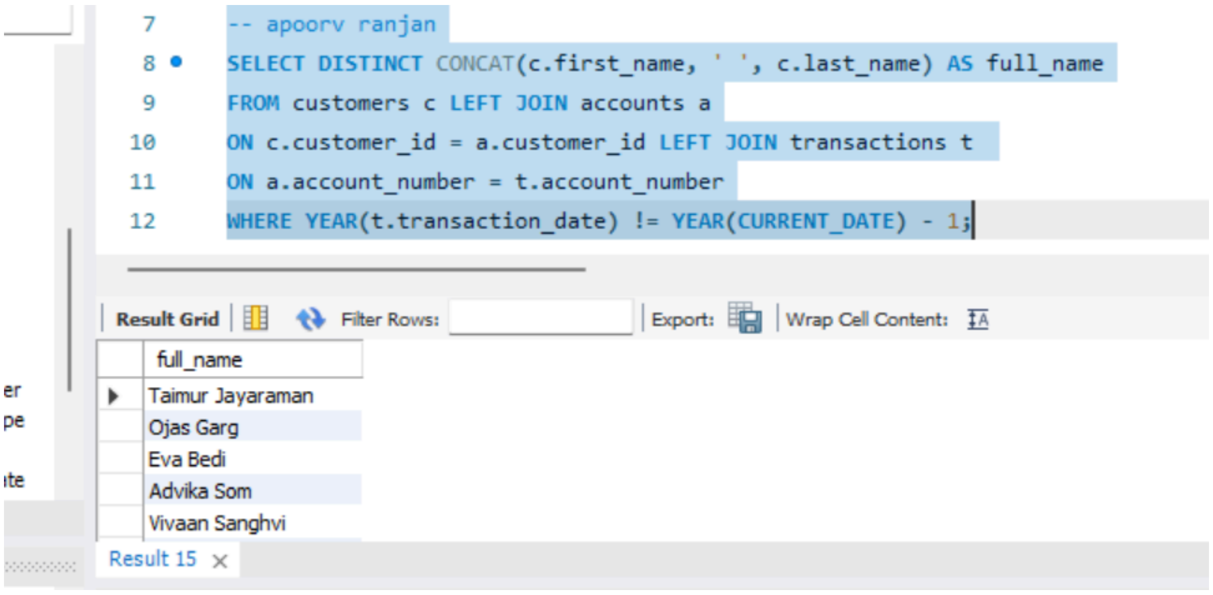
Banking Data Analysis in SQL

Q1: Write a query to list all customers who haven't made any transactions in the last year. How can we make them active again? Provide appropriate region.

Soln-

```
–apoorv ranjan
SELECT DISTINCT CONCAT(c.first_name, ' ', c.last_name) AS full_name
FROM customers c
LEFT JOIN accounts a
ON c.customer_id = a.customer_id
LEFT JOIN transactions t
ON a.account_number = t.account_number
WHERE YEAR(t.transaction_date) != YEAR(CURRENT_DATE) - 1;
```

Reason- Customers may have shifted to competitors due to better incentives, or they lack awareness of existing benefits. Poor engagement or irrelevant services could also contribute to inactivity.

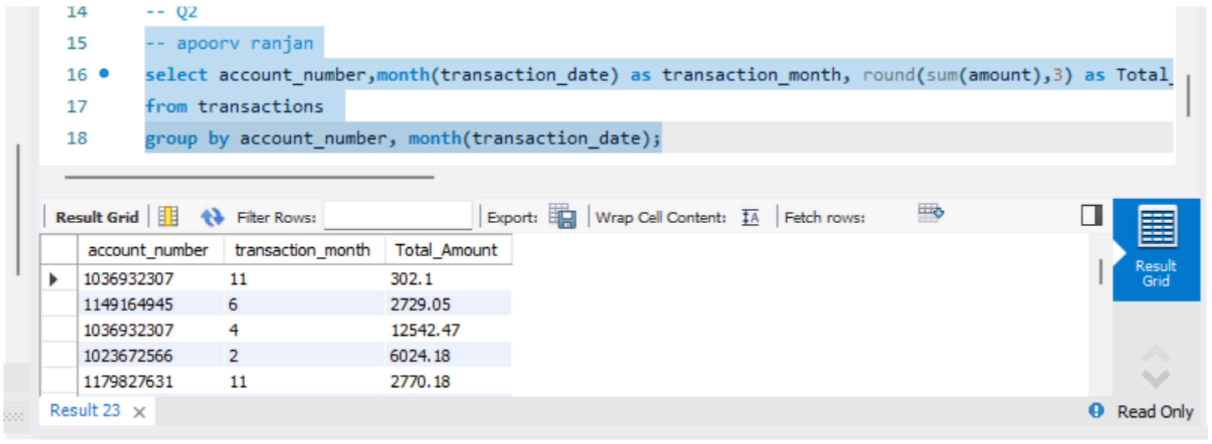


```
-- apoorv ranjan
SELECT DISTINCT CONCAT(c.first_name, ' ', c.last_name) AS full_name
FROM customers c LEFT JOIN accounts a
ON c.customer_id = a.customer_id LEFT JOIN transactions t
ON a.account_number = t.account_number
WHERE YEAR(t.transaction_date) != YEAR(CURRENT_DATE) - 1;
```

full_name
Taimur Jayaraman
Ojas Garg
Eva Bedi
Advika Som
Vivaan Sanghvi

Q2. Summarize the total transaction amount per account per month.

```
-- apoorv ranjan
select account_number,month(transaction_date) as transaction_month, round(sum(amount),3)
as Total_Amount
from transactions
group by account_number, month(transaction_date);
```



```
-- Q2
-- apoorv ranjan
select account_number,month(transaction_date) as transaction_month, round(sum(amount),3) as Total_
from transactions
group by account_number, month(transaction_date);
```

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

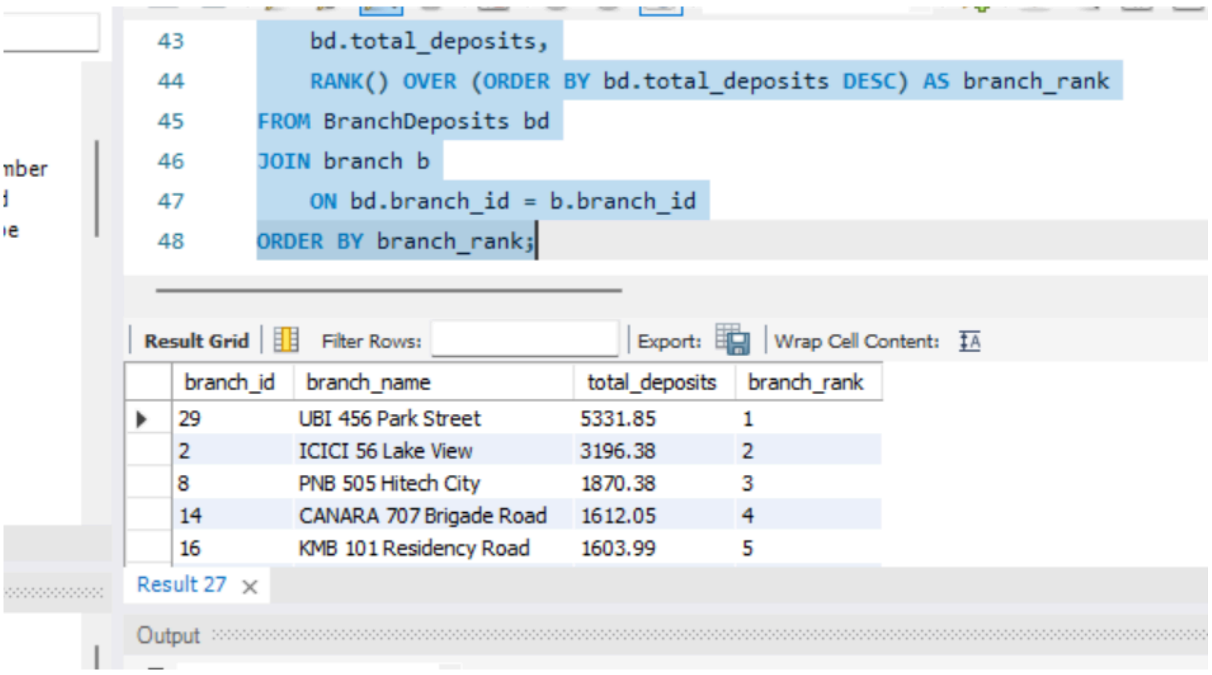
account_number	transaction_month	Total_Amount
1036932307	11	302.1
1149164945	6	2729.05
1036932307	4	12542.47
1023672566	2	6024.18
1179827631	11	2770.18

Result 23 × | Read Only

Q3. Rank branches based on the total amount of deposits made in the last quarter.

Solution-

```
-- apoorv ranjan
WITH LastQuarterTransactions AS (
    SELECT
        t.account_number,
        t.amount,
        t.transaction_date
    FROM transactions t
    WHERE t.transaction_type = 'deposit' -- Only consider deposits
      AND QUARTER(t.transaction_date) = QUARTER(CURRENT_DATE - INTERVAL 3 MONTH) --
Last quarter
      AND YEAR(t.transaction_date) = YEAR(CURRENT_DATE - INTERVAL 3 MONTH) -- Ensure it
matches the year of the last quarter
),
BranchDeposits AS (
    SELECT
        a.branch_id,
        SUM(lqt.amount) AS total_deposits
    FROM LastQuarterTransactions lqt
    JOIN accounts a
        ON lqt.account_number = a.account_number
    GROUP BY a.branch_id
)
SELECT
    b.branch_id,
    b.branch_name,
    bd.total_deposits,
    RANK() OVER (ORDER BY bd.total_deposits DESC) AS branch_rank
FROM BranchDeposits bd
JOIN branch b
    ON bd.branch_id = b.branch_id
ORDER BY branch_rank;
```

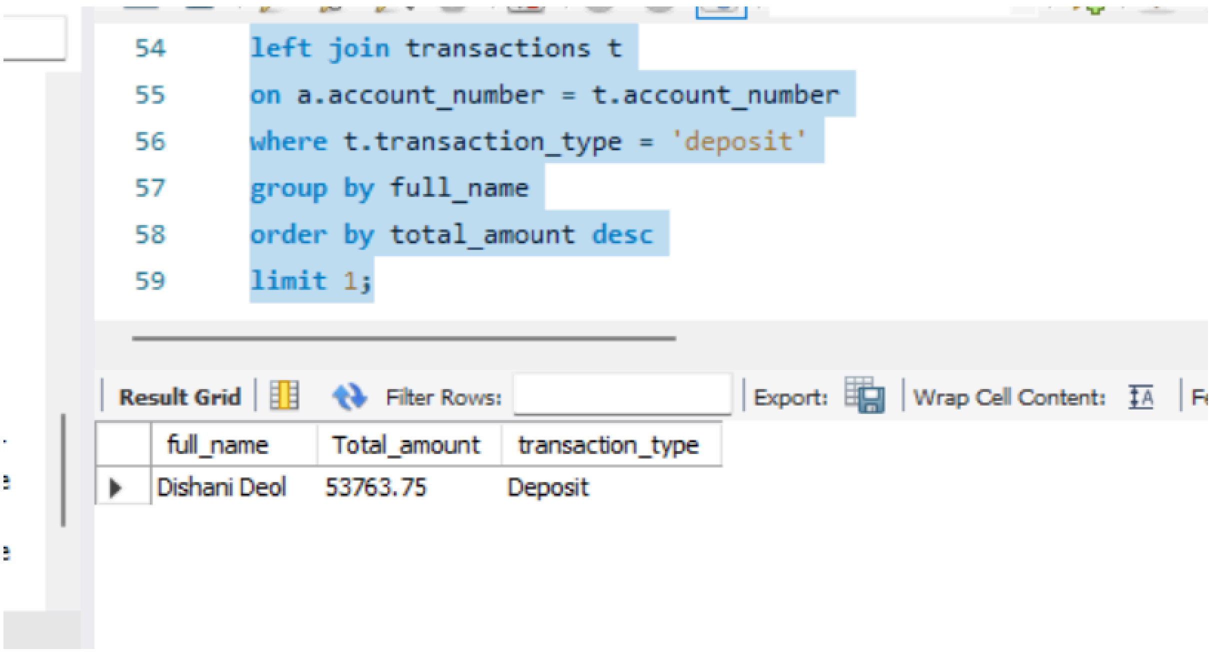


```
        bd.total_deposits,
        RANK() OVER (ORDER BY bd.total_deposits DESC) AS branch_rank
FROM BranchDeposits bd
JOIN branch b
    ON bd.branch_id = b.branch_id
ORDER BY branch_rank;
```

branch_id	branch_name	total_deposits	branch_rank
29	UBI 456 Park Street	5331.85	1
2	ICICI 56 Lake View	3196.38	2
8	PNB 505 Hitech City	1870.38	3
14	CANARA 707 Brigade Road	1612.05	4
16	KMB 101 Residency Road	1603.99	5

Q4. Find the name of the customer who has deposited the highest amount.

-- apoorv ranjan
select concat(c.first_name , ' ', c.last_name) as full_name, round(sum(t.amount),3) as Total_amount, t.transaction_type from customers c
left join accounts a
on c.customer_id = a.customer_id
left join transactions t
on a.account_number = t.account_number
where t.transaction_type = 'deposit'
group by full_name
order by total_amount desc
limit 1;



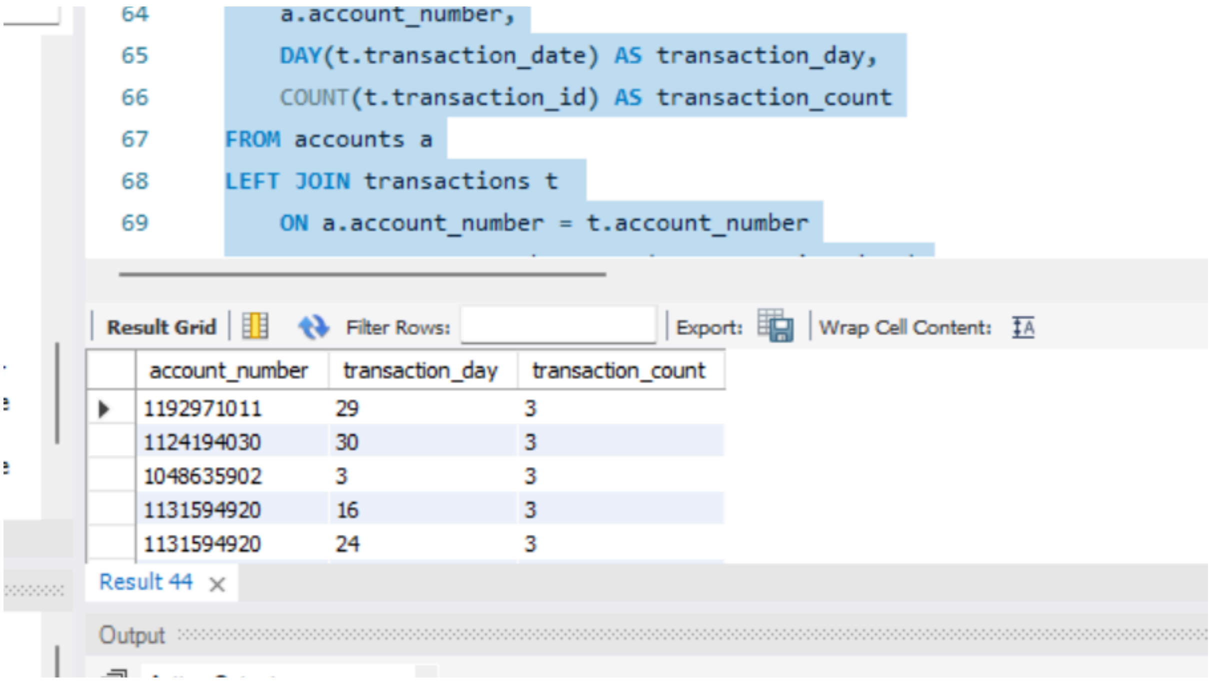
```
left join transactions t
on a.account_number = t.account_number
where t.transaction_type = 'deposit'
group by full_name
order by total_amount desc
limit 1;
```

	full_name	Total_amount	transaction_type
▶	Dishani Deol	53763.75	Deposit

Q5. Identify any accounts that have made more than two transactions in a single day, which could indicate fraudulent activity. How can you verify any fraudulent transactions?

```
-- apoorv ranjan
SELECT
    a.account_number,
    DAY(t.transaction_date) AS transaction_day,
    COUNT(t.transaction_id) AS transaction_count
FROM accounts a
LEFT JOIN transactions t
    ON a.account_number = t.account_number
GROUP BY a.account_number, DAY(t.transaction_date)
HAVING COUNT(t.transaction_id) > 2;
```

Verification: To verify potential fraud, cross-check transaction patterns, validate unusual amounts or frequencies, and confirm with the customer for authentication.



```
        a.account_number,
        DAY(t.transaction_date) AS transaction_day,
        COUNT(t.transaction_id) AS transaction_count
    FROM accounts a
    LEFT JOIN transactions t
        ON a.account_number = t.account_number
```

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

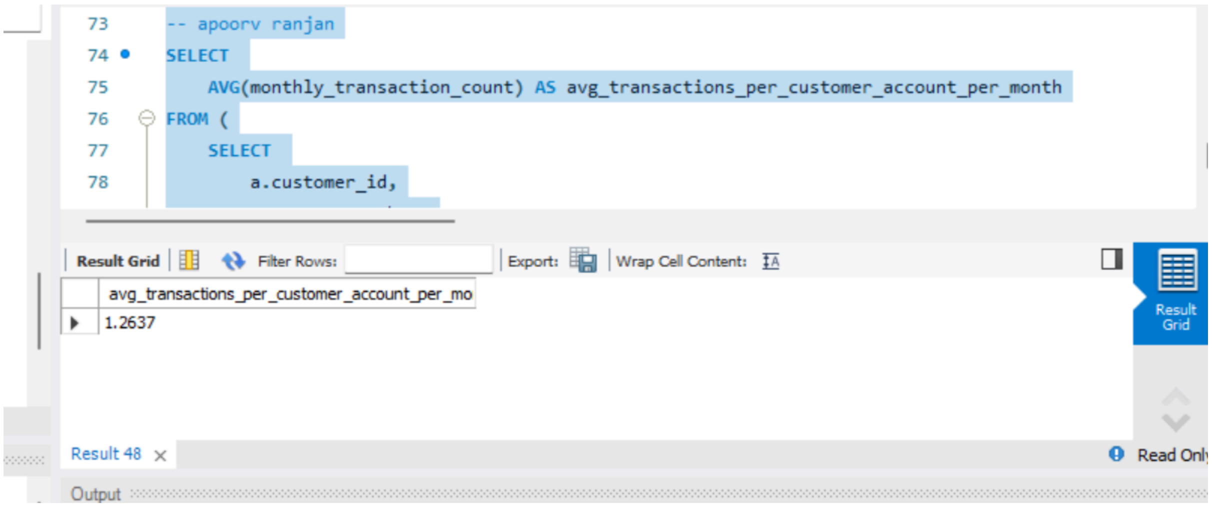
account_number	transaction_day	transaction_count
1192971011	29	3
1124194030	30	3
1048635902	3	3
1131594920	16	3
1131594920	24	3

Result 44

Output

Q6. Calculate the average number of transactions per customer per account per month over the last year.

```
-- apoorv ranjan
SELECT
   AVG(monthly_transaction_count) AS avg_transactions_per_customer_account_per_month
FROM (
   SELECT
      a.customer_id,
      a.account_number,
      MONTH(t.transaction_date) AS transaction_month,
      COUNT(t.transaction_id) AS monthly_transaction_count
   FROM accounts a
   LEFT JOIN transactions t
      ON a.account_number = t.account_number
   WHERE t.transaction_date >= DATE_SUB(CURDATE(), INTERVAL 12 MONTH) -- Only consider
the last 12 months
   GROUP BY a.customer_id, a.account_number, MONTH(t.transaction_date)
) AS monthly_summary;
```



```
-- apoorv ranjan
SELECT
    AVG(monthly_transaction_count) AS avg_transactions_per_customer_account_per_month
FROM (
    SELECT
        a.customer_id,
```

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

```
avg_transactions_per_customer_account_per_mo
1.2637
```

Result 48 × Read Only

Output

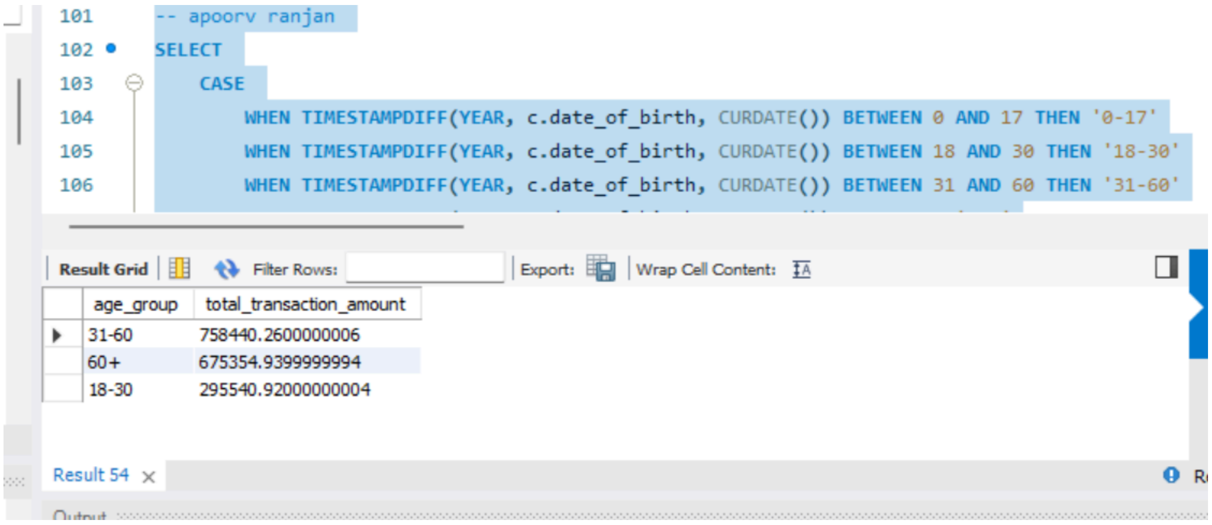
Q7. Write a query to find the daily transaction volume (total amount of all transactions) for the past month.

```
SELECT
   DATE(transaction_date) AS transaction_day,
   SUM(amount) AS total_transaction_volume
FROM transactions
WHERE
   YEAR(transaction_date) = YEAR(DATE_SUB(CURDATE(), INTERVAL 1 MONTH))
   AND MONTH(transaction_date) = MONTH(DATE_SUB(CURDATE(), INTERVAL 1 MONTH))
GROUP BY DATE(transaction_date)
ORDER BY transaction_day;
```

Q8. Calculate the total transaction amount performed by each age group in the past year. (Age groups: 0-17, 18-30, 31-60, 60+)

```
-- apoorv ranjan
SELECT
   CASE
      WHEN TIMESTAMPDIFF(YEAR, c.date_of_birth, CURDATE()) BETWEEN 0 AND 17 THEN
'0-17'
      WHEN TIMESTAMPDIFF(YEAR, c.date_of_birth, CURDATE()) BETWEEN 18 AND 30 THEN
'18-30'
      WHEN TIMESTAMPDIFF(YEAR, c.date_of_birth, CURDATE()) BETWEEN 31 AND 60 THEN
'31-60'
      WHEN TIMESTAMPDIFF(YEAR, c.date_of_birth, CURDATE()) > 60 THEN '60+'
   END AS age_group,
   SUM(t.amount) AS total_transaction_amount
FROM customers c
INNER JOIN accounts a
   ON c.customer_id = a.customer_id
INNER JOIN transactions t
```

```
    ON a.account_number = t.account_number
WHERE t.transaction_date BETWEEN DATE_SUB(CURDATE(), INTERVAL 1 YEAR) AND
CURDATE()
GROUP BY age_group
ORDER BY total_transaction_amount DESC;
```

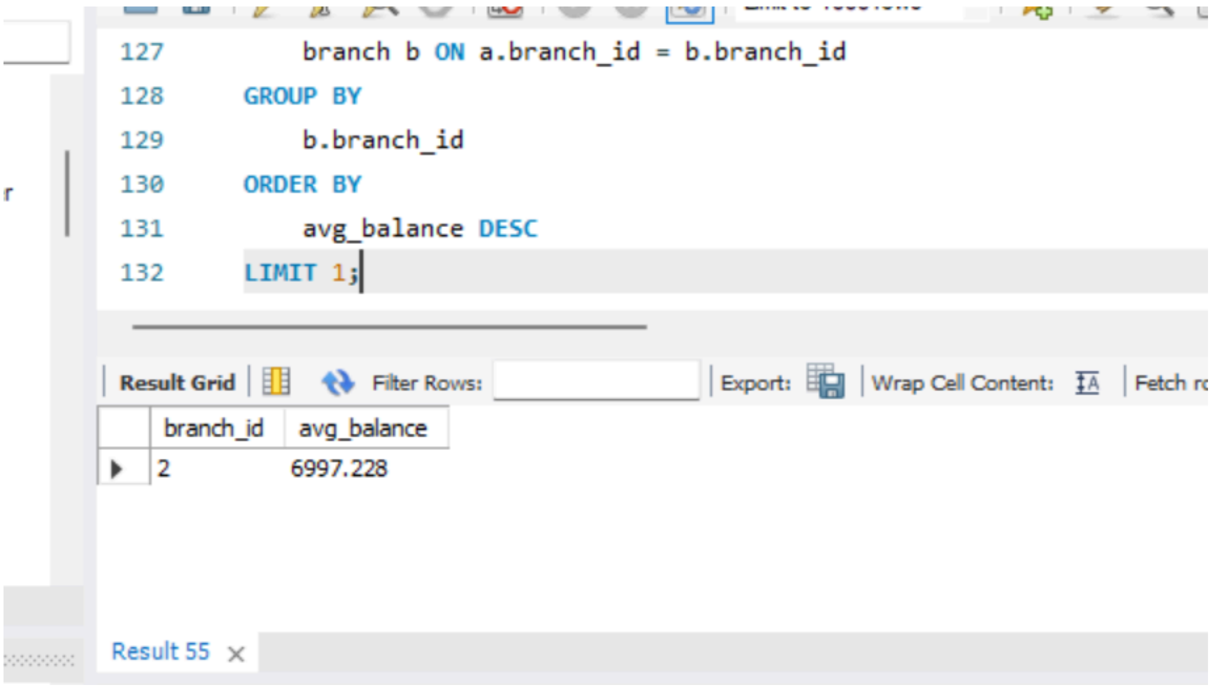


```
-- apoorv ranjan
SELECT
    CASE
        WHEN TIMESTAMPDIFF(YEAR, c.date_of_birth, CURDATE()) BETWEEN 0 AND 17 THEN '0-17'
        WHEN TIMESTAMPDIFF(YEAR, c.date_of_birth, CURDATE()) BETWEEN 18 AND 30 THEN '18-30'
        WHEN TIMESTAMPDIFF(YEAR, c.date_of_birth, CURDATE()) BETWEEN 31 AND 60 THEN '31-60'
```

age_group	total_transaction_amount
31-60	758440.2600000006
60+	675354.9399999994
18-30	295540.92000000004

Q9. Find the branch with the highest average account balance.

```
-- apoorv ranjan
SELECT
    b.branch_id,
    AVG(a.balance) AS avg_balance
FROM
    accounts a
INNER JOIN
    branch b ON a.branch_id = b.branch_id
GROUP BY
    b.branch_id
ORDER BY
    avg_balance DESC
LIMIT 1;
```



```
        branch b ON a.branch_id = b.branch_id
    GROUP BY
        b.branch_id
    ORDER BY
        avg_balance DESC
    LIMIT 1;
```

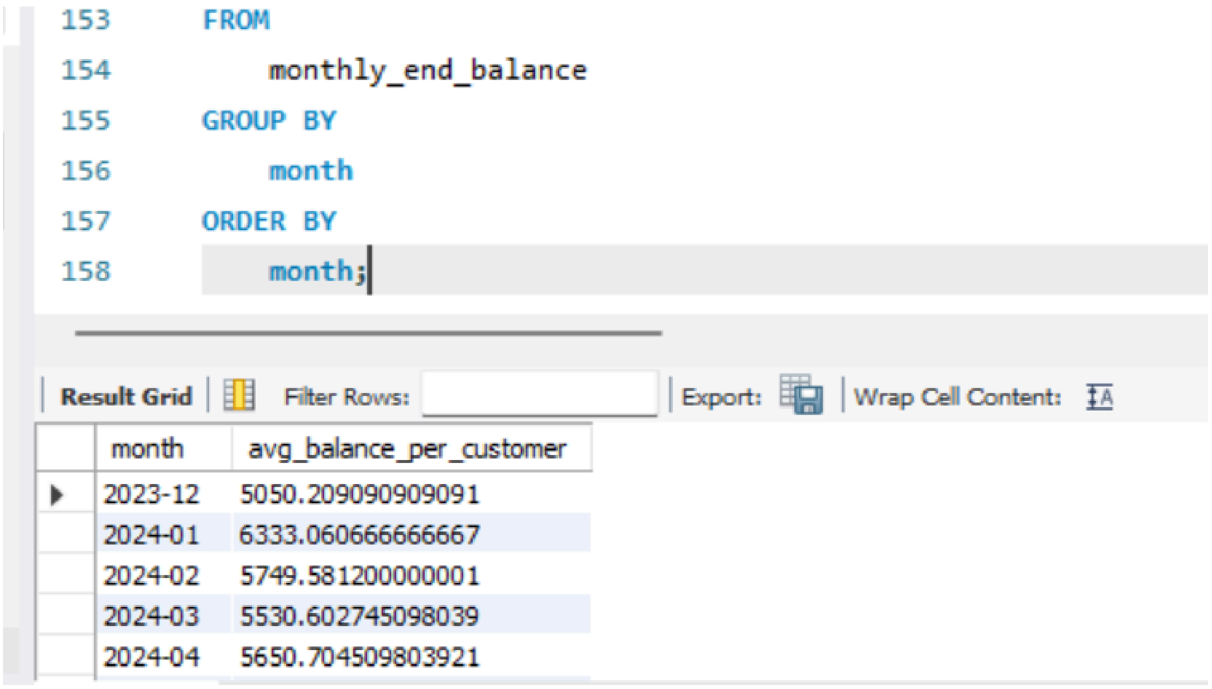
Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch r

```
branch_id  avg_balance
2          6997.228
```

Result 55

Q10. Calculate the average balance per customer at the end of each month in the last year.

```
-- apoorv ranjan
WITH monthly_end_balance AS (
   SELECT
      a.customer_id,
      DATE_FORMAT(t.transaction_date, '%Y-%m') AS month,  -- Extract year-month
      MAX(a.balance) AS end_month_balance
   FROM
      accounts a
   INNER JOIN
      transactions t ON a.account_number = t.account_number
   WHERE
      t.transaction_date BETWEEN DATE_SUB(CURDATE(), INTERVAL 12 MONTH) AND
CURDATE()
   GROUP BY
      a.customer_id, DATE_FORMAT(t.transaction_date, '%Y-%m')
)
SELECT
   month,
   AVG(end_month_balance) AS avg_balance_per_customer
FROM
   monthly_end_balance
GROUP BY
   month
ORDER BY
   month;
```



```
FROM
    monthly_end_balance
GROUP BY
    month
ORDER BY
    month;
```

month	avg_balance_per_customer
2023-12	5050.209090909091
2024-01	6333.060666666667
2024-02	5749.581200000001
2024-03	5530.602745098039
2024-04	5650.704509803921