	200.0000

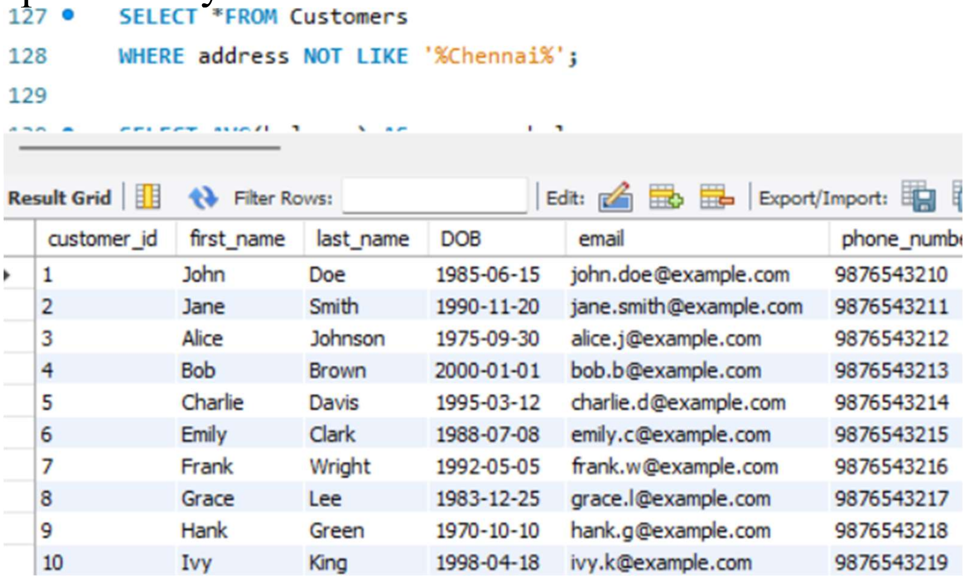
11. Write a SQL query to Identify accounts where the balance is less than a specified overdraft limit.



```sql
SELECT *FROM Accounts
WHERE balance < 500;
```

account_id	customer_id	account_type	balance
107	7	zero_balance	0.00
NULL	NULL	NULL	NULL

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

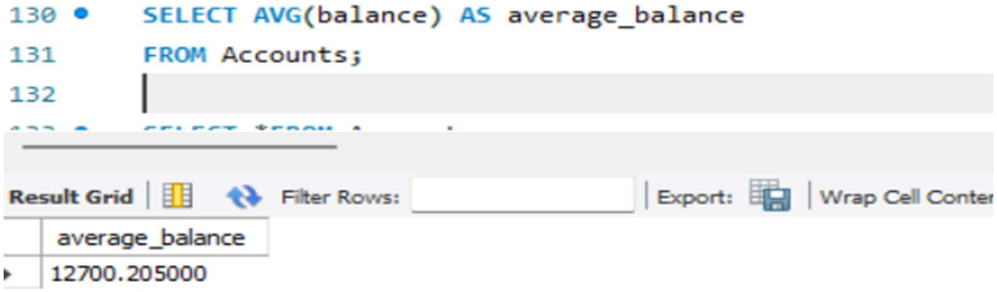


```
SELECT *FROM Customers
WHERE address NOT LIKE '%Chennai%';
```

customer_id	first_name	last_name	DOB	email	phone_numb
1	John	Doe	1985-06-15	john.doe@example.com	9876543210
2	Jane	Smith	1990-11-20	jane.smith@example.com	9876543211
3	Alice	Johnson	1975-09-30	alice.j@example.com	9876543212
4	Bob	Brown	2000-01-01	bob.b@example.com	9876543213
5	Charlie	Davis	1995-03-12	charlie.d@example.com	9876543214
6	Emily	Clark	1988-07-08	emily.c@example.com	9876543215
7	Frank	Wright	1992-05-05	frank.w@example.com	9876543216
8	Grace	Lee	1983-12-25	grace.l@example.com	9876543217
9	Hank	Green	1970-10-10	hank.g@example.com	9876543218
10	Ivy	King	1998-04-18	ivy.k@example.com	9876543219

Tasks 3: Aggregate functions, Having, Order By, Group By and Joins:

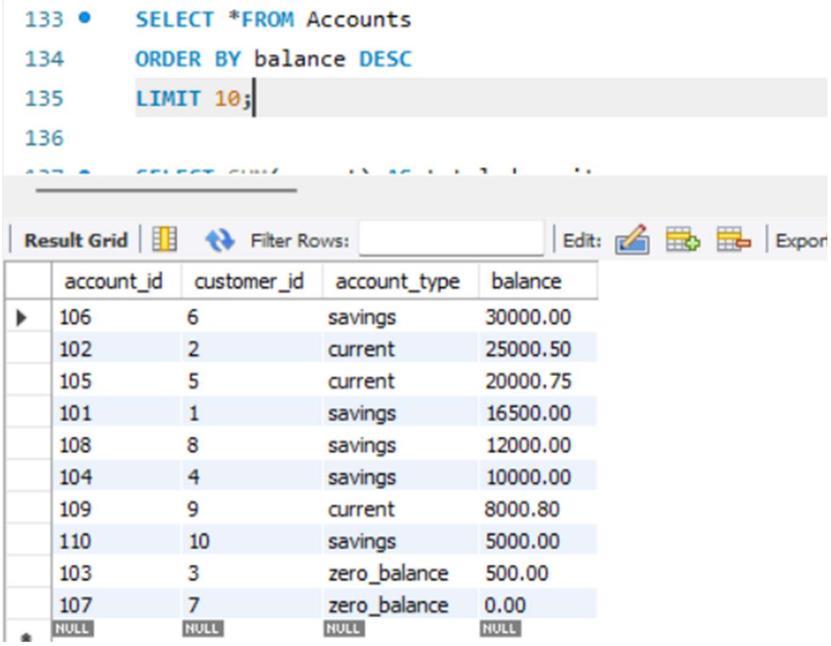
1. Write a SQL query to Find the average account balance for all customers.



```
SELECT AVG(balance) AS average_balance
FROM Accounts;
```

```
average_balance
12700.205000
```

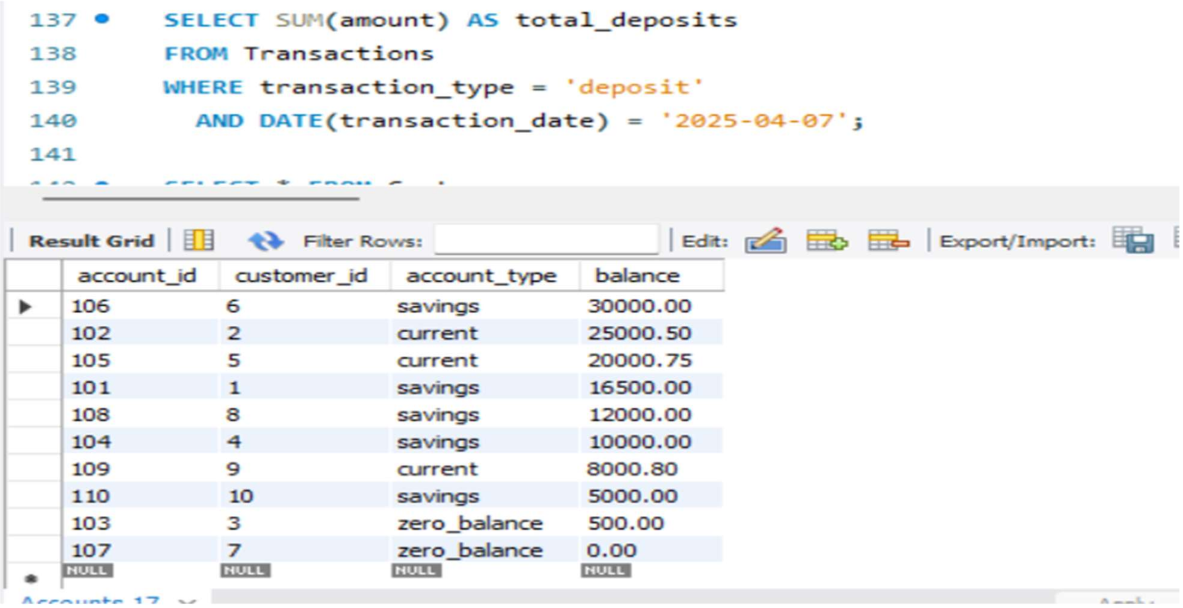
2. Write a SQL query to Retrieve the top 10 highest account balances.



```
SELECT *FROM Accounts
ORDER BY balance DESC
LIMIT 10;
```

account_id	customer_id	account_type	balance
106	6	savings	30000.00
102	2	current	25000.50
105	5	current	20000.75
101	1	savings	16500.00
108	8	savings	12000.00
104	4	savings	10000.00
109	9	current	8000.80
110	10	savings	5000.00
103	3	zero_balance	500.00
107	7	zero_balance	0.00
NULL	NULL	NULL	NULL

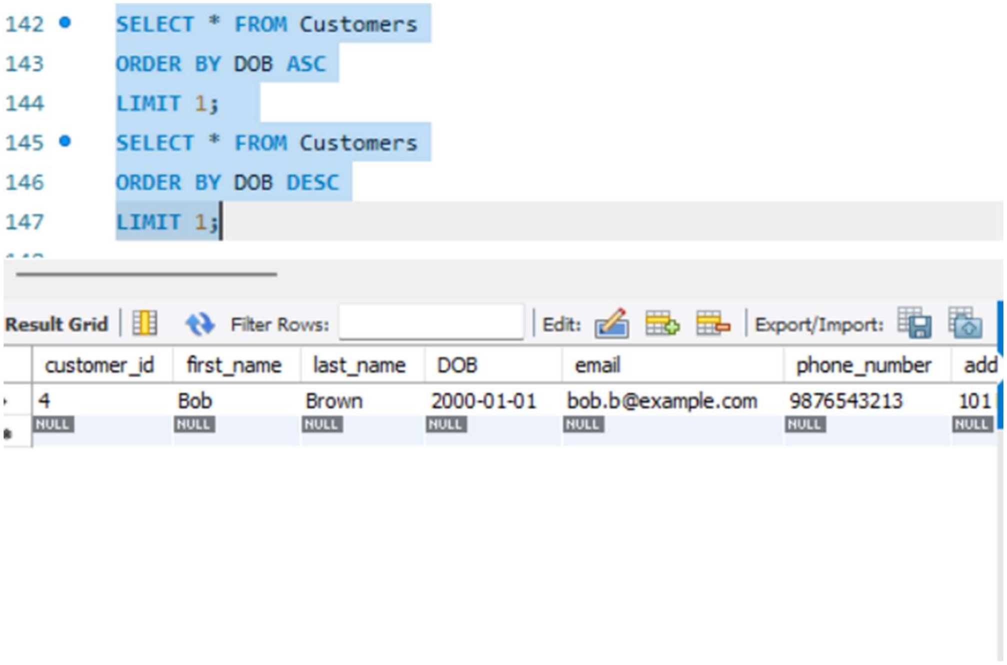
3. Write a SQL query to Calculate Total Deposits for All Customers in specific date.



```
SELECT SUM(amount) AS total_deposits
FROM Transactions
WHERE transaction_type = 'deposit'
  AND DATE(transaction_date) = '2025-04-07';
```

account_id	customer_id	account_type	balance
106	6	savings	30000.00
102	2	current	25000.50
105	5	current	20000.75
101	1	savings	16500.00
108	8	savings	12000.00
104	4	savings	10000.00
109	9	current	8000.80
110	10	savings	5000.00
103	3	zero_balance	500.00
107	7	zero_balance	0.00
NULL	NULL	NULL	NULL

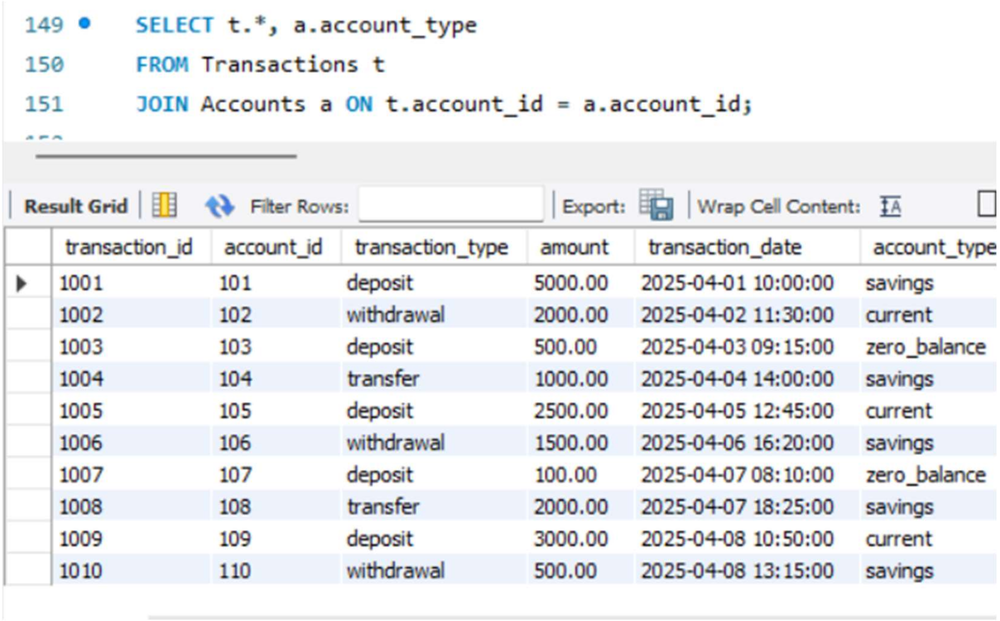
4. Write a SQL query to Find the Oldest and Newest Customers.



```sql
SELECT * FROM Customers
ORDER BY DOB ASC
LIMIT 1;
SELECT * FROM Customers
ORDER BY DOB DESC
LIMIT 1;
```

customer_id	first_name	last_name	DOB	email	phone_number	add
4	Bob	Brown	2000-01-01	bob.b@example.com	9876543213	101
NULL	NULL	NULL	NULL	NULL	NULL	NULL

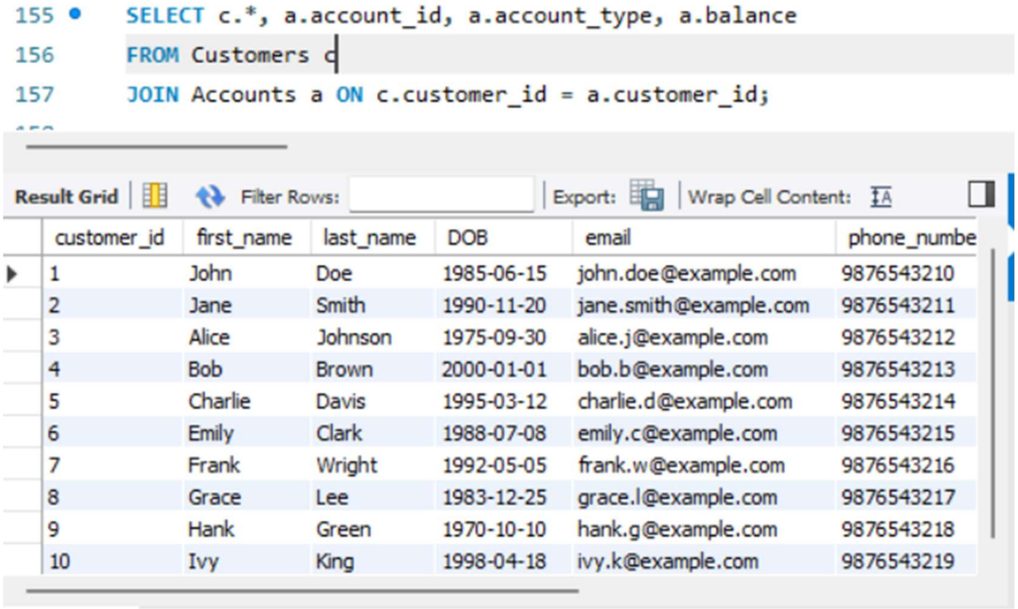
5. Write a SQL query to Retrieve transaction details along with the account type.



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

transaction_id	account_id	transaction_type	amount	transaction_date	account_type
1001	101	deposit	5000.00	2025-04-01 10:00:00	savings
1002	102	withdrawal	2000.00	2025-04-02 11:30:00	current
1003	103	deposit	500.00	2025-04-03 09:15:00	zero_balance
1004	104	transfer	1000.00	2025-04-04 14:00:00	savings
1005	105	deposit	2500.00	2025-04-05 12:45:00	current
1006	106	withdrawal	1500.00	2025-04-06 16:20:00	savings
1007	107	deposit	100.00	2025-04-07 08:10:00	zero_balance
1008	108	transfer	2000.00	2025-04-07 18:25:00	savings
1009	109	deposit	3000.00	2025-04-08 10:50:00	current
1010	110	withdrawal	500.00	2025-04-08 13:15:00	savings

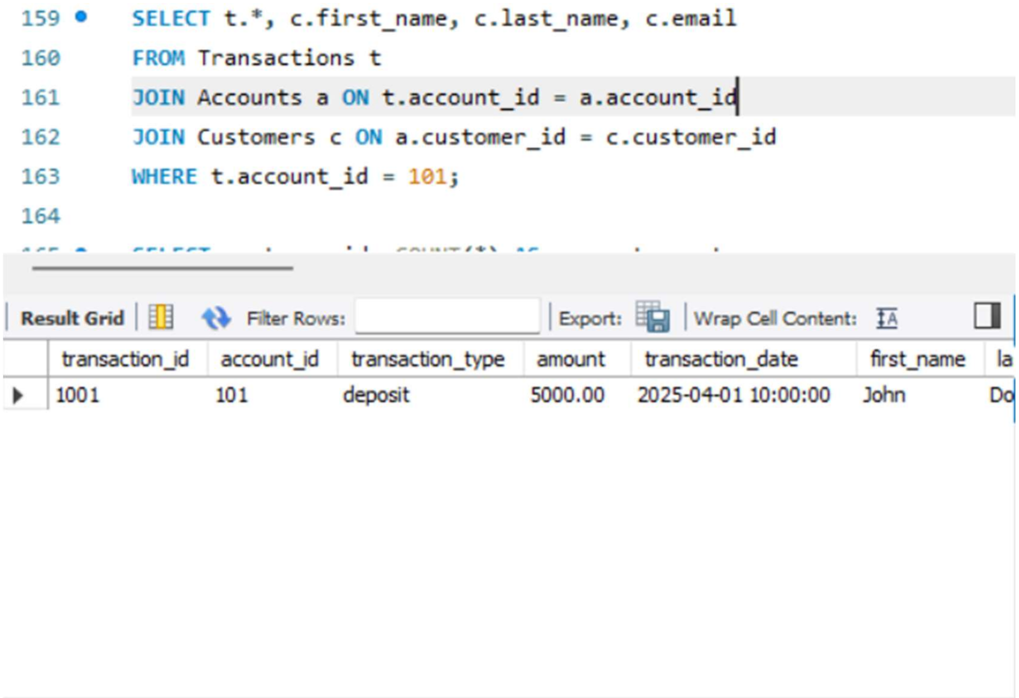
6. Write a SQL query to Get a list of customers along with their account details.



```sql
SELECT c.*, a.account_id, a.account_type, a.balance
FROM Customers c
JOIN Accounts a ON c.customer_id = a.customer_id;
```

customer_id	first_name	last_name	DOB	email	phone_numbe
1	John	Doe	1985-06-15	john.doe@example.com	9876543210
2	Jane	Smith	1990-11-20	jane.smith@example.com	9876543211
3	Alice	Johnson	1975-09-30	alice.j@example.com	9876543212
4	Bob	Brown	2000-01-01	bob.b@example.com	9876543213
5	Charlie	Davis	1995-03-12	charlie.d@example.com	9876543214
6	Emily	Clark	1988-07-08	emily.c@example.com	9876543215
7	Frank	Wright	1992-05-05	frank.w@example.com	9876543216
8	Grace	Lee	1983-12-25	grace.l@example.com	9876543217
9	Hank	Green	1970-10-10	hank.g@example.com	9876543218
10	Ivy	King	1998-04-18	ivy.k@example.com	9876543219

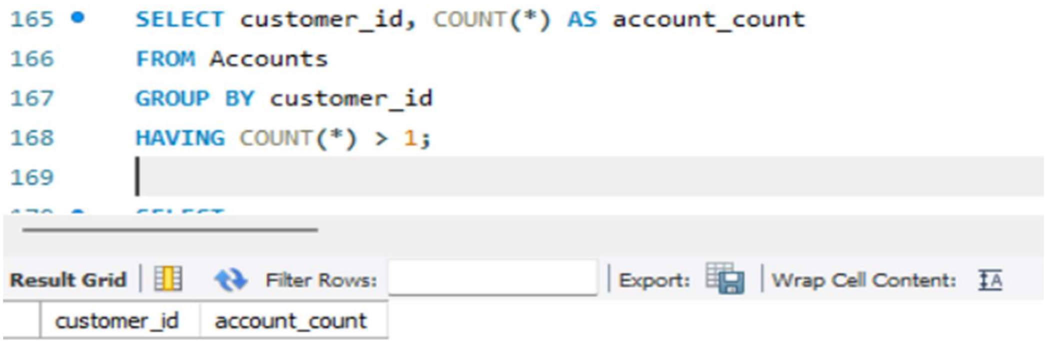
7. Write a SQL query to Retrieve transaction details along with customer information for a specific account.



```sql
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 = 101;
```

transaction_id	account_id	transaction_type	amount	transaction_date	first_name	la
1001	101	deposit	5000.00	2025-04-01 10:00:00	John	Do

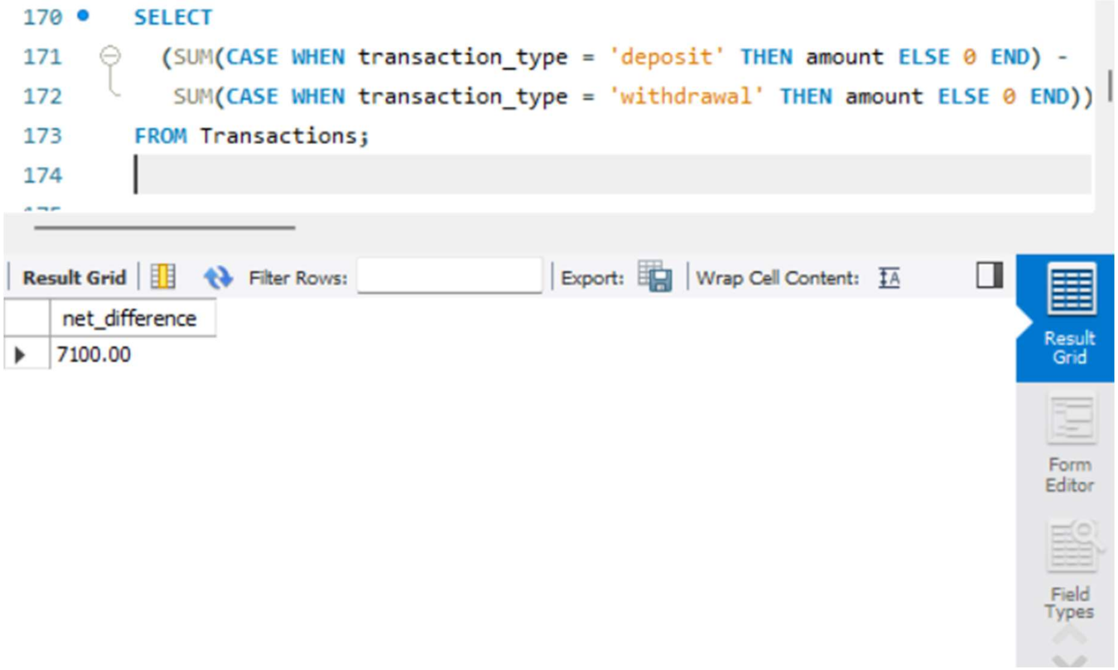
8. Write a SQL query to Identify customers who have more than one account.



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

customer_id	account_count

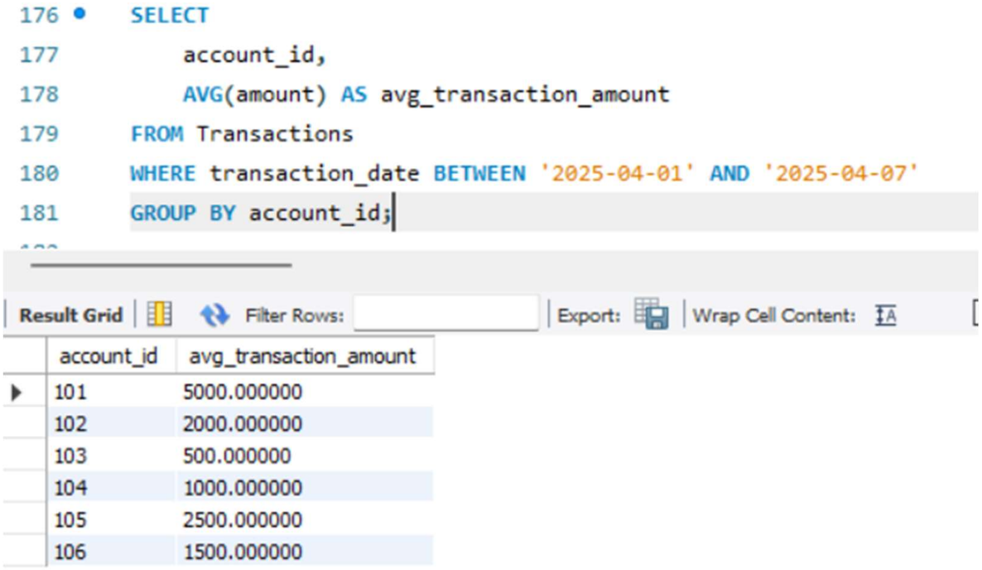
9. Write a SQL query to Calculate the difference in transaction amounts between deposits and withdrawals.



```sql
SELECT
   (SUM(CASE WHEN transaction_type = 'deposit' THEN amount ELSE 0 END) -
    SUM(CASE WHEN transaction_type = 'withdrawal' THEN amount ELSE 0 END))
FROM Transactions;
```

net_difference
7100.00

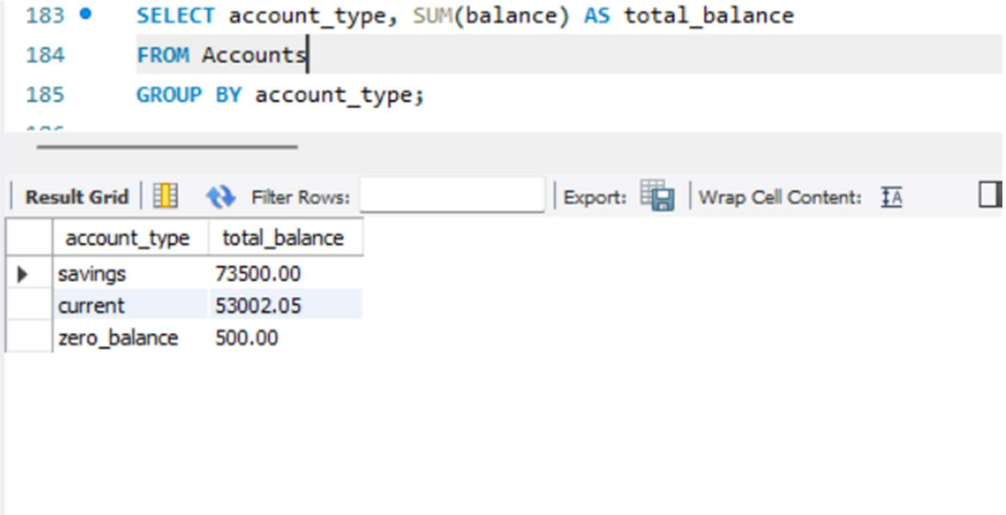
10. Write a SQL query to Calculate the average daily balance for each account over a specified period.



```
SELECT
    account_id,
    AVG(amount) AS avg_transaction_amount
FROM Transactions
WHERE transaction_date BETWEEN '2025-04-01' AND '2025-04-07'
GROUP BY account_id;
```

account_id	avg_transaction_amount
101	5000.000000
102	2000.000000
103	500.000000
104	1000.000000
105	2500.000000
106	1500.000000

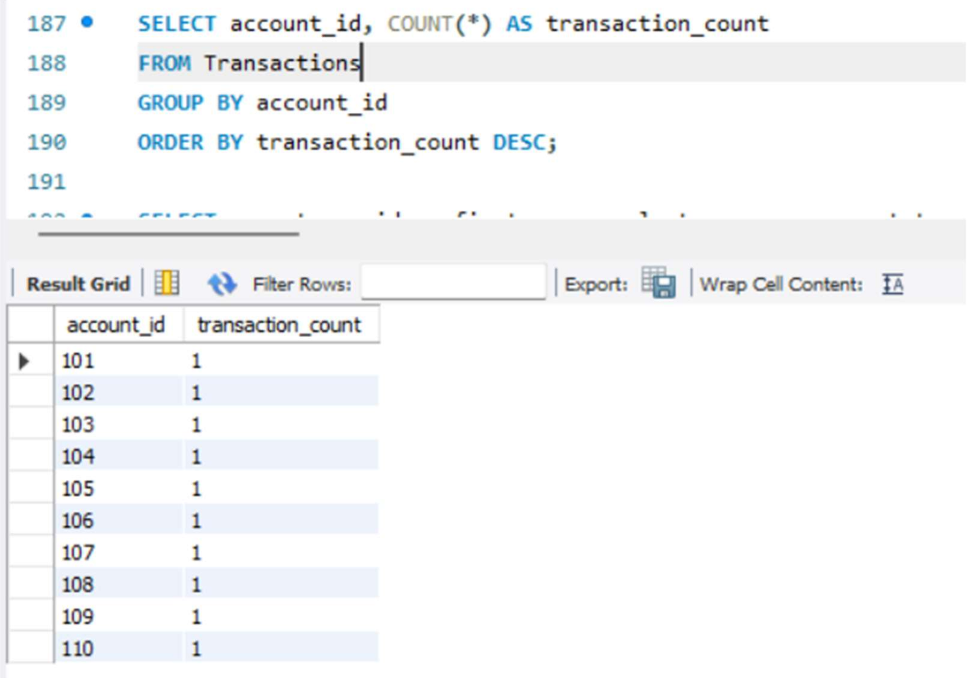
11. Calculate the total balance for each account type.



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

account_type	total_balance
savings	73500.00
current	53002.05
zero_balance	500.00

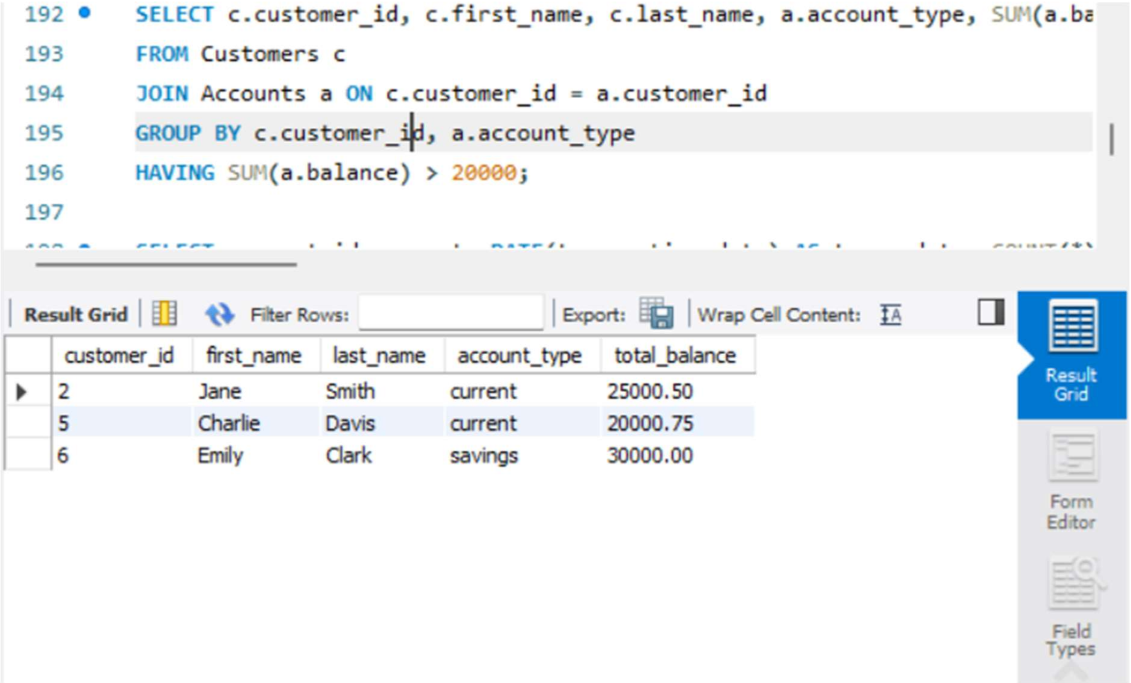
12. Identify accounts with the highest number of transactions order by descending order.



```sql
SELECT account_id, COUNT(*) 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

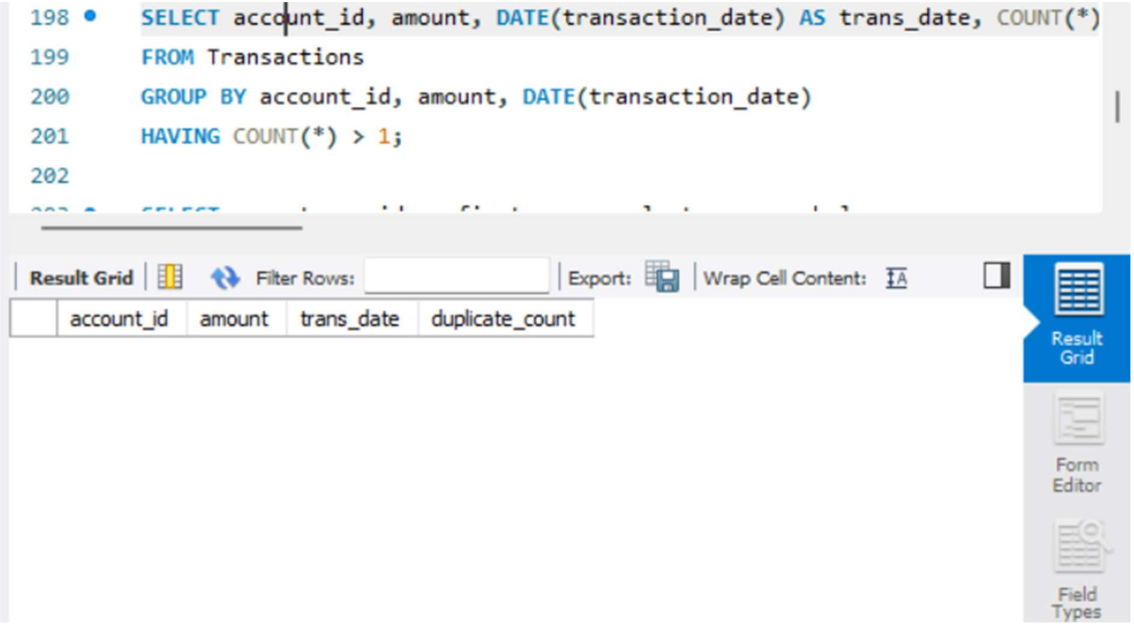
13. List customers with high aggregate account balances, along with their account types.



```sql
SELECT c.customer_id, c.first_name, c.last_name, a.account_type, SUM(a.ba
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;
```

customer_id	first_name	last_name	account_type	total_balance
2	Jane	Smith	current	25000.50
5	Charlie	Davis	current	20000.75
6	Emily	Clark	savings	30000.00

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

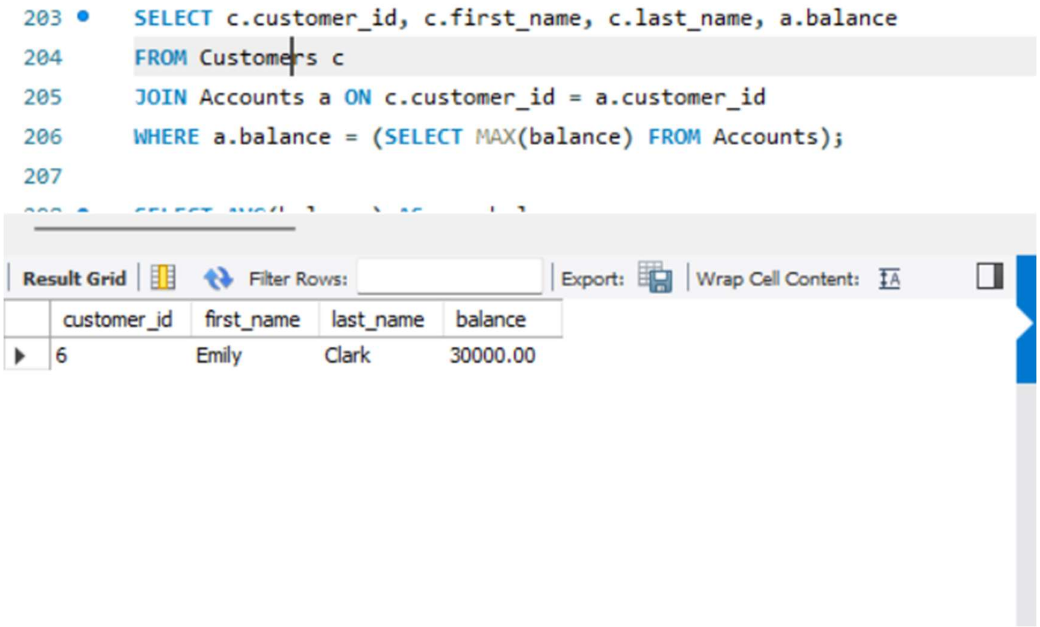


```sql
SELECT account_id, amount, DATE(transaction_date) AS trans_date, COUNT(*)
FROM Transactions
GROUP BY account_id, amount, DATE(transaction_date)
HAVING COUNT(*) > 1;
```

account_id	amount	trans_date	duplicate_count

Tasks 4: Subquery and its type:

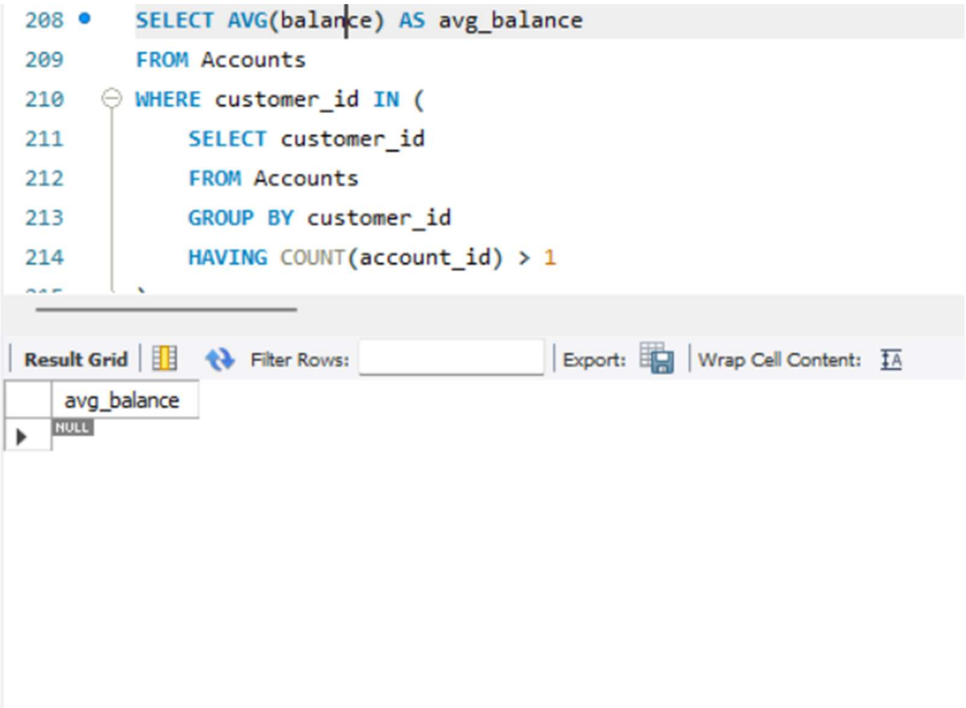
1. Retrieve the customer(s) with the highest account balance.



```sql
SELECT c.customer_id, c.first_name, c.last_name, a.balance
FROM Customers c
JOIN Accounts a ON c.customer_id = a.customer_id
WHERE a.balance = (SELECT MAX(balance) FROM Accounts);
```

customer_id	first_name	last_name	balance
6	Emily	Clark	30000.00

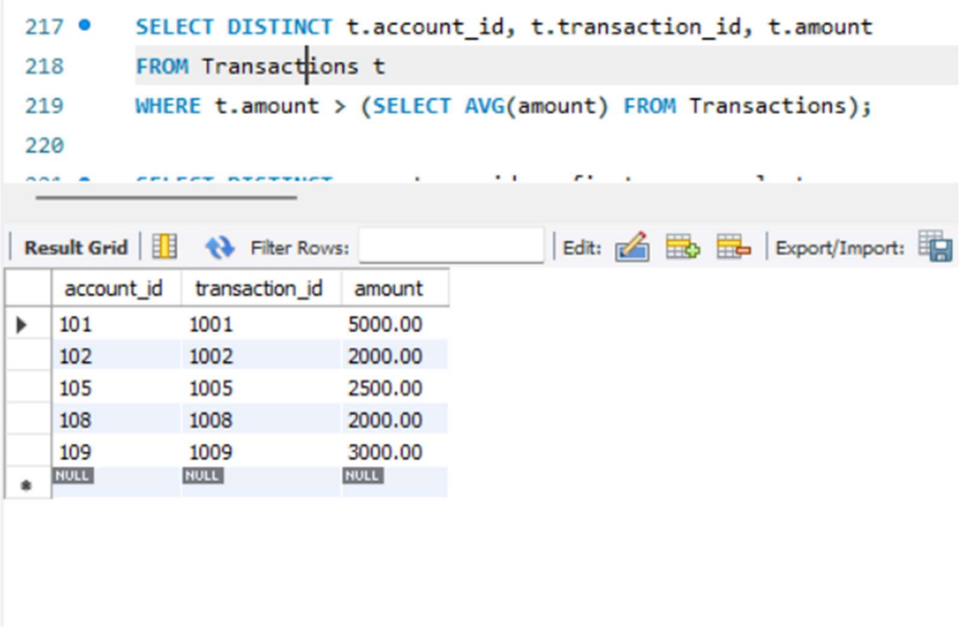
2. Calculate the average account balance for customers who have more than one account.



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

avg_balance
NULL

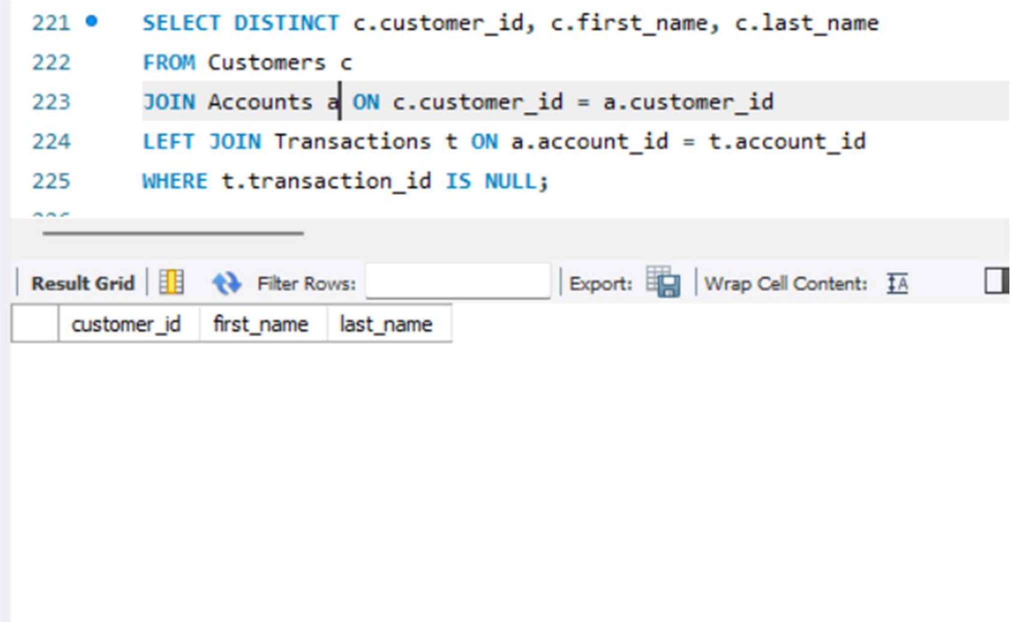
3. Retrieve accounts with transactions whose amounts exceed the average transaction amount.



```
SELECT DISTINCT t.account_id, t.transaction_id, t.amount
FROM Transactions t
WHERE t.amount > (SELECT AVG(amount) FROM Transactions);
```

account_id	transaction_id	amount
101	1001	5000.00
102	1002	2000.00
105	1005	2500.00
108	1008	2000.00
109	1009	3000.00
NULL	NULL	NULL

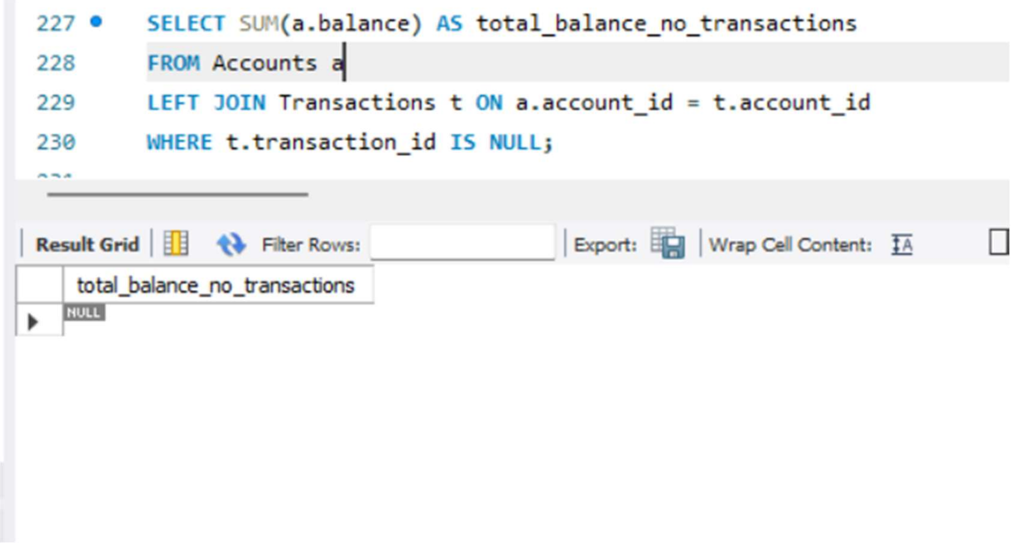
4. Identify customers who have no recorded transactions.



```sql
SELECT DISTINCT c.customer_id, c.first_name, c.last_name
FROM Customers c
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;
```

```
customer_id | first_name | last_name
```

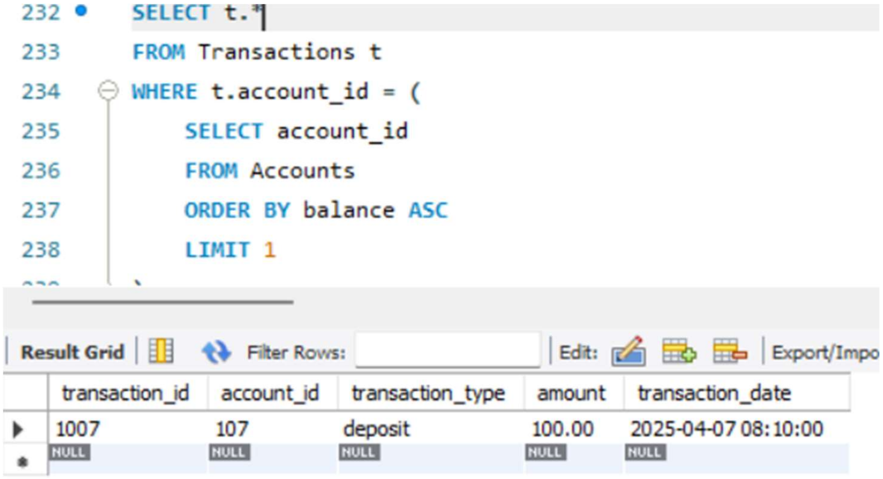
5. Calculate the total balance of accounts with no recorded transactions.



```sql
SELECT SUM(a.balance) AS total_balance_no_transactions
FROM Accounts a
LEFT JOIN Transactions t ON a.account_id = t.account_id
WHERE t.transaction_id IS NULL;
```

```
total_balance_no_transactions
NULL
```

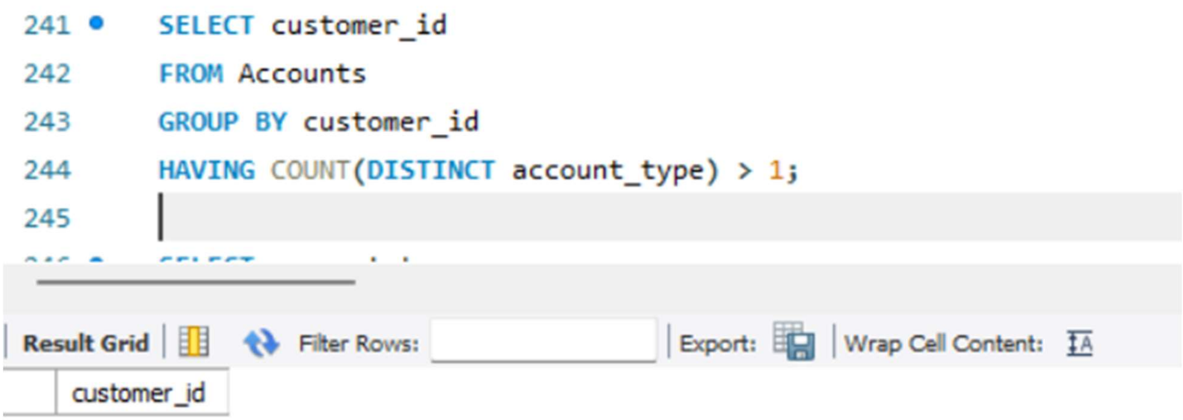
6. Retrieve transactions for accounts with the lowest balance.



```
SELECT t.*
FROM Transactions t
WHERE t.account_id = (
    SELECT account_id
    FROM Accounts
    ORDER BY balance ASC
    LIMIT 1
)
```

transaction_id	account_id	transaction_type	amount	transaction_date
1007	107	deposit	100.00	2025-04-07 08:10:00
NULL	NULL	NULL	NULL	NULL

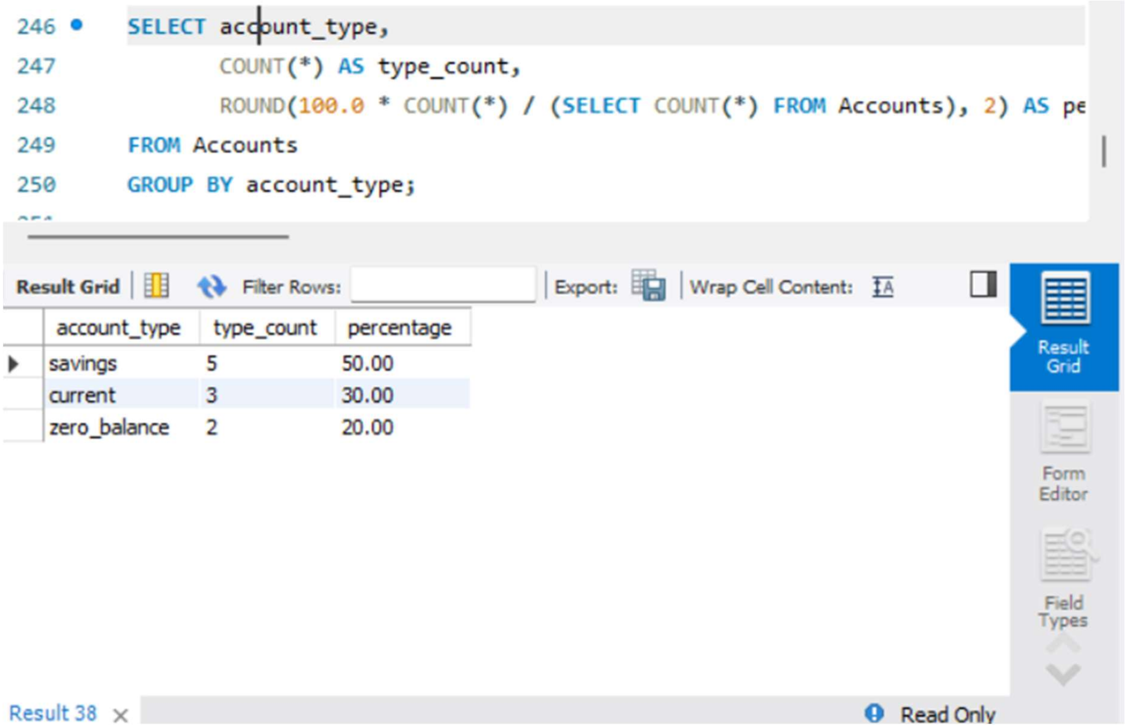
7. Identify customers who have accounts of multiple types.



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

```
customer_id
```

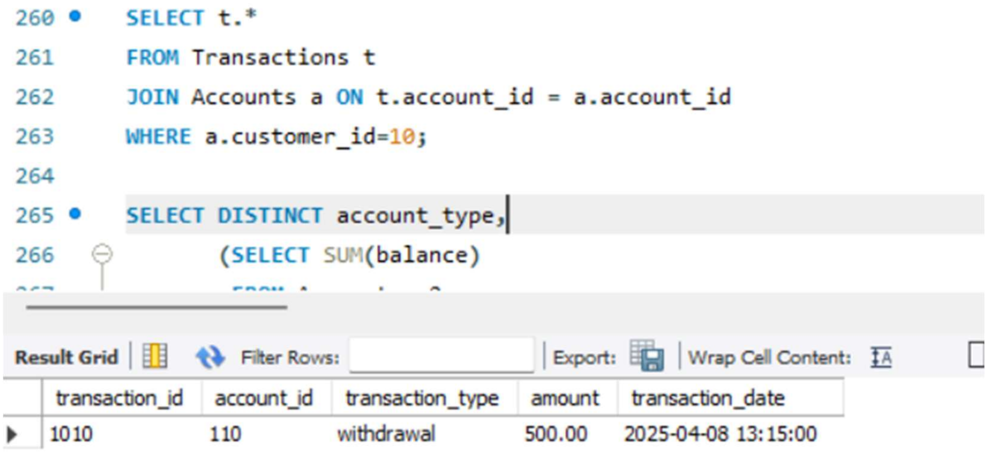
8. Calculate the percentage of each account type out of the total number of accounts.



```sql
SELECT account_type,
        COUNT(*) AS type_count,
        ROUND(100.0 * COUNT(*) / (SELECT COUNT(*) FROM Accounts), 2) AS pe
FROM Accounts
GROUP BY account_type;
```

account_type	type_count	percentage
savings	5	50.00
current	3	30.00
zero_balance	2	20.00

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

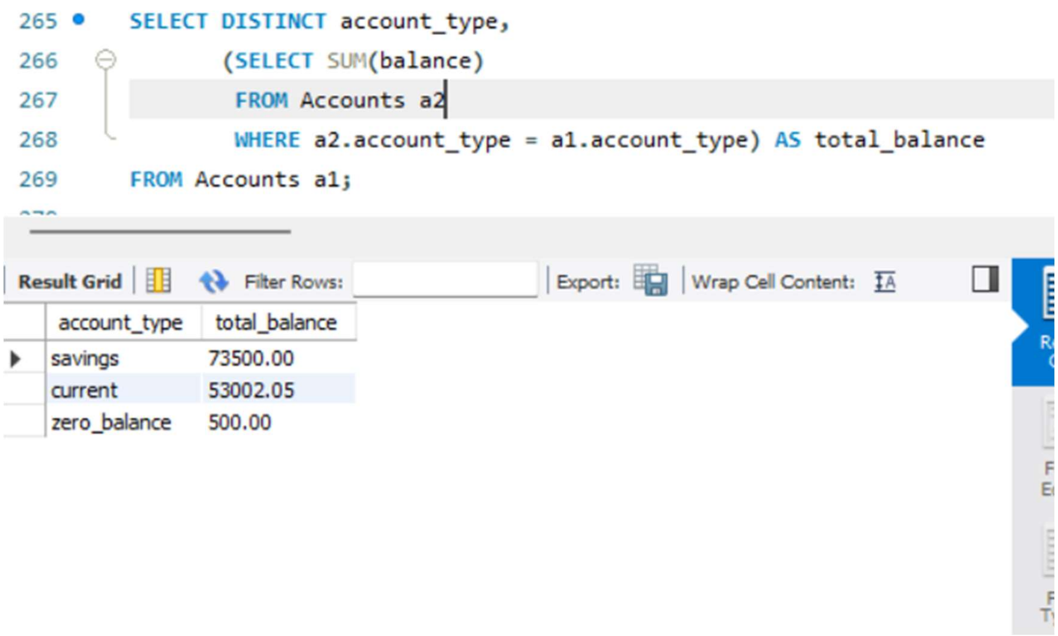


```sql
SELECT t.*
FROM Transactions t
JOIN Accounts a ON t.account_id = a.account_id
WHERE a.customer_id=10;

SELECT DISTINCT account_type,
        (SELECT SUM(balance)
```

transaction_id	account_id	transaction_type	amount	transaction_date
1010	110	withdrawal	500.00	2025-04-08 13:15:00

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



```sql
SELECT DISTINCT account_type,
       (SELECT SUM(balance)
        FROM Accounts a2
        WHERE a2.account_type = a1.account_type) AS total_balance
FROM Accounts a1;
```

account_type	total_balance
savings	73500.00
current	53002.05
zero_balance	500.00