Capstone Project

Data Analytics at Walmart

By Farook Mohammad

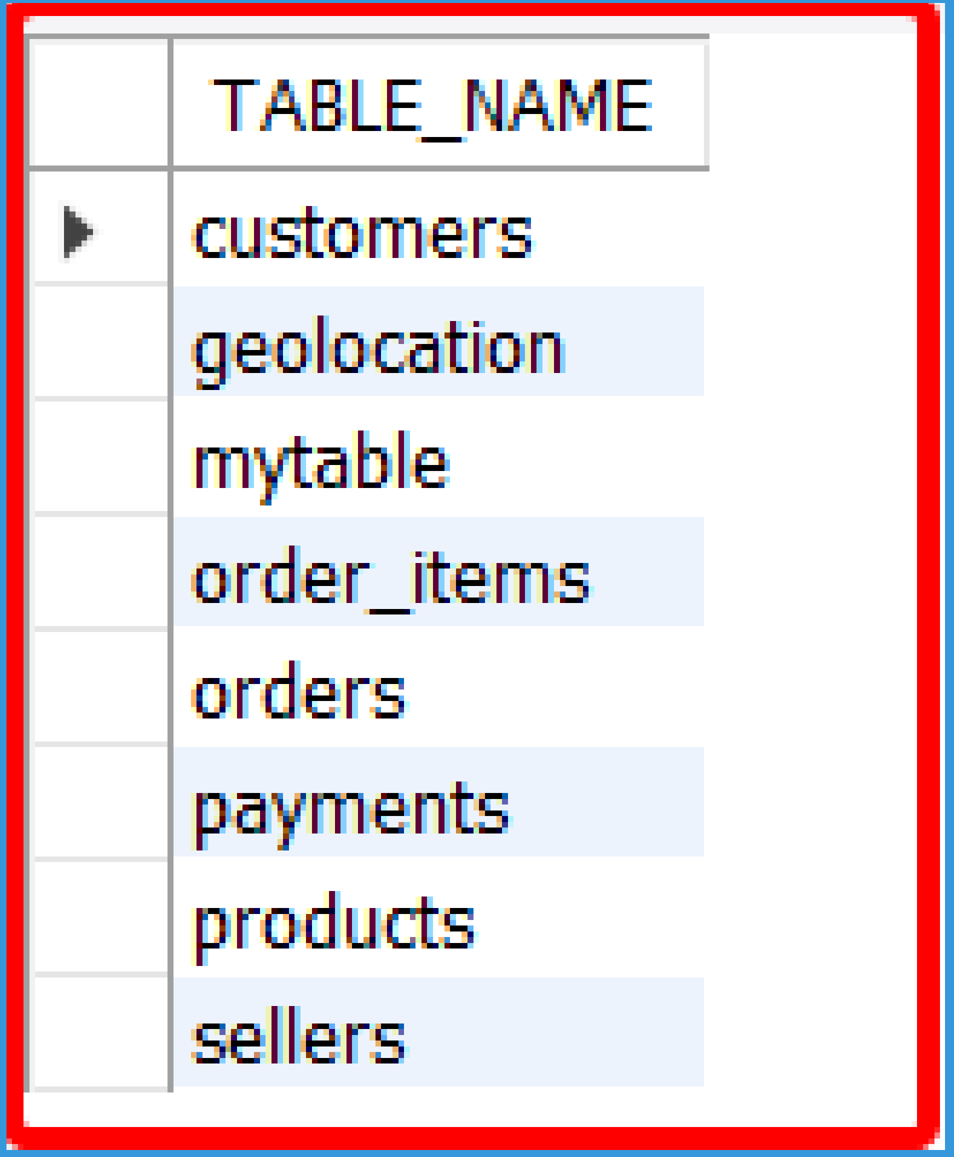
Introduction

Walmart Inc. is a leading American multinational retail corporation that operates a chain of hypermarkets, discount department stores, and grocery stores. It was founded by Sam Walton in 1962 in Rogers, Arkansas, and has since grown into one of the largest companies in the world by revenue.

Project Overview

Transform Walmart's data into insights for strategic decisions

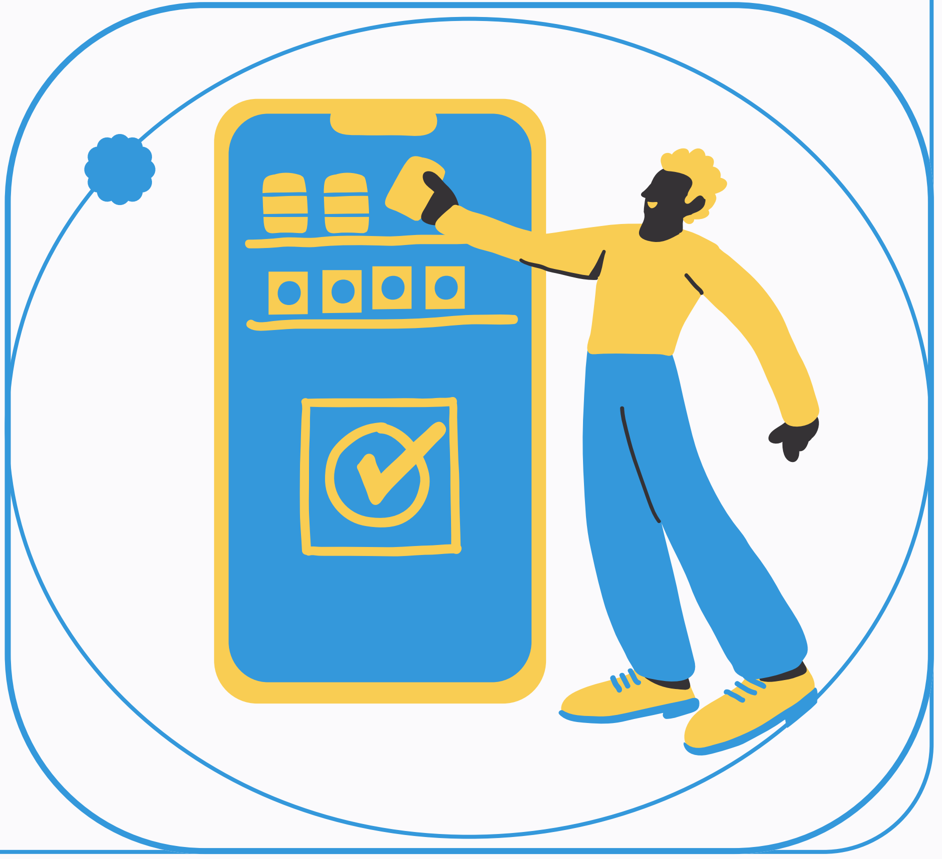
Dataset Overview



	TABLE_NAME
▶	customers
	geolocation
	mytable
	order_items
	orders
	payments
	products
	sellers

🚀 Phase 1: Advanced SQL Analysis

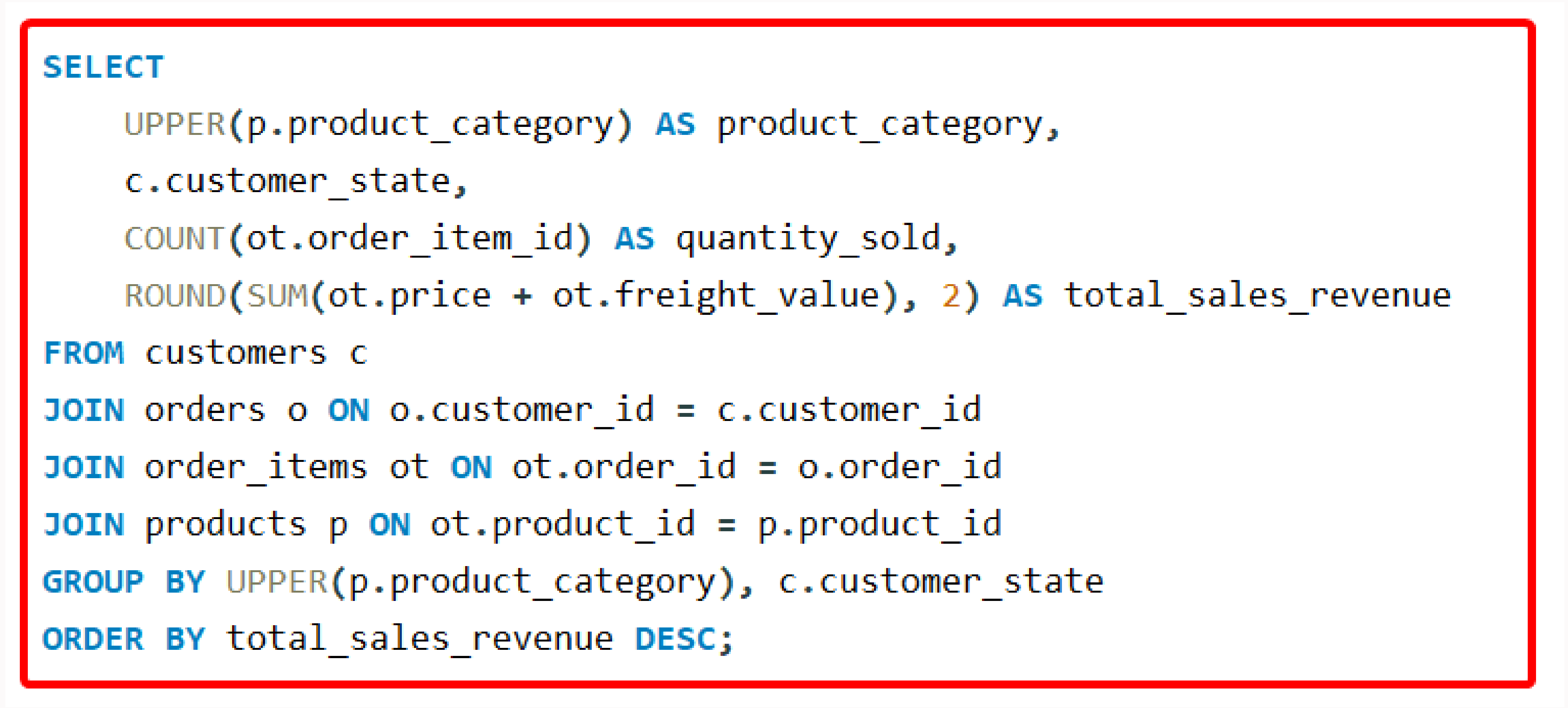
Tool :



Problem Statements

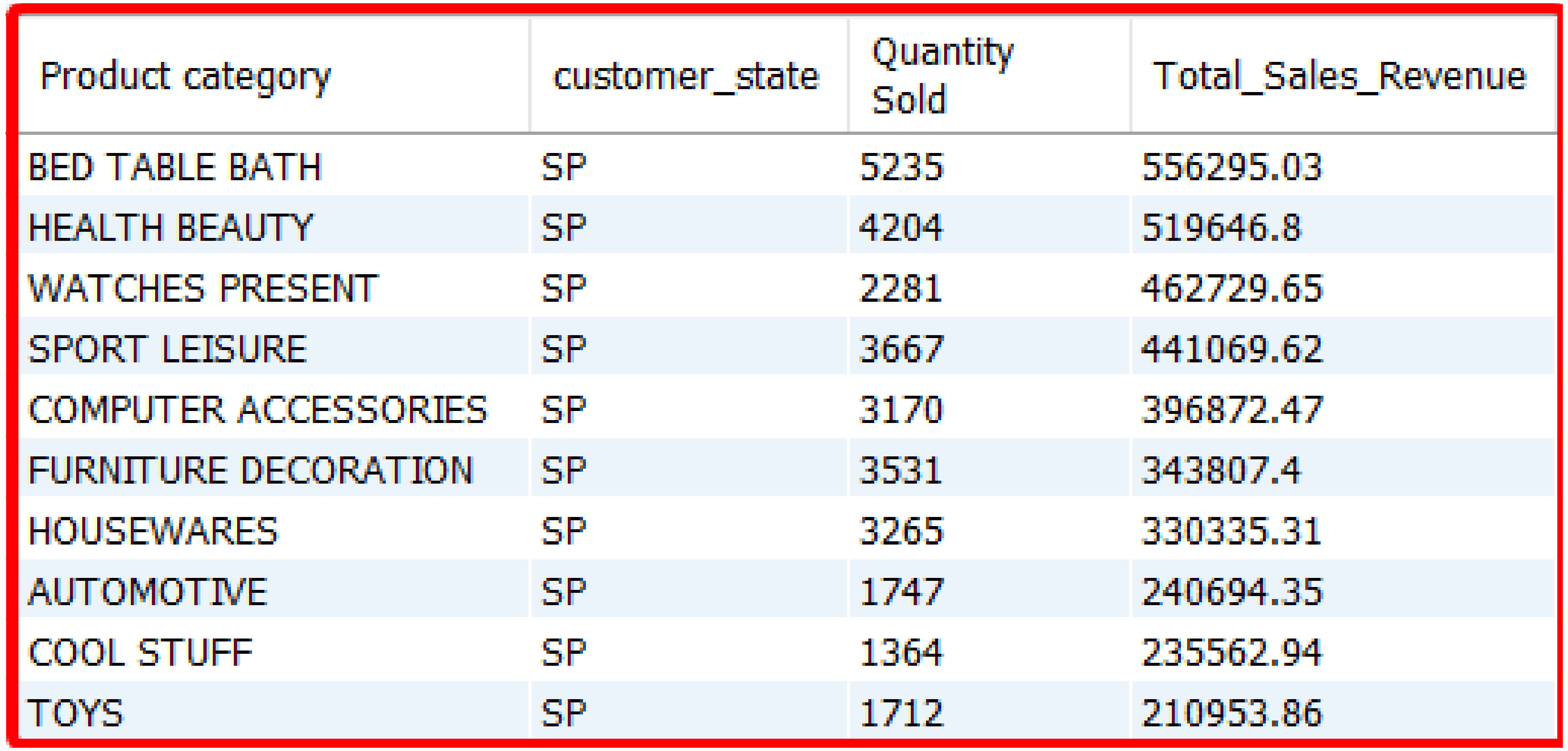
1. Calculate total sales revenue and quantity sold by product category and customer_state.
2. Identify the top 5 products by total sales revenue across all Walmart regions.
3. Find customers with the highest number of orders and total spend, ranking them as Walmart's most valuable customers.
4. Determine customer states with the highest average order value (AOV).
5. Compute average delivery time (in days) by seller state, calculated as the difference between order_purchase_timestamp and order_delivered_customer_date.
6. List the top 5 sellers based on total revenue earned.
7. Analyze the monthly revenue trend over the last 12 months to track Walmart's growth.
8. Calculate the number of new unique customers acquired each month, based on customer_unique_id.
9. Rank customers by lifetime spend within each customer state using SQL window functions.
10. Compute the rolling 3-month average revenue trend, to visualize sales momentum.

1. Calculate total sales revenue and quantity sold by product category and customer_state.



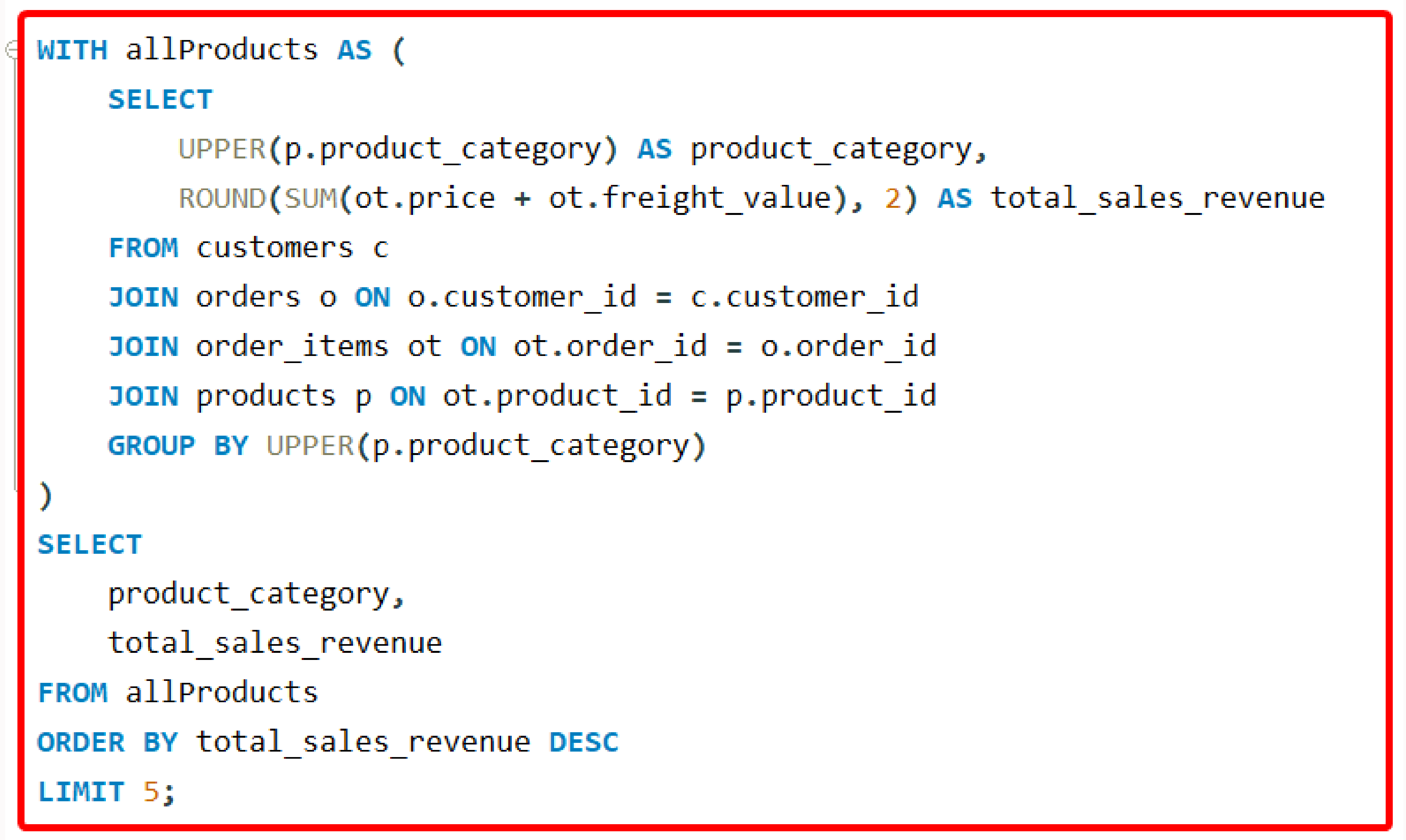
```
SELECT
    UPPER(p.product_category) AS product_category,
    c.customer_state,
    COUNT(ot.order_item_id) AS quantity_sold,
    ROUND(SUM(ot.price + ot.freight_value), 2) AS total_sales_revenue
FROM customers c
JOIN orders o ON o.customer_id = c.customer_id
JOIN order_items ot ON ot.order_id = o.order_id
JOIN products p ON ot.product_id = p.product_id
GROUP BY UPPER(p.product_category), c.customer_state
ORDER BY total_sales_revenue DESC;
```

Output



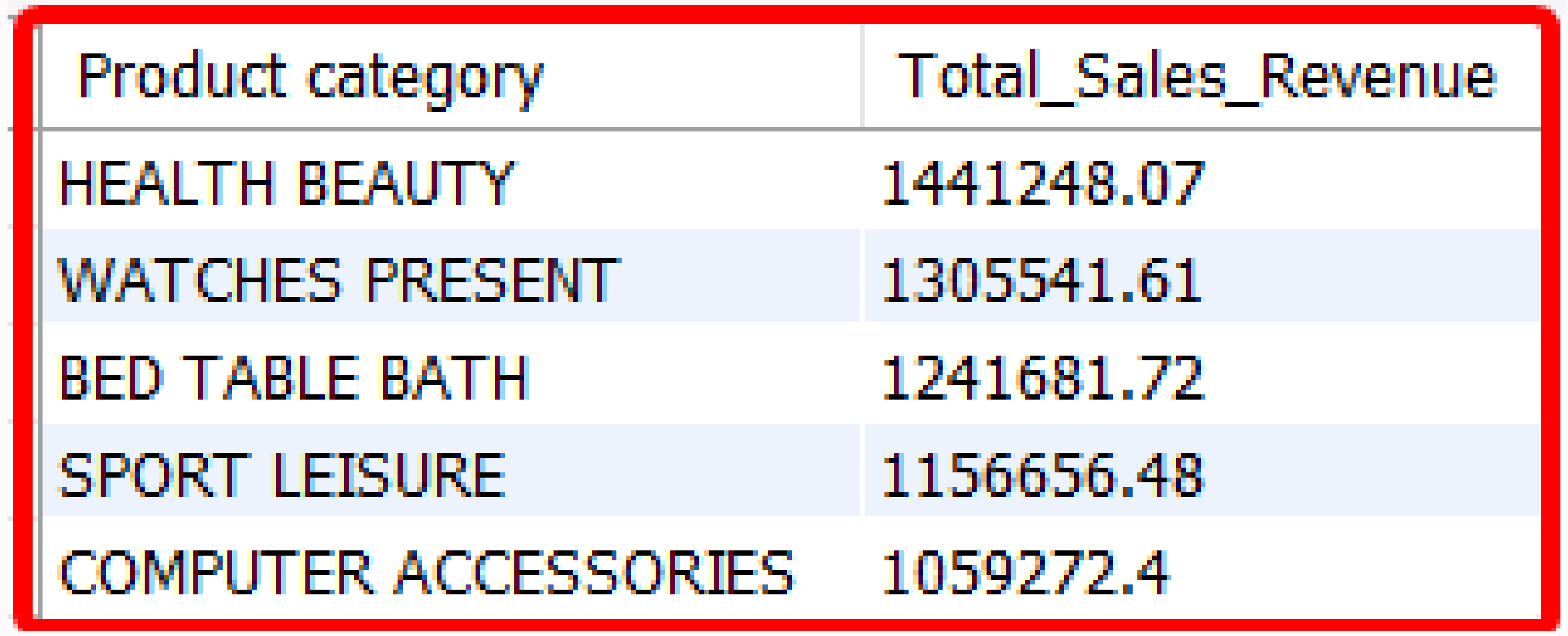
Product category	customer_state	Quantity Sold	Total_Sales_Revenue
BED TABLE BATH	SP	5235	556295.03
HEALTH BEAUTY	SP	4204	519646.8
WATCHES PRESENT	SP	2281	462729.65
SPORT LEISURE	SP	3667	441069.62
COMPUTER ACCESSORIES	SP	3170	396872.47
FURNITURE DECORATION	SP	3531	343807.4
HOUSEWARES	SP	3265	330335.31
AUTOMOTIVE	SP	1747	240694.35
COOL STUFF	SP	1364	235562.94
TOYS	SP	1712	210953.86

2. Identify the top 5 products by total sales revenue across all Walmart regions.



```sql
WITH allProducts AS (
    SELECT
        UPPER(p.product_category) AS product_category,
        ROUND(SUM(ot.price + ot.freight_value), 2) AS total_sales_revenue
    FROM customers c
    JOIN orders o ON o.customer_id = c.customer_id
    JOIN order_items ot ON ot.order_id = o.order_id
    JOIN products p ON ot.product_id = p.product_id
    GROUP BY UPPER(p.product_category)
)
SELECT
    product_category,
    total_sales_revenue
FROM allProducts
ORDER BY total_sales_revenue DESC
LIMIT 5;
```

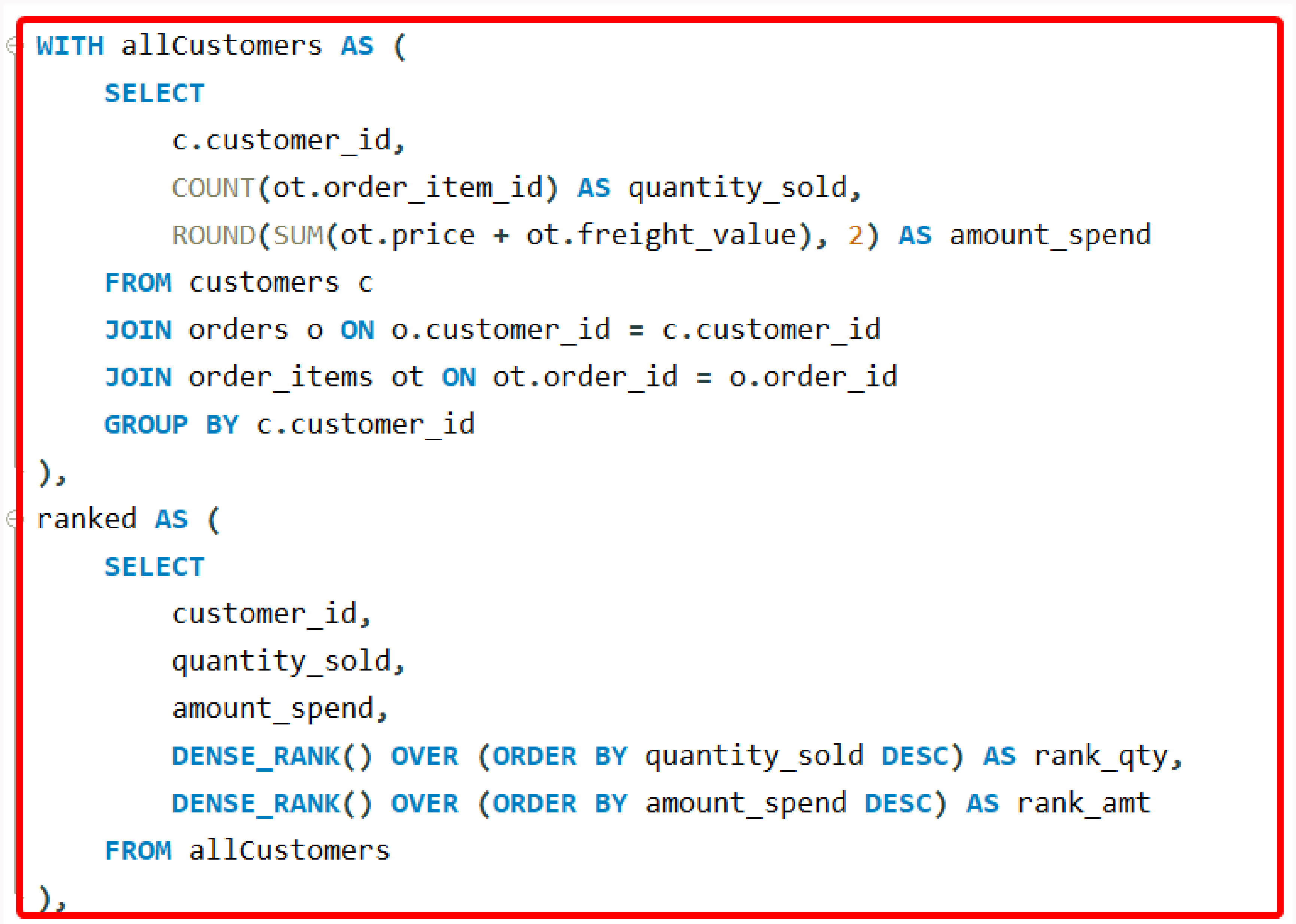
Output



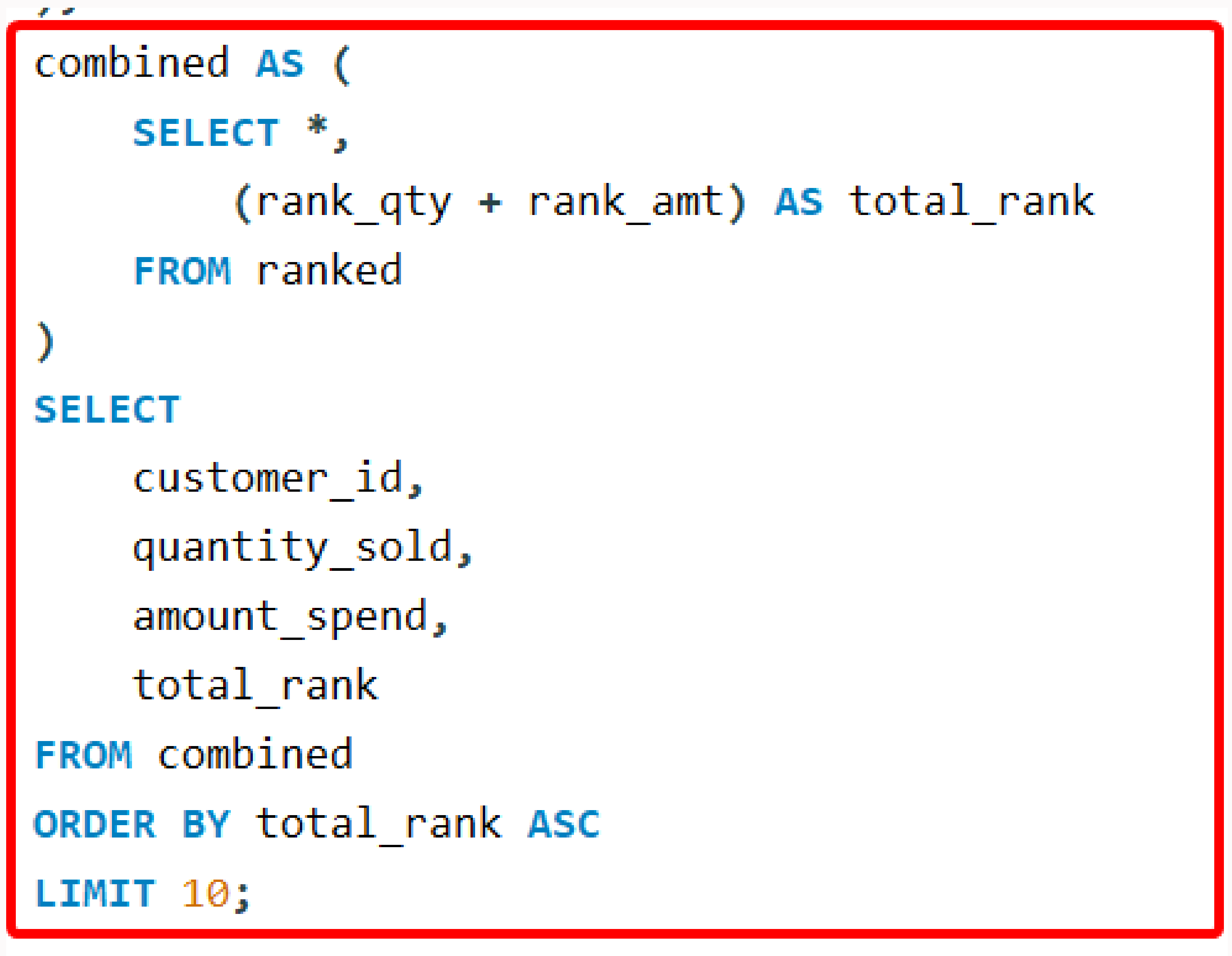
Product category	Total_Sales_Revenue
HEALTH BEAUTY	1441248.07
WATCHES PRESENT	1305541.61
BED TABLE BATH	1241681.72
SPORT LEISURE	1156656.48
COMPUTER ACCESSORIES	1059272.4

"Health Beauty" product category is being at 1st with $14,41,248.07 total sales revenue, at 2nd postion "Watches Present" with $13,05,541.61, at 3rd position "Bed Table Bath" with $12,41,681.72, at 4th "Sport Leisure" with $11,56,656.48 and at 5th "COmputer Accessories" with $10,59,272.4 .

3. Find customers with the highest number of orders and total spend, ranking them as Walmart's most valuable customers.

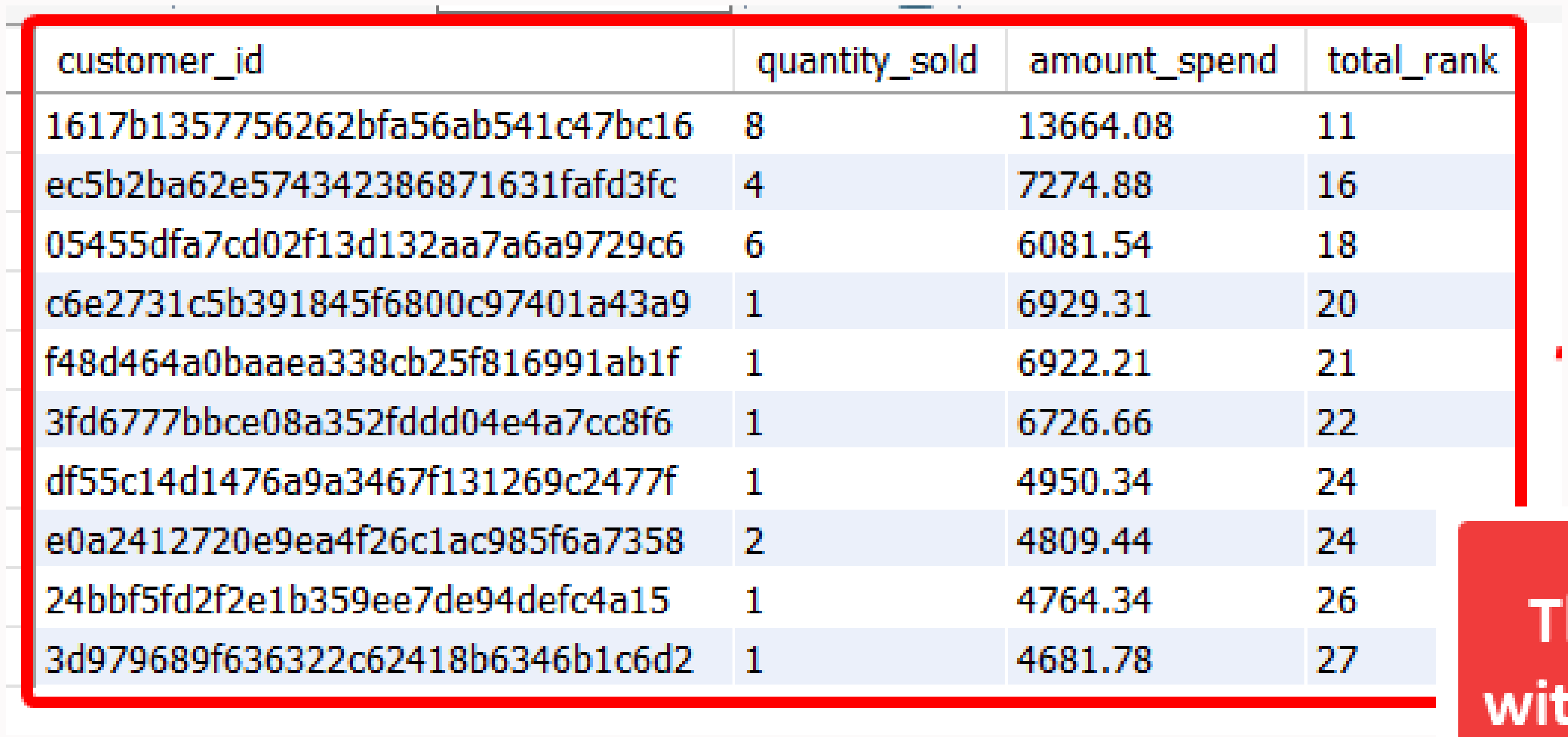


```sql
WITH allCustomers AS (
    SELECT
        c.customer_id,
        COUNT(ot.order_item_id) AS quantity_sold,
        ROUND(SUM(ot.price + ot.freight_value), 2) AS amount_spend
    FROM customers c
    JOIN orders o ON o.customer_id = c.customer_id
    JOIN order_items ot ON ot.order_id = o.order_id
    GROUP BY c.customer_id
),
ranked AS (
    SELECT
        customer_id,
        quantity_sold,
        amount_spend,
        DENSE_RANK() OVER (ORDER BY quantity_sold DESC) AS rank_qty,
        DENSE_RANK() OVER (ORDER BY amount_spend DESC) AS rank_amt
    FROM allCustomers
),
```



```sql
combined AS (
    SELECT *,
        (rank_qty + rank_amt) AS total_rank
    FROM ranked
)
SELECT
    customer_id,
    quantity_sold,
    amount_spend,
    total_rank
FROM combined
ORDER BY total_rank ASC
LIMIT 10;
```

Output

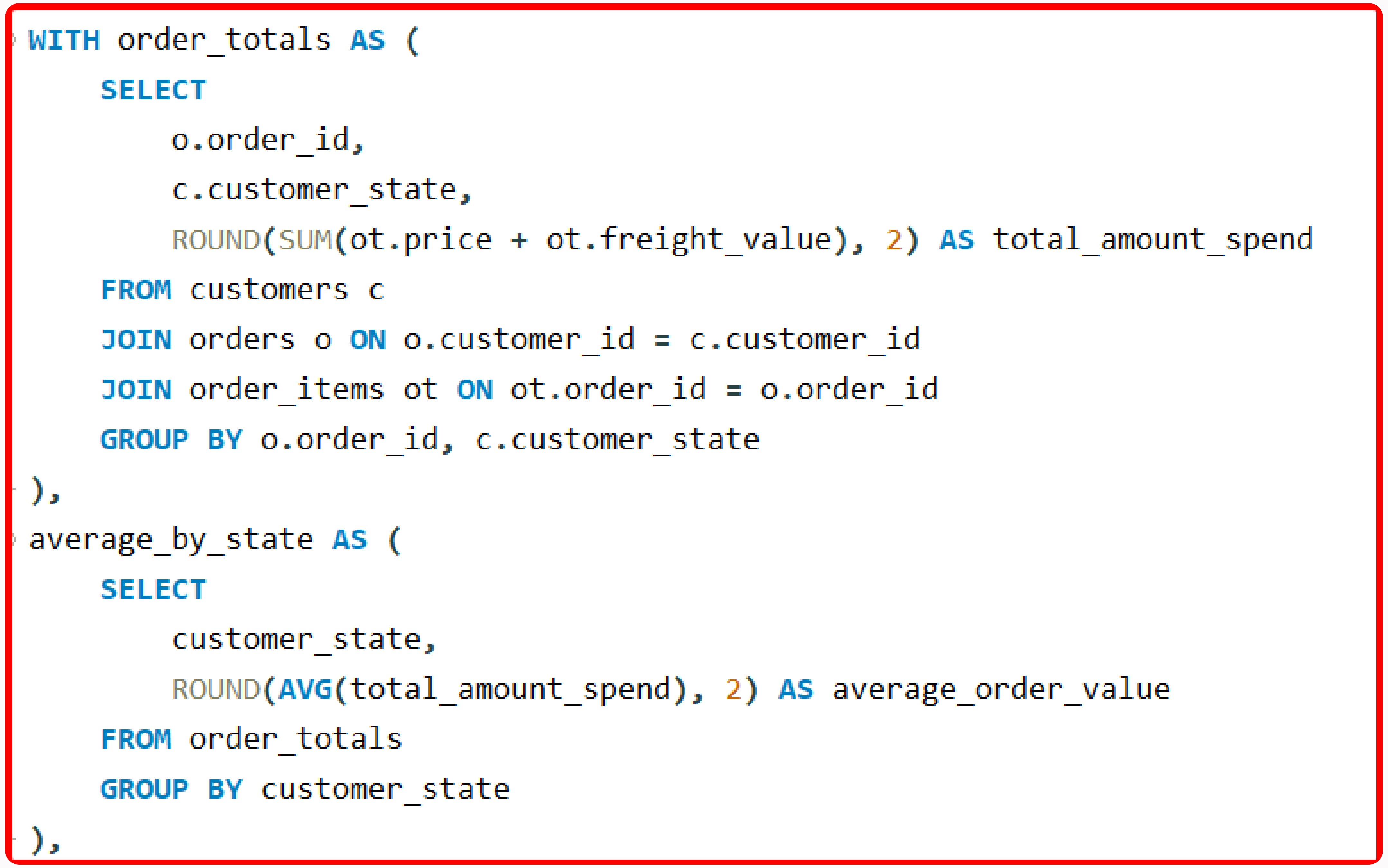


customer_id	quantity_sold	amount_spend	total_rank
1617b1357756262bfa56ab541c47bc16	8	13664.08	11
ec5b2ba62e574342386871631fafd3fc	4	7274.88	16
05455dfa7cd02f13d132aa7a6a9729c6	6	6081.54	18
c6e2731c5b391845f6800c97401a43a9	1	6929.31	20
f48d464a0baaea338cb25f816991ab1f	1	6922.21	21
3fd6777bbce08a352fddd04e4a7cc8f6	1	6726.66	22
df55c14d1476a9a3467f131269c2477f	1	4950.34	24
e0a2412720e9ea4f26c1ac985f6a7358	2	4809.44	24
24bbf5fd2f2e1b359ee7de94defc4a15	1	4764.34	26
3d979689f636322c62418b6346b1c6d2	1	4681.78	27

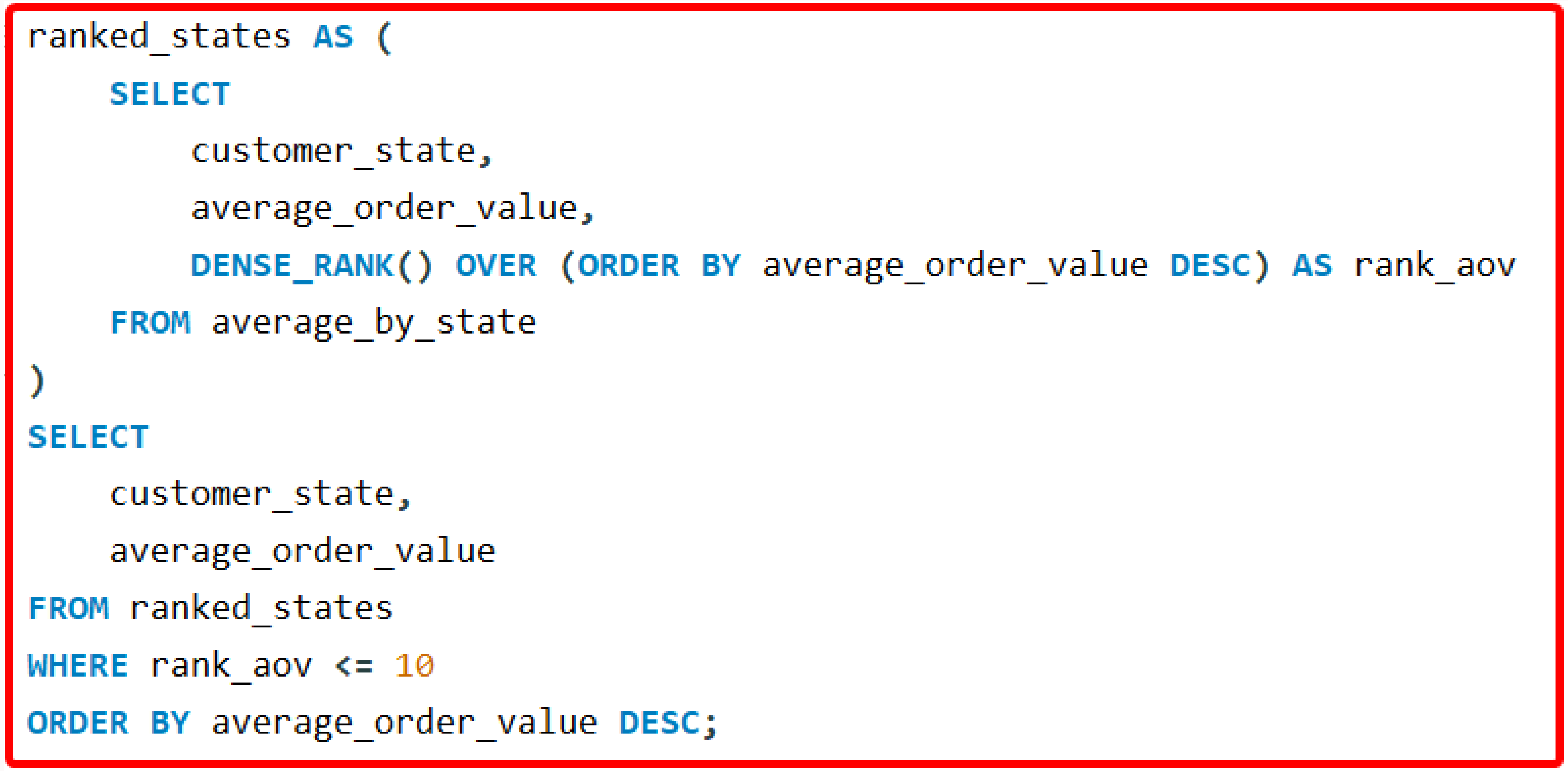
These are the top 10 customers with highest number of orders and total spend combine, making them walmart's most valuable customers .

4.

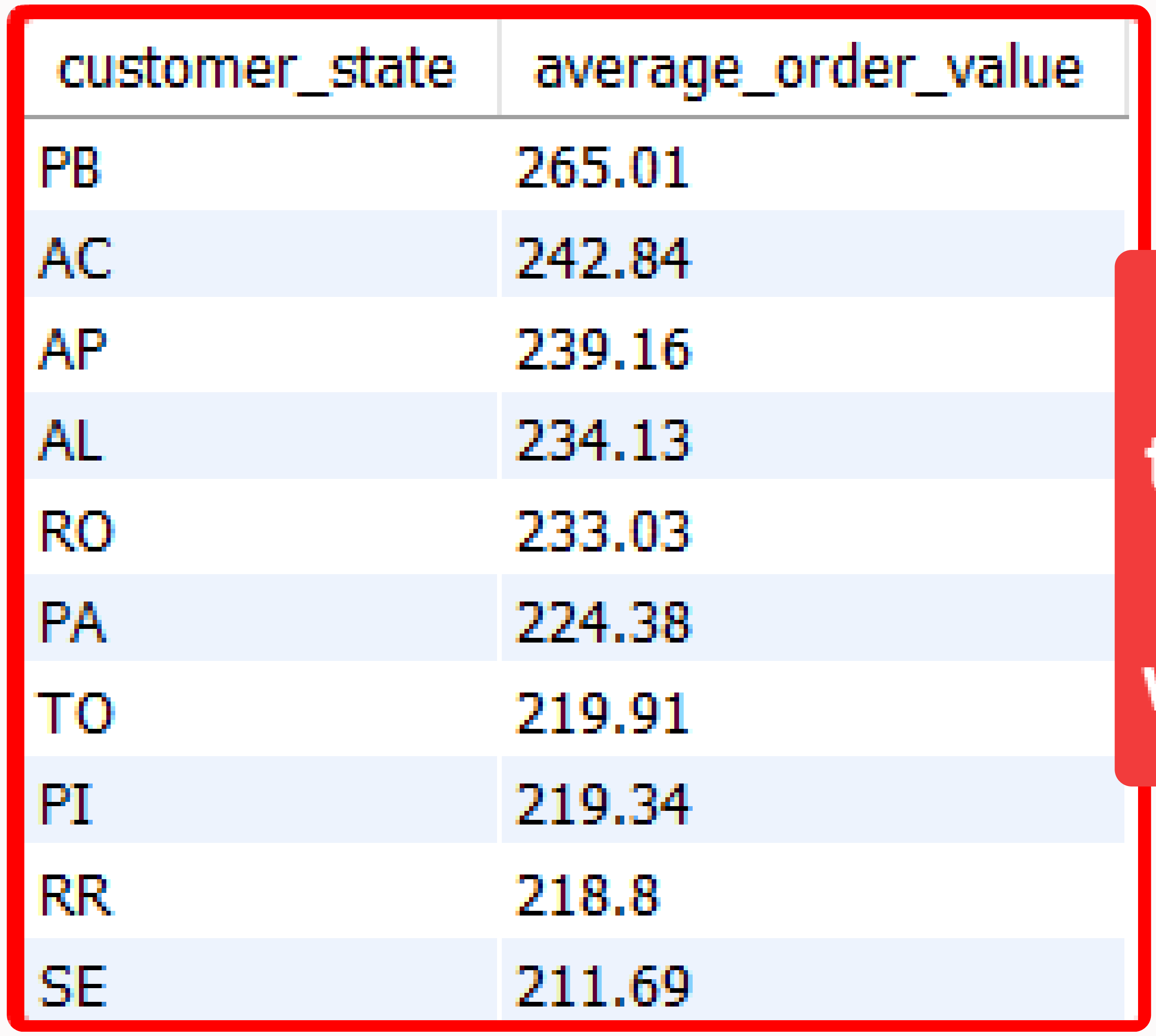
Determine customer states with the highest average order value (AOV).



```sql
WITH order_totals AS (
    SELECT
        o.order_id,
        c.customer_state,
        ROUND(SUM(ot.price + ot.freight_value), 2) AS total_amount_spend
    FROM customers c
    JOIN orders o ON o.customer_id = c.customer_id
    JOIN order_items ot ON ot.order_id = o.order_id
    GROUP BY o.order_id, c.customer_state
),
average_by_state AS (
    SELECT
        customer_state,
        ROUND(AVG(total_amount_spend), 2) AS average_order_value
    FROM order_totals
    GROUP BY customer_state
),
```


```sql
ranked_states AS (
    SELECT
        customer_state,
        average_order_value,
        DENSE_RANK() OVER (ORDER BY average_order_value DESC) AS rank_aov
    FROM average_by_state
)
SELECT
    customer_state,
    average_order_value
FROM ranked_states
WHERE rank_aov <= 10
ORDER BY average_order_value DESC;
```

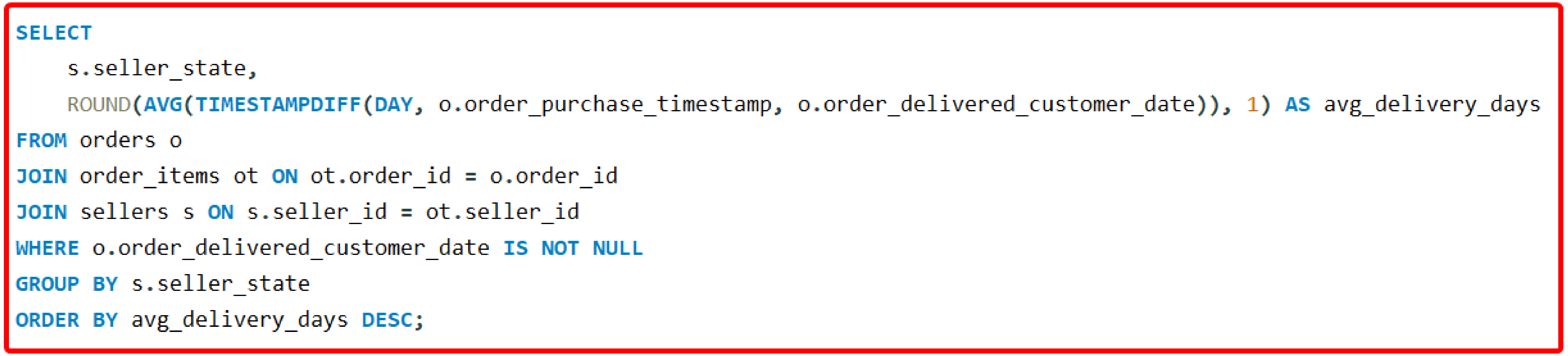
Output



customer_state	average_order_value
PB	265.01
AC	242.84
AP	239.16
AL	234.13
RO	233.03
PA	224.38
TO	219.91
PI	219.34
RR	218.8
SE	211.69

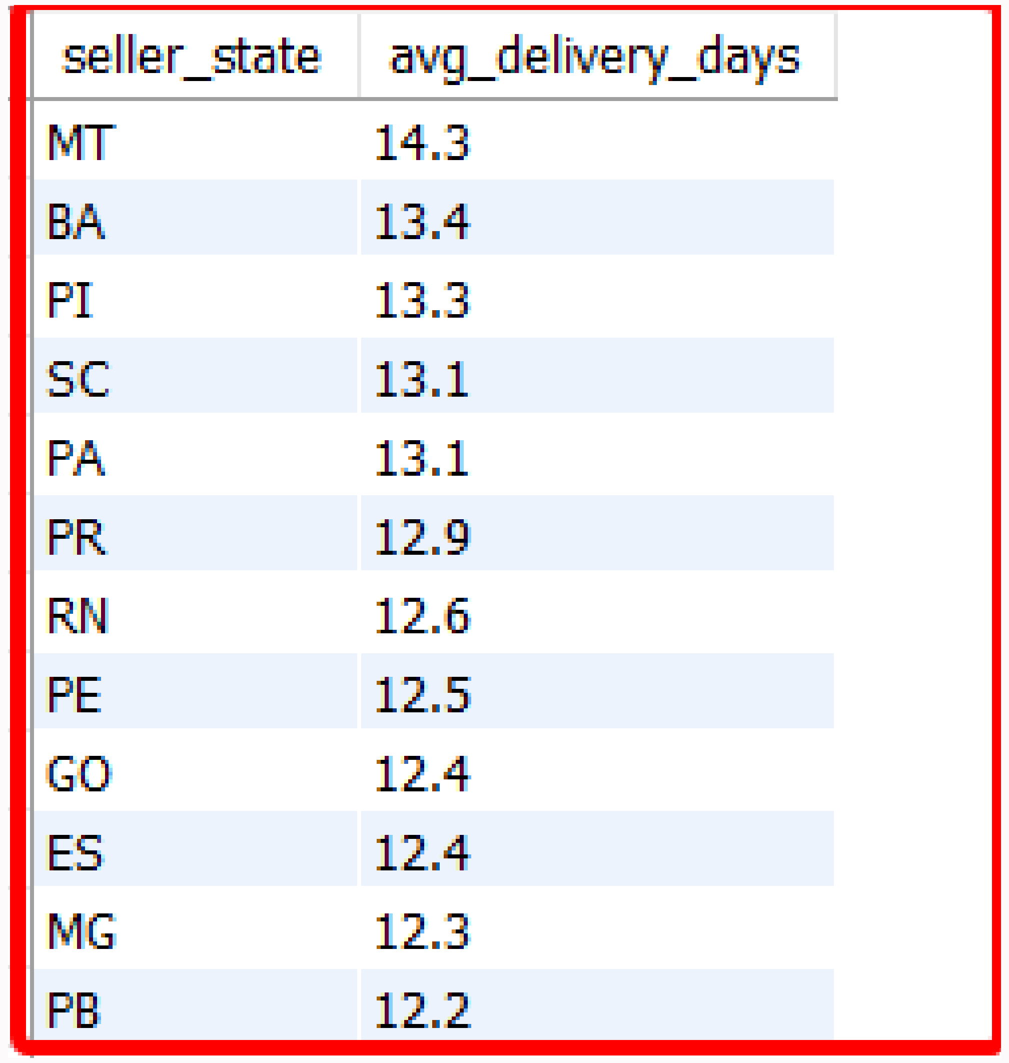
"PB" customer state is beign at top position with 265.01 average order value and at last position with 211.69 average order value.

5. Compute average delivery time (in days) by seller state, calculated as the difference between order_purchase_timestamp and order_delivered_customer_date.

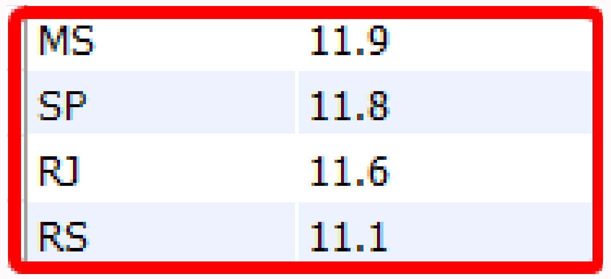


```sql
SELECT
    s.seller_state,
    ROUND(AVG(TIMESTAMPDIFF(DAY, o.order_purchase_timestamp, o.order_delivered_customer_date)), 1) AS avg_delivery_days
FROM orders o
JOIN order_items ot ON ot.order_id = o.order_id
JOIN sellers s ON s.seller_id = ot.seller_id
WHERE o.order_delivered_customer_date IS NOT NULL
GROUP BY s.seller_state
ORDER BY avg_delivery_days DESC;
```

Output



seller_state	avg_delivery_days
MT	14.3
BA	13.4
PI	13.3
SC	13.1
PA	13.1
PR	12.9
RN	12.6
PE	12.5
GO	12.4
ES	12.4
MG	12.3
PB	12.2

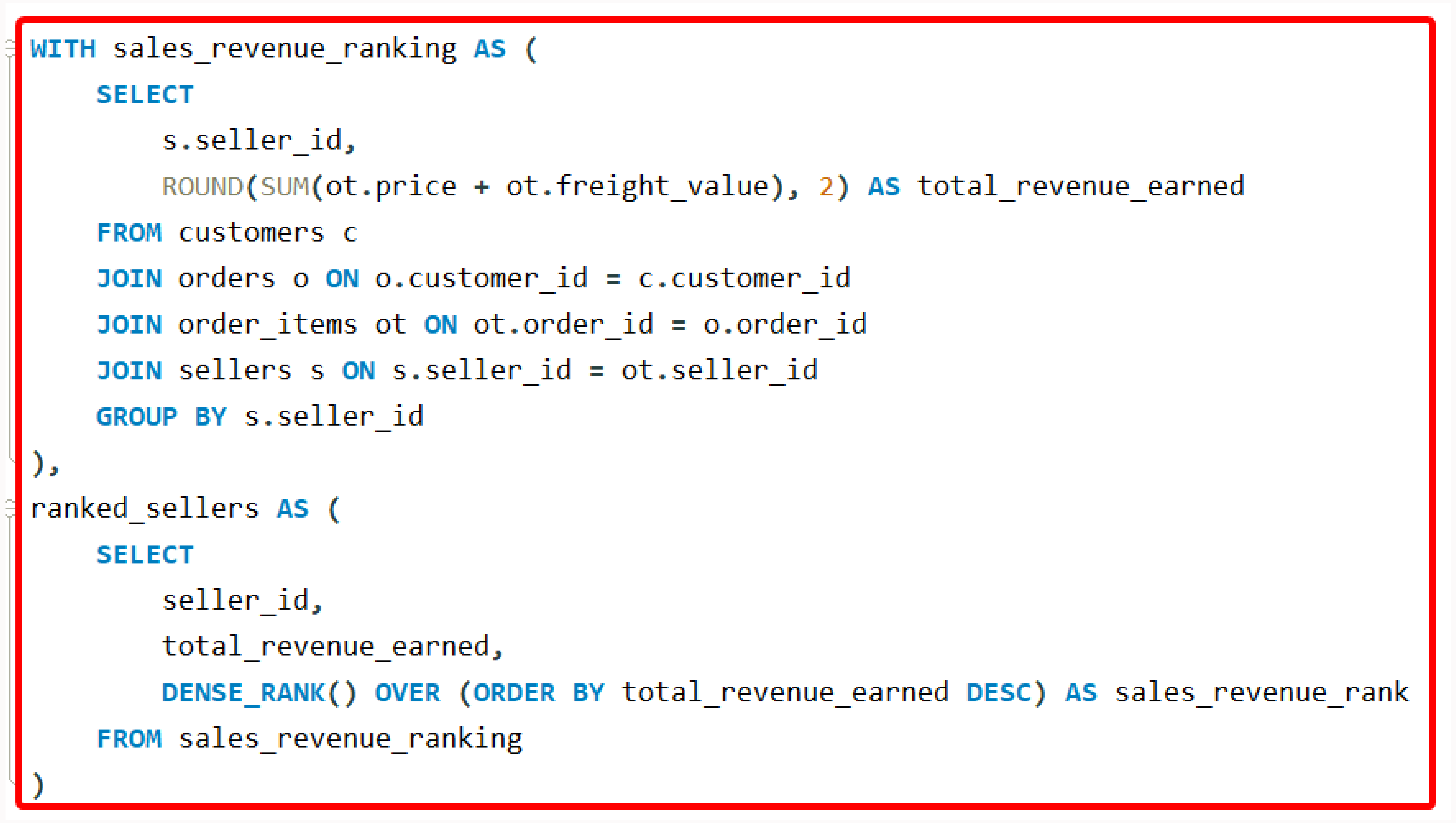


MS	11.9
SP	11.8
RJ	11.6
RS	11.1

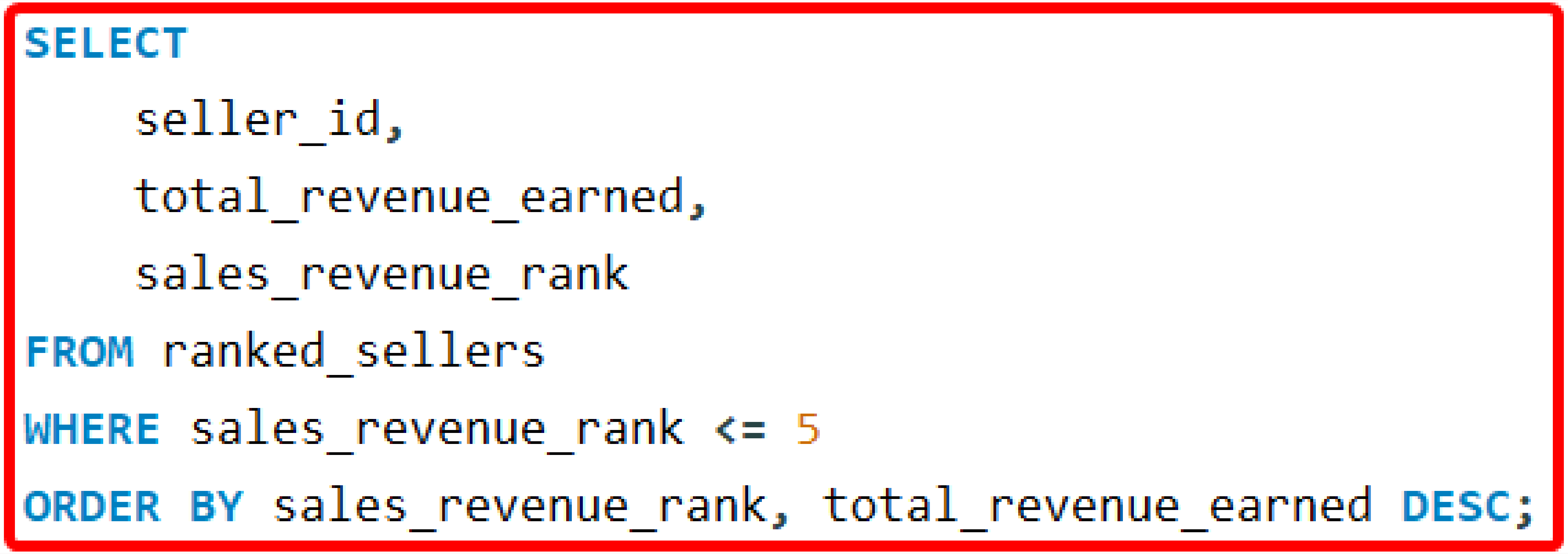
These are the seller states with there average delivery days where state "MT" holds the first position with 14.3 average delivery days and at last position state "RS" with 11.1 average delivery days.

6.

List the top 5 sellers based on total revenue earned.

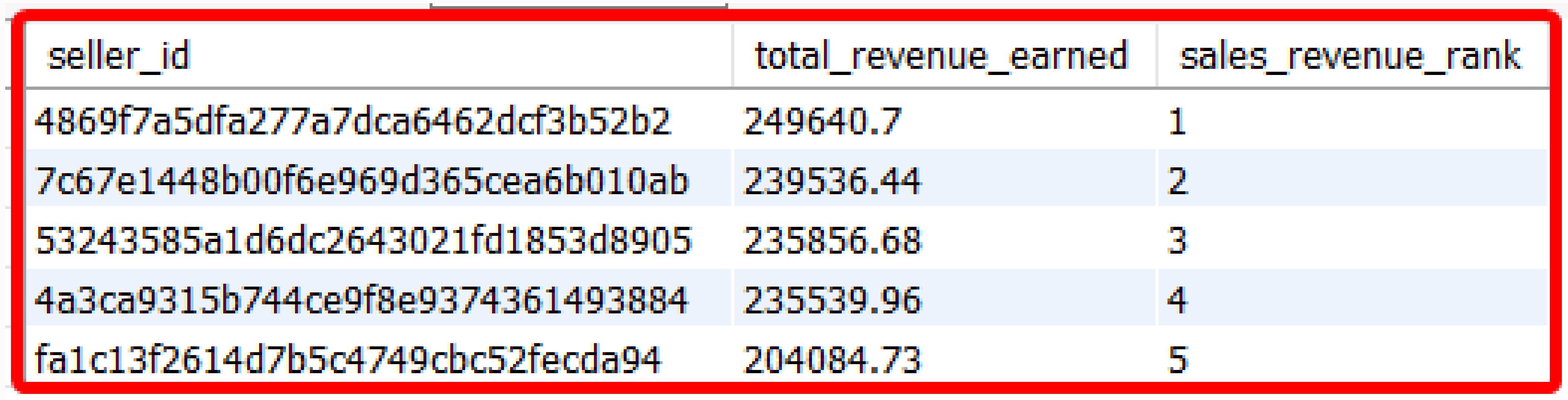


```sql
WITH sales_revenue_ranking AS (
    SELECT
        s.seller_id,
        ROUND(SUM(ot.price + ot.freight_value), 2) AS total_revenue_earned
    FROM customers c
    JOIN orders o ON o.customer_id = c.customer_id
    JOIN order_items ot ON ot.order_id = o.order_id
    JOIN sellers s ON s.seller_id = ot.seller_id
    GROUP BY s.seller_id
),
ranked_sellers AS (
    SELECT
        seller_id,
        total_revenue_earned,
        DENSE_RANK() OVER (ORDER BY total_revenue_earned DESC) AS sales_revenue_rank
    FROM sales_revenue_ranking
)
```



```sql
SELECT
    seller_id,
    total_revenue_earned,
    sales_revenue_rank
FROM ranked_sellers
WHERE sales_revenue_rank <= 5
ORDER BY sales_revenue_rank, total_revenue_earned DESC;
```

Output

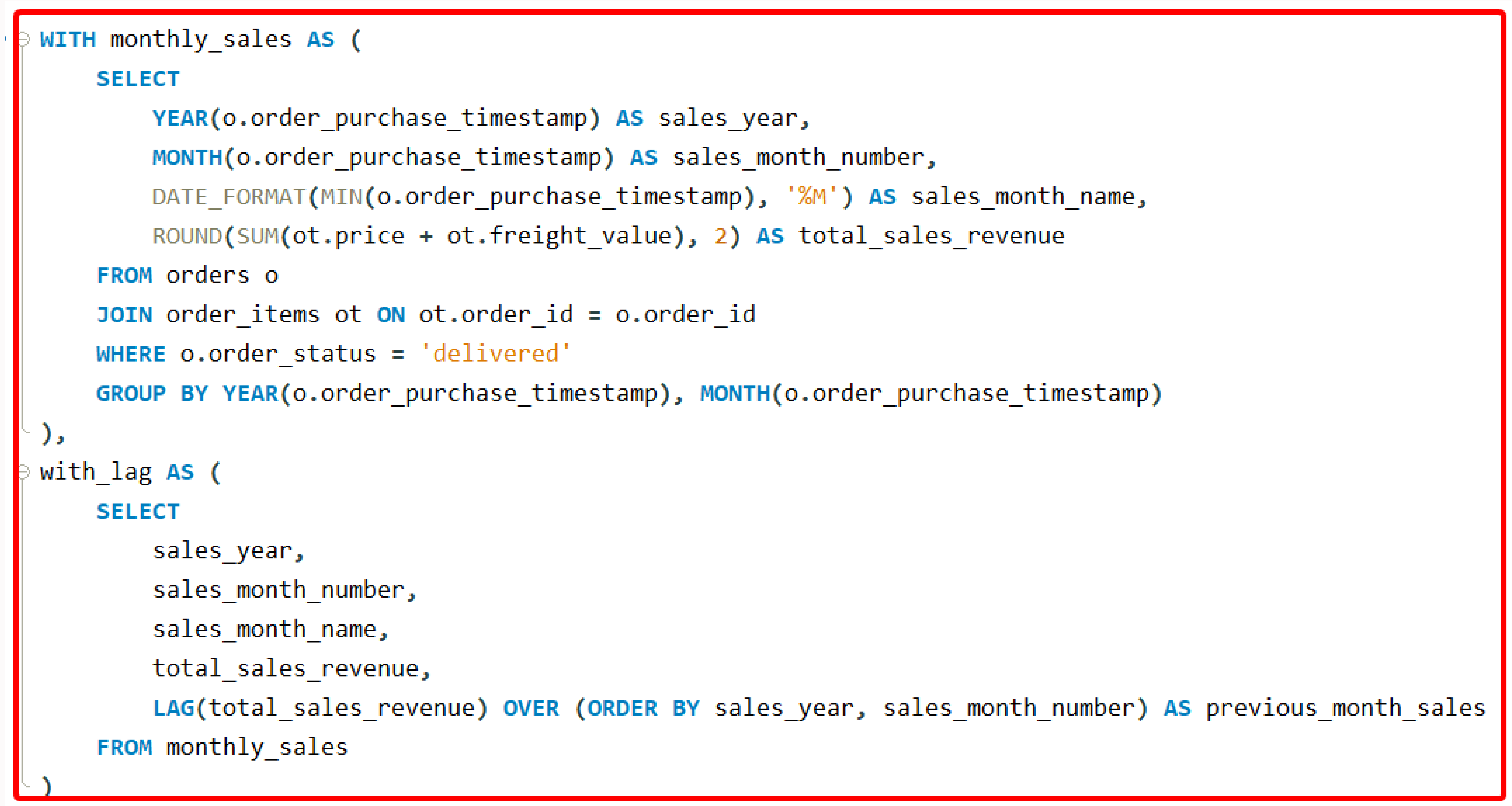


seller_id	total_revenue_earned	sales_revenue_rank
4869f7a5dfa277a7dca6462dcf3b52b2	249640.7	1
7c67e1448b00f6e969d365cea6b010ab	239536.44	2
53243585a1d6dc2643021fd1853d8905	235856.68	3
4a3ca9315b744ce9f8e9374361493884	235539.96	4
fa1c13f2614d7b5c4749cbc52fecda94	204084.73	5

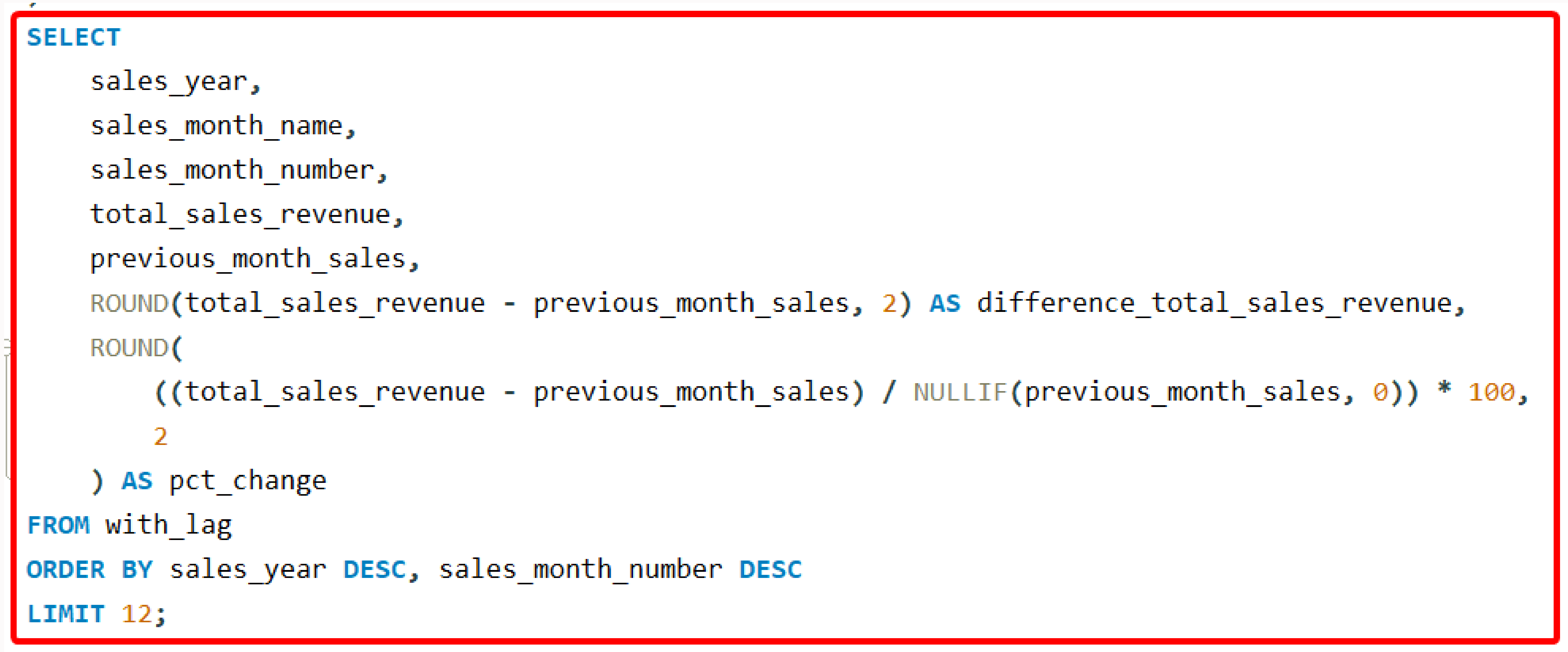
These are the sellers with highest revenue generated .

7.

Analyze the monthly revenue trend over the last 12 months to track Walmart’s growth.

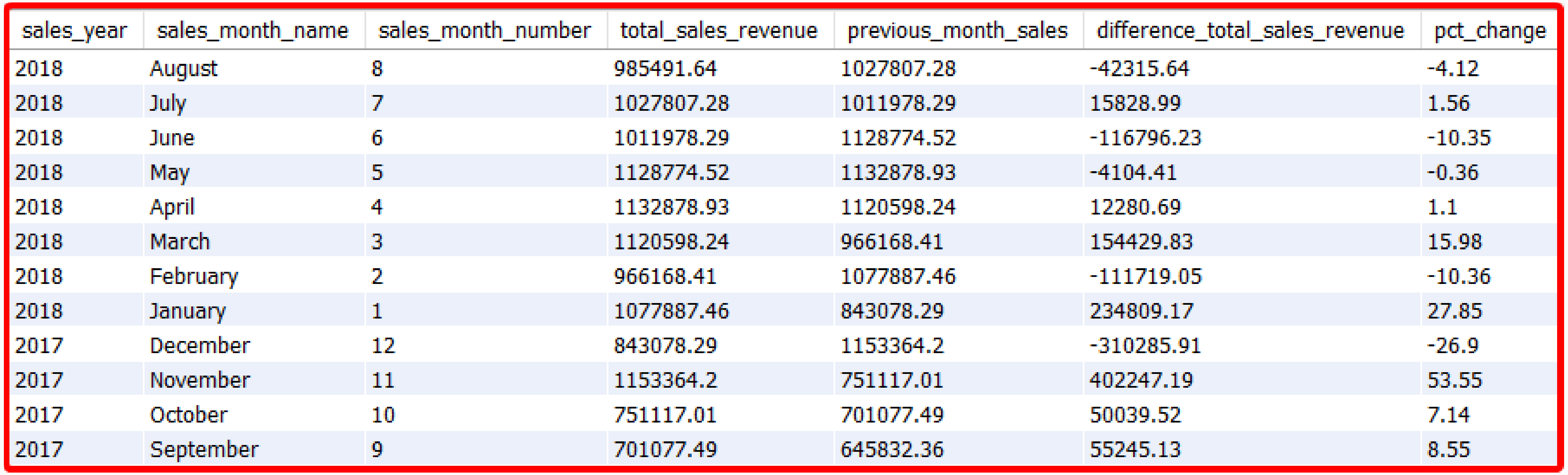


```sql
WITH monthly_sales AS (
    SELECT
        YEAR(o.order_purchase_timestamp) AS sales_year,
        MONTH(o.order_purchase_timestamp) AS sales_month_number,
        DATE_FORMAT(MIN(o.order_purchase_timestamp), '%M') AS sales_month_name,
        ROUND(SUM(ot.price + ot.freight_value), 2) AS total_sales_revenue
    FROM orders o
    JOIN order_items ot ON ot.order_id = o.order_id
    WHERE o.order_status = 'delivered'
    GROUP BY YEAR(o.order_purchase_timestamp), MONTH(o.order_purchase_timestamp)
),
with_lag AS (
    SELECT
        sales_year,
        sales_month_number,
        sales_month_name,
        total_sales_revenue,
        LAG(total_sales_revenue) OVER (ORDER BY sales_year, sales_month_number) AS previous_month_sales
    FROM monthly_sales
)
```



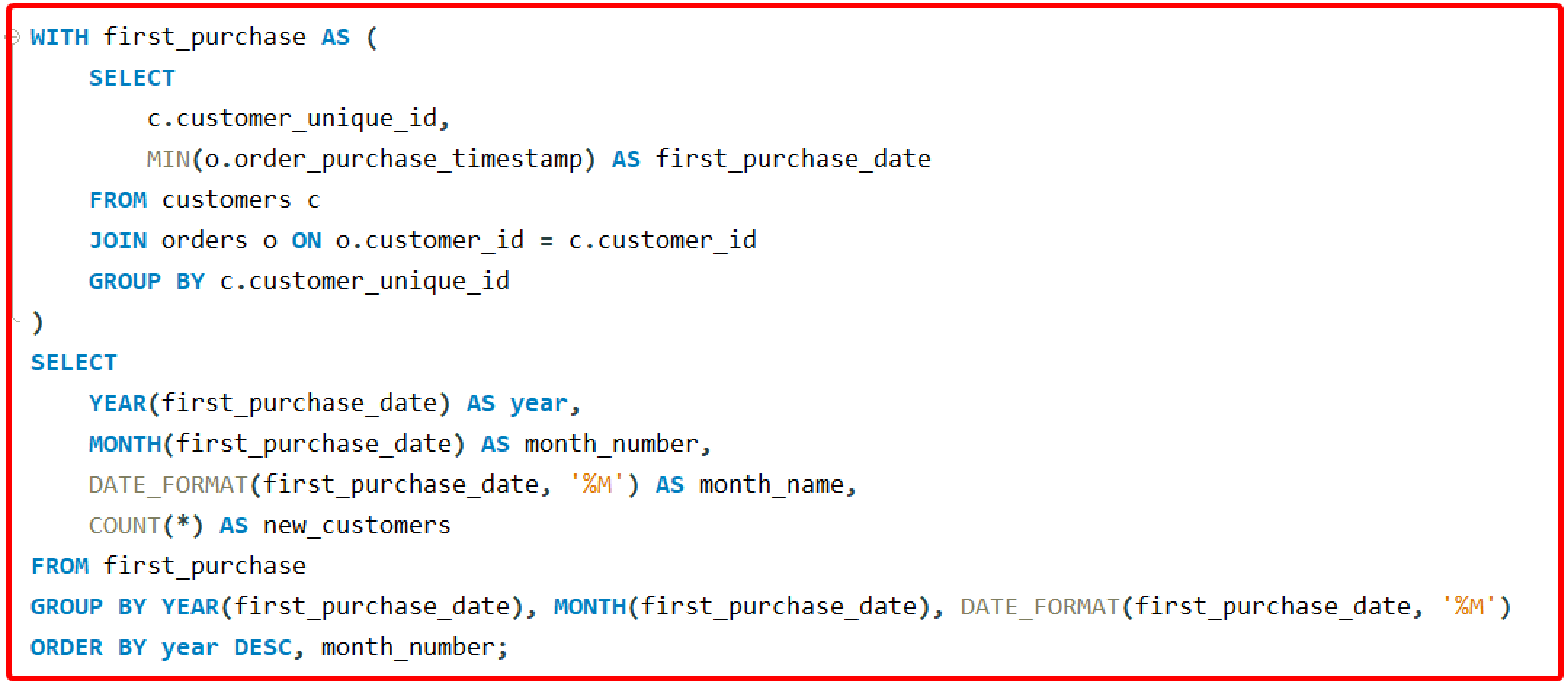
```sql
SELECT
    sales_year,
    sales_month_name,
    sales_month_number,
    total_sales_revenue,
    previous_month_sales,
    ROUND(total_sales_revenue - previous_month_sales, 2) AS difference_total_sales_revenue,
    ROUND(
        ((total_sales_revenue - previous_month_sales) / NULLIF(previous_month_sales, 0)) * 100,
        2
    ) AS pct_change
FROM with_lag
ORDER BY sales_year DESC, sales_month_number DESC
LIMIT 12;
```

Output



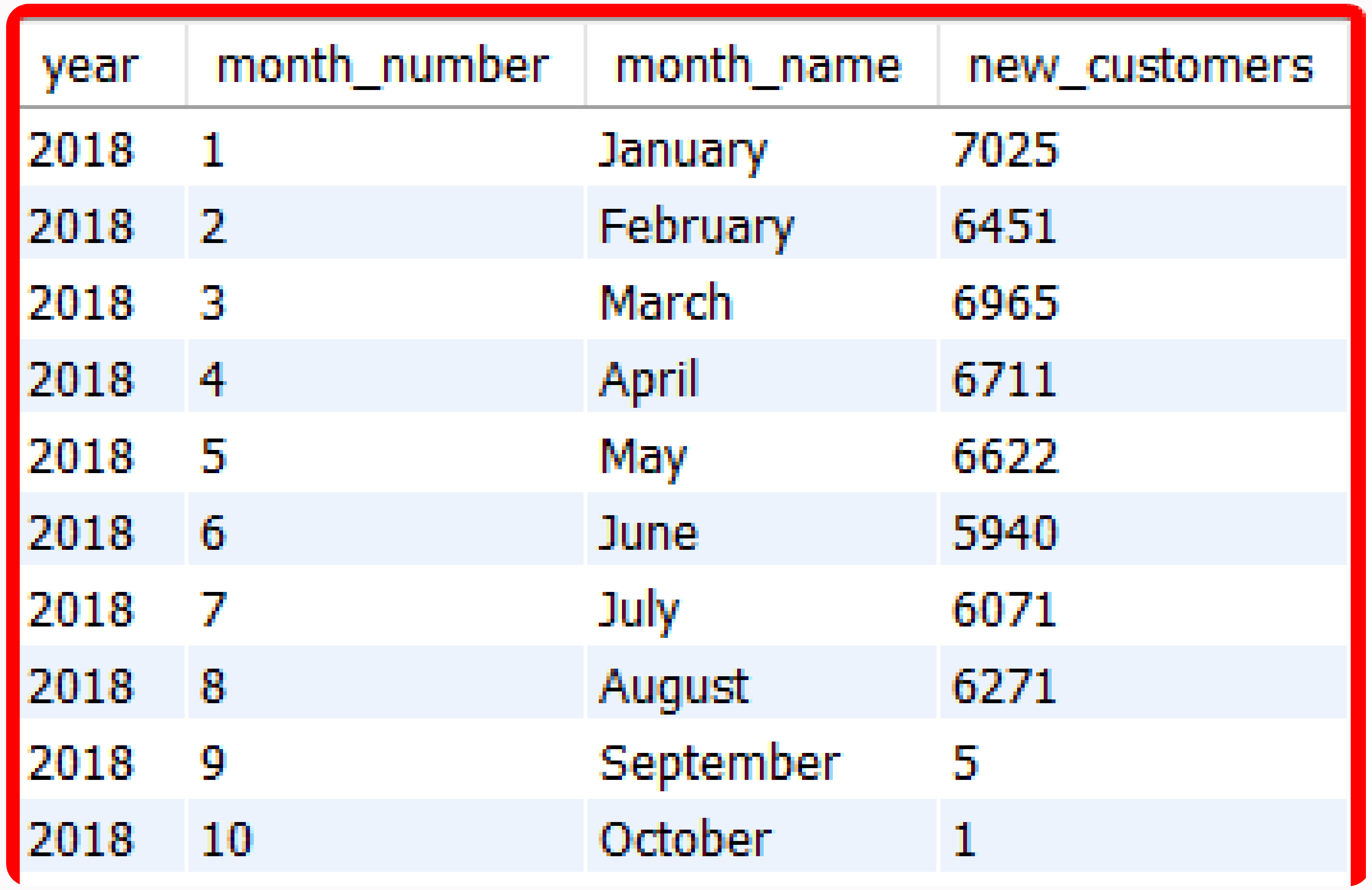
sales_year	sales_month_name	sales_month_number	total_sales_revenue	previous_month_sales	difference_total_sales_revenue	pct_change
2018	August	8	985491.64	1027807.28	-42315.64	-4.12
2018	July	7	1027807.28	1011978.29	15828.99	1.56
2018	June	6	1011978.29	1128774.52	-116796.23	-10.35
2018	May	5	1128774.52	1132878.93	-4104.41	-0.36
2018	April	4	1132878.93	1120598.24	12280.69	1.1
2018	March	3	1120598.24	966168.41	154429.83	15.98
2018	February	2	966168.41	1077887.46	-111719.05	-10.36
2018	January	1	1077887.46	843078.29	234809.17	27.85
2017	December	12	843078.29	1153364.2	-310285.91	-26.9
2017	November	11	1153364.2	751117.01	402247.19	53.55
2017	October	10	751117.01	701077.49	50039.52	7.14
2017	September	9	701077.49	645832.36	55245.13	8.55

8. Calculate the number of new unique customers acquired each month, based on customer_unique_id.

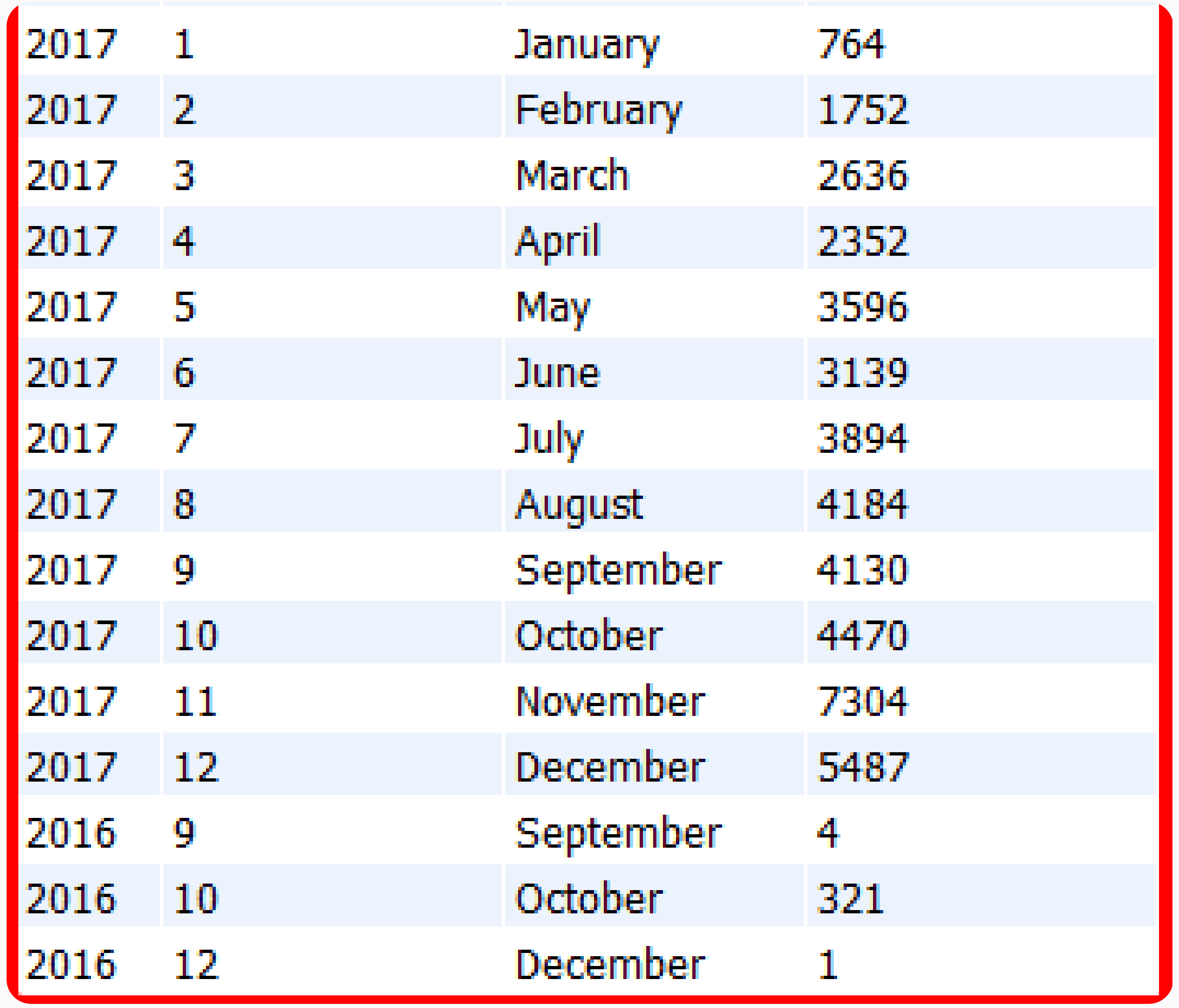


```
WITH first_purchase AS (
    SELECT
        c.customer_unique_id,
        MIN(o.order_purchase_timestamp) AS first_purchase_date
    FROM customers c
    JOIN orders o ON o.customer_id = c.customer_id
    GROUP BY c.customer_unique_id
)
SELECT
    YEAR(first_purchase_date) AS year,
    MONTH(first_purchase_date) AS month_number,
    DATE_FORMAT(first_purchase_date, '%M') AS month_name,
    COUNT(*) AS new_customers
FROM first_purchase
GROUP BY YEAR(first_purchase_date), MONTH(first_purchase_date), DATE_FORMAT(first_purchase_date, '%M')
ORDER BY year DESC, month_number;
```

Output



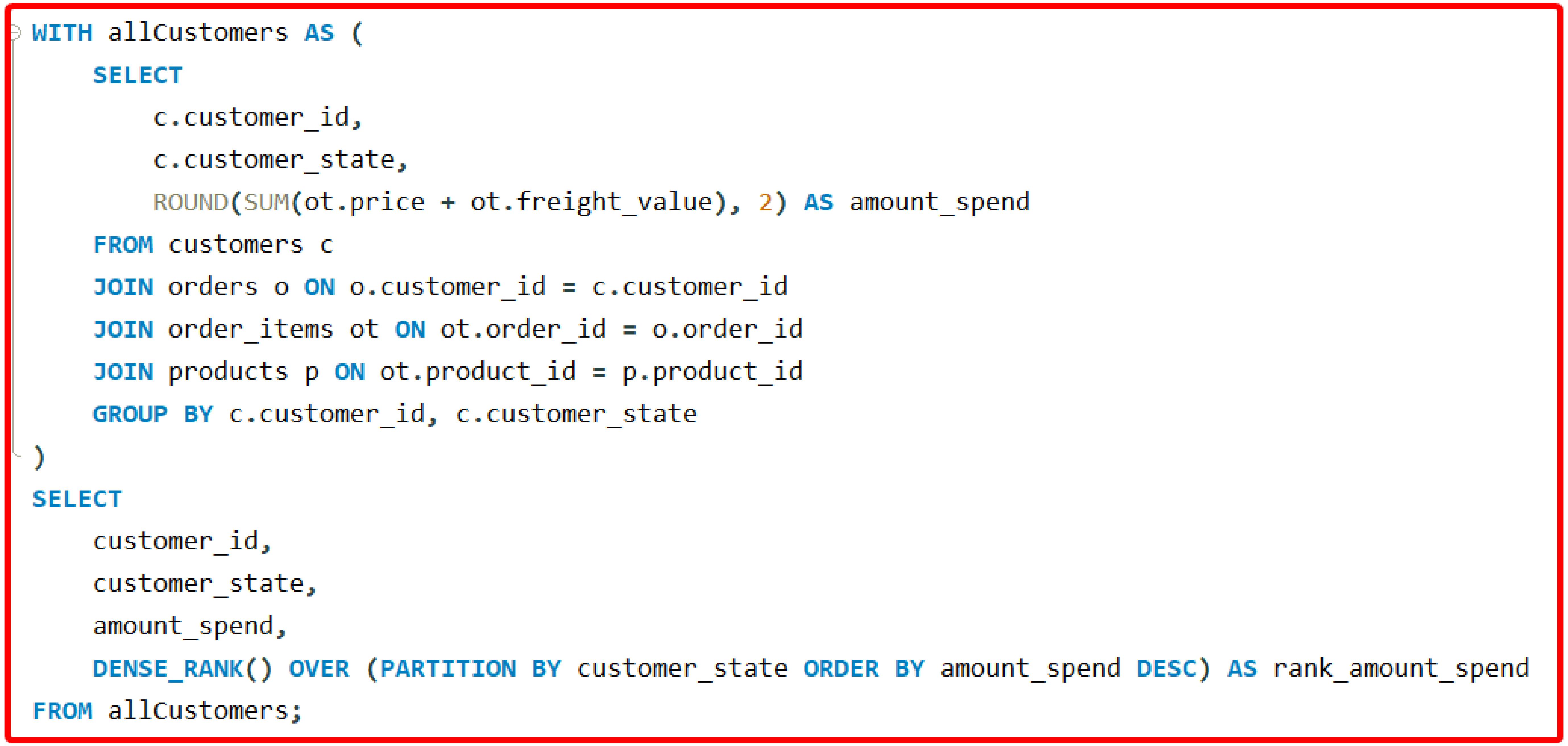
year	month_number	month_name	new_customers
2018	1	January	7025
2018	2	February	6451
2018	3	March	6965
2018	4	April	6711
2018	5	May	6622
2018	6	June	5940
2018	7	July	6071
2018	8	August	6271
2018	9	September	5
2018	10	October	1



2017	1	January	764
2017	2	February	1752
2017	3	March	2636
2017	4	April	2352
2017	5	May	3596
2017	6	June	3139
2017	7	July	3894
2017	8	August	4184
2017	9	September	4130
2017	10	October	4470
2017	11	November	7304
2017	12	December	5487
2016	9	September	4
2016	10	October	321
2016	12	December	1

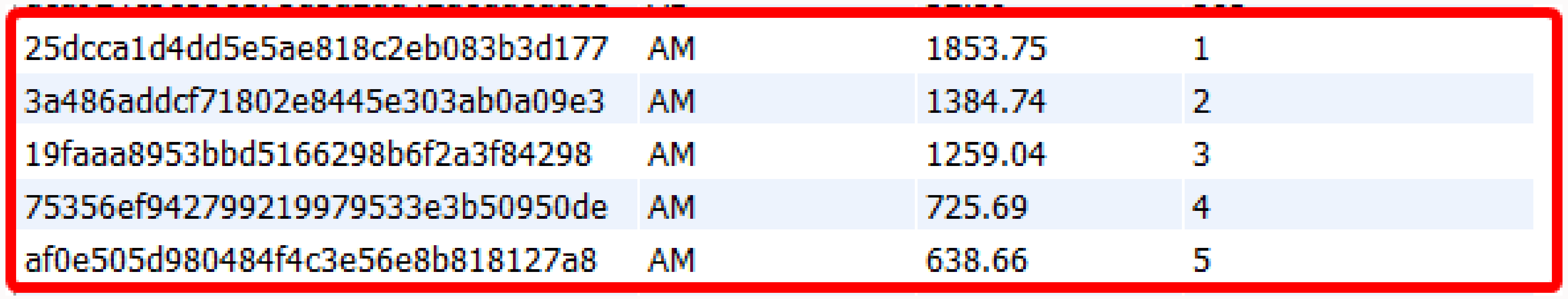
9.

Rank customers by lifetime spend within each customer state using SQL window functions.

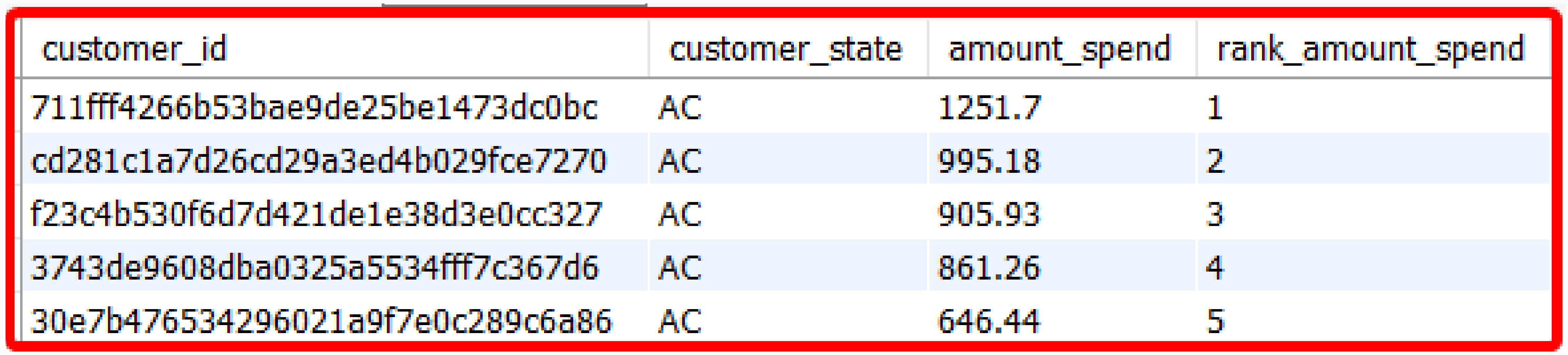


```sql
WITH allCustomers AS (
    SELECT
        c.customer_id,
        c.customer_state,
        ROUND(SUM(ot.price + ot.freight_value), 2) AS amount_spend
    FROM customers c
    JOIN orders o ON o.customer_id = c.customer_id
    JOIN order_items ot ON ot.order_id = o.order_id
    JOIN products p ON ot.product_id = p.product_id
    GROUP BY c.customer_id, c.customer_state
)
SELECT
    customer_id,
    customer_state,
    amount_spend,
    DENSE_RANK() OVER (PARTITION BY customer_state ORDER BY amount_spend DESC) AS rank_amount_spend
FROM allCustomers;
```

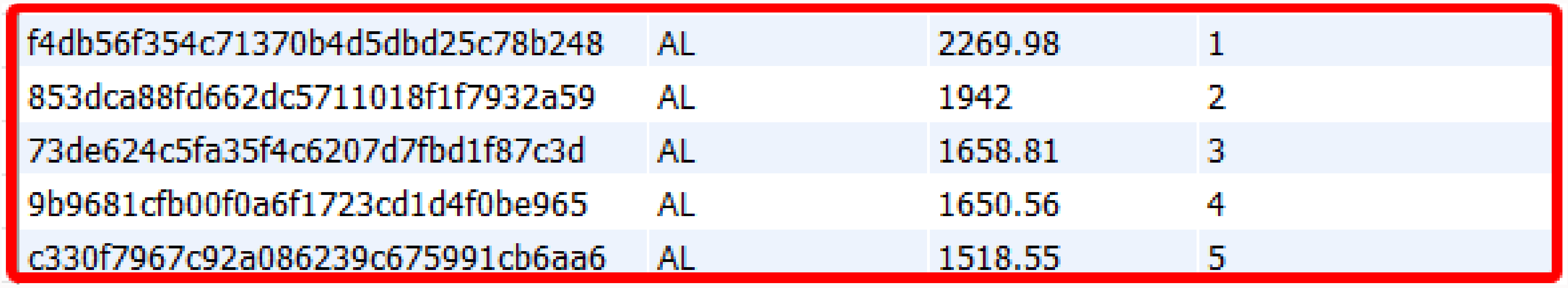
Output



25dcca1d4dd5e5ae818c2eb083b3d177	AM	1853.75	1
3a486addcf71802e8445e303ab0a09e3	AM	1384.74	2
19faaa8953bbd5166298b6f2a3f84298	AM	1259.04	3
75356ef942799219979533e3b50950de	AM	725.69	4
af0e505d980484f4c3e56e8b818127a8	AM	638.66	5

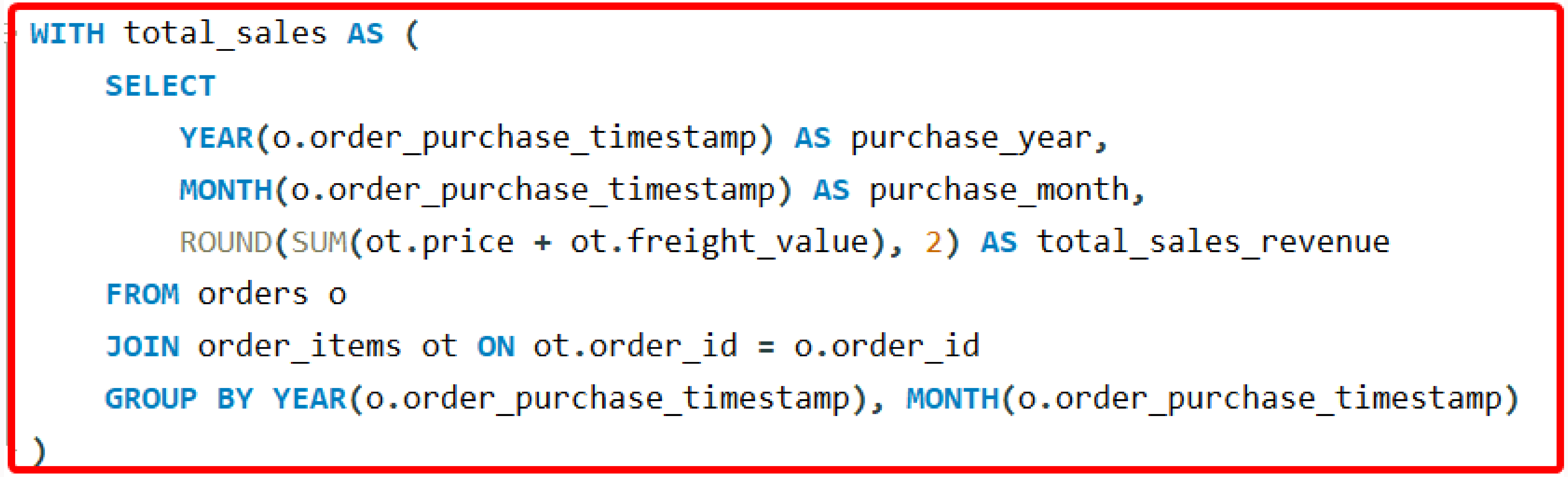


customer_id	customer_state	amount_spend	rank_amount_spend
711fff4266b53bae9de25be1473dc0bc	AC	1251.7	1
cd281c1a7d26cd29a3ed4b029fce7270	AC	995.18	2
f23c4b530f6d7d421de1e38d3e0cc327	AC	905.93	3
3743de9608dba0325a5534fff7c367d6	AC	861.26	4
30e7b476534296021a9f7e0c289c6a86	AC	646.44	5

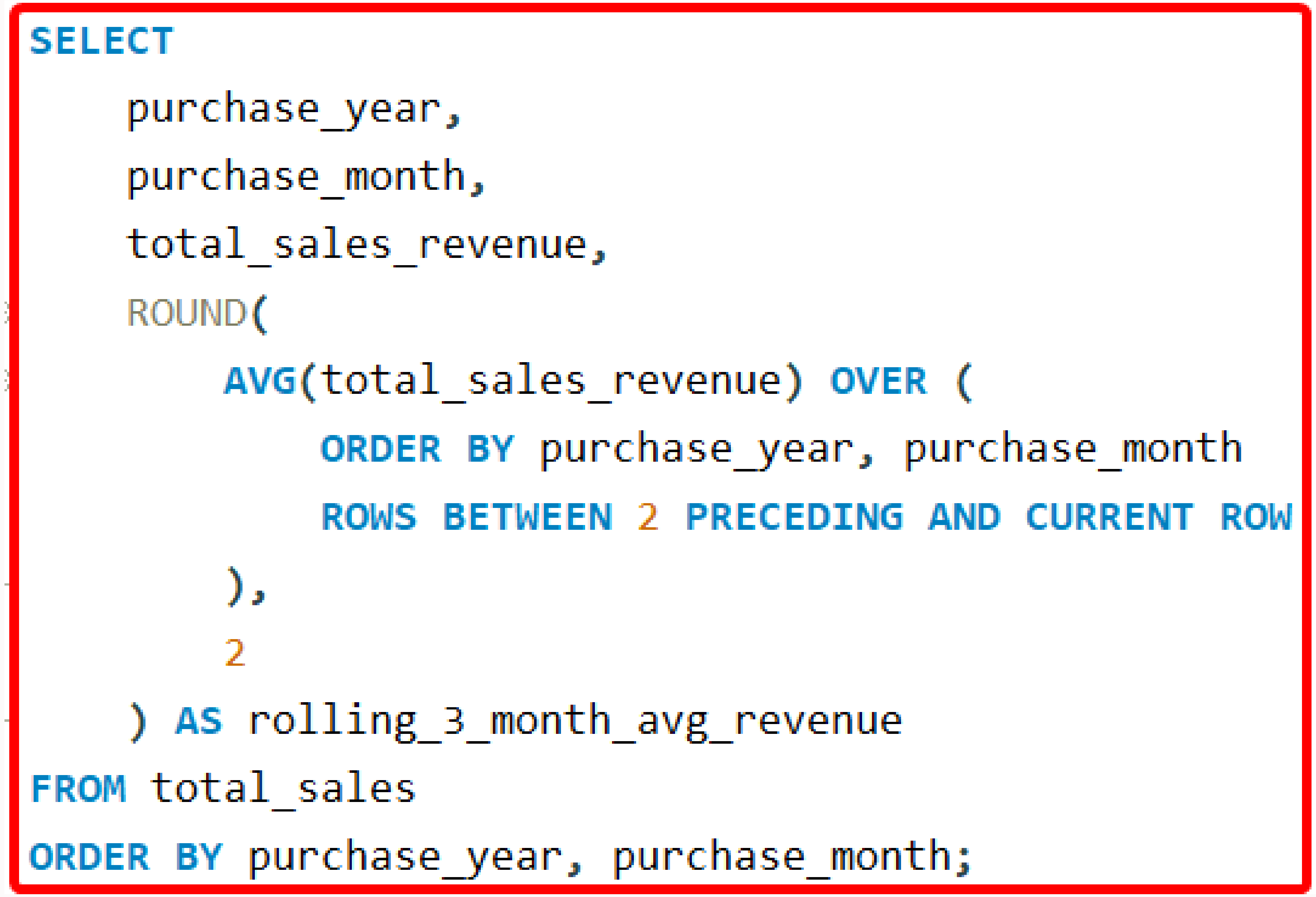


f4db56f354c71370b4d5dbd25c78b248	AL	2269.98	1
853dca88fd662dc5711018f1f7932a59	AL	1942	2
73de624c5fa35f4c6207d7fbd1f87c3d	AL	1658.81	3
9b9681cfb00f0a6f1723cd1d4f0be965	AL	1650.56	4
c330f7967c92a086239c675991cb6aa6	AL	1518.55	5

10. **Compute the rolling 3-month average revenue trend, to visualize sales momentum.**

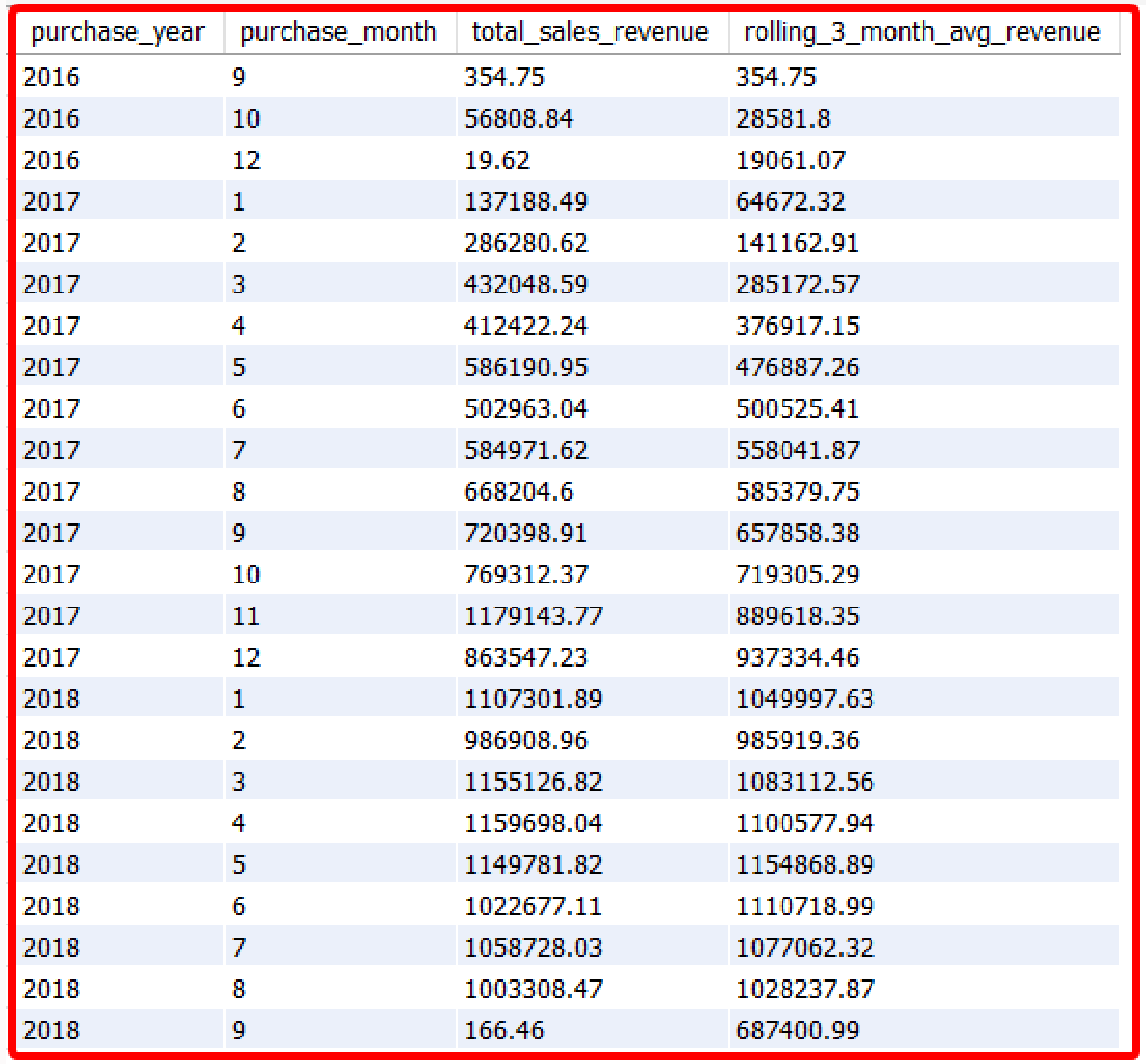


```sql
WITH total_sales AS (
    SELECT
        YEAR(o.order_purchase_timestamp) AS purchase_year,
        MONTH(o.order_purchase_timestamp) AS purchase_month,
        ROUND(SUM(ot.price + ot.freight_value), 2) AS total_sales_revenue
    FROM orders o
    JOIN order_items ot ON ot.order_id = o.order_id
    GROUP BY YEAR(o.order_purchase_timestamp), MONTH(o.order_purchase_timestamp)
)
```



```sql
SELECT
    purchase_year,
    purchase_month,
    total_sales_revenue,
    ROUND(
        AVG(total_sales_revenue) OVER (
            ORDER BY purchase_year, purchase_month
            ROWS BETWEEN 2 PRECEDING AND CURRENT ROW
        ),
        2
    ) AS rolling_3_month_avg_revenue
FROM total_sales
ORDER BY purchase_year, purchase_month;
```

Output



purchase_year	purchase_month	total_sales_revenue	rolling_3_month_avg_revenue
2016	9	354.75	354.75
2016	10	56808.84	28581.8
2016	12	19.62	19061.07
2017	1	137188.49	64672.32
2017	2	286280.62	141162.91
2017	3	432048.59	285172.57
2017	4	412422.24	376917.15
2017	5	586190.95	476887.26
2017	6	502963.04	500525.41
2017	7	584971.62	558041.87
2017	8	668204.6	585379.75
2017	9	720398.91	657858.38
2017	10	769312.37	719305.29
2017	11	1179143.77	889618.35
2017	12	863547.23	937334.46
2018	1	1107301.89	1049997.63
2018	2	986908.96	985919.36
2018	3	1155126.82	1083112.56
2018	4	1159698.04	1100577.94
2018	5	1149781.82	1154868.89
2018	6	1022677.11	1110718.99
2018	7	1058728.03	1077062.32
2018	8	1003308.47	1028237.87
2018	9	166.46	687400.99

Data Analytics at Walmart

Thank You for Joining Us!

Connect **Farook Mohammad**