Addressing Feedback & DeliveringResults to Re-earn Your Consideration

GOAL: TO SHOWCASE IMPROVED SQL PROFICIENCY AND BUSINESS INSIGHT FOR A SECOND CHANCE

Rui Pan - Joining Fetch has been one of my dreams since coming to the U.S.

AGENDA

- My strengths!
- Feedback from Initial Interview
- Revised SQL Code with CTEs
- Data Visualization
- Data Visualization- Business Insights

MY STRENGTHS!

- Exceptional Communication Skills

Explained ideas in a way that's easy for everyone to understand and gave examples to support my answers.

- Strong analytical Abilities

Asked good questions to fully understand the task, making sure I understood before starting the work.

- Growth Mindset

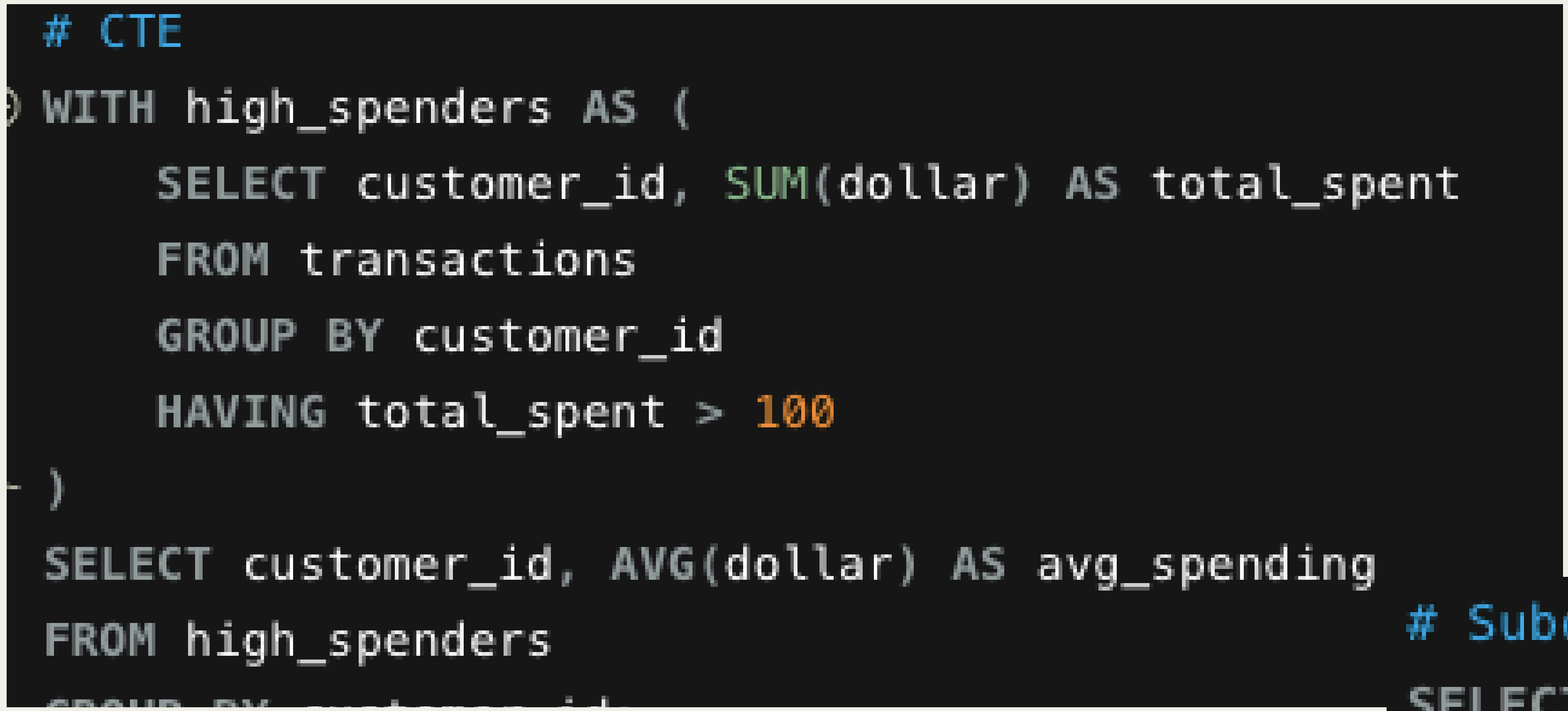
Aware of areas to work on, like CTEs and filtering, actively practicing to make my SQL skills stronger and more organized.

FEEDBACK FROM INTERVIEW

- CTE V.S. Subquery
- Having V.S. Where

CTE V.S. SUBQUERY

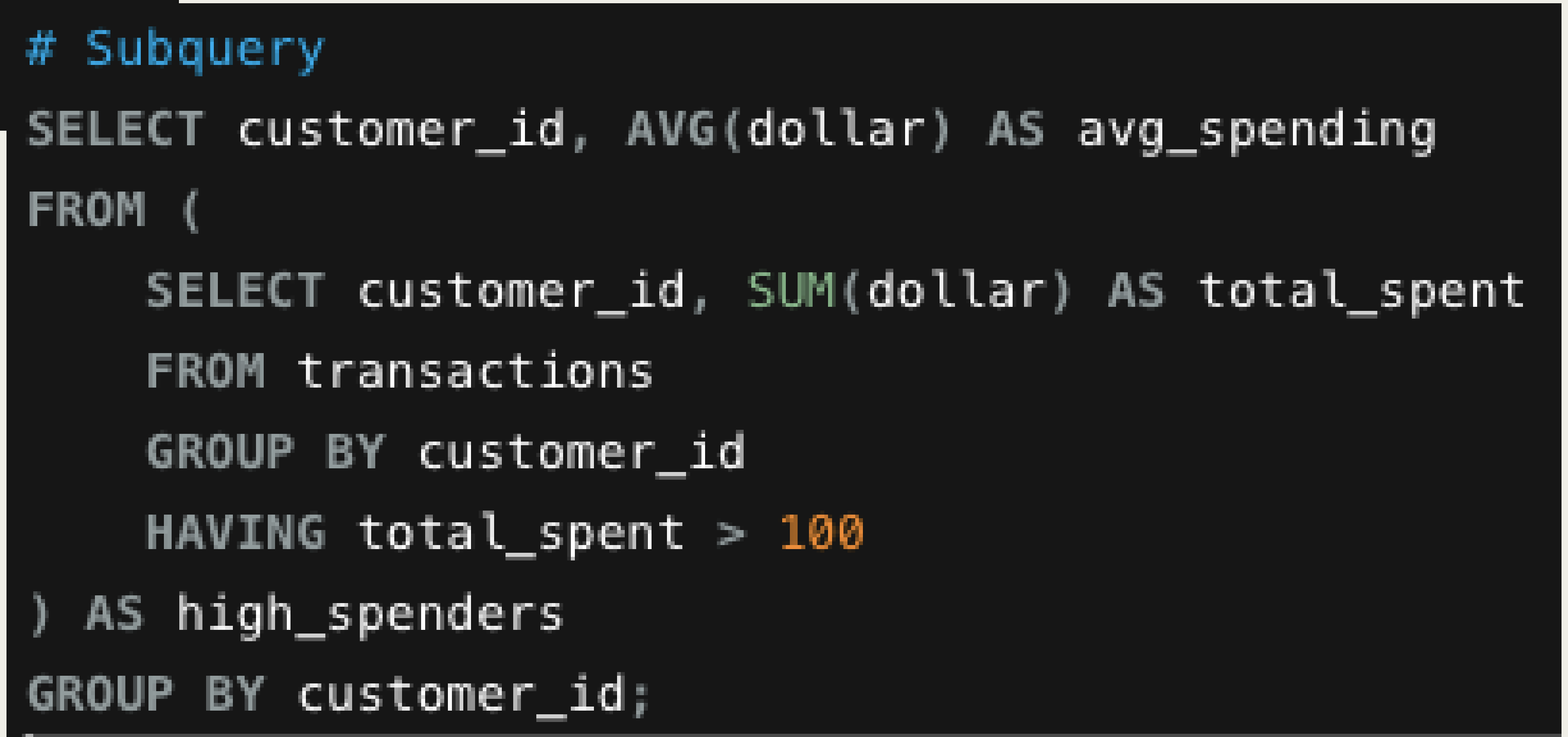
Goal: Find customers who scanned receipts worth over $100 and then calculate their average spending.



```
# CTE
WITH high_spenders AS (
    SELECT customer_id, SUM(dollar) AS total_spent
    FROM transactions
    GROUP BY customer_id
    HAVING total_spent > 100
)
SELECT customer_id, AVG(dollar) AS avg_spending
FROM high_spenders
```

CTE:

- Step 1: CTE (high_spenders) filters customers with spending > $100.
- Step 2: Main query calculates average spending for these customers.
- Each part is separate and easy to follow.



```
# Subquery
SELECT customer_id, AVG(dollar) AS avg_spending
FROM (
    SELECT customer_id, SUM(dollar) AS total_spent
    FROM transactions
    GROUP BY customer_id
    HAVING total_spent > 100
) AS high_spenders
GROUP BY customer_id;
```

Subquery:

- Both steps are combined in one long query.
- The code is messy and more difficult to modify.

CTE V.S. SUBQUERY

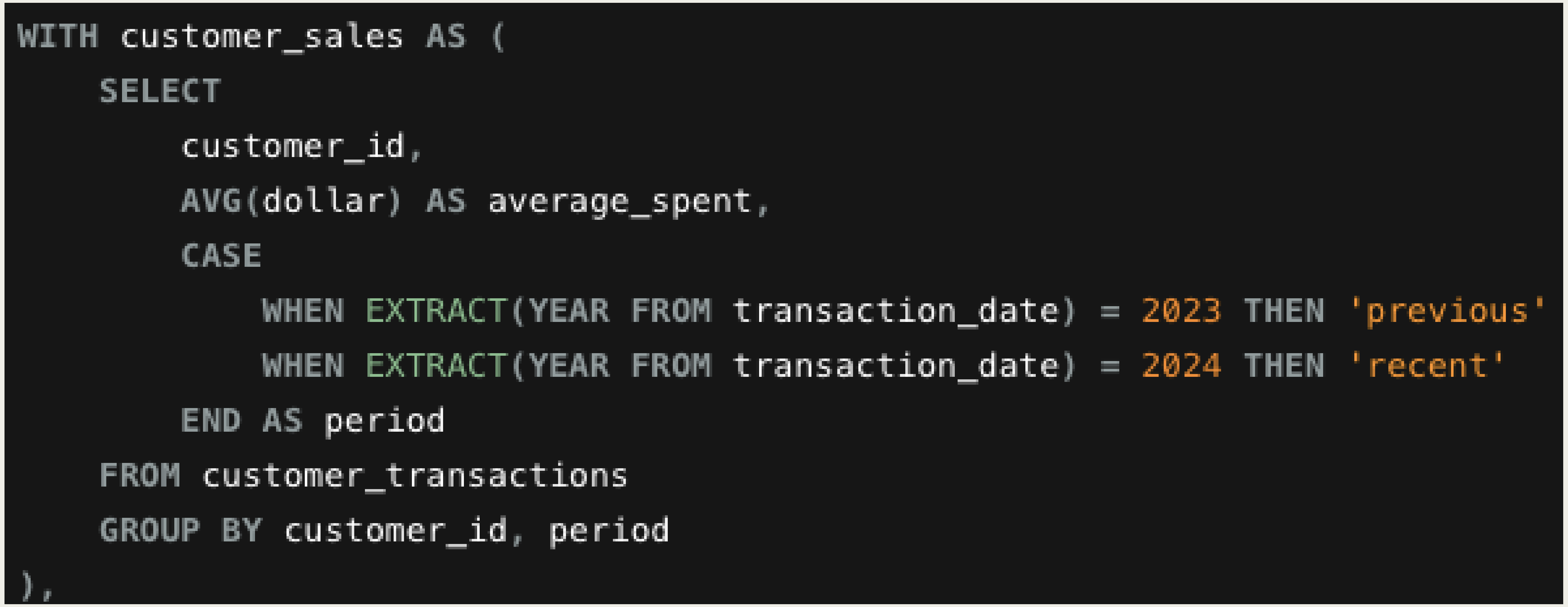
Criteria	CTE	Subquery
Readability	Clear, Step-by-Step	Nested, Confusing
Maintainability	Easy to Modify	Hard to Update
Logic Structure	Logical Flow	Complex Layers
Performance	Efficient	Can be Repetitive
Use Case Example	Separate Steps	Combined in One Query

WHERE V.S. HAVING

Criteria	**Having**	**Where**
Purpose	Filters groups **after** aggregation	Filters rows **before** aggregation
Use With	Aggregated columns (like SUM)	Non-aggregated columns
Example	HAVING SUM(dollar) > 100	WHERE dollar > 100

REVISED SQL CODE WITH CTES (STEP1)

Columns:
customer_id, transaction_date, dollar, product_category, location, receipt_status

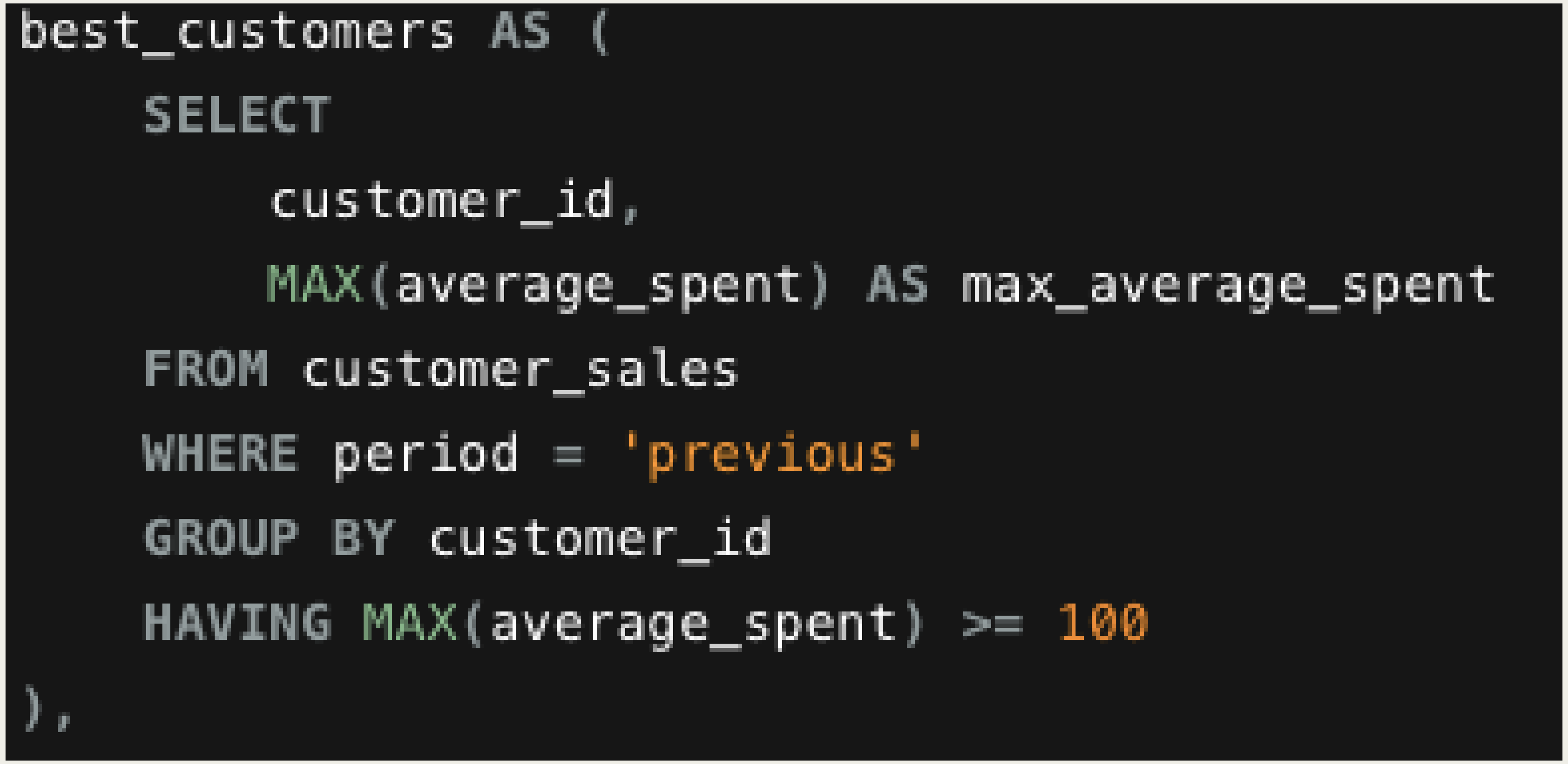


```
WITH customer_sales AS (
    SELECT
        customer_id,
        AVG(dollar) AS average_spent,
        CASE
            WHEN EXTRACT(YEAR FROM transaction_date) = 2023 THEN 'previous'
            WHEN EXTRACT(YEAR FROM transaction_date) = 2024 THEN 'recent'
        END AS period
    FROM customer_transactions
    GROUP BY customer_id, period
),
```

Calculates the average spending for each customer by year. Labels 2023 as "previous" and 2024 as "recent."

REVISED SQL CODE WITH CTES(STEP2)

Columns:
customer_id, transaction_date, dollar, product_category, location, receipt_status

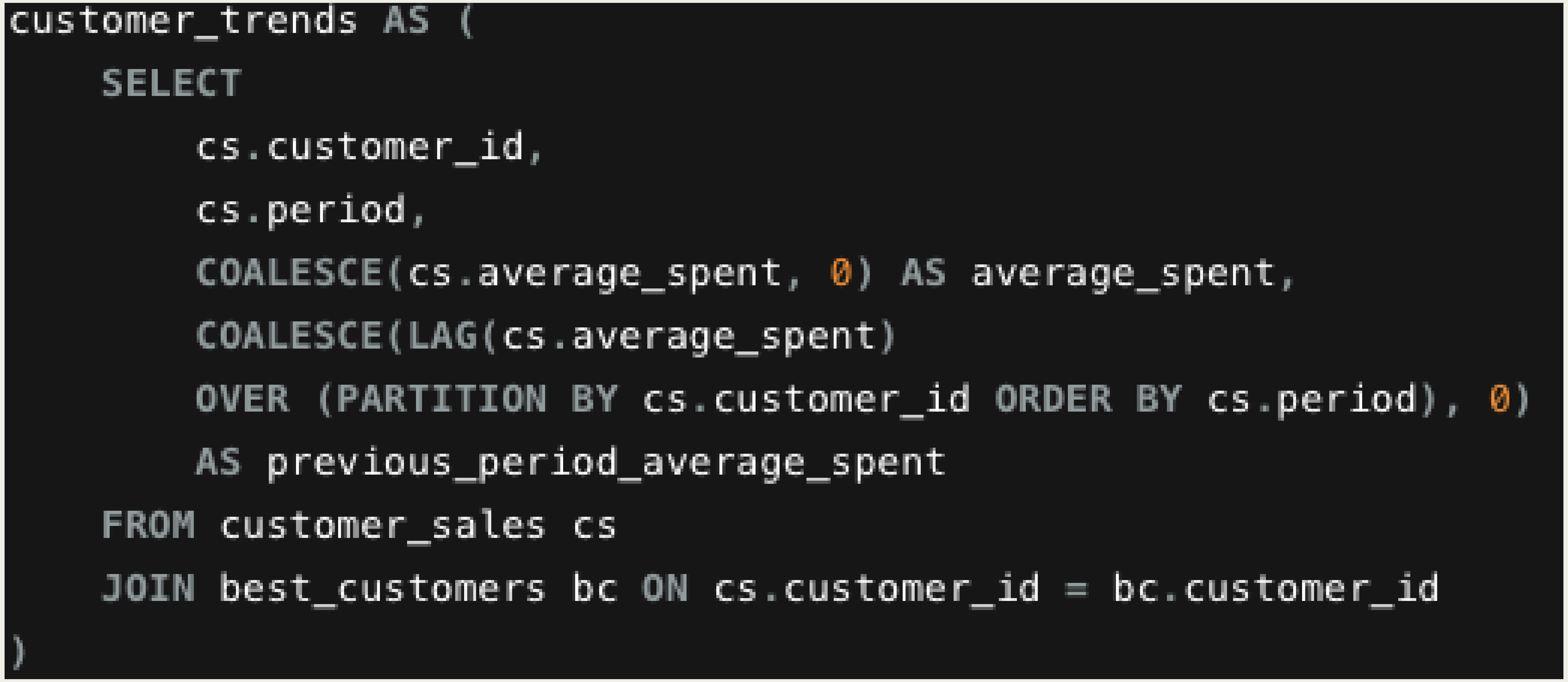


```sql
best_customers AS (
    SELECT
        customer_id,
        MAX(average_spent) AS max_average_spent
    FROM customer_sales
    WHERE period = 'previous'
    GROUP BY customer_id
    HAVING MAX(average_spent) >= 100
),
```

Identifies "best customers" who spent at least $100 in 2023.

REVISED SQL CODE WITH CTES(STEP3)

Columns:
customer_id, transaction_date, dollar, product_category, location, receipt_status

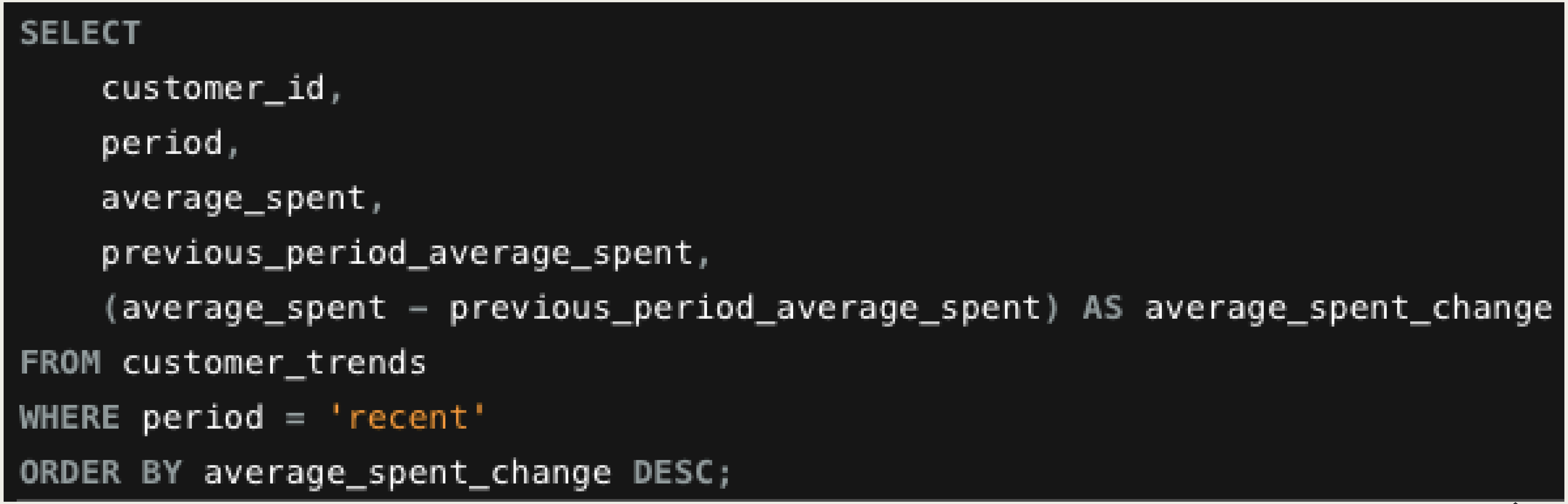


```sql
customer_trends AS (
    SELECT
        cs.customer_id,
        cs.period,
        COALESCE(cs.average_spent, 0) AS average_spent,
        COALESCE(LAG(cs.average_spent)
        OVER (PARTITION BY cs.customer_id ORDER BY cs.period), 0)
        AS previous_period_average_spent
    FROM customer_sales cs
    JOIN best_customers bc ON cs.customer_id = bc.customer_id
)
```

Includes all "best customers" and their spending trends across years. If a year is missing, average_spent is set to 0.

REVISED SQL CODE WITH CTES(STEP4)

Columns:
customer_id, transaction_date, dollar, product_category, location, receipt_status

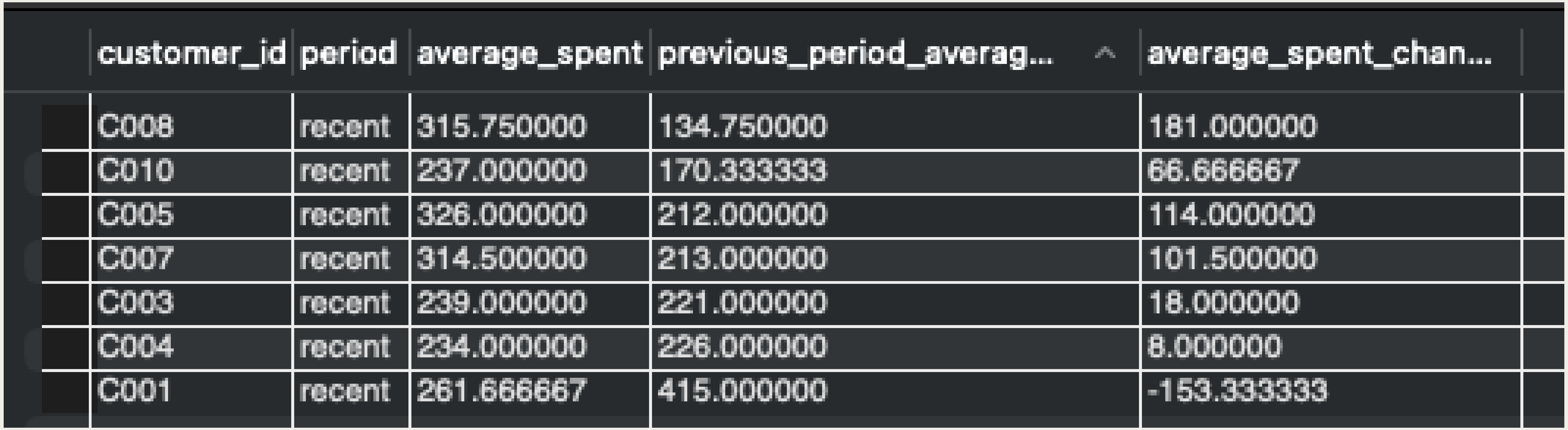


```sql
SELECT
    customer_id,
    period,
    average_spent,
    previous_period_average_spent,
    (average_spent - previous_period_average_spent) AS average_spent_change
FROM customer_trends
WHERE period = 'recent'
ORDER BY average_spent_change DESC;
```

Retrieves the most recent spending data and compares it to the previous year, calculating the spending change.

REVISED SQL CODE WITH CTES(OUTPUT1)

Columns:
customer_id, period, average_spent, previous_period_average_spent, average_spent_change

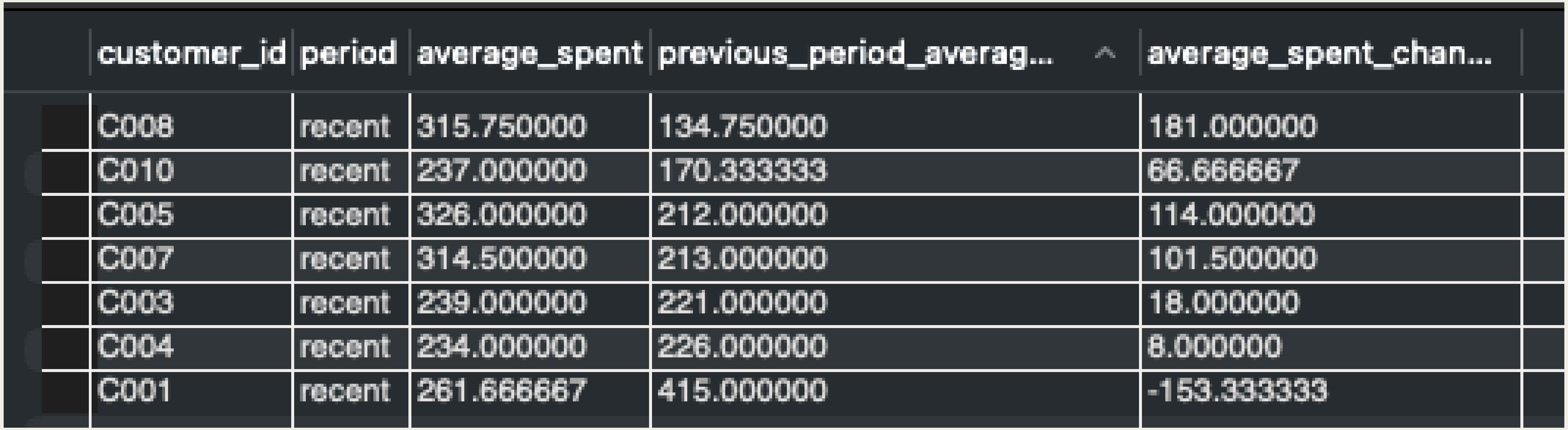


customer_id	period	average_spent	previous_period_averag...	average_spent_chan...
C008	recent	315.750000	134.750000	181.000000
C010	recent	237.000000	170.333333	66.666667
C005	recent	326.000000	212.000000	114.000000
C007	recent	314.500000	213.000000	101.500000
C003	recent	239.000000	221.000000	18.000000
C004	recent	234.000000	226.000000	8.000000
C001	recent	261.666667	415.000000	-153.333333

Reward top spenders, keep steady customers engaged, and win back those who are spending less.

REVISED SQL CODE WITH CTES(OUTPUT2)

Columns:
customer_id, period, average_spent, previous_period_average_spent, average_spent_change



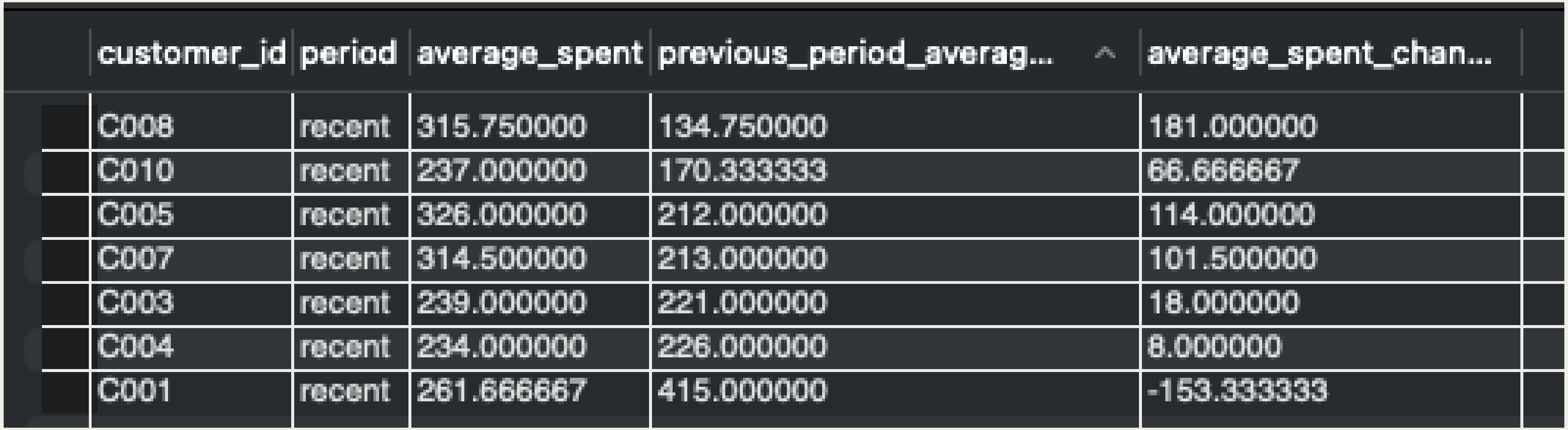
customer_id	period	average_spent	previous_period_averag...	average_spent_chan...
C008	recent	315.750000	134.750000	181.000000
C010	recent	237.000000	170.333333	66.666667
C005	recent	326.000000	212.000000	114.000000
C007	recent	314.500000	213.000000	101.500000
C003	recent	239.000000	221.000000	18.000000
C004	recent	234.000000	226.000000	8.000000
C001	recent	261.666667	415.000000	-153.333333

Increased Spending Customers: C008, C005
These are engaged customers likely responding well to rewards.

Recommendation: Offer loyalty programs or targeted rewards to maintain high spending.

REVISED SQL CODE WITH CTES(OUTPUT3)

Columns:
customer_id, period, average_spent, previous_period_average_spent, average_spent_change



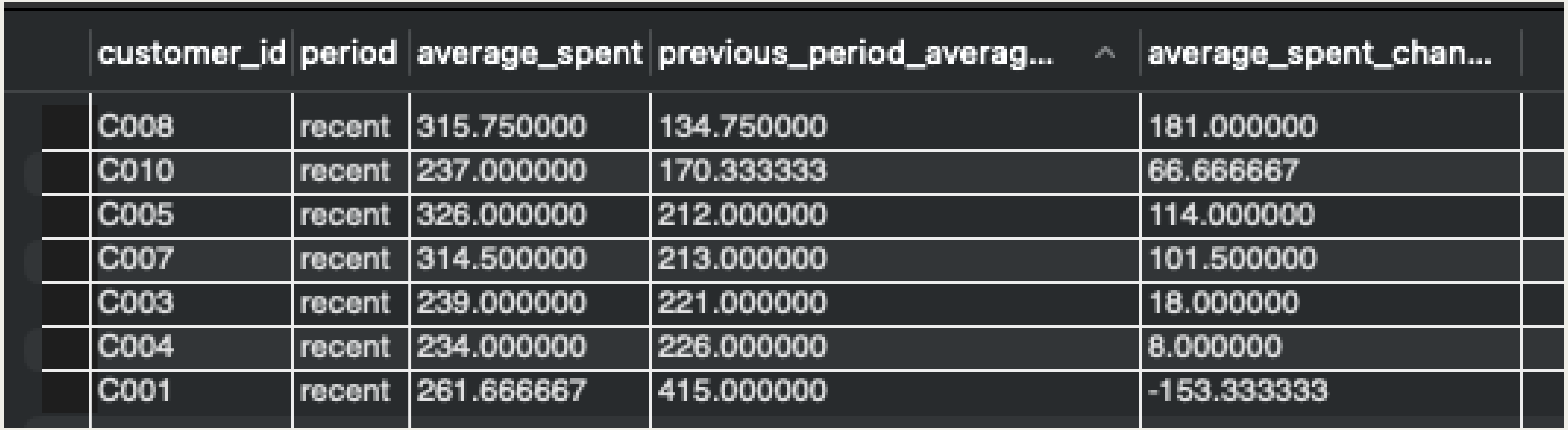
customer_id	period	average_spent	previous_period_averag...	average_spent_chan...
C008	recent	315.750000	134.750000	181.000000
C010	recent	237.000000	170.333333	66.666667
C005	recent	326.000000	212.000000	114.000000
C007	recent	314.500000	213.000000	101.500000
C003	recent	239.000000	221.000000	18.000000
C004	recent	234.000000	226.000000	8.000000
C001	recent	261.666667	415.000000	-153.333333

Stable Spending Customers: C004, C003
These customers are consistent but not highly engaged.

Recommendation: Continue with regular engagement to maintain their spending.

REVISED SQL CODE WITH CTES(OUTPUT4)

Columns:
customer_id, period, average_spent, previous_period_average_spent, average_spent_change

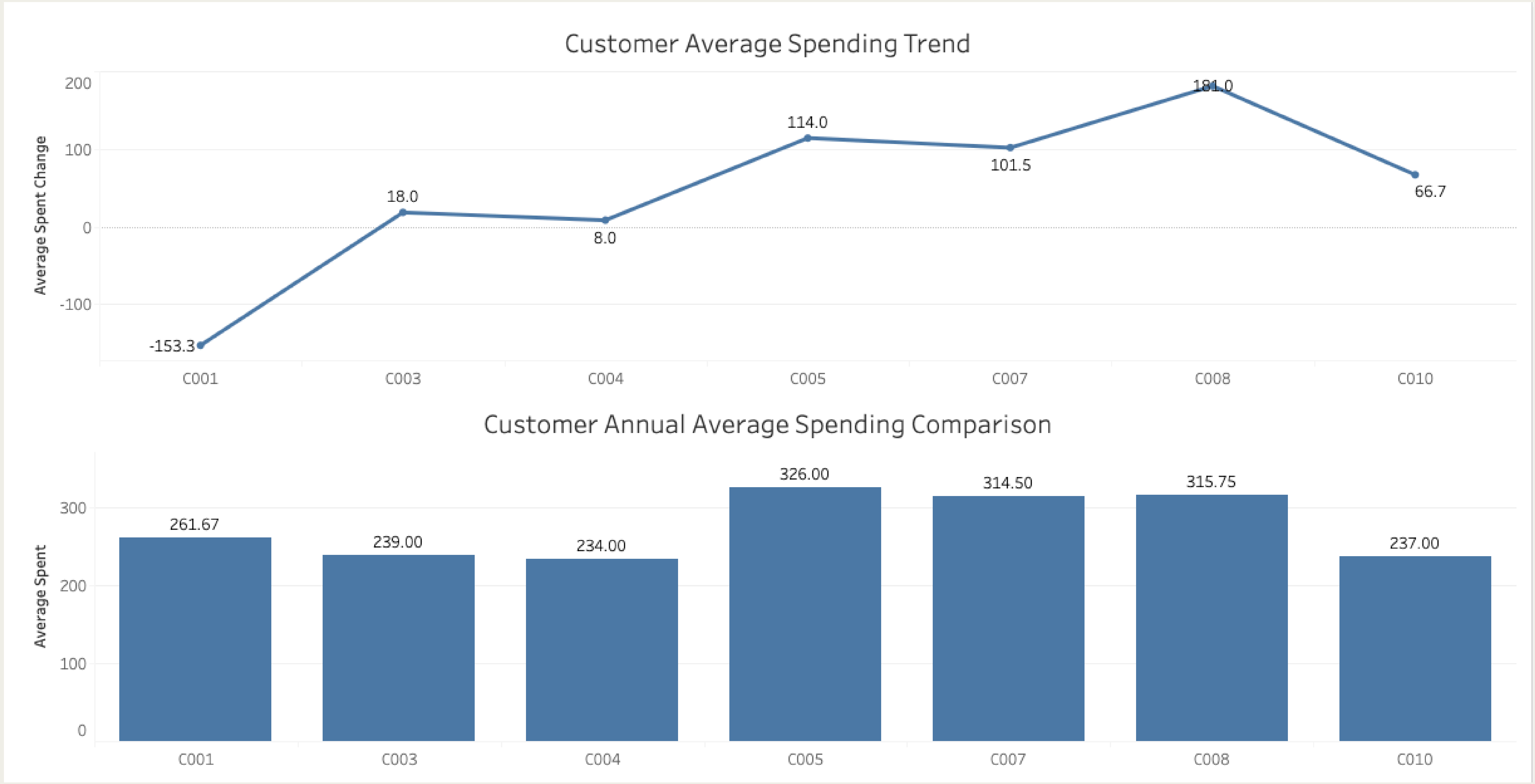


customer_id	period	average_spent	previous_period_averag...	average_spent_chan...
C008	recent	315.750000	134.750000	181.000000
C010	recent	237.000000	170.333333	66.666667
C005	recent	326.000000	212.000000	114.000000
C007	recent	314.500000	213.000000	101.500000
C003	recent	239.000000	221.000000	18.000000
C004	recent	234.000000	226.000000	8.000000
C001	recent	261.666667	415.000000	-153.333333

Decreased Spending Customer: C001
Sharp decline in spending, signaling a potential drop in engagement.

Recommendation: Implement retention strategies, such as personalized promotions, to re-engage this customer.

DATA VISUALIZATION



link: https://public.tableau.com/authoring/SampleAnnualSpendingAnalysis-Fetch/Dashboard1/Annual%20Spending%20Analysis#1

DATA VISUALIZATION- BUSINESS INSIGHTS

- **Focus on Declining Customers (C001):**

This customer shows a significant drop in spending, suggesting the need for re-engagement strategies. Personalized offers or targeted campaigns could help regain interest.

- **Reward High-Growth Customers (C005 and C008):**

These customers show high spending growth, indicating strong engagement. Offering additional rewards or loyalty incentives could sustain their interest.

- **Maintain Engagement for Stable Customers (C003 and C004):**

Customers with small spending changes may be maintained with existing reward structures to keep them engaged.

- **Re-engage Low-Spending Customers (C010):**

With low average spending, this customer could benefit from new offers or incentives to boost engagement.

Thank you!

GOAL: TO SHOWCASE IMPROVED SQL PROFICIENCY AND BUSINESS INSIGHT FOR A SECOND CHANCE

Rui Pan - Joining Fetch has been one of my dreams since coming to the U.S.