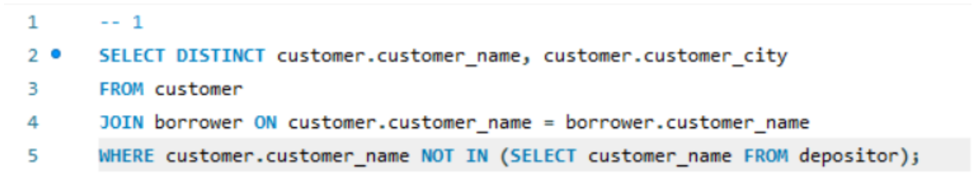
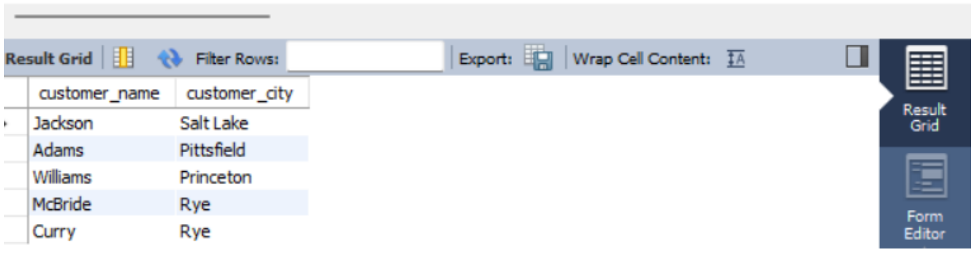
SQL Query Report(Lab 3)

1. Customers with Loans but No Accounts

This query retrieves a list of customers who have taken out loans but do not hold any accounts with the bank. It helps in identifying potential customers who may be in need of banking services.



```sql
-- 1
SELECT DISTINCT customer.customer_name, customer.customer_city
FROM customer
JOIN borrower ON customer.customer_name = borrower.customer_name
WHERE customer.customer_name NOT IN (SELECT customer_name FROM depositor);
```

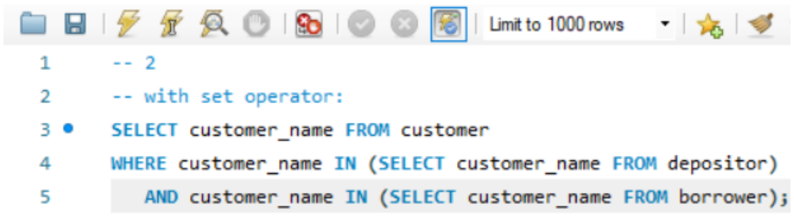


customer_name	customer_city
Jackson	Salt Lake
Adams	Pittsfield
Williams	Princeton
McBride	Rye
Curry	Rye

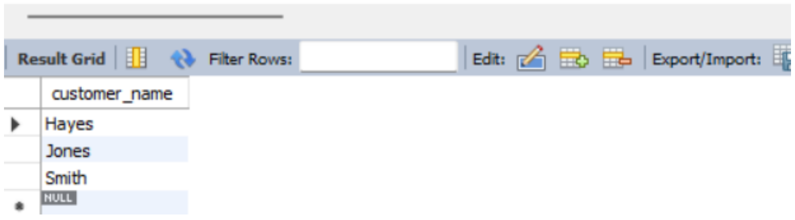
2. Customers with Both Accounts and Loans

Two approaches were used to find customers who have both an account and a loan:

- **Set Operator**: This method uses a subquery to find customer names present in both the borrower and depositor tables.

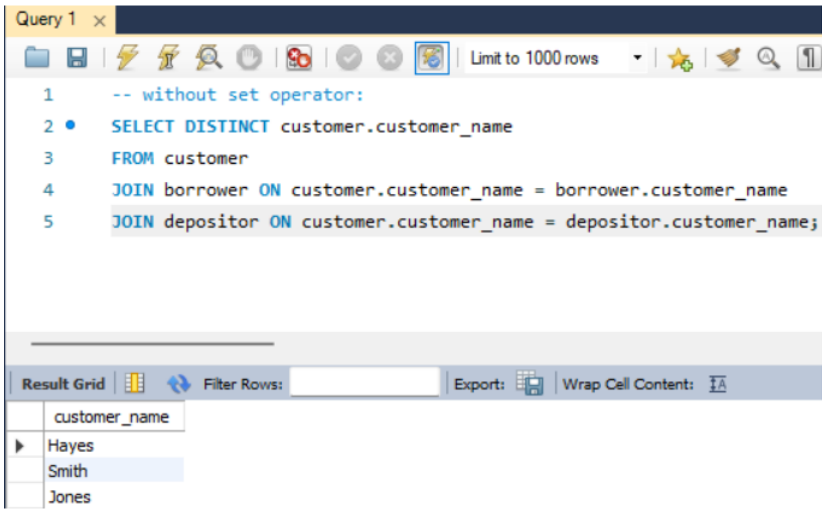


```sql
-- 2
-- with set operator:
SELECT customer_name FROM customer
WHERE customer_name IN (SELECT customer_name FROM depositor)
   AND customer_name IN (SELECT customer_name FROM borrower);
```



customer_name
Hayes
Jones
Smith
NULL

- **Without Set Operator**: This approach utilizes joins to directly obtain customer names from both tables, providing a more efficient execution plan.



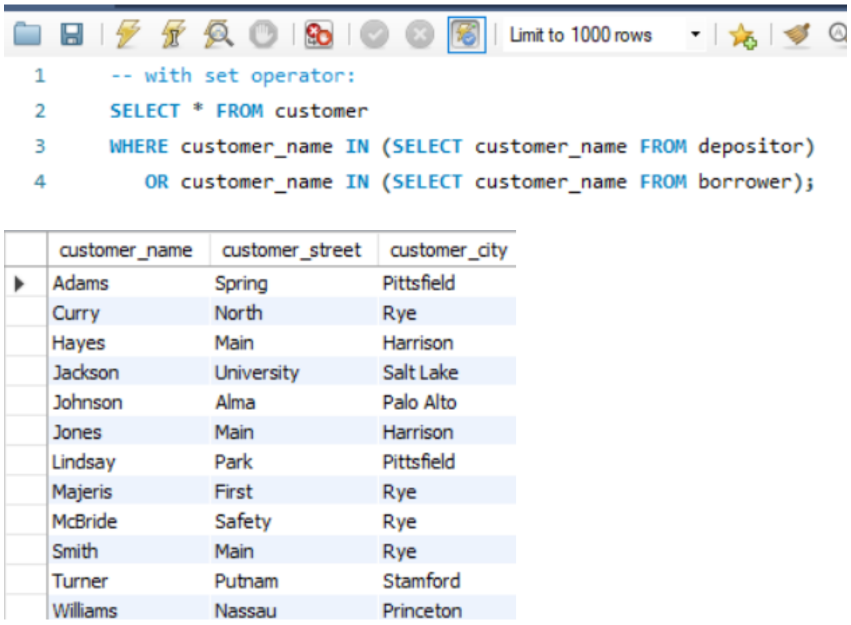
```sql
-- without set operator:
SELECT DISTINCT customer.customer_name
FROM customer
JOIN borrower ON customer.customer_name = borrower.customer_name
JOIN depositor ON customer.customer_name = depositor.customer_name;
```

customer_name
Hayes
Smith
Jones

3. Customers with Accounts or Loans

Similar to the previous task, this query identifies customers who have either an account or a loan:

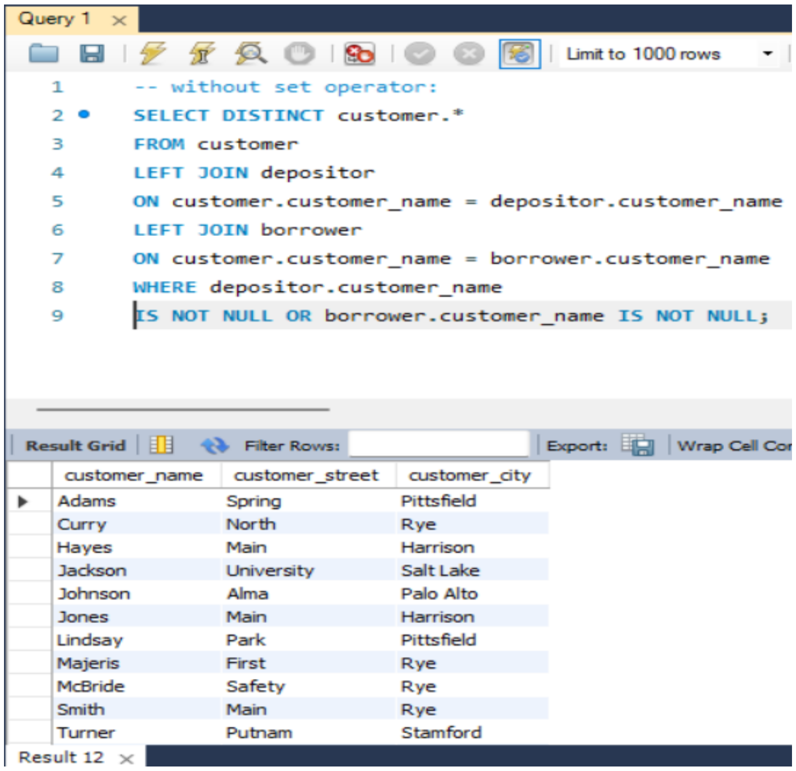
- **With Set Operator**: It uses subqueries to find customers in either table.



```sql
-- with set operator:
SELECT * FROM customer
WHERE customer_name IN (SELECT customer_name FROM depositor)
   OR customer_name IN (SELECT customer_name FROM borrower);
```

customer_name	customer_street	customer_city
Adams	Spring	Pittsfield
Curry	North	Rye
Hayes	Main	Harrison
Jackson	University	Salt Lake
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Majeris	First	Rye
McBride	Safety	Rye
Smith	Main	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

- **Without Set Operator**: This method employs left joins to gather all customer information where they have an account or a loan.

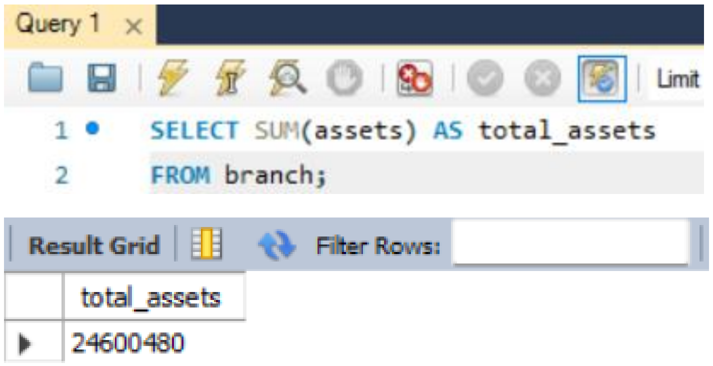


```sql
-- without set operator:
SELECT DISTINCT customer.*
FROM customer
LEFT JOIN depositor
ON customer.customer_name = depositor.customer_name
LEFT JOIN borrower
ON customer.customer_name = borrower.customer_name
WHERE depositor.customer_name
IS NOT NULL OR borrower.customer_name IS NOT NULL;
```

customer_name	customer_street	customer_city
Adams	Spring	Pittsfield
Curry	North	Rye
Hayes	Main	Harrison
Jackson	University	Salt Lake
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Majeris	First	Rye
McBride	Safety	Rye
Smith	Main	Rye
Turner	Putnam	Stamford

4. Total Assets of Branches

This query computes the total assets held across all branches, providing insight into the overall financial health of the bank.

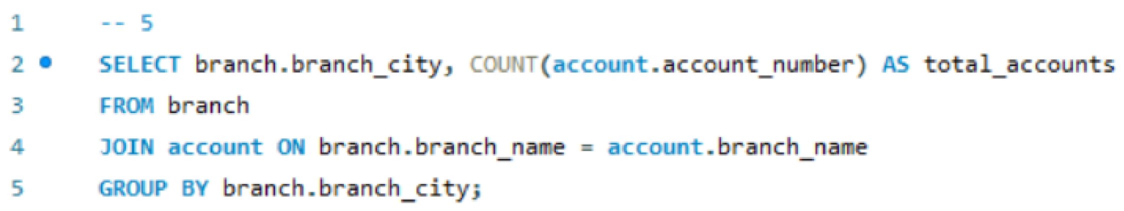


```sql
SELECT SUM(assets) AS total_assets
FROM branch;
```

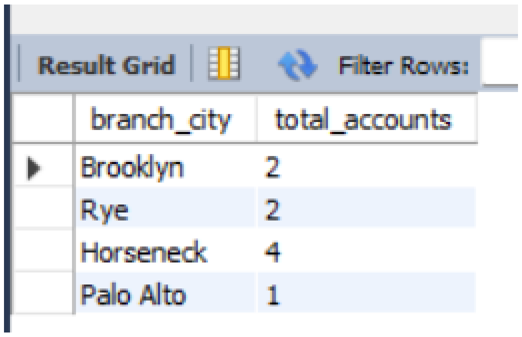
total_assets
24600480

5. Total Number of Accounts by Branch City

This query counts the number of accounts in each branch city, allowing for geographical analysis of account distribution.



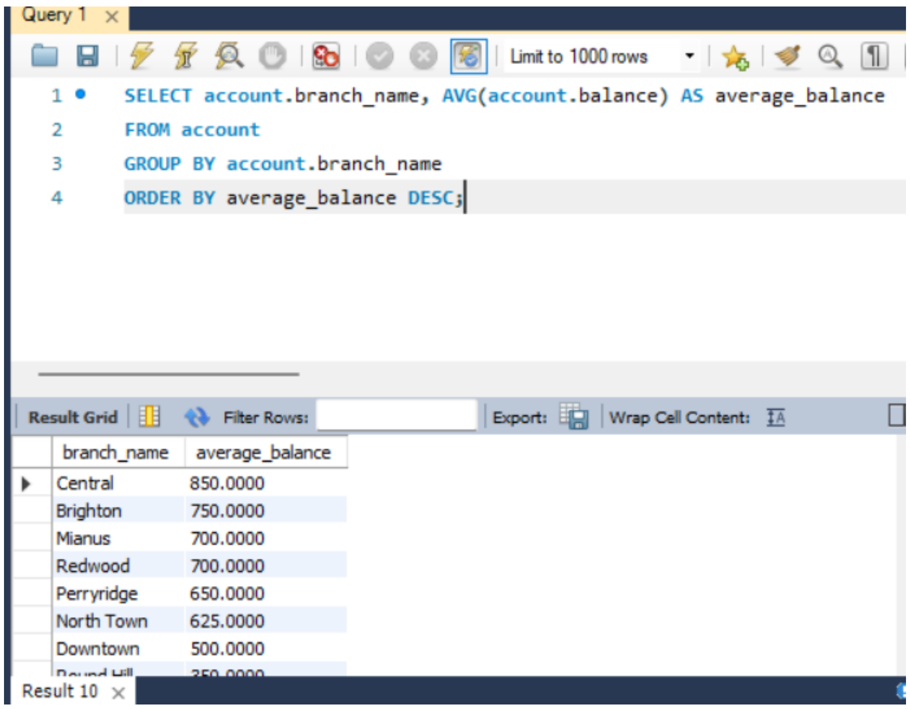
```sql
-- 5
SELECT branch.branch_city, COUNT(account.account_number) AS total_accounts
FROM branch
JOIN account ON branch.branch_name = account.branch_name
GROUP BY branch.branch_city;
```



branch_city	total_accounts
Brooklyn	2
Rye	2
Horseneck	4
Palo Alto	1

6. Average Balance of Accounts by Branch

This query calculates the average balance of accounts at each branch and sorts the results in descending order. It is valuable for assessing the financial status of different branches.

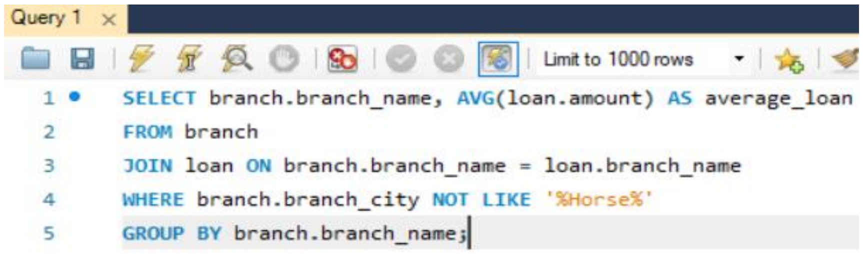


```sql
SELECT account.branch_name, AVG(account.balance) AS average_balance
FROM account
GROUP BY account.branch_name
ORDER BY average_balance DESC;
```

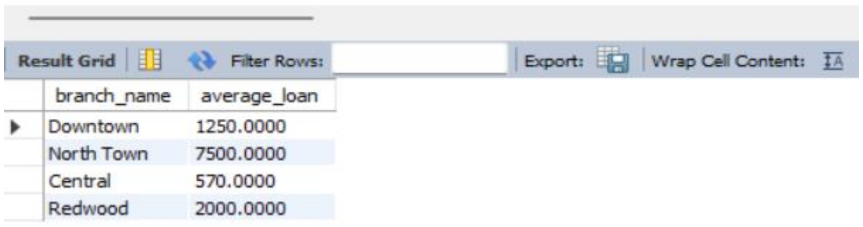
branch_name	average_balance
Central	850.0000
Brighton	750.0000
Mianus	700.0000
Redwood	700.0000
Perryridge	650.0000
North Town	625.0000
Downtown	500.0000
Round Hill	350.0000

7. Average Loan Amount by Branch Excluding Certain Cities

This query finds the average loan amount for each branch while excluding branches located in cities containing "Horse" in their names. This can help focus on specific geographical areas.



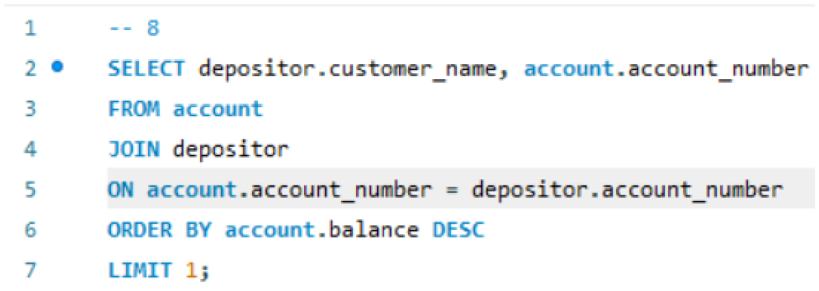
```sql
SELECT branch.branch_name, AVG(loan.amount) AS average_loan
FROM branch
JOIN loan ON branch.branch_name = loan.branch_name
WHERE branch.branch_city NOT LIKE '%Horse%'
GROUP BY branch.branch_name;
```



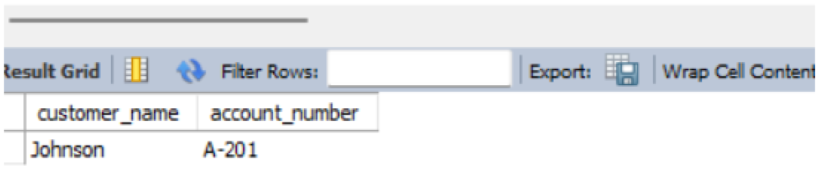
branch_name	average_loan
Downtown	1250.0000
North Town	7500.0000
Central	570.0000
Redwood	2000.0000

8. Account with the Highest Balance

This query identifies the customer and their account number for the account with the highest balance, assisting in recognizing top clients.



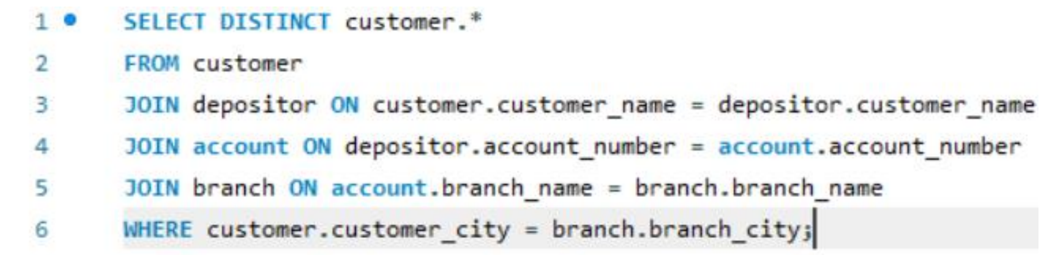
```sql
-- 8
SELECT depositor.customer_name, account.account_number
FROM account
JOIN depositor
ON account.account_number = depositor.account_number
ORDER BY account.balance DESC
LIMIT 1;
```



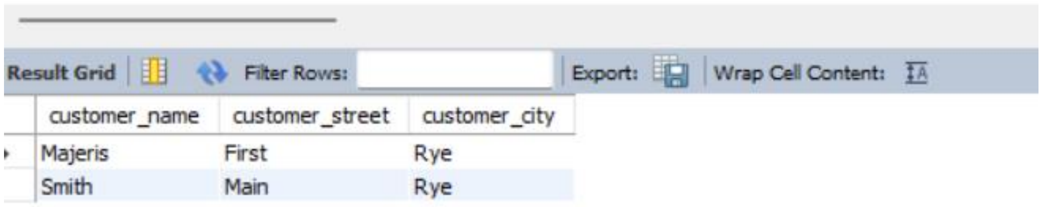
customer_name	account_number
Johnson	A-201

9. Customers with Accounts in Their City of Residence

This query retrieves customer information for those who have accounts in branches located in the same city where they reside, promoting personalized banking services.



```
SELECT DISTINCT customer.*
FROM customer
JOIN depositor ON customer.customer_name = depositor.customer_name
JOIN account ON depositor.account_number = account.account_number
JOIN branch ON account.branch_name = branch.branch_name
WHERE customer.customer_city = branch.branch_city;
```

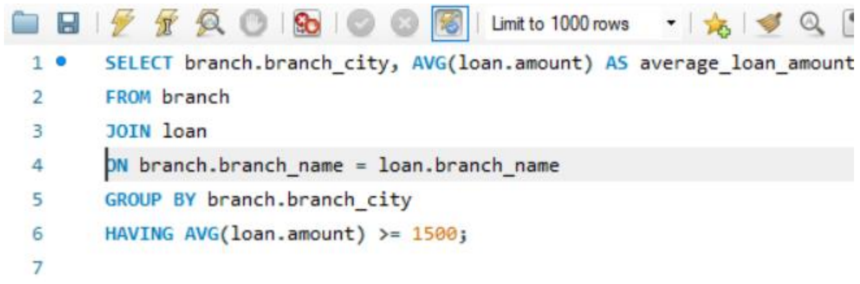


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_name	customer_street	customer_city
Majeris	First	Rye
Smith	Main	Rye

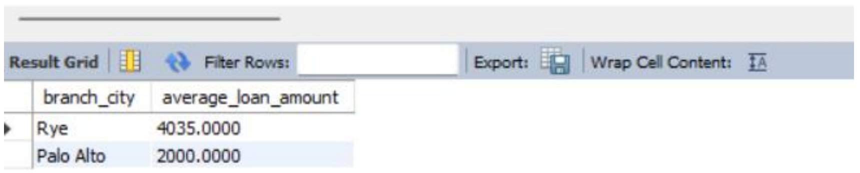
10. Average Loan Amount by Branch City with a Minimum Threshold

This query calculates the average loan amount for loans opened in each branch city, filtering out cities where the average amount is below 1500. This helps identify cities with significant lending activity.



Limit to 1000 rows

```
SELECT branch.branch_city, AVG(loan.amount) AS average_loan_amount
FROM branch
JOIN loan
ON branch.branch_name = loan.branch_name
GROUP BY branch.branch_city
HAVING AVG(loan.amount) >= 1500;
```

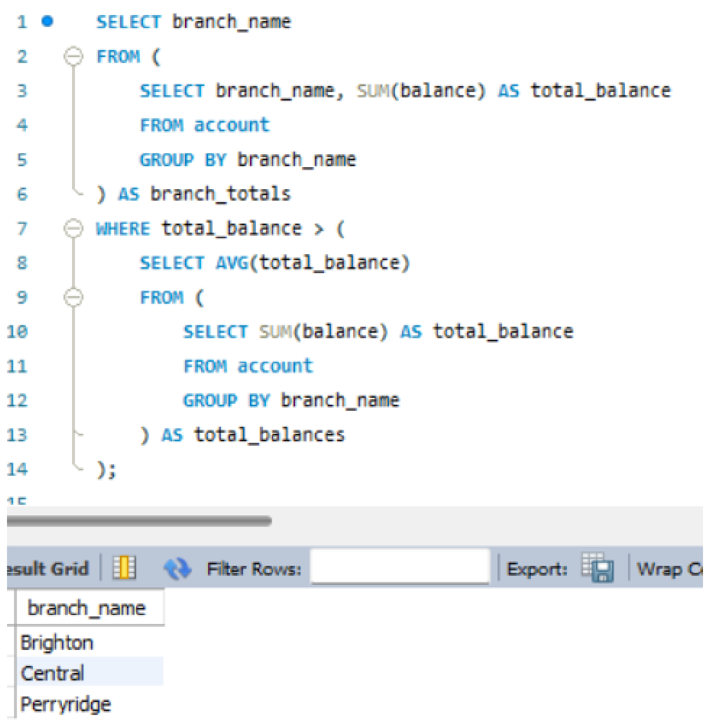


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

branch_city	average_loan_amount
Rye	4035.0000
Palo Alto	2000.0000

11. Branches with Higher Total Account Balances

This query determines which branches have a total account balance exceeding the average balance among all branches, highlighting financially stronger branches.

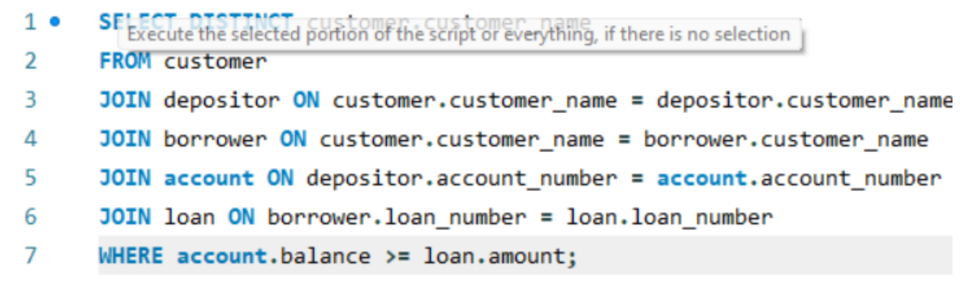


```sql
SELECT branch_name
FROM (
    SELECT branch_name, SUM(balance) AS total_balance
    FROM account
    GROUP BY branch_name
) AS branch_totals
WHERE total_balance > (
    SELECT AVG(total_balance)
    FROM (
        SELECT SUM(balance) AS total_balance
        FROM account
        GROUP BY branch_name
    ) AS total_balances
);
```

```
branch_name
Brighton
Central
Perryridge
```

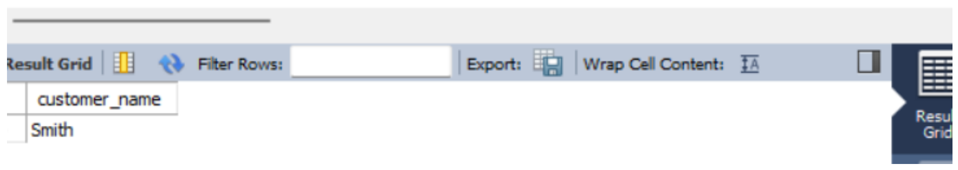
12. Customers Who Can Pay Off Their Loans

This query finds customers who have at least one loan that can be paid off using their total account balance, indicating their financial stability.



Execute the selected portion of the script or everything, if there is no selection

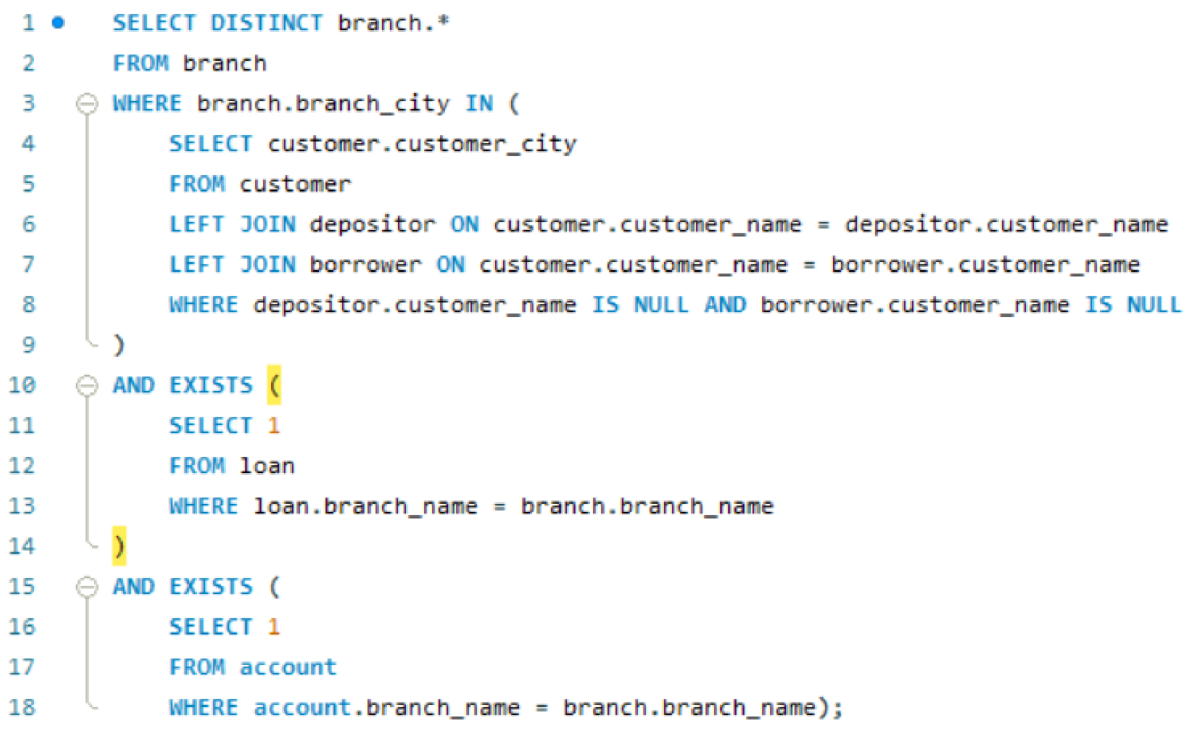
```sql
SELECT DISTINCT customer.customer_name
FROM customer
JOIN depositor ON customer.customer_name = depositor.customer_name
JOIN borrower ON customer.customer_name = borrower.customer_name
JOIN account ON depositor.account_number = account.account_number
JOIN loan ON borrower.loan_number = loan.loan_number
WHERE account.balance >= loan.amount;
```



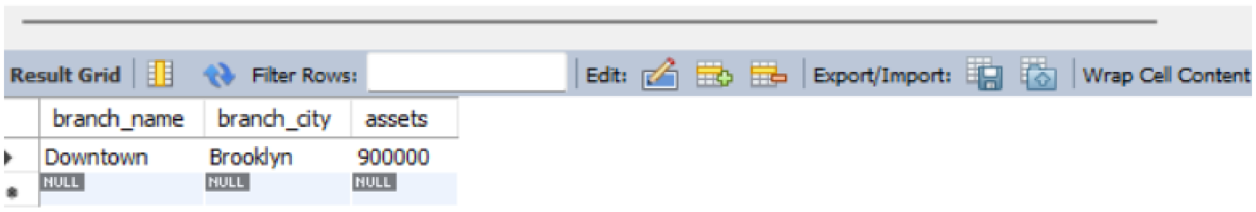
```
customer_name
Smith
```

13. Branch Information for Cities with Customers Lacking Accounts or Loans

This query retrieves branch information for cities where at least one customer lives without any accounts or loans, provided the branch has active accounts and loans from other customers. This can highlight underserved markets.



```sql
SELECT DISTINCT branch.*
FROM branch
WHERE branch.branch_city IN (
    SELECT customer.customer_city
    FROM customer
    LEFT JOIN depositor ON customer.customer_name = depositor.customer_name
    LEFT JOIN borrower ON customer.customer_name = borrower.customer_name
    WHERE depositor.customer_name IS NULL AND borrower.customer_name IS NULL
)
AND EXISTS (
    SELECT 1
    FROM loan
    WHERE loan.branch_name = branch.branch_name
)
AND EXISTS (
    SELECT 1
    FROM account
    WHERE account.branch_name = branch.branch_name);
```



branch_name	branch_city	assets
Downtown	Brooklyn	900000
NULL	NULL	NULL