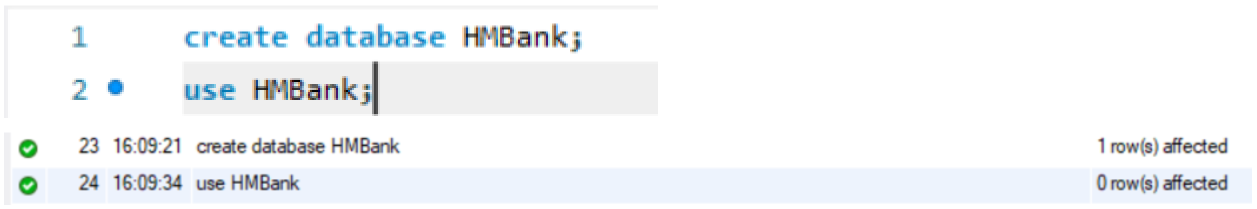
Banking System

Task 1

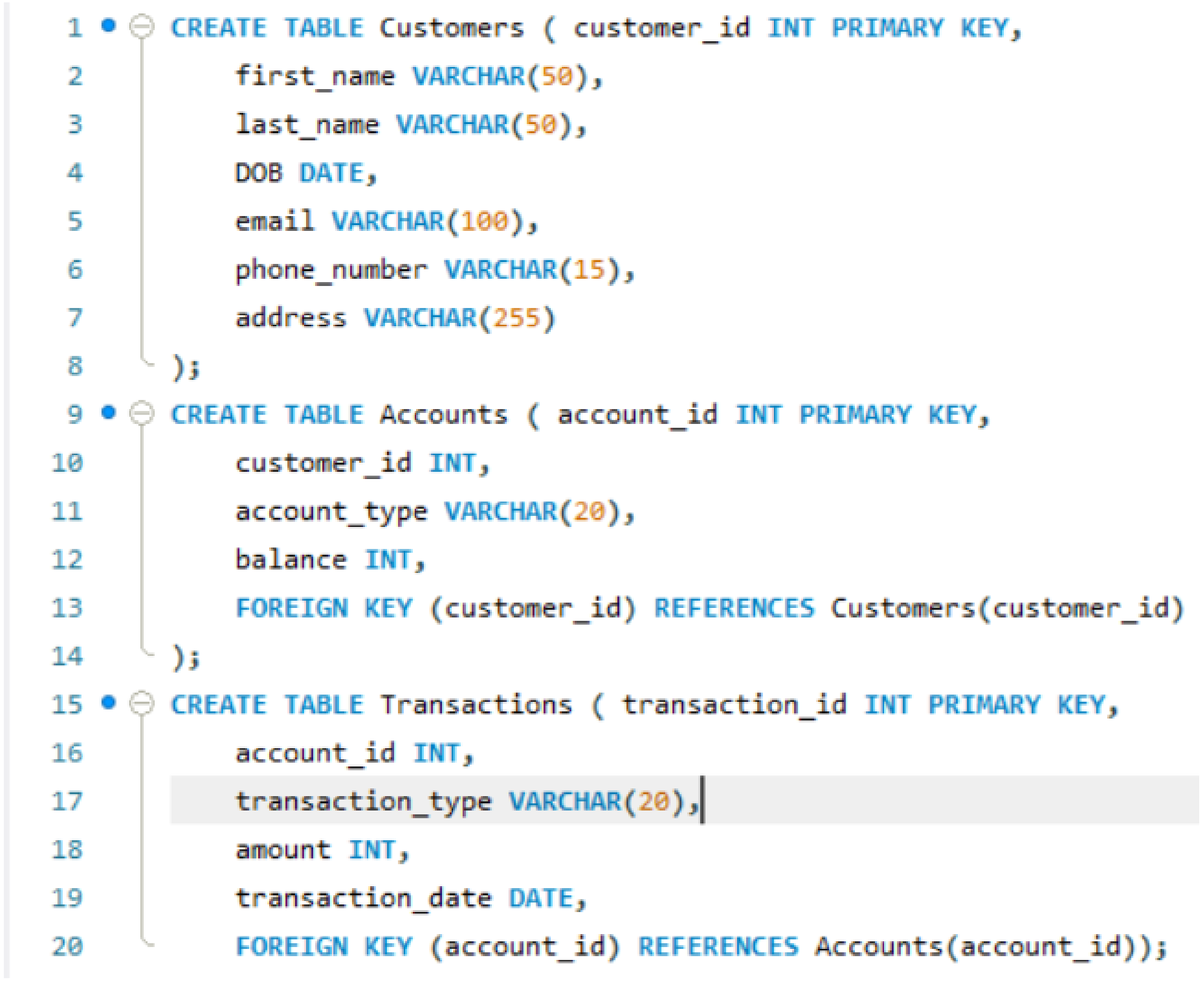
a.) Create the database named "HMBank"



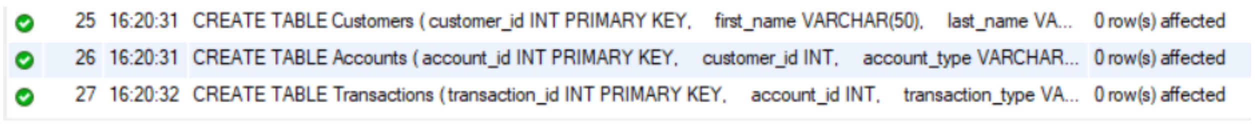
```
create database HMBank;
use HMBank;
```

```
23  16:09:21  create database HMBank    1 row(s) affected
24  16:09:34  use HMBank                0 row(s) affected
```

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

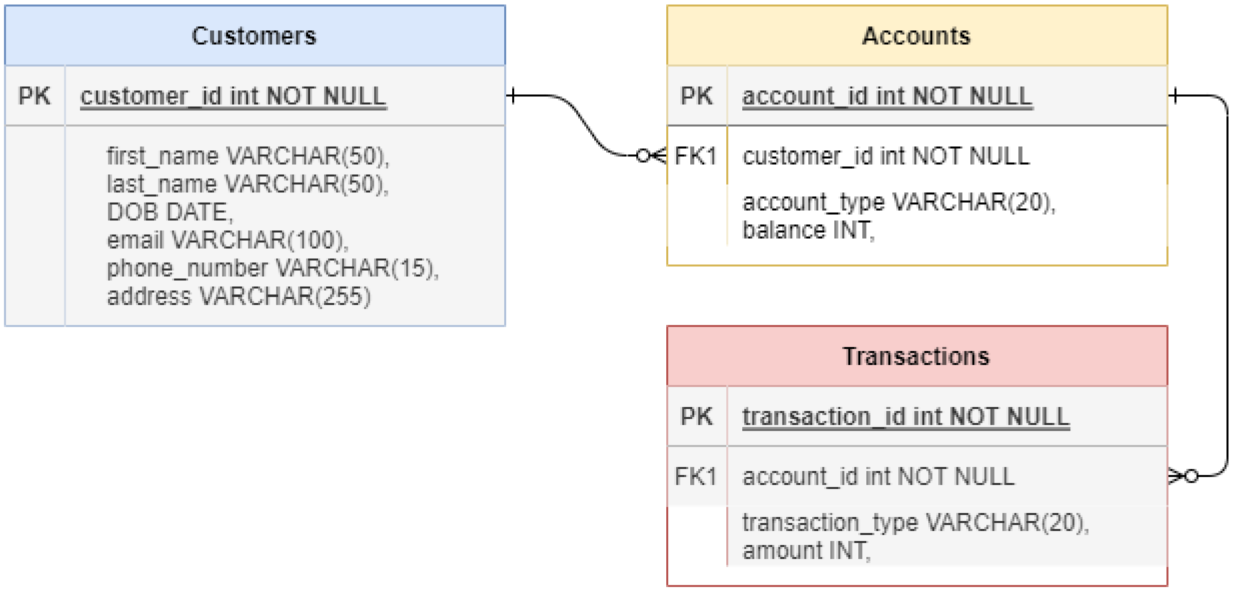


```
CREATE TABLE Customers ( customer_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    DOB DATE,
    email VARCHAR(100),
    phone_number VARCHAR(15),
    address VARCHAR(255)
);
CREATE TABLE Accounts ( account_id INT PRIMARY KEY,
    customer_id INT,
    account_type VARCHAR(20),
    balance INT,
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);
CREATE TABLE Transactions ( transaction_id INT PRIMARY KEY,
    account_id INT,
    transaction_type VARCHAR(20),
    amount INT,
    transaction_date DATE,
    FOREIGN KEY (account_id) REFERENCES Accounts(account_id));
```



```
✓ 25 16:20:31 CREATE TABLE Customers ( customer_id INT PRIMARY KEY,   first_name VARCHAR(50),   last_name VA...   0 row(s) affected
✓ 26 16:20:31 CREATE TABLE Accounts ( account_id INT PRIMARY KEY,   customer_id INT,   account_type VARCHAR...   0 row(s) affected
✓ 27 16:20:32 CREATE TABLE Transactions ( transaction_id INT PRIMARY KEY,   account_id INT,   transaction_type VA...   0 row(s) affected
```

c,) Create an ERD (Entity Relationship Diagram) for the database



d.) Create appropriate Primary Key and Foreign Key constraints for referential integrity. **(Done Above)**

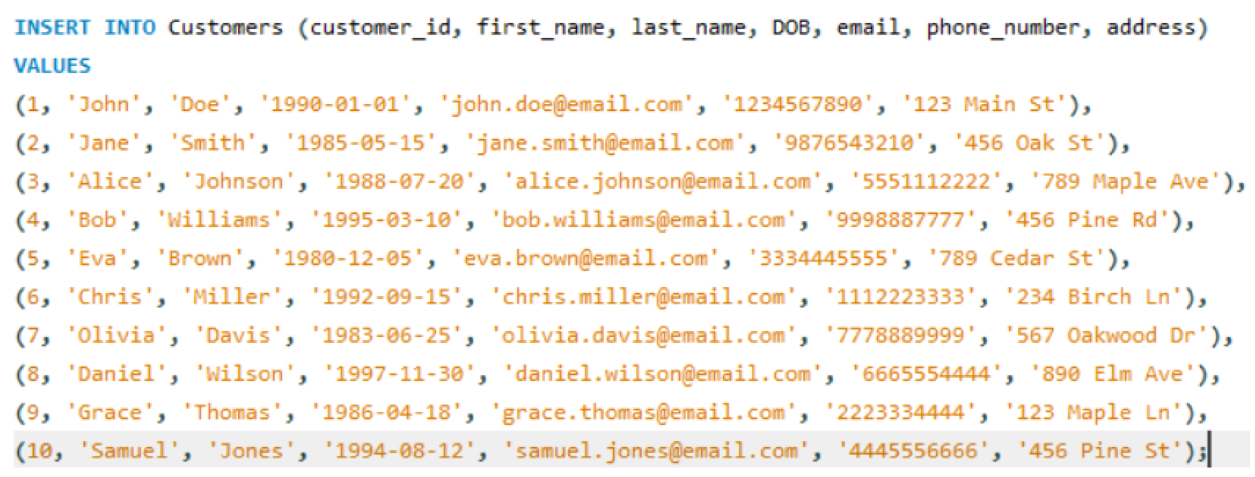
e.) Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships. **(Done Above)**

Customers
Accounts
Transactions

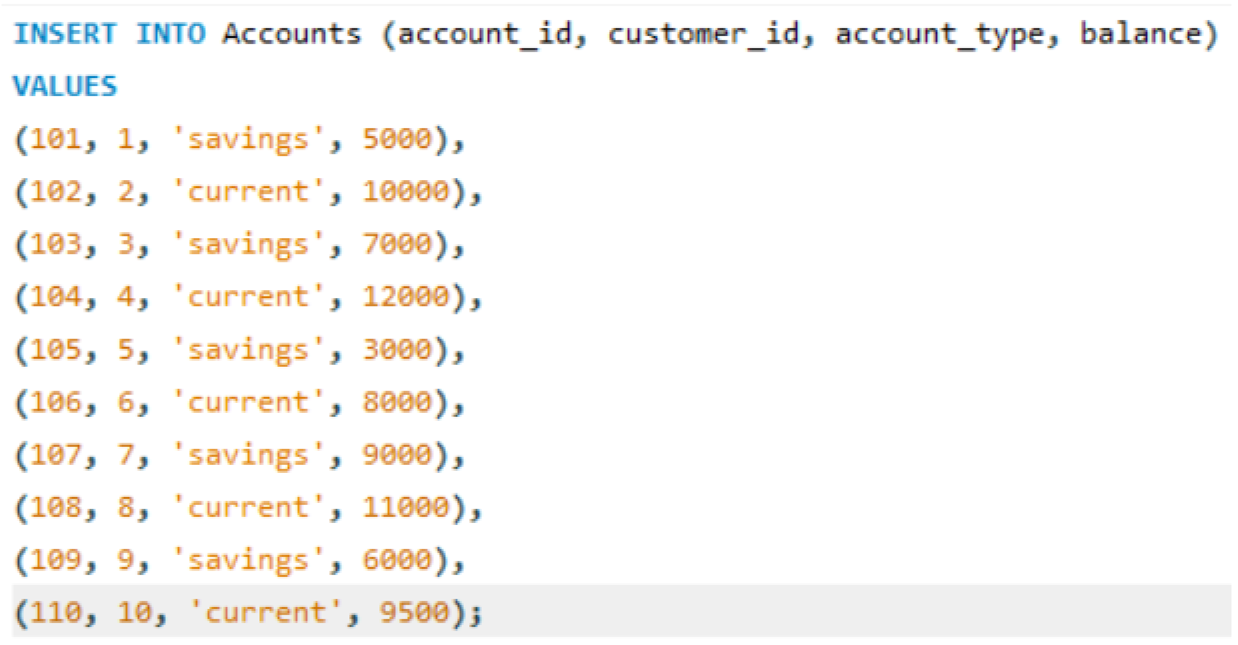
Task 2

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

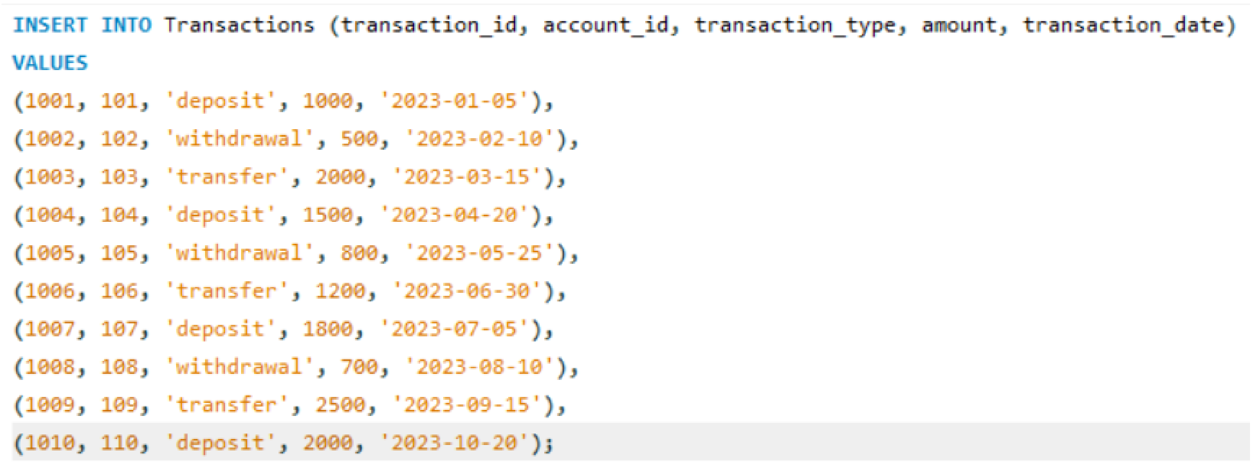
Customers
Accounts
Transactions



```sql
INSERT INTO Customers (customer_id, first_name, last_name, DOB, email, phone_number, address)
VALUES
(1, 'John', 'Doe', '1990-01-01', 'john.doe@email.com', '1234567890', '123 Main St'),
(2, 'Jane', 'Smith', '1985-05-15', 'jane.smith@email.com', '9876543210', '456 Oak St'),
(3, 'Alice', 'Johnson', '1988-07-20', 'alice.johnson@email.com', '5551112222', '789 Maple Ave'),
(4, 'Bob', 'Williams', '1995-03-10', 'bob.williams@email.com', '9998887777', '456 Pine Rd'),
(5, 'Eva', 'Brown', '1980-12-05', 'eva.brown@email.com', '3334445555', '789 Cedar St'),
(6, 'Chris', 'Miller', '1992-09-15', 'chris.miller@email.com', '1112223333', '234 Birch Ln'),
(7, 'Olivia', 'Davis', '1983-06-25', 'olivia.davis@email.com', '7778889999', '567 Oakwood Dr'),
(8, 'Daniel', 'Wilson', '1997-11-30', 'daniel.wilson@email.com', '6665554444', '890 Elm Ave'),
(9, 'Grace', 'Thomas', '1986-04-18', 'grace.thomas@email.com', '2223334444', '123 Maple Ln'),
(10, 'Samuel', 'Jones', '1994-08-12', 'samuel.jones@email.com', '4445556666', '456 Pine St');
```



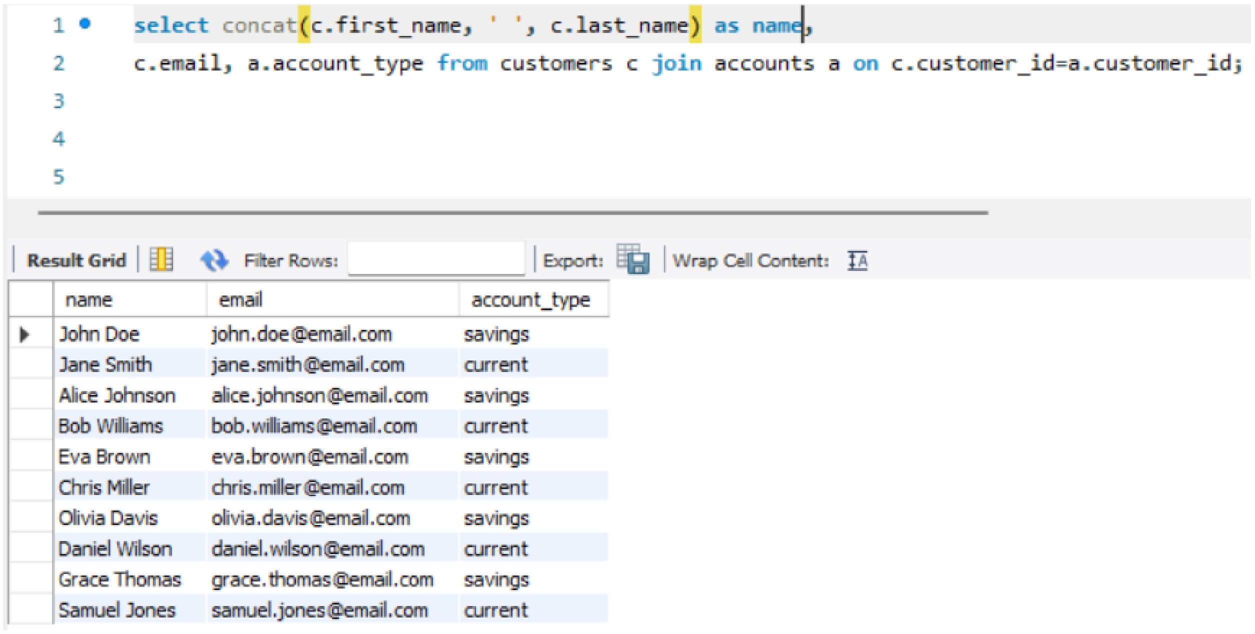
```sql
INSERT INTO Accounts (account_id, customer_id, account_type, balance)
VALUES
(101, 1, 'savings', 5000),
(102, 2, 'current', 10000),
(103, 3, 'savings', 7000),
(104, 4, 'current', 12000),
(105, 5, 'savings', 3000),
(106, 6, 'current', 8000),
(107, 7, 'savings', 9000),
(108, 8, 'current', 11000),
(109, 9, 'savings', 6000),
(110, 10, 'current', 9500);
```



```sql
INSERT INTO Transactions (transaction_id, account_id, transaction_type, amount, transaction_date)
VALUES
(1001, 101, 'deposit', 1000, '2023-01-05'),
(1002, 102, 'withdrawal', 500, '2023-02-10'),
(1003, 103, 'transfer', 2000, '2023-03-15'),
(1004, 104, 'deposit', 1500, '2023-04-20'),
(1005, 105, 'withdrawal', 800, '2023-05-25'),
(1006, 106, 'transfer', 1200, '2023-06-30'),
(1007, 107, 'deposit', 1800, '2023-07-05'),
(1008, 108, 'withdrawal', 700, '2023-08-10'),
(1009, 109, 'transfer', 2500, '2023-09-15'),
(1010, 110, 'deposit', 2000, '2023-10-20');
```

2.)

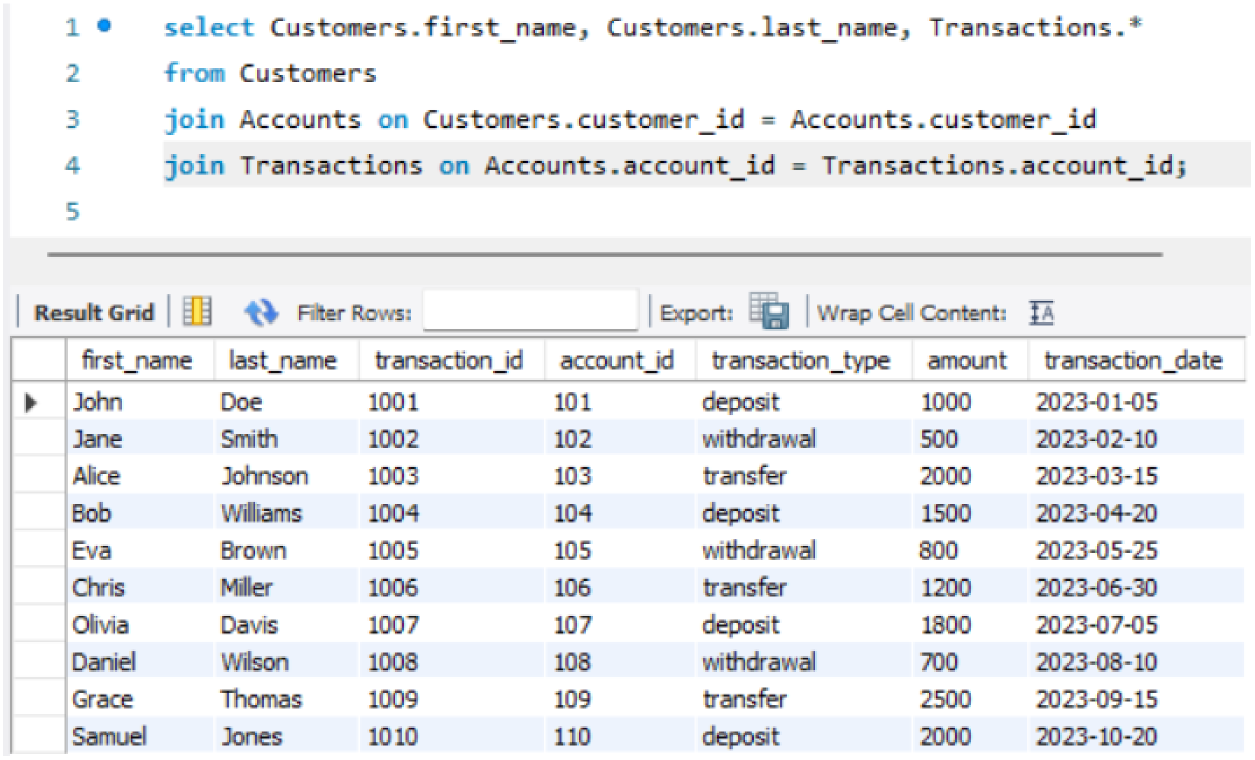
i.) Write a SQL query to retrieve the name, account type and email of all customers



```sql
select concat(c.first_name, ' ', c.last_name) as name,
c.email, a.account_type from customers c join accounts a on c.customer_id=a.customer_id;
```

name	email	account_type
John Doe	john.doe@email.com	savings
Jane Smith	jane.smith@email.com	current
Alice Johnson	alice.johnson@email.com	savings
Bob Williams	bob.williams@email.com	current
Eva Brown	eva.brown@email.com	savings
Chris Miller	chris.miller@email.com	current
Olivia Davis	olivia.davis@email.com	savings
Daniel Wilson	daniel.wilson@email.com	current
Grace Thomas	grace.thomas@email.com	savings
Samuel Jones	samuel.jones@email.com	current

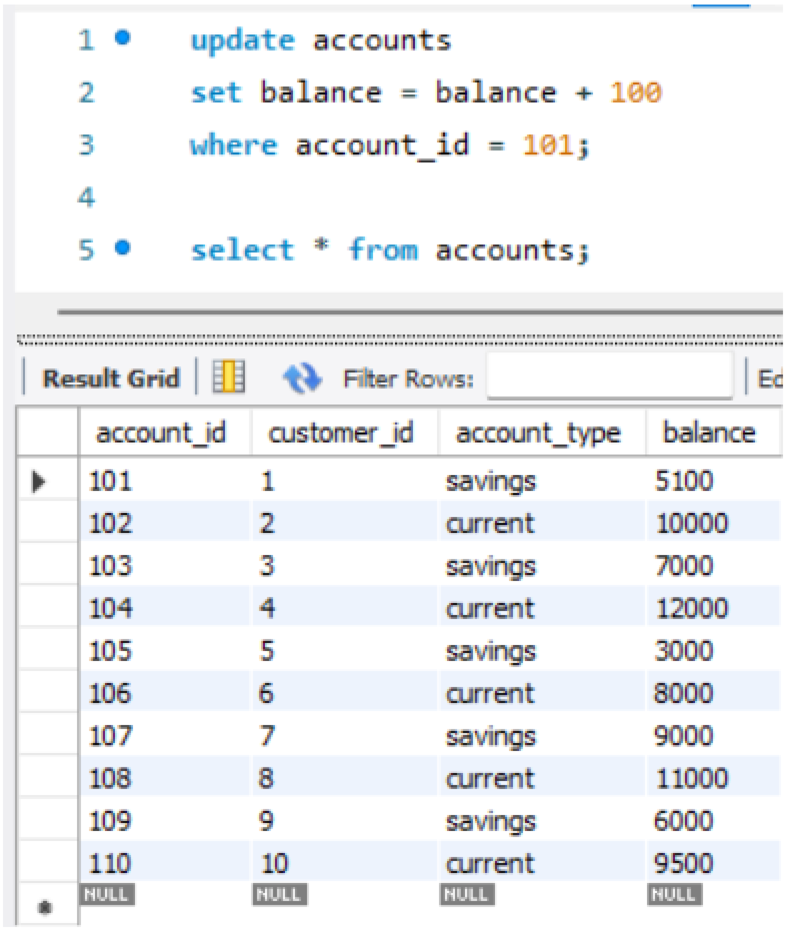
ii.) Write a SQL query to list all transaction corresponding customer.



```sql
select Customers.first_name, Customers.last_name, Transactions.*
from Customers
join Accounts on Customers.customer_id = Accounts.customer_id
join Transactions on Accounts.account_id = Transactions.account_id;
```

first_name	last_name	transaction_id	account_id	transaction_type	amount	transaction_date
John	Doe	1001	101	deposit	1000	2023-01-05
Jane	Smith	1002	102	withdrawal	500	2023-02-10
Alice	Johnson	1003	103	transfer	2000	2023-03-15
Bob	Williams	1004	104	deposit	1500	2023-04-20
Eva	Brown	1005	105	withdrawal	800	2023-05-25
Chris	Miller	1006	106	transfer	1200	2023-06-30
Olivia	Davis	1007	107	deposit	1800	2023-07-05
Daniel	Wilson	1008	108	withdrawal	700	2023-08-10
Grace	Thomas	1009	109	transfer	2500	2023-09-15
Samuel	Jones	1010	110	deposit	2000	2023-10-20

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

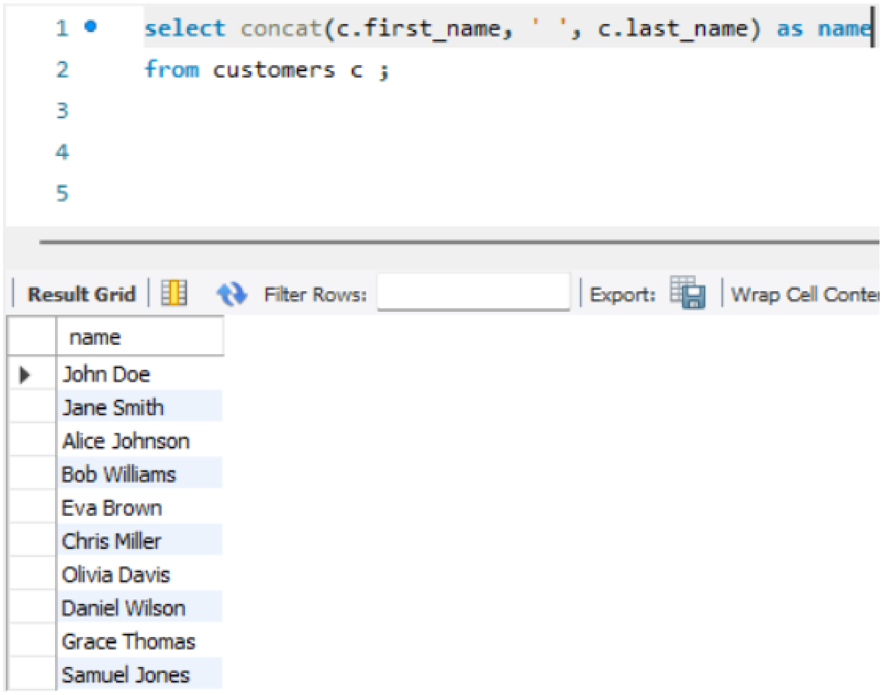


```
update accounts
set balance = balance + 100
where account_id = 101;

select * from accounts;
```

	account_id	customer_id	account_type	balance
▶	101	1	savings	5100
	102	2	current	10000
	103	3	savings	7000
	104	4	current	12000
	105	5	savings	3000
	106	6	current	8000
	107	7	savings	9000
	108	8	current	11000
	109	9	savings	6000
	110	10	current	9500
●	NULL	NULL	NULL	NULL

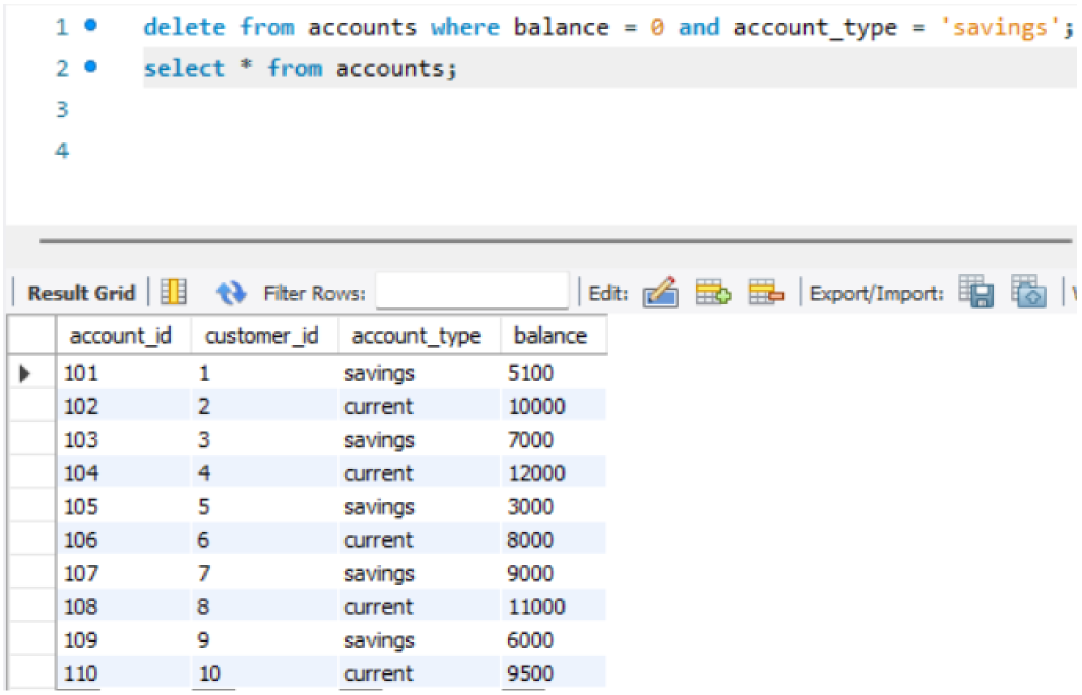
iv.) Write a SQL query to Combine first and last names of customers as a full_name.



```sql
select concat(c.first_name, ' ', c.last_name) as name
from customers c ;
```

name
John Doe
Jane Smith
Alice Johnson
Bob Williams
Eva Brown
Chris Miller
Olivia Davis
Daniel Wilson
Grace Thomas
Samuel Jones

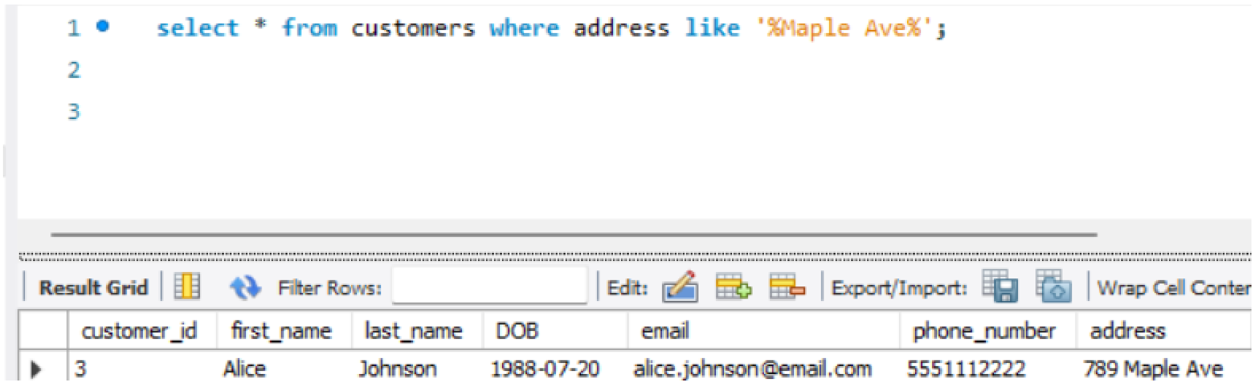
v.) Write a SQL query to remove accounts with a balance of zero where the account type is savings.



```sql
delete from accounts where balance = 0 and account_type = 'savings';
select * from accounts;
```

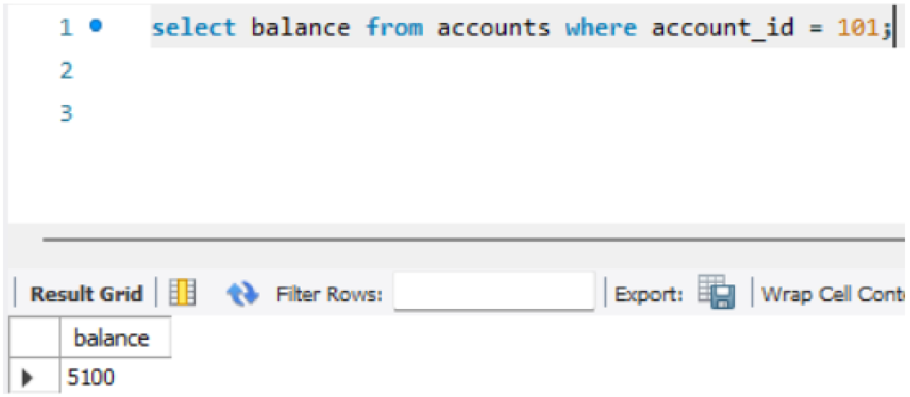
account_id	customer_id	account_type	balance
101	1	savings	5100
102	2	current	10000
103	3	savings	7000
104	4	current	12000
105	5	savings	3000
106	6	current	8000
107	7	savings	9000
108	8	current	11000
109	9	savings	6000
110	10	current	9500

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


```
select * from customers where address like '%Maple Ave%';
```

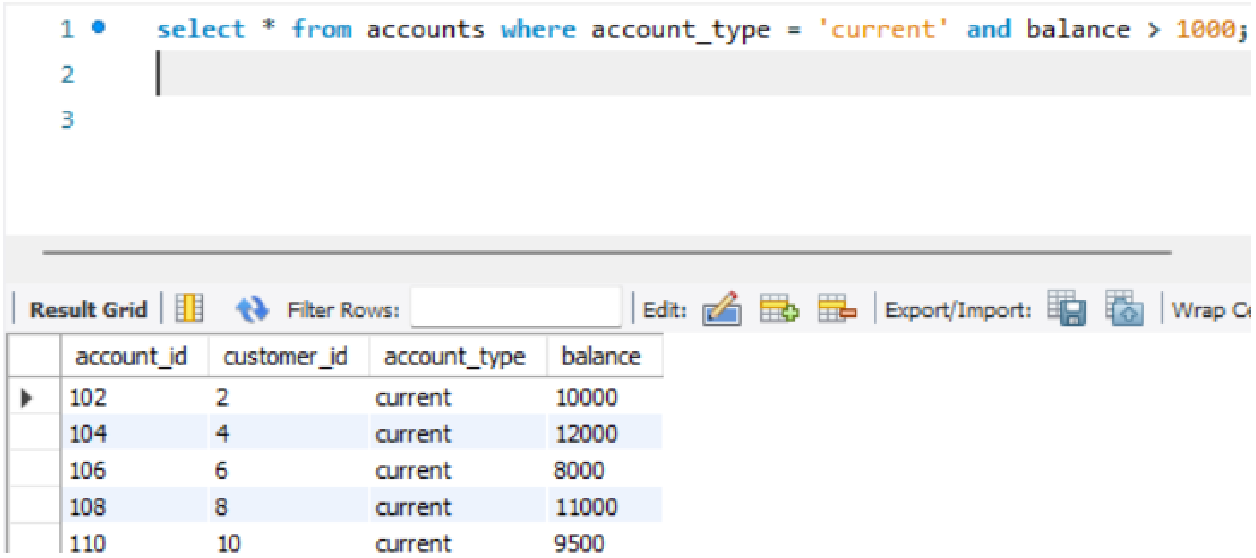
customer_id	first_name	last_name	DOB	email	phone_number	address
3	Alice	Johnson	1988-07-20	alice.johnson@email.com	5551112222	789 Maple Ave

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


```
select balance from accounts where account_id = 101;
```

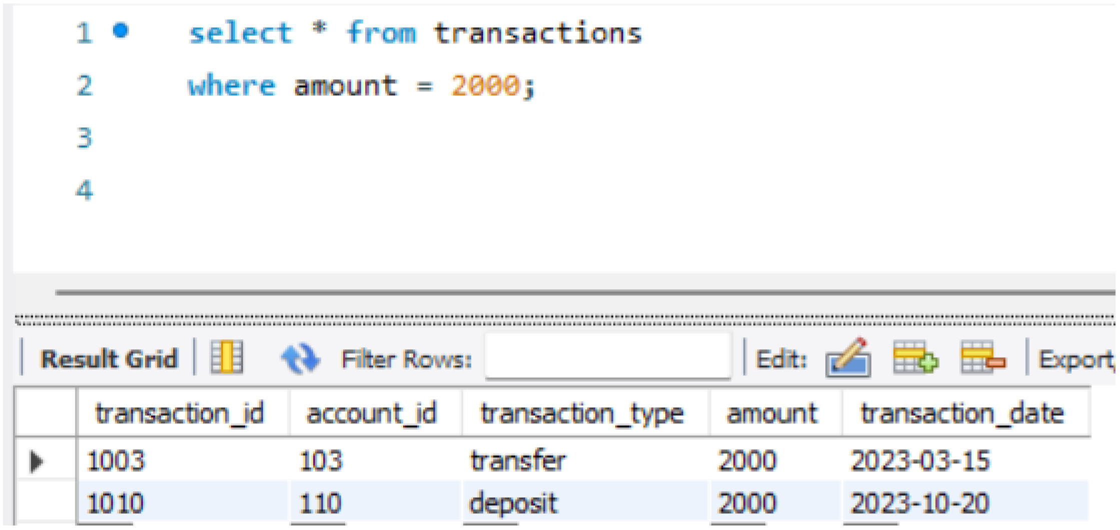
balance
5100

viii.) 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;
```

account_id	customer_id	account_type	balance
102	2	current	10000
104	4	current	12000
106	6	current	8000
108	8	current	11000
110	10	current	9500

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

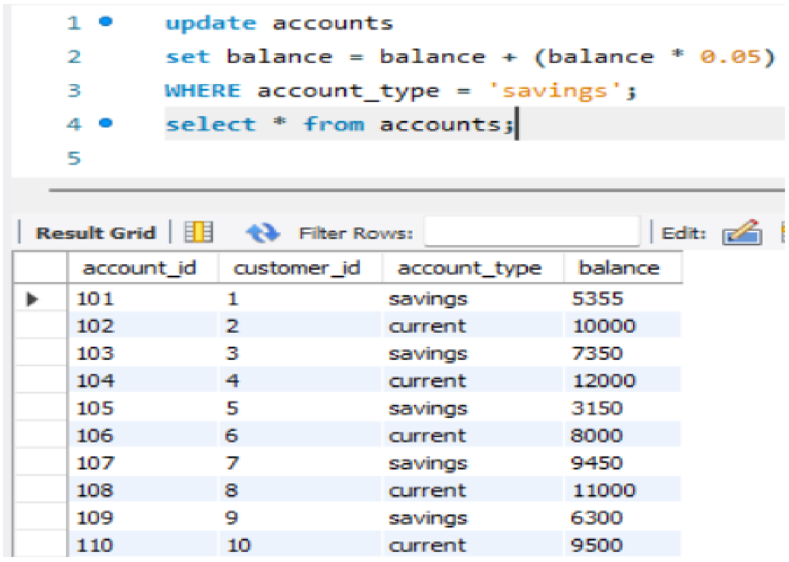


```sql
select * from transactions
where amount = 2000;
```

Result Grid | Filter Rows: | Edit: | Export

transaction_id	account_id	transaction_type	amount	transaction_date
1003	103	transfer	2000	2023-03-15
1010	110	deposit	2000	2023-10-20

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

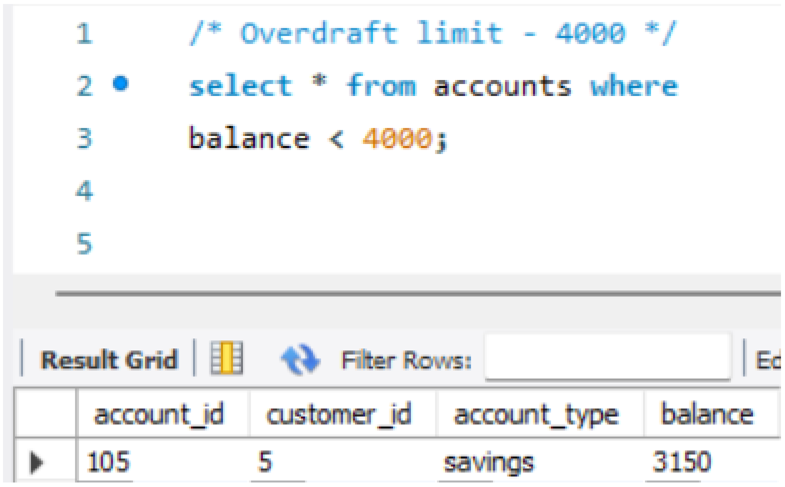


```sql
update accounts
set balance = balance + (balance * 0.05)
WHERE account_type = 'savings';
select * from accounts;
```

Result Grid | Filter Rows: | Edit:

account_id	customer_id	account_type	balance
101	1	savings	5355
102	2	current	10000
103	3	savings	7350
104	4	current	12000
105	5	savings	3150
106	6	current	8000
107	7	savings	9450
108	8	current	11000
109	9	savings	6300
110	10	current	9500

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

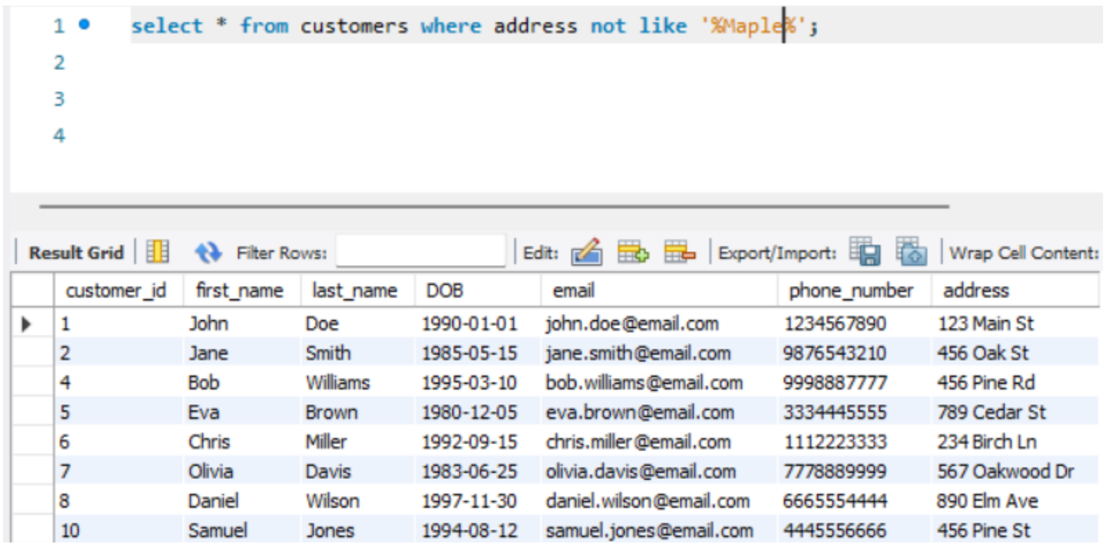


```
/* Overdraft limit - 4000 */
select * from accounts where
balance < 4000;
```

Result Grid | Filter Rows: | Ed

	account_id	customer_id	account_type	balance
▶	105	5	savings	3150

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



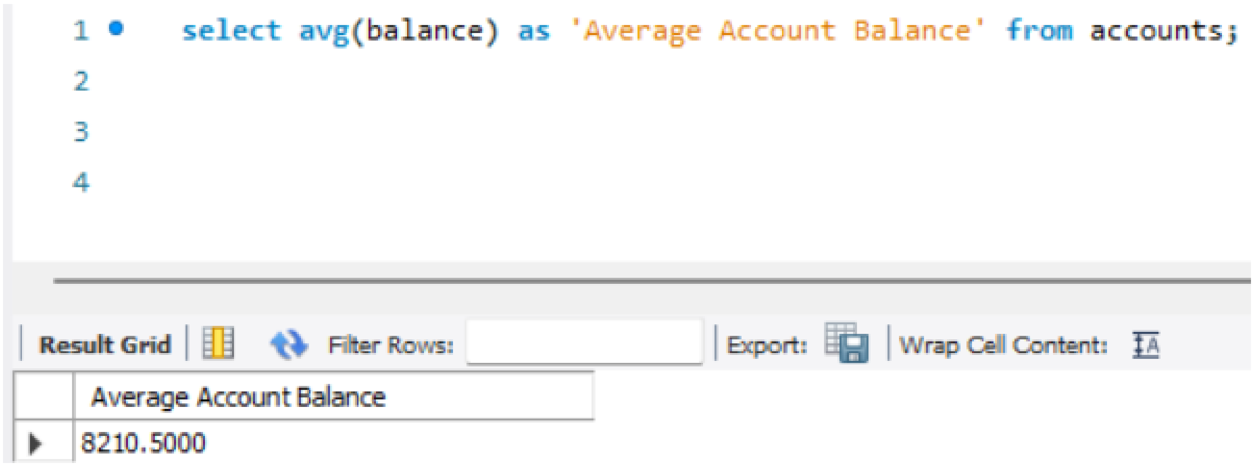
```
select * from customers where address not like '%Maple%';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

	customer_id	first_name	last_name	DOB	email	phone_number	address
▶	1	John	Doe	1990-01-01	john.doe@email.com	1234567890	123 Main St
	2	Jane	Smith	1985-05-15	jane.smith@email.com	9876543210	456 Oak St
	4	Bob	Williams	1995-03-10	bob.williams@email.com	9998887777	456 Pine Rd
	5	Eva	Brown	1980-12-05	eva.brown@email.com	3334445555	789 Cedar St
	6	Chris	Miller	1992-09-15	chris.miller@email.com	1112223333	234 Birch Ln
	7	Olivia	Davis	1983-06-25	olivia.davis@email.com	7778889999	567 Oakwood Dr
	8	Daniel	Wilson	1997-11-30	daniel.wilson@email.com	6665554444	890 Elm Ave
	10	Samuel	Jones	1994-08-12	samuel.jones@email.com	4445556666	456 Pine St

Task 3

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

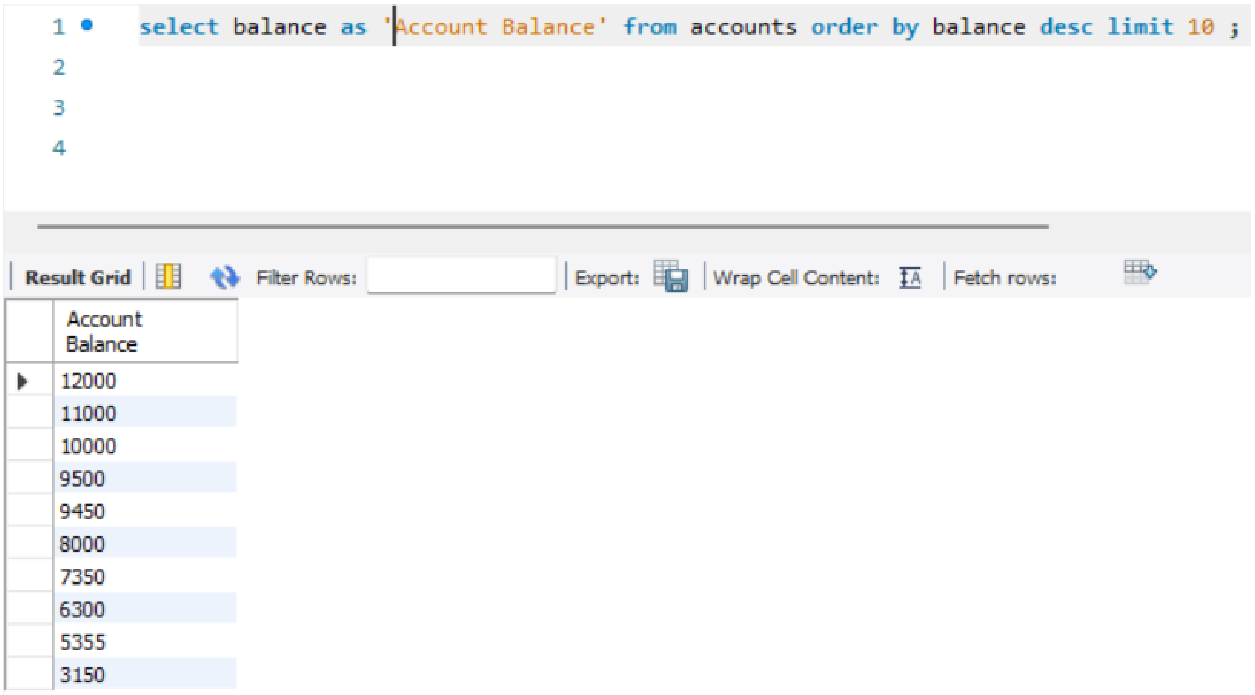


```
select avg(balance) as 'Average Account Balance' from accounts;
```

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

Average Account Balance
8210.5000

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

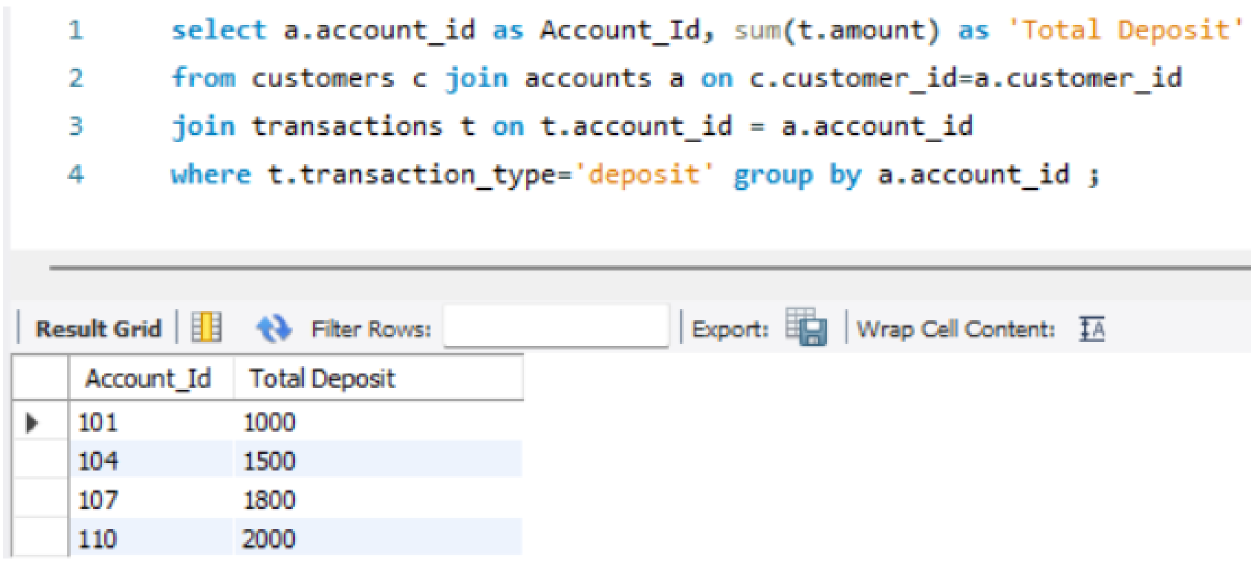


```
select balance as 'Account Balance' from accounts order by balance desc limit 10 ;
```

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

Account Balance
12000
11000
10000
9500
9450
8000
7350
6300
5355
3150

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

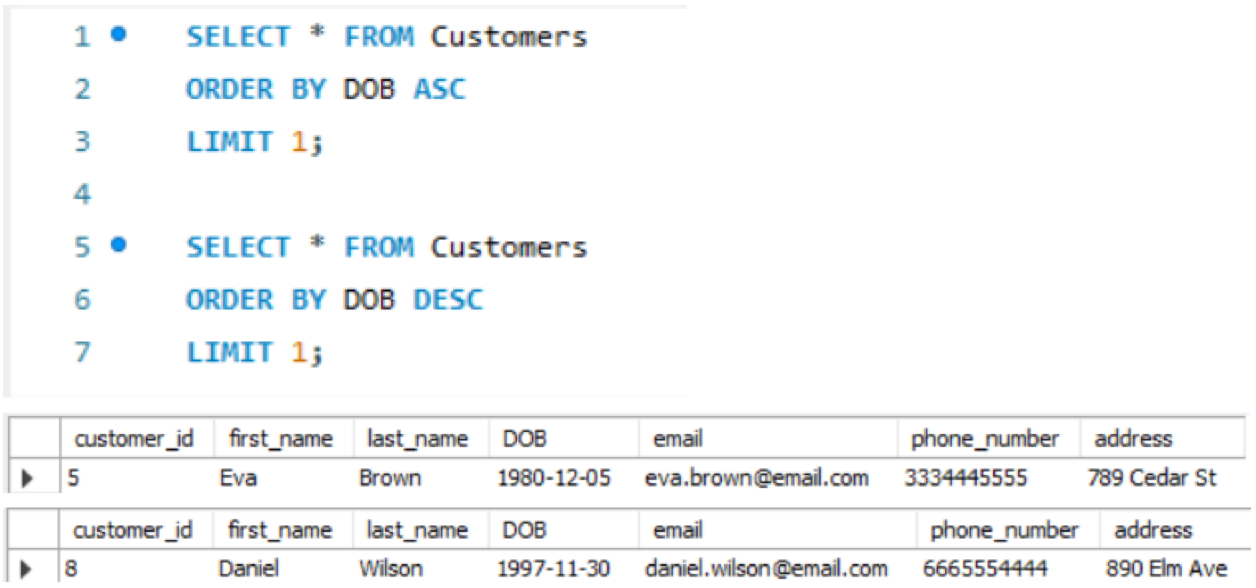


```sql
select a.account_id as Account_Id, sum(t.amount) as 'Total Deposit'
from customers c join accounts a on c.customer_id=a.customer_id
join transactions t on t.account_id = a.account_id
where t.transaction_type='deposit' group by a.account_id ;
```

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

Account_Id	Total Deposit
101	1000
104	1500
107	1800
110	2000

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



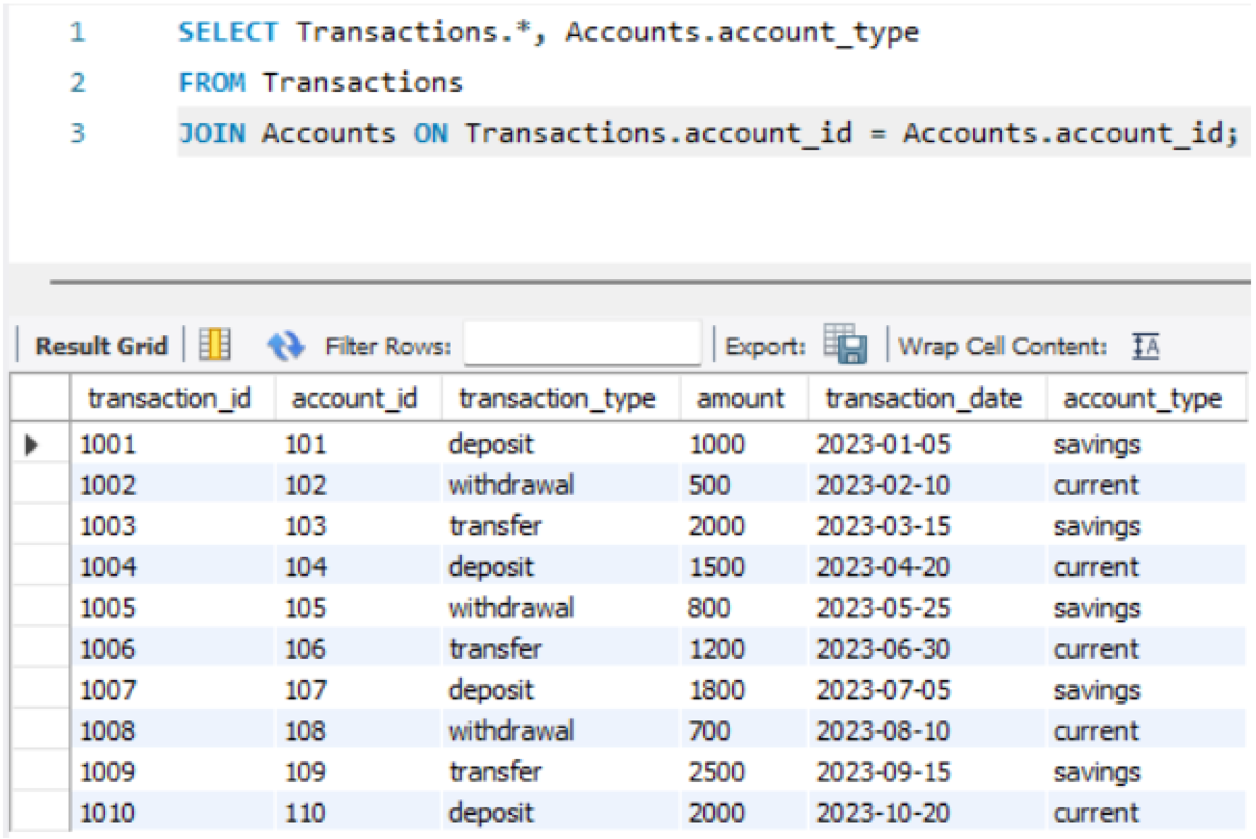
```sql
SELECT * FROM Customers
ORDER BY DOB ASC
LIMIT 1;

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

customer_id	first_name	last_name	DOB	email	phone_number	address
5	Eva	Brown	1980-12-05	eva.brown@email.com	3334445555	789 Cedar St

customer_id	first_name	last_name	DOB	email	phone_number	address
8	Daniel	Wilson	1997-11-30	daniel.wilson@email.com	6665554444	890 Elm Ave

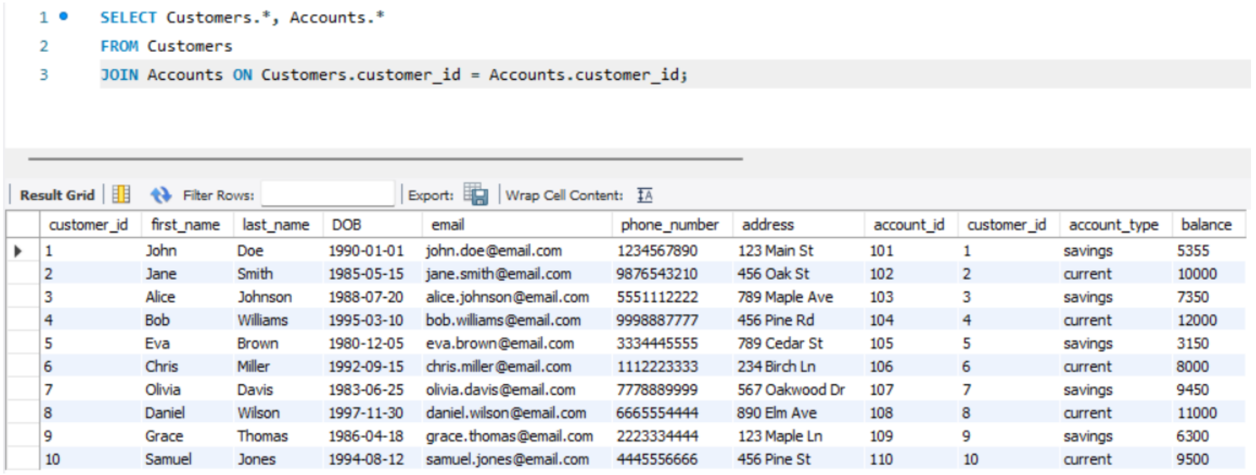
v.) Write a SQL query to Retrieve transaction details along with the account type.



```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	1000	2023-01-05	savings
1002	102	withdrawal	500	2023-02-10	current
1003	103	transfer	2000	2023-03-15	savings
1004	104	deposit	1500	2023-04-20	current
1005	105	withdrawal	800	2023-05-25	savings
1006	106	transfer	1200	2023-06-30	current
1007	107	deposit	1800	2023-07-05	savings
1008	108	withdrawal	700	2023-08-10	current
1009	109	transfer	2500	2023-09-15	savings
1010	110	deposit	2000	2023-10-20	current

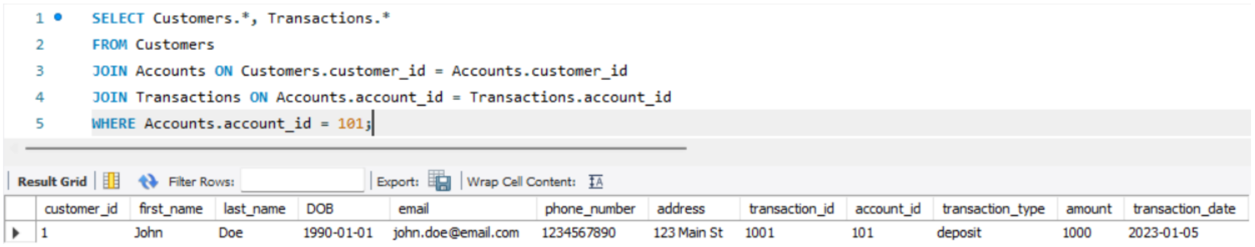
vi.) Write a SQL query to Get a list of customers along with their account details.



```sql
SELECT Customers.*, Accounts.*
FROM Customers
JOIN Accounts ON Customers.customer_id = Accounts.customer_id;
```

customer_id	first_name	last_name	DOB	email	phone_number	address	account_id	customer_id	account_type	balance
1	John	Doe	1990-01-01	john.doe@email.com	1234567890	123 Main St	101	1	savings	5355
2	Jane	Smith	1985-05-15	jane.smith@email.com	9876543210	456 Oak St	102	2	current	10000
3	Alice	Johnson	1988-07-20	alice.johnson@email.com	5551112222	789 Maple Ave	103	3	savings	7350
4	Bob	Williams	1995-03-10	bob.williams@email.com	9998887777	456 Pine Rd	104	4	current	12000
5	Eva	Brown	1980-12-05	eva.brown@email.com	3334445555	789 Cedar St	105	5	savings	3150
6	Chris	Miller	1992-09-15	chris.miller@email.com	1112223333	234 Birch Ln	106	6	current	8000
7	Olivia	Davis	1983-06-25	olivia.davis@email.com	7778889999	567 Oakwood Dr	107	7	savings	9450
8	Daniel	Wilson	1997-11-30	daniel.wilson@email.com	6665554444	890 Elm Ave	108	8	current	11000
9	Grace	Thomas	1986-04-18	grace.thomas@email.com	2223334444	123 Maple Ln	109	9	savings	6300
10	Samuel	Jones	1994-08-12	samuel.jones@email.com	4445556666	456 Pine St	110	10	current	9500

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

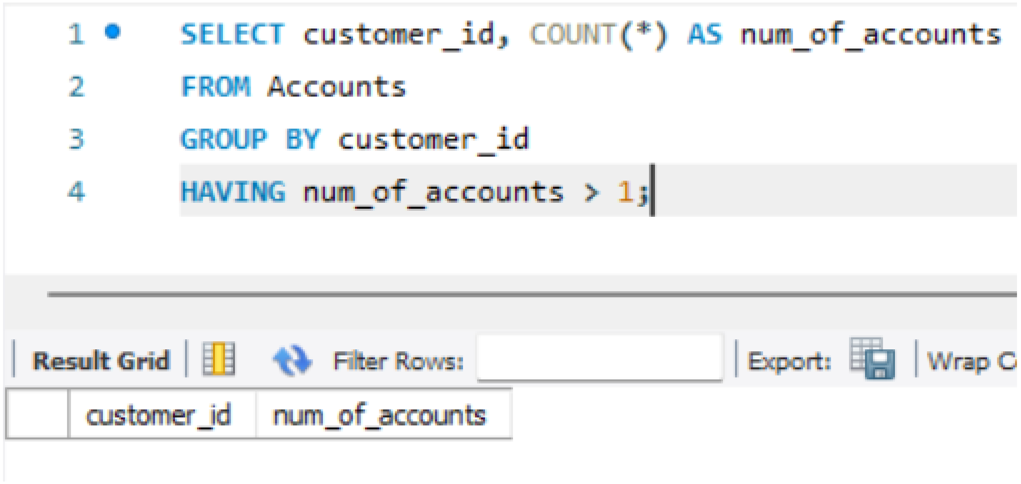


```sql
SELECT Customers.*, Transactions.*
FROM Customers
JOIN Accounts ON Customers.customer_id = Accounts.customer_id
JOIN Transactions ON Accounts.account_id = Transactions.account_id
WHERE Accounts.account_id = 101;
```

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

customer_id	first_name	last_name	DOB	email	phone_number	address	transaction_id	account_id	transaction_type	amount	transaction_date
1	John	Doe	1990-01-01	john.doe@email.com	1234567890	123 Main St	1001	101	deposit	1000	2023-01-05

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

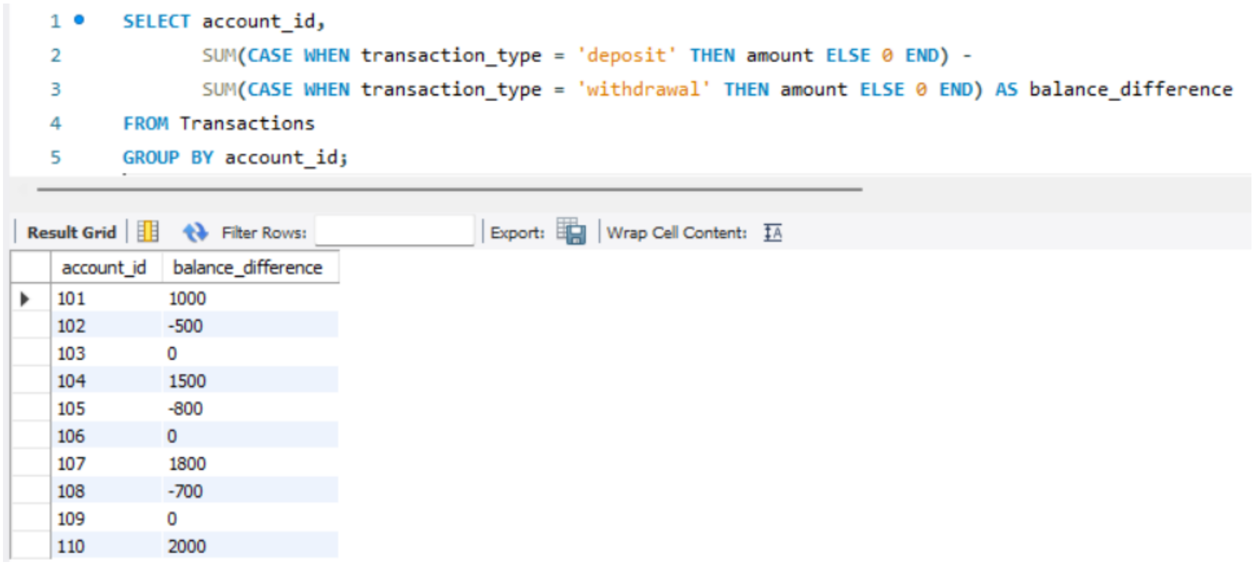


```sql
SELECT customer_id, COUNT(*) AS num_of_accounts
FROM Accounts
GROUP BY customer_id
HAVING num_of_accounts > 1;
```

Result Grid | Filter Rows: | Export: | Wrap C

customer_id	num_of_accounts

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



```sql
SELECT account_id,
        SUM(CASE WHEN transaction_type = 'deposit' THEN amount ELSE 0 END) -
        SUM(CASE WHEN transaction_type = 'withdrawal' THEN amount ELSE 0 END) AS balance_difference
FROM Transactions
GROUP BY account_id;
```

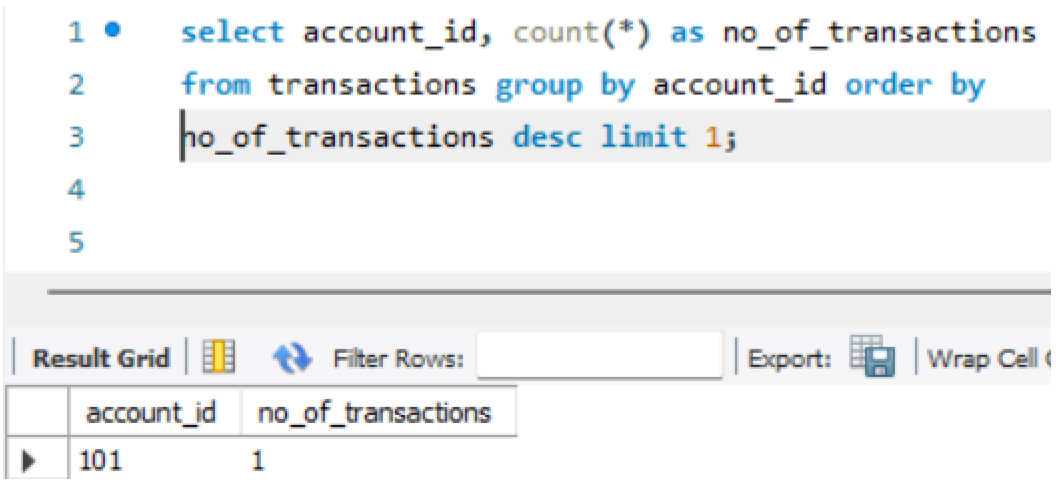
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

account_id	balance_difference
101	1000
102	-500
103	0
104	1500
105	-800
106	0
107	1800
108	-700
109	0
110	2000

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

To do

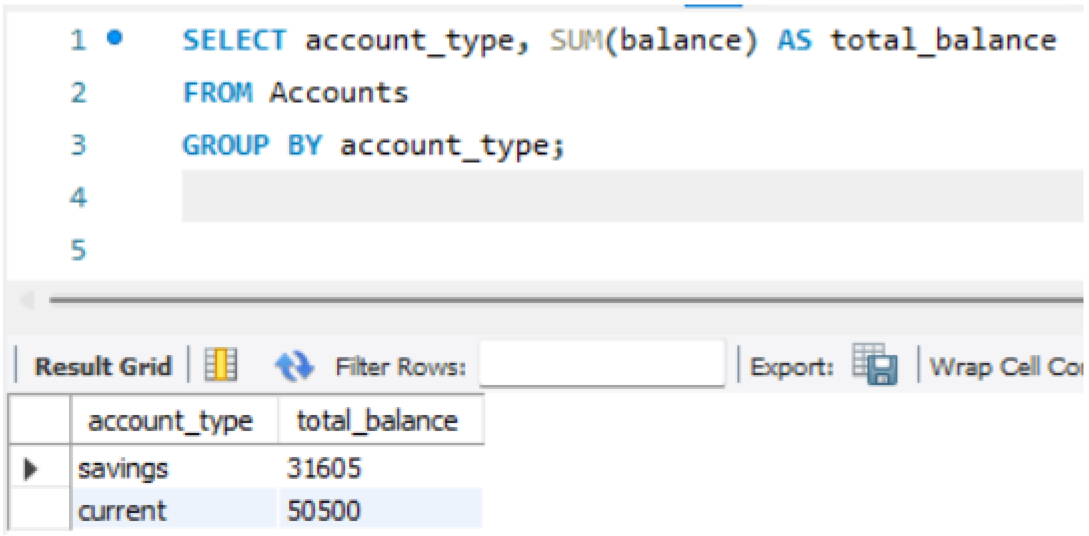
xi.) Calculate the total balance for each account type



```sql
select account_id, count(*) as no_of_transactions
from transactions group by account_id order by
no_of_transactions desc limit 1;
```

account_id	no_of_transactions
101	1

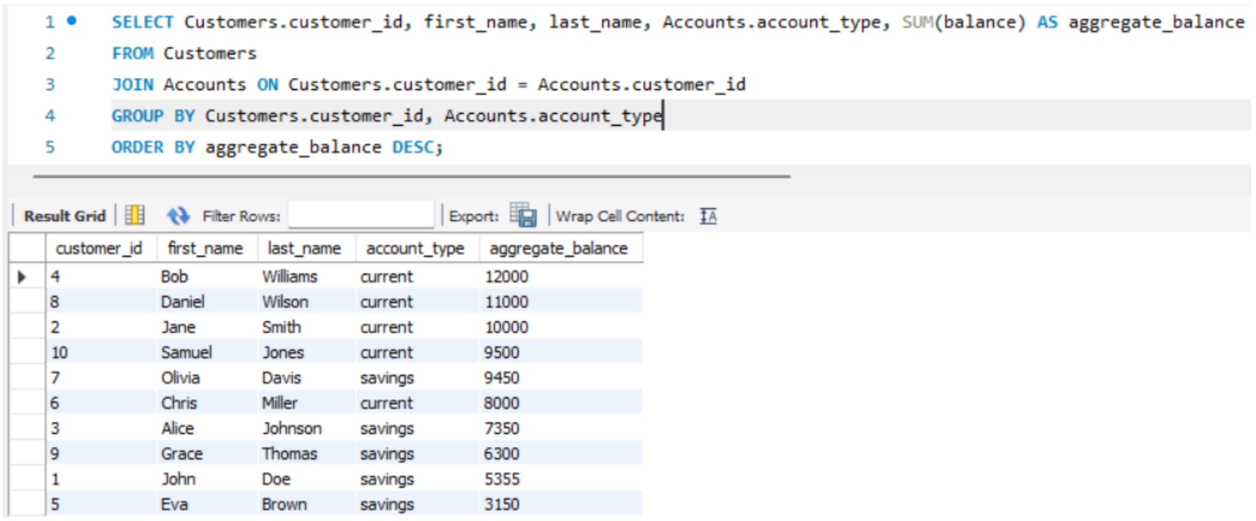
xii.) Identify accounts with the highest number of transactions order by descending order



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

account_type	total_balance
savings	31605
current	50500

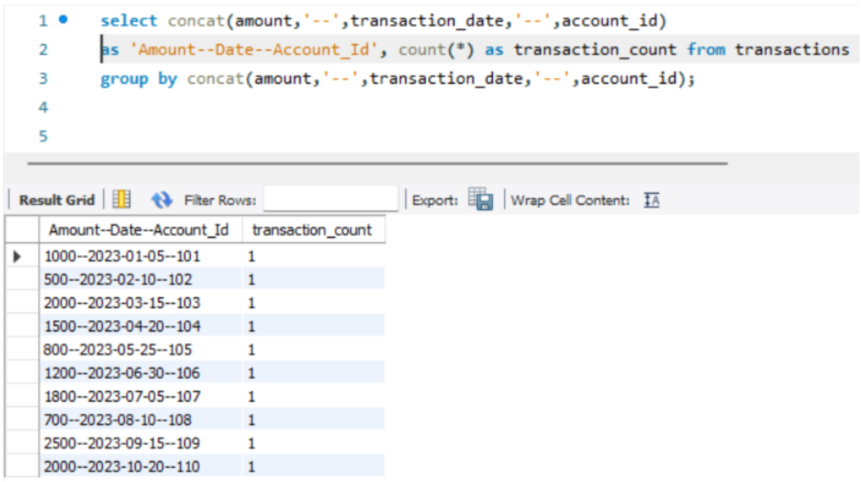
xiii.) List customers with high aggregate account balances, along with their account types.



```sql
SELECT Customers.customer_id, first_name, last_name, Accounts.account_type, SUM(balance) AS aggregate_balance
FROM Customers
JOIN Accounts ON Customers.customer_id = Accounts.customer_id
GROUP BY Customers.customer_id, Accounts.account_type
ORDER BY aggregate_balance DESC;
```

customer_id	first_name	last_name	account_type	aggregate_balance
4	Bob	Williams	current	12000
8	Daniel	Wilson	current	11000
2	Jane	Smith	current	10000
10	Samuel	Jones	current	9500
7	Olivia	Davis	savings	9450
6	Chris	Miller	current	8000
3	Alice	Johnson	savings	7350
9	Grace	Thomas	savings	6300
1	John	Doe	savings	5355
5	Eva	Brown	savings	3150

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

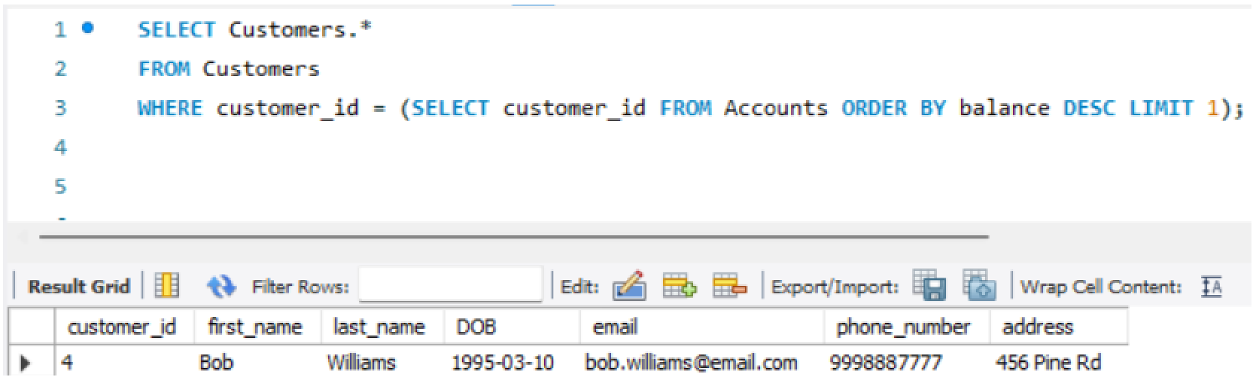


```
select concat(amount,'--',transaction_date,'--',account_id)
as 'Amount--Date--Account_Id', count(*) as transaction_count from transactions
group by concat(amount,'--',transaction_date,'--',account_id);
```

Amount--Date--Account_Id	transaction_count
1000--2023-01-05--101	1
500--2023-02-10--102	1
2000--2023-03-15--103	1
1500--2023-04-20--104	1
800--2023-05-25--105	1
1200--2023-06-30--106	1
1800--2023-07-05--107	1
700--2023-08-10--108	1
2500--2023-09-15--109	1
2000--2023-10-20--110	1

Task 4

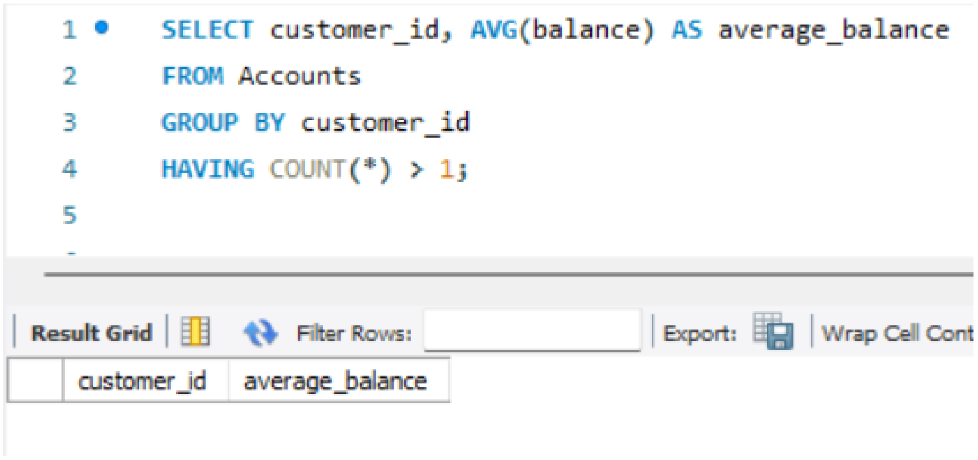
1. Retrieve the customer(s) with the highest account balance



```
SELECT Customers.*
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	Bob	Williams	1995-03-10	bob.williams@email.com	9998887777	456 Pine Rd

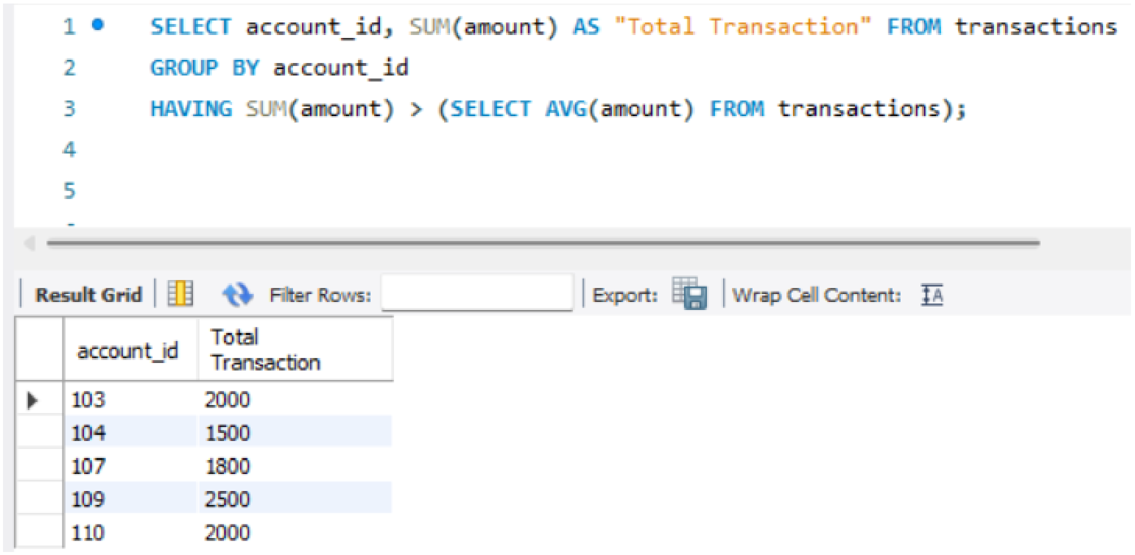
2. Calculate the average account balance for customers who have more than one account.


```
SELECT customer_id, AVG(balance) AS average_balance
FROM Accounts
GROUP BY customer_id
HAVING COUNT(*) > 1;
```

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

customer_id	average_balance

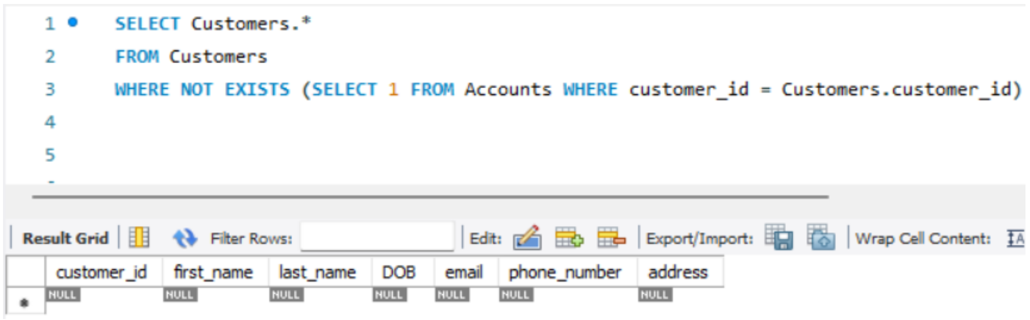
3. Retrieve accounts with transactions whose amounts exceed the average transaction amount


```
SELECT account_id, SUM(amount) AS "Total Transaction" FROM transactions
GROUP BY account_id
HAVING SUM(amount) > (SELECT AVG(amount) FROM transactions);
```

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

account_id	Total Transaction
103	2000
104	1500
107	1800
109	2500
110	2000

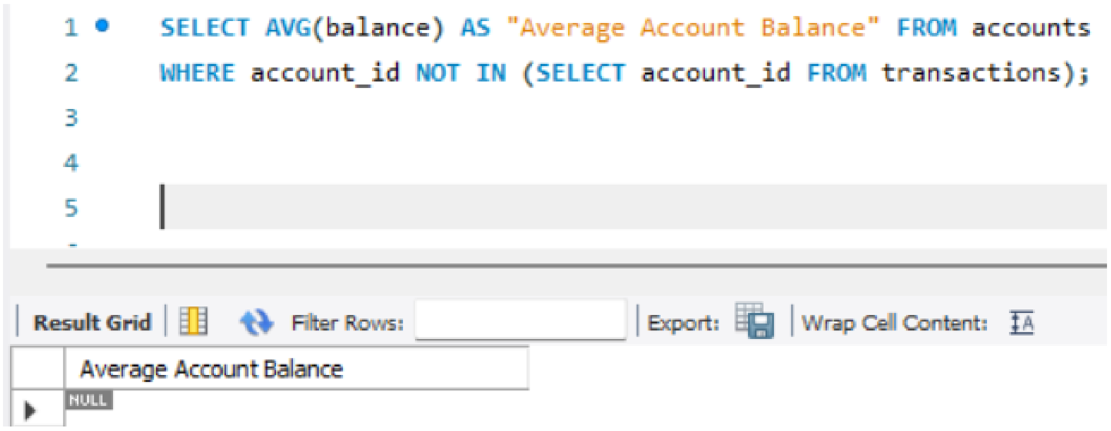
4. Identify customers who have no recorded transaction


```
SELECT Customers.*
FROM Customers
WHERE NOT EXISTS (SELECT 1 FROM Accounts WHERE customer_id = Customers.customer_id)
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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.

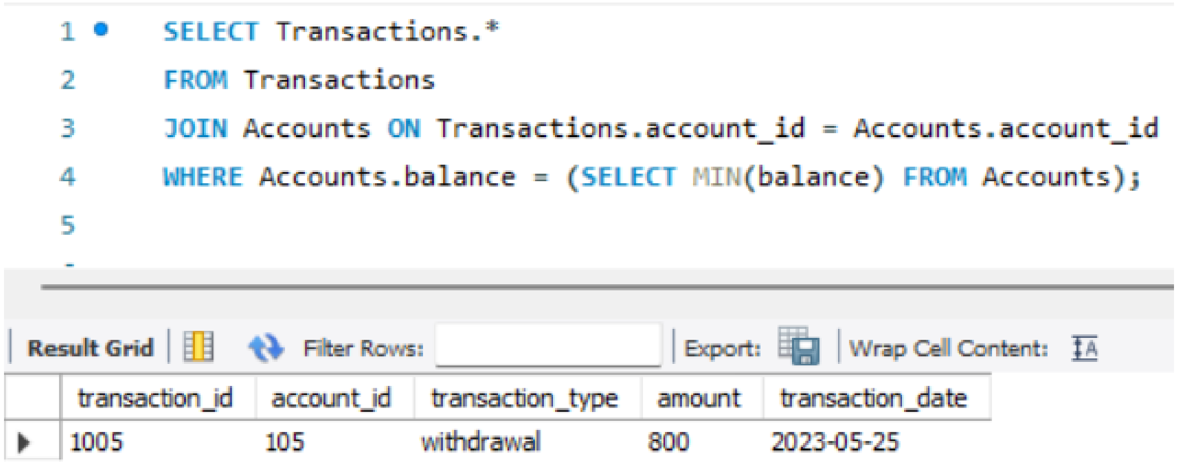


```
SELECT AVG(balance) AS "Average Account Balance" FROM accounts
WHERE account_id NOT IN (SELECT account_id FROM transactions);
```

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

	Average Account Balance
▶	NULL

6. Retrieve transactions for accounts with the lowest balance.

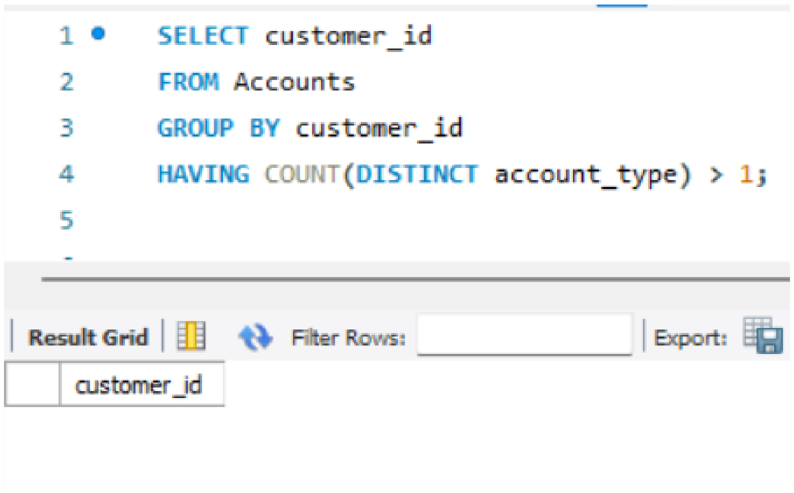


```
SELECT Transactions.*
FROM Transactions
JOIN Accounts ON Transactions.account_id = Accounts.account_id
WHERE Accounts.balance = (SELECT MIN(balance) FROM Accounts);
```

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

	transaction_id	account_id	transaction_type	amount	transaction_date
▶	1005	105	withdrawal	800	2023-05-25

7. Identify customers who have accounts of multiple types.

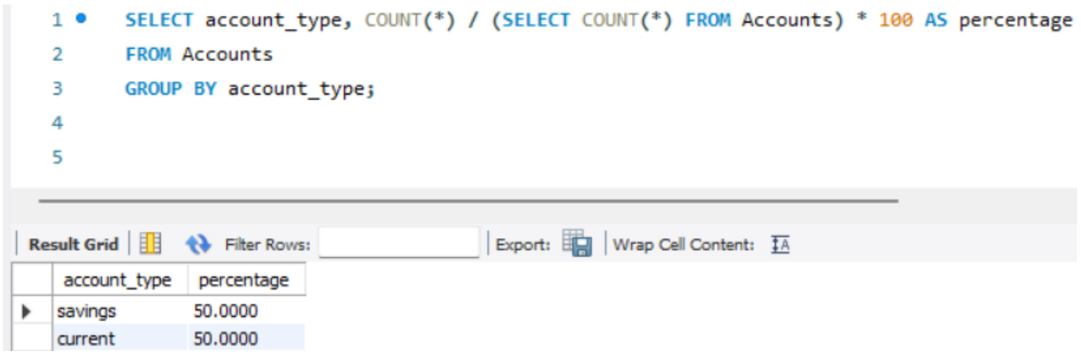


```
SELECT customer_id
FROM Accounts
GROUP BY customer_id
HAVING COUNT(DISTINCT account_type) > 1;
```

Result Grid | Filter Rows: | Export:

	customer_id

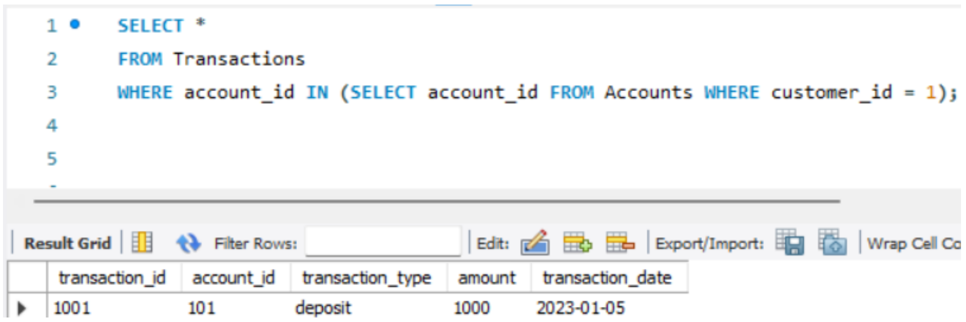
8. Calculate the percentage of each account type out of the total number of accounts.



```
SELECT account_type, COUNT(*) / (SELECT COUNT(*) FROM Accounts) * 100 AS percentage
FROM Accounts
GROUP BY account_type;
```

account_type	percentage
savings	50.0000
current	50.0000

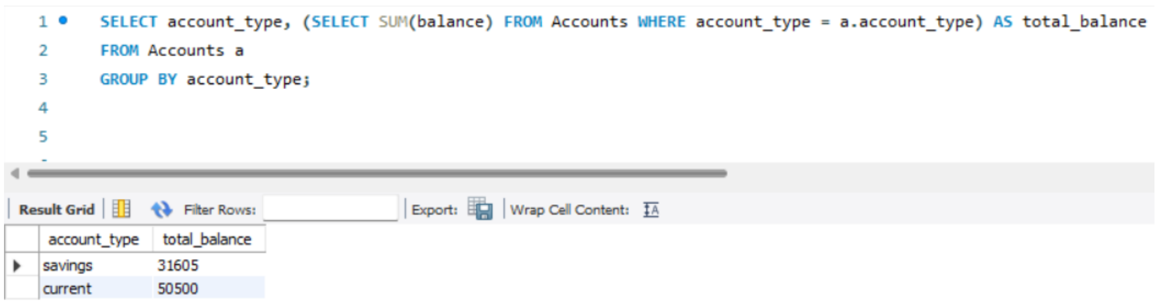
9. Retrieve all transactions for a customer with a given customer_id.



```
SELECT *
FROM Transactions
WHERE account_id IN (SELECT account_id FROM Accounts WHERE customer_id = 1);
```

transaction_id	account_id	transaction_type	amount	transaction_date
1001	101	deposit	1000	2023-01-05

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



```
SELECT account_type, (SELECT SUM(balance) FROM Accounts WHERE account_type = a.account_type) AS total_balance
FROM Accounts a
GROUP BY account_type;
```

account_type	total_balance
savings	31605
current	50500