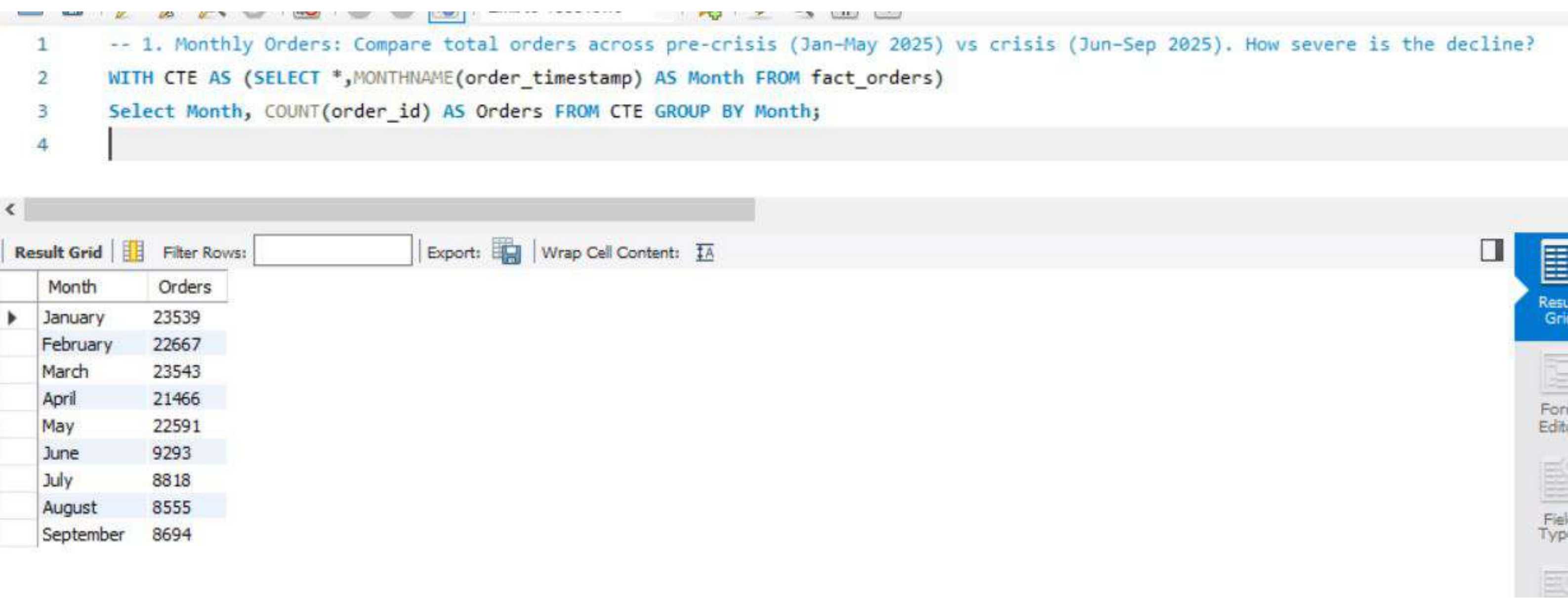
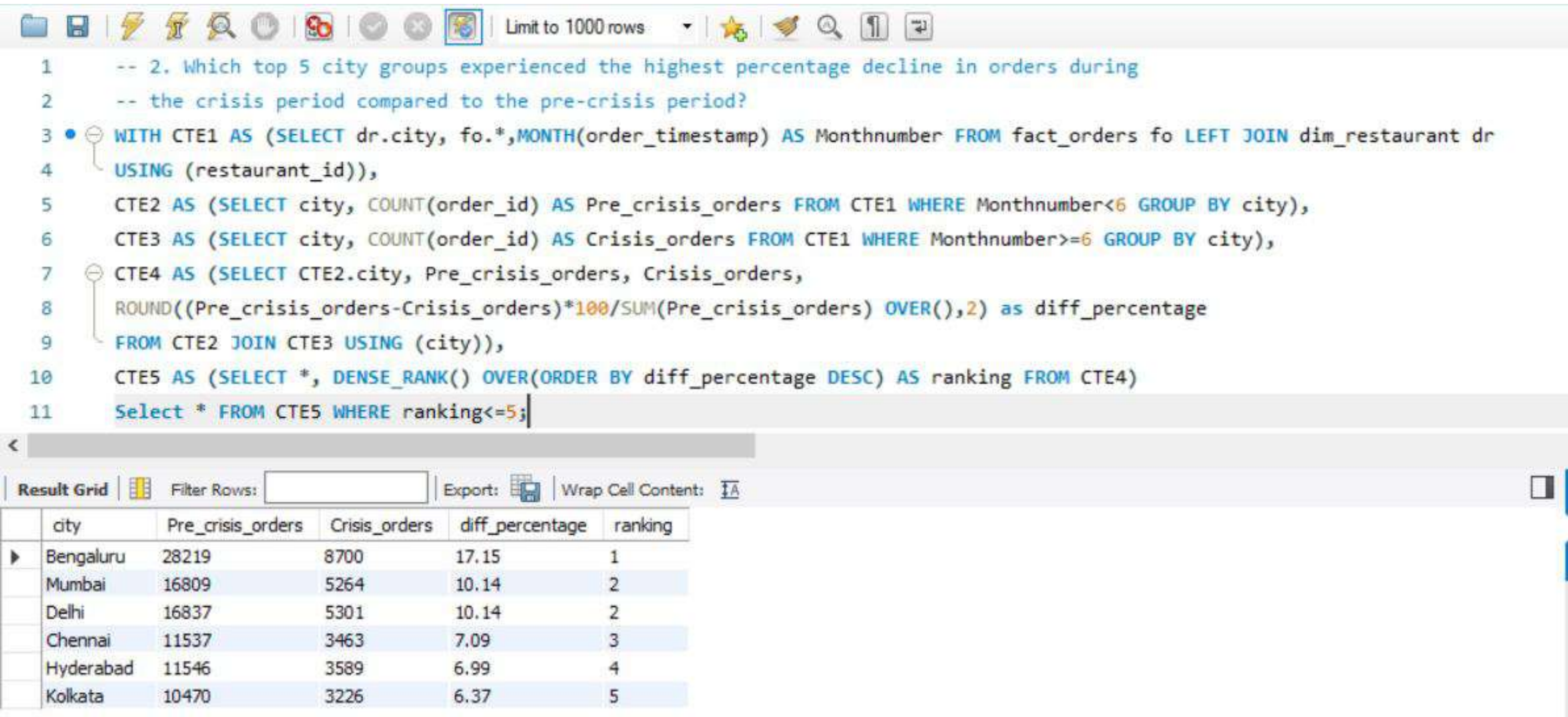


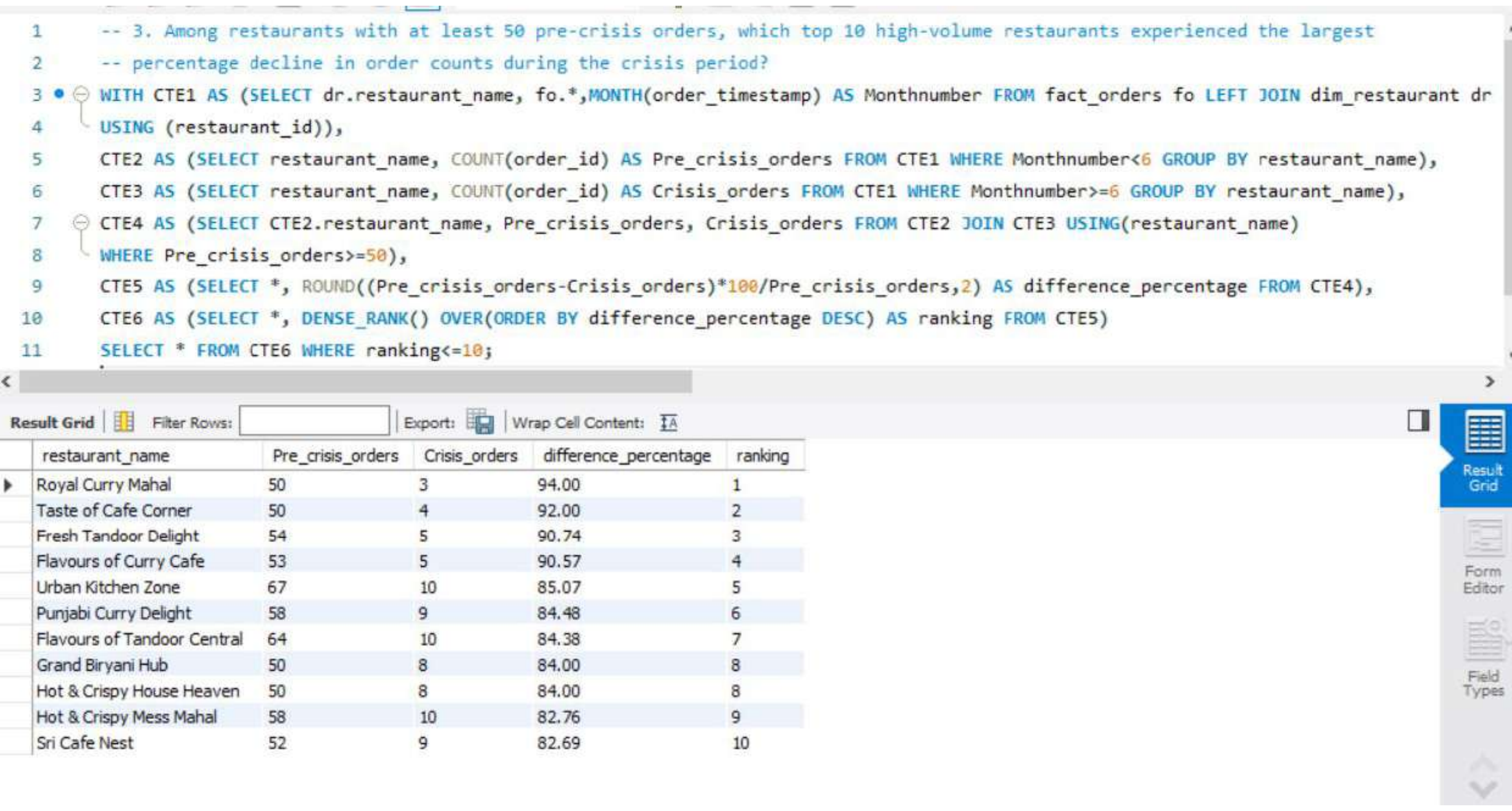
```
-- 1. Monthly Orders: Compare total orders across pre-crisis (Jan-May 2025) vs crisis (Jun-Sep 2025). How severe is the decline?
WITH CTE AS (SELECT *,MONTHNAME(order_timestamp) AS Month FROM fact_orders)
Select Month, COUNT(order_id) AS Orders FROM CTE GROUP BY Month;
```

Month	Orders
January	23539
February	22667
March	23543
April	21466
May	22591
June	9293
July	8818
August	8555
September	8694



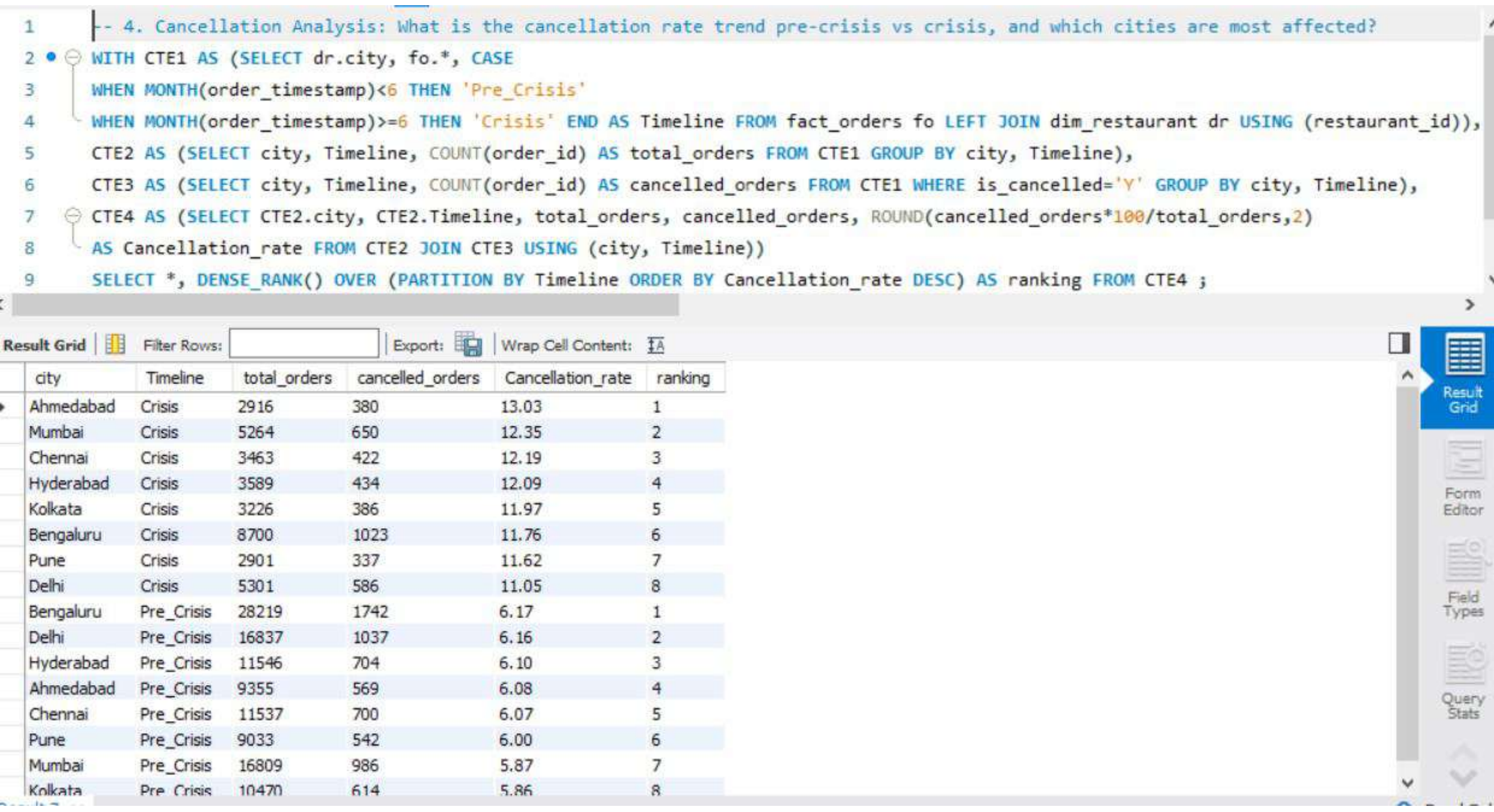
```sql
-- 2. Which top 5 city groups experienced the highest percentage decline in orders during
-- the crisis period compared to the pre-crisis period?
WITH CTE1 AS (SELECT dr.city, fo.*,MONTH(order_timestamp) AS Monthnumber FROM fact_orders fo LEFT JOIN dim_restaurant dr
USING (restaurant_id)),
CTE2 AS (SELECT city, COUNT(order_id) AS Pre_crisis_orders FROM CTE1 WHERE Monthnumber<6 GROUP BY city),
CTE3 AS (SELECT city, COUNT(order_id) AS Crisis_orders FROM CTE1 WHERE Monthnumber>=6 GROUP BY city),
CTE4 AS (SELECT CTE2.city, Pre_crisis_orders, Crisis_orders,
ROUND((Pre_crisis_orders-Crisis_orders)*100/SUM(Pre_crisis_orders) OVER(),2) as diff_percentage
FROM CTE2 JOIN CTE3 USING (city)),
CTE5 AS (SELECT *, DENSE_RANK() OVER(ORDER BY diff_percentage DESC) AS ranking FROM CTE4)
Select * FROM CTE5 WHERE ranking<=5;
```

city	Pre_crisis_orders	Crisis_orders	diff_percentage	ranking
Bengaluru	28219	8700	17.15	1
Mumbai	16809	5264	10.14	2
Delhi	16837	5301	10.14	2
Chennai	11537	3463	7.09	3
Hyderabad	11546	3589	6.99	4
Kolkata	10470	3226	6.37	5



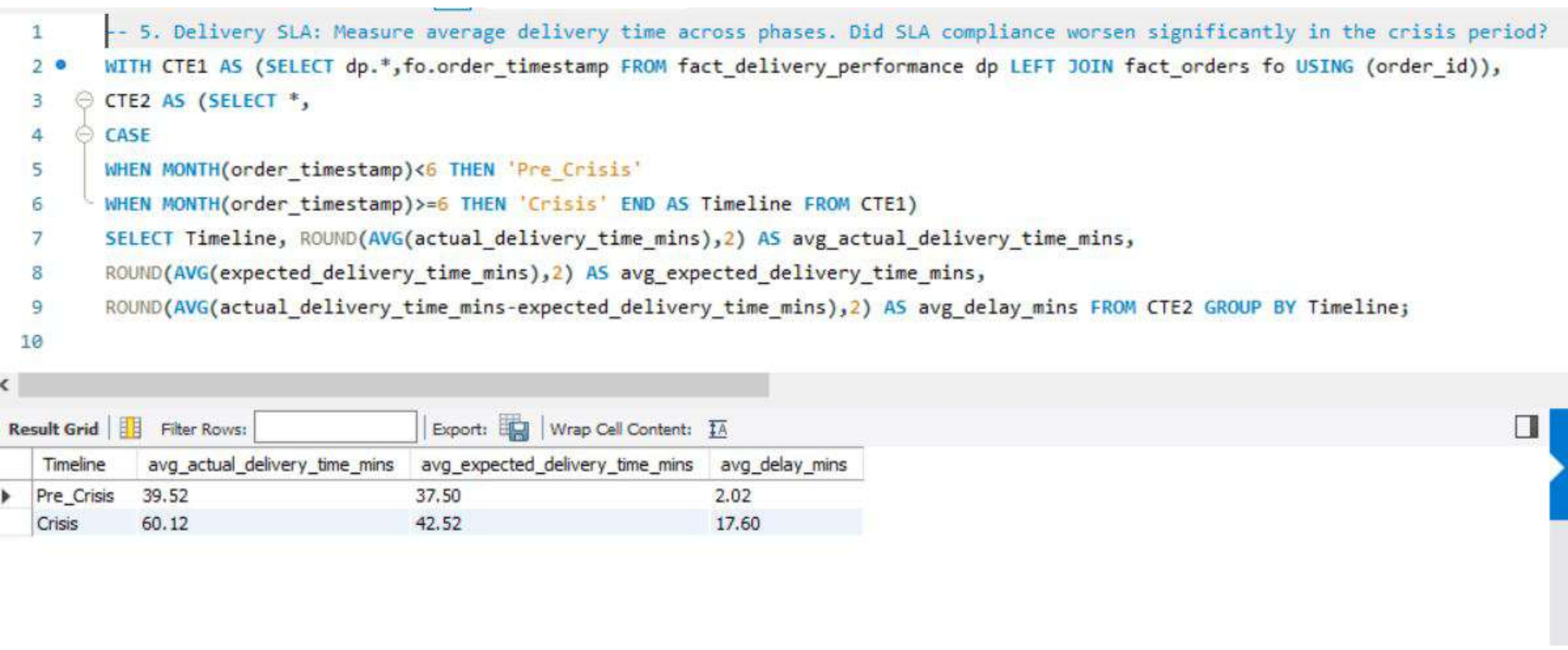
```sql
-- 3. Among restaurants with at least 50 pre-crisis orders, which top 10 high-volume restaurants experienced the largest
-- percentage decline in order counts during the crisis period?
WITH CTE1 AS (SELECT dr.restaurant_name, fo.*,MONTH(order_timestamp) AS Monthnumber FROM fact_orders fo LEFT JOIN dim_restaurant dr
USING (restaurant_id)),
CTE2 AS (SELECT restaurant_name, COUNT(order_id) AS Pre_crisis_orders FROM CTE1 WHERE Monthnumber<6 GROUP BY restaurant_name),
CTE3 AS (SELECT restaurant_name, COUNT(order_id) AS Crisis_orders FROM CTE1 WHERE Monthnumber>=6 GROUP BY restaurant_name),
CTE4 AS (SELECT CTE2.restaurant_name, Pre_crisis_orders, Crisis_orders FROM CTE2 JOIN CTE3 USING(restaurant_name)
WHERE Pre_crisis_orders>=50),
CTE5 AS (SELECT *, ROUND((Pre_crisis_orders-Crisis_orders)*100/Pre_crisis_orders,2) AS difference_percentage FROM CTE4),
CTE6 AS (SELECT *, DENSE_RANK() OVER(ORDER BY difference_percentage DESC) AS ranking FROM CTE5)
SELECT * FROM CTE6 WHERE ranking<=10;
```

restaurant_name	Pre_crisis_orders	Crisis_orders	difference_percentage	ranking
Royal Curry Mahal	50	3	94.00	1
Taste of Cafe Corner	50	4	92.00	2
Fresh Tandoor Delight	54	5	90.74	3
Flavours of Curry Cafe	53	5	90.57	4
Urban Kitchen Zone	67	10	85.07	5
Punjabi Curry Delight	58	9	84.48	6
Flavours of Tandoor Central	64	10	84.38	7
Grand Biryani Hub	50	8	84.00	8
Hot & Crispy House Heaven	50	8	84.00	8
Hot & Crispy Mess Mahal	58	10	82.76	9
Sri Cafe Nest	52	9	82.69	10



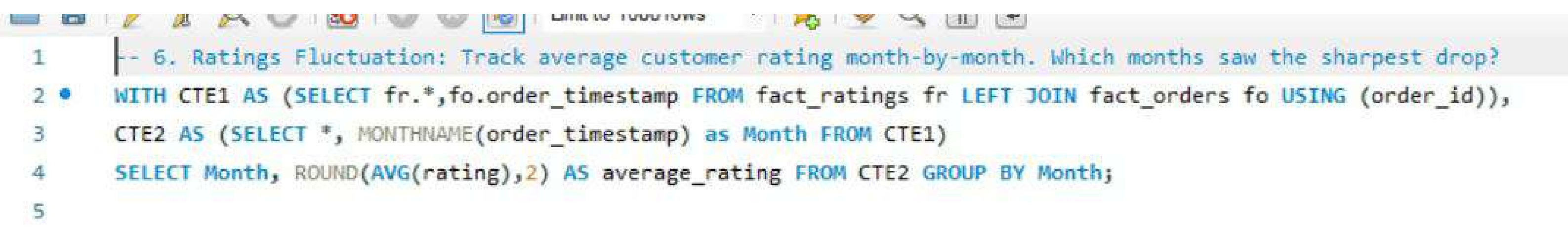
```sql
-- 4. Cancellation Analysis: What is the cancellation rate trend pre-crisis vs crisis, and which cities are most affected?
WITH CTE1 AS (SELECT dr.city, fo.*, CASE
WHEN MONTH(order_timestamp)<6 THEN 'Pre_Crisis'
WHEN MONTH(order_timestamp)>=6 THEN 'Crisis' END AS Timeline FROM fact_orders fo LEFT JOIN dim_restaurant dr USING (restaurant_id)),
CTE2 AS (SELECT city, Timeline, COUNT(order_id) AS total_orders FROM CTE1 GROUP BY city, Timeline),
CTE3 AS (SELECT city, Timeline, COUNT(order_id) AS cancelled_orders FROM CTE1 WHERE is_cancelled='Y' GROUP BY city, Timeline),
CTE4 AS (SELECT CTE2.city, CTE2.Timeline, total_orders, cancelled_orders, ROUND(cancelled_orders*100/total_orders,2)
AS Cancellation_rate FROM CTE2 JOIN CTE3 USING (city, Timeline))
SELECT *, DENSE_RANK() OVER (PARTITION BY Timeline ORDER BY Cancellation_rate DESC) AS ranking FROM CTE4 ;
```

city	Timeline	total_orders	cancelled_orders	Cancellation_rate	ranking
Ahmedabad	Crisis	2916	380	13.03	1
Mumbai	Crisis	5264	650	12.35	2
Chennai	Crisis	3463	422	12.19	3
Hyderabad	Crisis	3589	434	12.09	4
Kolkata	Crisis	3226	386	11.97	5
Bengaluru	Crisis	8700	1023	11.76	6
Pune	Crisis	2901	337	11.62	7
Delhi	Crisis	5301	586	11.05	8
Bengaluru	Pre_Crisis	28219	1742	6.17	1
Delhi	Pre_Crisis	16837	1037	6.16	2
Hyderabad	Pre_Crisis	11546	704	6.10	3
Ahmedabad	Pre_Crisis	9355	569	6.08	4
Chennai	Pre_Crisis	11537	700	6.07	5
Pune	Pre_Crisis	9033	542	6.00	6
Mumbai	Pre_Crisis	16809	986	5.87	7
Kolkata	Pre_Crisis	10470	614	5.86	8

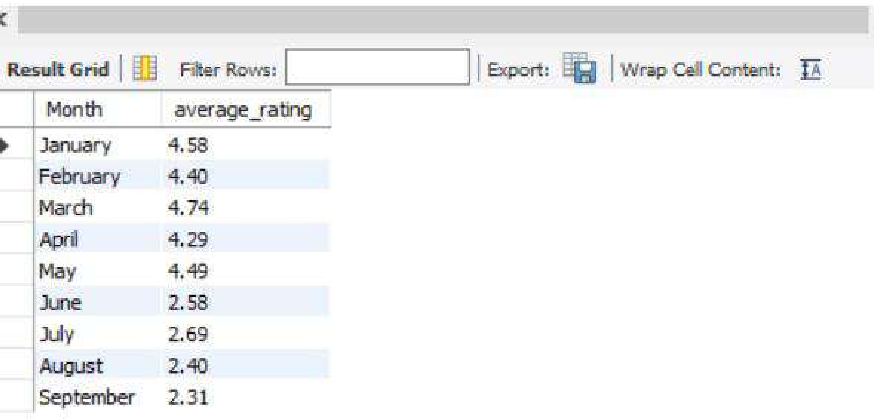


```sql
-- 5. Delivery SLA: Measure average delivery time across phases. Did SLA compliance worsen significantly in the crisis period?
WITH CTE1 AS (SELECT dp.*,fo.order_timestamp FROM fact_delivery_performance dp LEFT JOIN fact_orders fo USING (order_id)),
CTE2 AS (SELECT *,
CASE
WHEN MONTH(order_timestamp)<6 THEN 'Pre_Crisis'
WHEN MONTH(order_timestamp)>=6 THEN 'Crisis' END AS Timeline FROM CTE1)
SELECT Timeline, ROUND(AVG(actual_delivery_time_mins),2) AS avg_actual_delivery_time_mins,
ROUND(AVG(expected_delivery_time_mins),2) AS avg_expected_delivery_time_mins,
ROUND(AVG(actual_delivery_time_mins-expected_delivery_time_mins),2) AS avg_delay_mins FROM CTE2 GROUP BY Timeline;
```

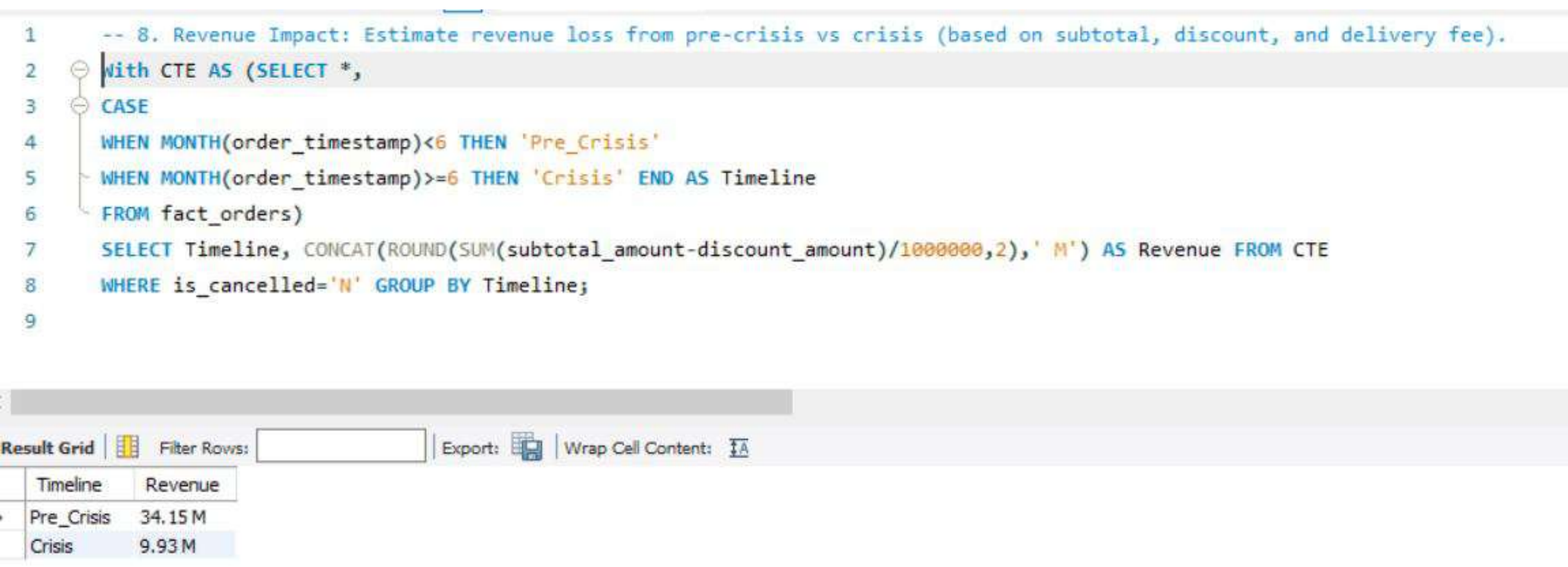
Timeline	avg_actual_delivery_time_mins	avg_expected_delivery_time_mins	avg_delay_mins
Pre_Crisis	39.52	37.50	2.02
Crisis	60.12	42.52	17.60



```sql
-- 6. Ratings Fluctuation: Track average customer rating month-by-month. Which months saw the sharpest drop?
WITH CTE1 AS (SELECT fr.*,fo.order_timestamp FROM fact_ratings fr LEFT JOIN fact_orders fo USING (order_id)),
CTE2 AS (SELECT *, MONTHNAME(order_timestamp) as Month FROM CTE1)
SELECT Month, ROUND(AVG(rating),2) AS average_rating FROM CTE2 GROUP BY Month;
```

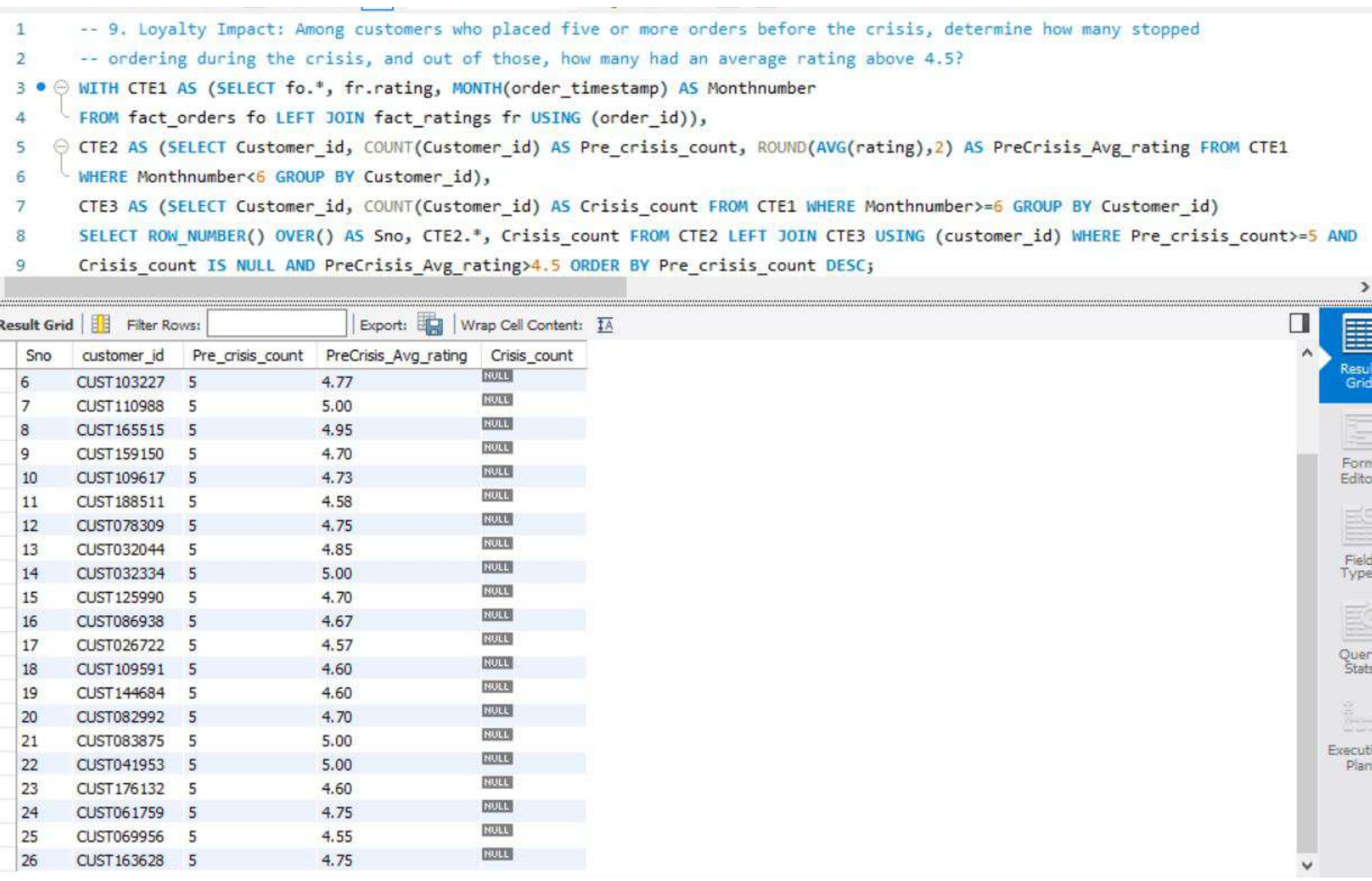


Month	average_rating
January	4.58
February	4.40
March	4.74
April	4.29
May	4.49
June	2.58
July	2.69
August	2.40
September	2.31



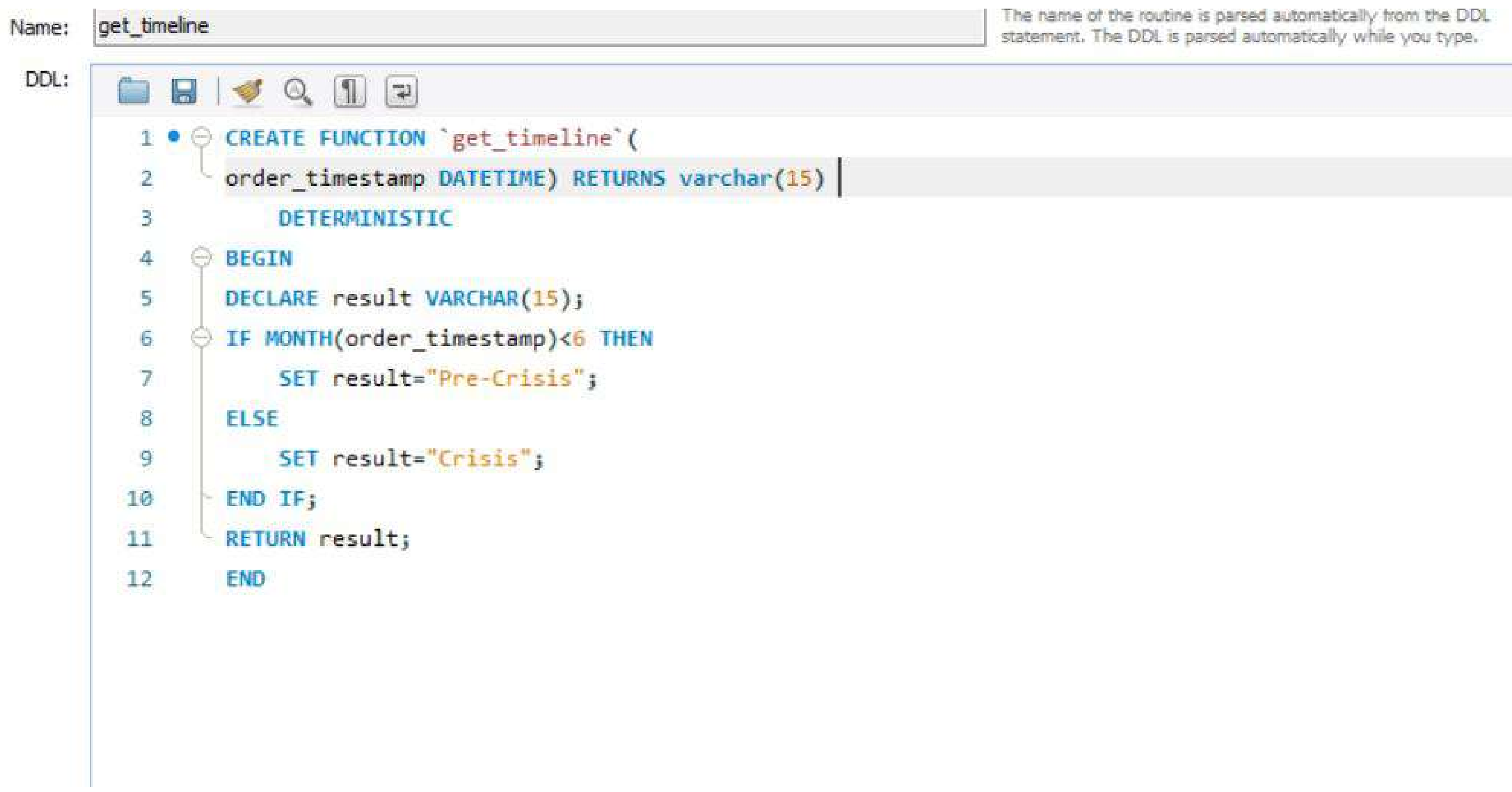
```sql
-- 8. Revenue Impact: Estimate revenue loss from pre-crisis vs crisis (based on subtotal, discount, and delivery fee).
With CTE AS (SELECT *,
CASE
WHEN MONTH(order_timestamp)<6 THEN 'Pre_Crisis'
WHEN MONTH(order_timestamp)>=6 THEN 'Crisis' END AS Timeline
FROM fact_orders)
SELECT Timeline, CONCAT(ROUND(SUM(subtotal_amount-discount_amount)/1000000,2),' M') AS Revenue FROM CTE
WHERE is_cancelled='N' GROUP BY Timeline;
```

Timeline	Revenue
Pre_Crisis	34.15 M
Crisis	9.93 M



```sql
-- 9. Loyalty Impact: Among customers who placed five or more orders before the crisis, determine how many stopped
-- ordering during the crisis, and out of those, how many had an average rating above 4.5?
WITH CTE1 AS (SELECT fo.*, fr.rating, MONTH(order_timestamp) AS Monthnumber
FROM fact_orders fo LEFT JOIN fact_ratings fr USING (order_id)),
CTE2 AS (SELECT Customer_id, COUNT(Customer_id) AS Pre_crisis_count, ROUND(AVG(rating),2) AS PreCrisis_Avg_rating FROM CTE1
WHERE Monthnumber<6 GROUP BY Customer_id),
CTE3 AS (SELECT Customer_id, COUNT(Customer_id) AS Crisis_count FROM CTE1 WHERE Monthnumber>=6 GROUP BY Customer_id)
SELECT ROW_NUMBER() OVER() AS Sno, CTE2.*, Crisis_count FROM CTE2 LEFT JOIN CTE3 USING (customer_id) WHERE Pre_crisis_count>=5 AND
Crisis_count IS NULL AND PreCrisis_Avg_rating>4.5 ORDER BY Pre_crisis_count DESC;
```

Sno	customer_id	Pre_crisis_count	PreCrisis_Avg_rating	Crisis_count
6	CUST103227	5	4.77	NULL
7	CUST110988	5	5.00	NULL
8	CUST165515	5	4.95	NULL
9	CUST159150	5	4.70	NULL
10	CUST109617	5	4.73	NULL
11	CUST188511	5	4.58	NULL
12	CUST078309	5	4.75	NULL
13	CUST032044	5	4.85	NULL
14	CUST032334	5	5.00	NULL
15	CUST125990	5	4.70	NULL
16	CUST086938	5	4.67	NULL
17	CUST026722	5	4.57	NULL
18	CUST109591	5	4.60	NULL
19	CUST144684	5	4.60	NULL
20	CUST082992	5	4.70	NULL
21	CUST083875	5	5.00	NULL
22	CUST041953	5	5.00	NULL
23	CUST176132	5	4.60	NULL
24	CUST061759	5	4.75	NULL
25	CUST069956	5	4.55	NULL
26	CUST163628	5	4.75	NULL



Name: get_timeline

The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:

```
CREATE FUNCTION `get_timeline`(
order_timestamp DATETIME) RETURNS varchar(15)
    DETERMINISTIC
BEGIN
DECLARE result VARCHAR(15);
IF MONTH(order_timestamp)<6 THEN
    SET result="Pre-Crisis";
ELSE
    SET result="Crisis";
END IF;
RETURN result;
END
```