SQL - PROJECT

Business Case - Target SQL

Made By – Sanjesh Chourasia

Submission Date – 02/06/2024

Table of Contents

Business Case: Target SQL

1 Context:

Target is a globally renowned brand and a prominent retailer in the United States. Target makes itself a preferred shopping destination by offering outstanding value, inspiration, innovation and an exceptional guest experience that no other retailer can deliver.

This particular business case focuses on the operations of Target in Brazil and provides insightful information about 100,000 orders placed between 2016 and 2018. The dataset offers a comprehensive view of various dimensions including the order status, price, payment and freight performance, customer location, product attributes, and customer reviews.

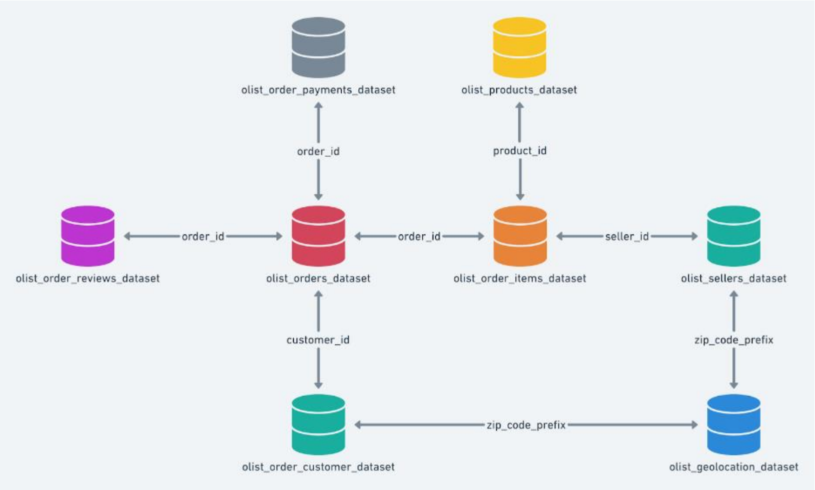
By analyzing this extensive dataset, it becomes possible to gain valuable insights into Target's operations in Brazil. The information can shed light on various aspects of the business, such as order processing, pricing strategies, payment and shipping efficiency, customer demographics, product characteristics, and customer satisfaction levels.

2 Dataset:

https://drive.google.com/drive/folders/1TGEc66YKbD443nslRi1bWgVd238gJCnb

The data is available in 8 csv files:

1. customers.csv
2. sellers.csv
3. order_items.csv
4. geolocation.csv
5. payments.csv
6. reviews.csv
7. orders.csv
8. products.cs

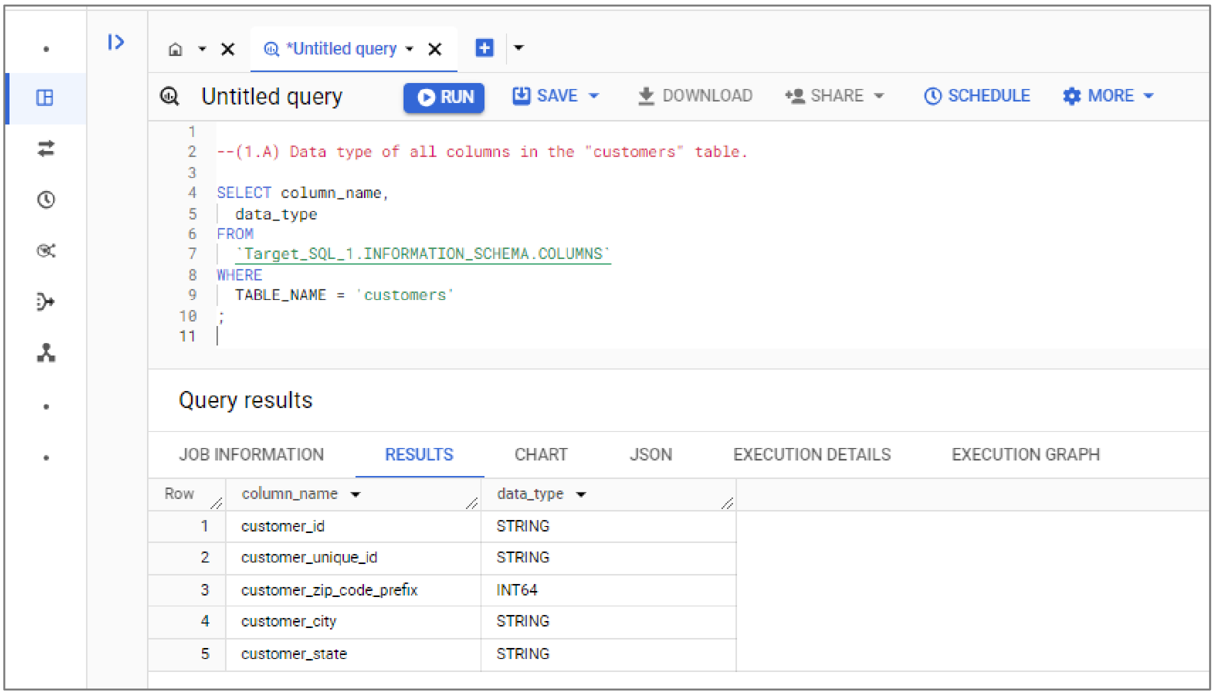


3 Questions

1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset
   a) Data type of all columns in the "customers" table.

Answer:-

SELECT column_name,
data_type
FROM
`Target_SQL_1.INFORMATION_SCHEMA.COLUMNS`
WHERE
TABLE_NAME = 'customers';



```

--(1.A) Data type of all columns in the "customers" table.

SELECT column_name,
  data_type
FROM
  `Target_SQL_1.INFORMATION_SCHEMA.COLUMNS`
WHERE
  TABLE_NAME = 'customers'
;

```

Query results

Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

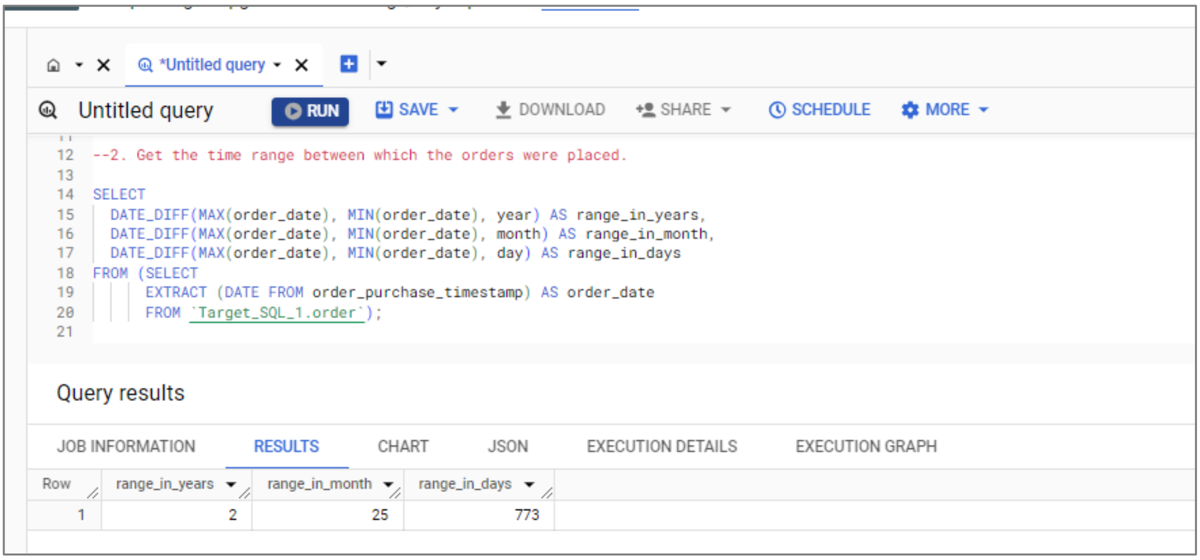
Insights:-

We checked the datatype of each column in the 'customers' table to ensure data integrity and understand the table's structure. All columns in the 'customers' table have a string data type, except for the “customer_zip_code_prefix” column, which has an integer data type.

b) Get the time range between which the orders were placed.

Answer:-

```
SELECT
  DATE_DIFF(MAX(order_date), MIN(order_date), year) AS range_in_years,
  DATE_DIFF(MAX(order_date), MIN(order_date), month) AS range_in_month,
  DATE_DIFF(MAX(order_date), MIN(order_date), day) AS range_in_days
FROM (SELECT
      EXTRACT (DATE FROM order_purchase_timestamp) AS order_date
      FROM `Target_SQL_1.order`);
```



```
--2. Get the time range between which the orders were placed.

SELECT
  DATE_DIFF(MAX(order_date), MIN(order_date), year) AS range_in_years,
  DATE_DIFF(MAX(order_date), MIN(order_date), month) AS range_in_month,
  DATE_DIFF(MAX(order_date), MIN(order_date), day) AS range_in_days
FROM (SELECT
      EXTRACT (DATE FROM order_purchase_timestamp) AS order_date
      FROM `Target_SQL_1.order`);
```

Query results

Row	range_in_years	range_in_month	range_in_days
1	2	25	773

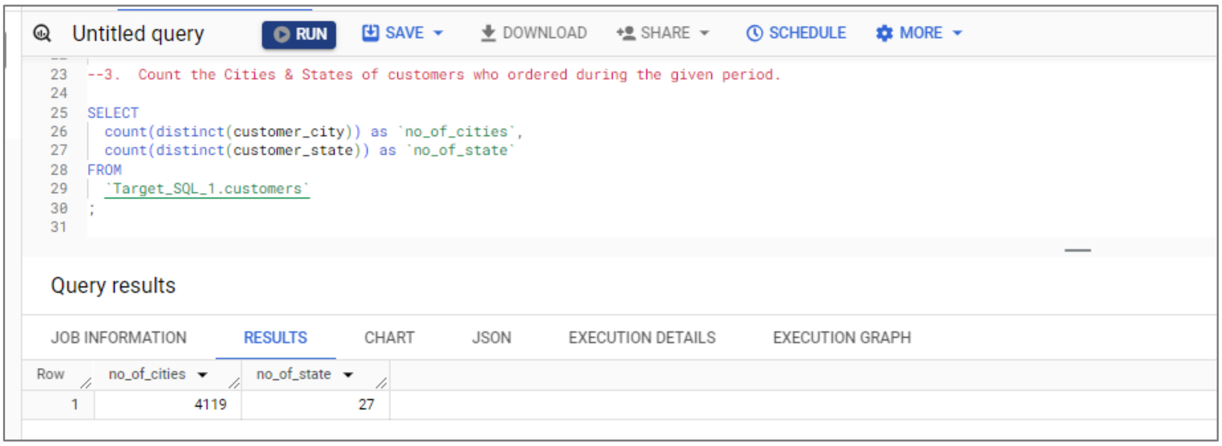
Insights:-

We checked the time period between the first and last orders placed, which spans 2 years, 25 months, or 773 days. Knowing this time range helps analyze trends, seasonality, and overall order patterns.

c) Count the Cities & States of customers who ordered during the given period.

Answer:-

```
SELECT
 count(distinct(customer_city)) as `no_of_cities`,
 count(distinct(customer_state)) as `no_of_state`
FROM
 `Target_SQL_1.customers`;
```



```
--3.  Count the Cities & States of customers who ordered during the given period.

SELECT
  count(distinct(customer_city)) as `no_of_cities`,
  count(distinct(customer_state)) as `no_of_state`
FROM
  `Target_SQL_1.customers`
;
```

Query results

Row	no_of_cities	no_of_state
1	4119	27

Insights:-

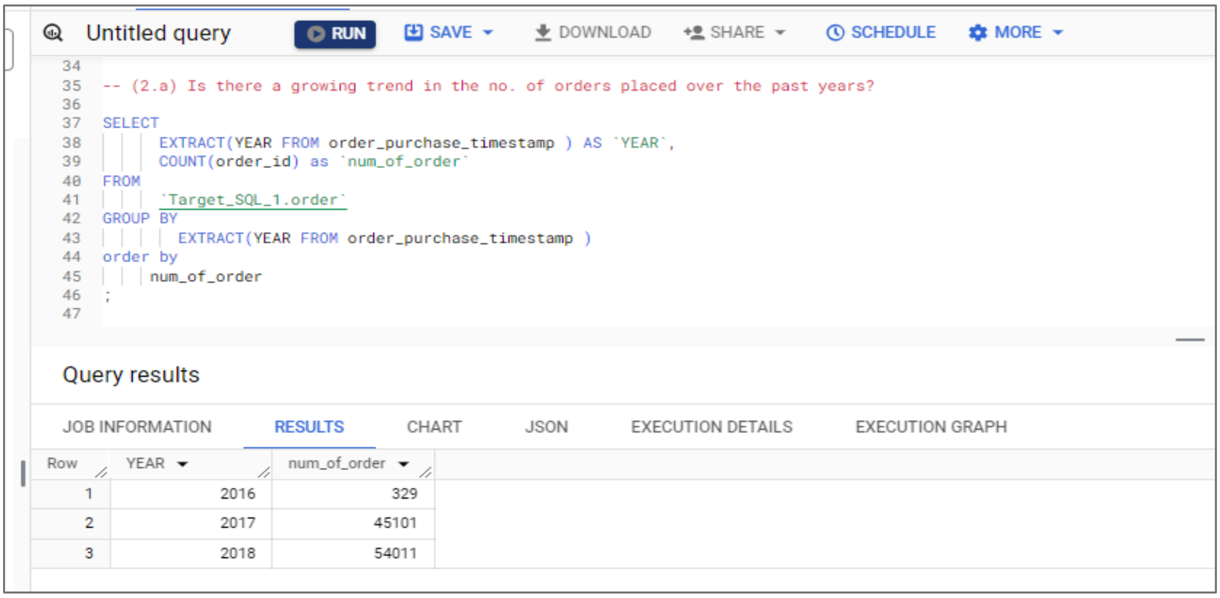
The dataset includes 27 states and 4,119 cities. Analyzing the distribution of these cities and states can provide insights into the diversity or concentration of our customers in various locations. This information can be useful for regional analysis, identifying hotspots, and understanding how far our company has spread across the nation or the globe.

2. In-depth Exploration:

a) Is there a growing trend in the no. of orders placed over the past years?

Answer:-

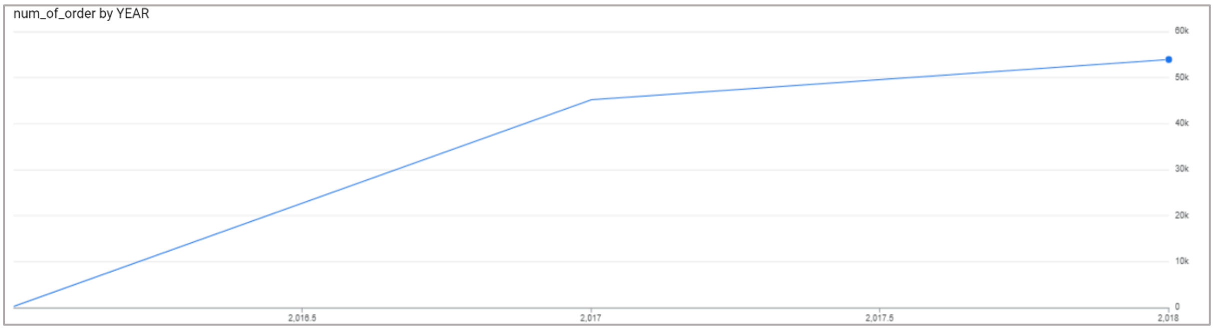
```
SELECT
     EXTRACT(YEAR FROM order_purchase_timestamp ) AS `YEAR`,
     COUNT(order_id) as `num_of_order`
FROM
     `Target_SQL_1.order`
GROUP BY
       EXTRACT(YEAR FROM order_purchase_timestamp )
order by
    num_of_order;
```



```
-- (2.a) Is there a growing trend in the no. of orders placed over the past years?

SELECT
      EXTRACT(YEAR FROM order_purchase_timestamp ) AS `YEAR`,
      COUNT(order_id) as `num_of_order`
FROM
      `Target_SQL_1.order`
GROUP BY
        EXTRACT(YEAR FROM order_purchase_timestamp )
order by
    num_of_order
;
```

Row	YEAR	num_of_order
1	2016	329
2	2017	45101
3	2018	54011



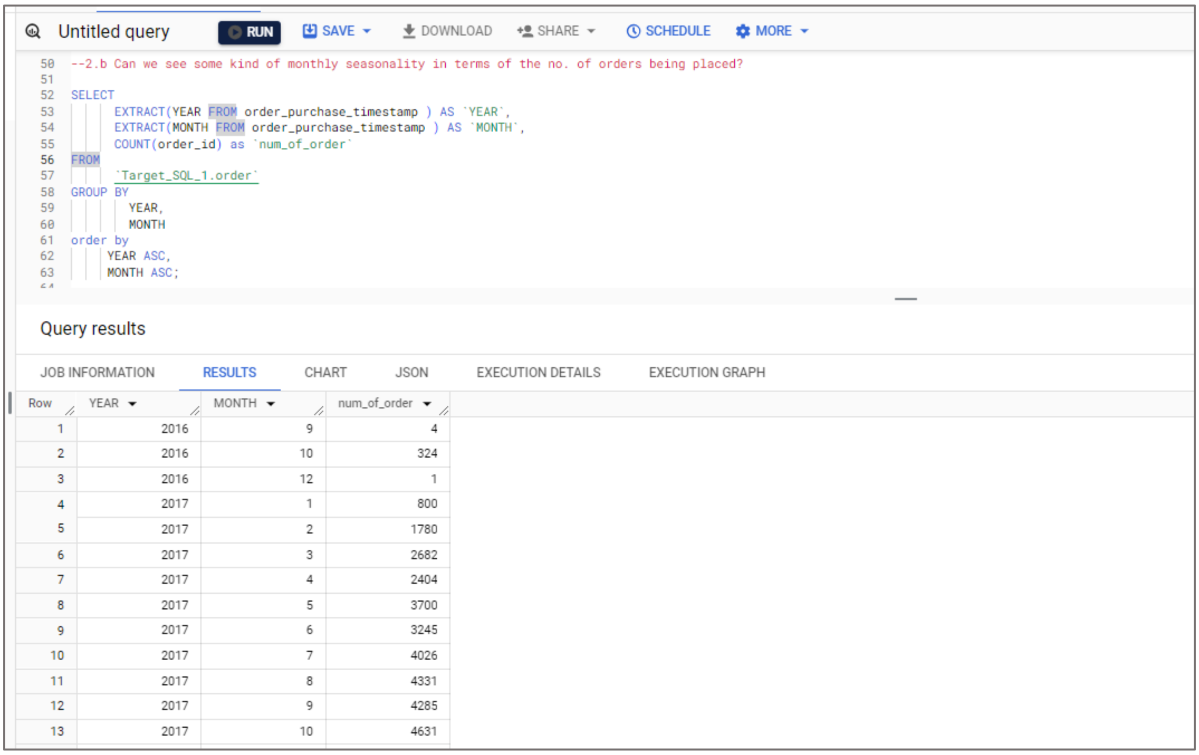
Insights:-

The number of orders has been increasing over the past few years. If the number of orders keeps going up each year, it's a positive trend. However, if there are fluctuations or a decline, it indicates a different pattern.

b) Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

Answer:-

```
SELECT
    EXTRACT(YEAR FROM order_purchase_timestamp ) AS `YEAR`,
    EXTRACT(MONTH FROM order_purchase_timestamp ) AS `MONTH`,
    COUNT(order_id) as `num_of_order`
FROM
    `Target_SQL_1.order`
GROUP BY
      YEAR,
      MONTH
order by
     YEAR ASC,
     MONTH ASC;
```

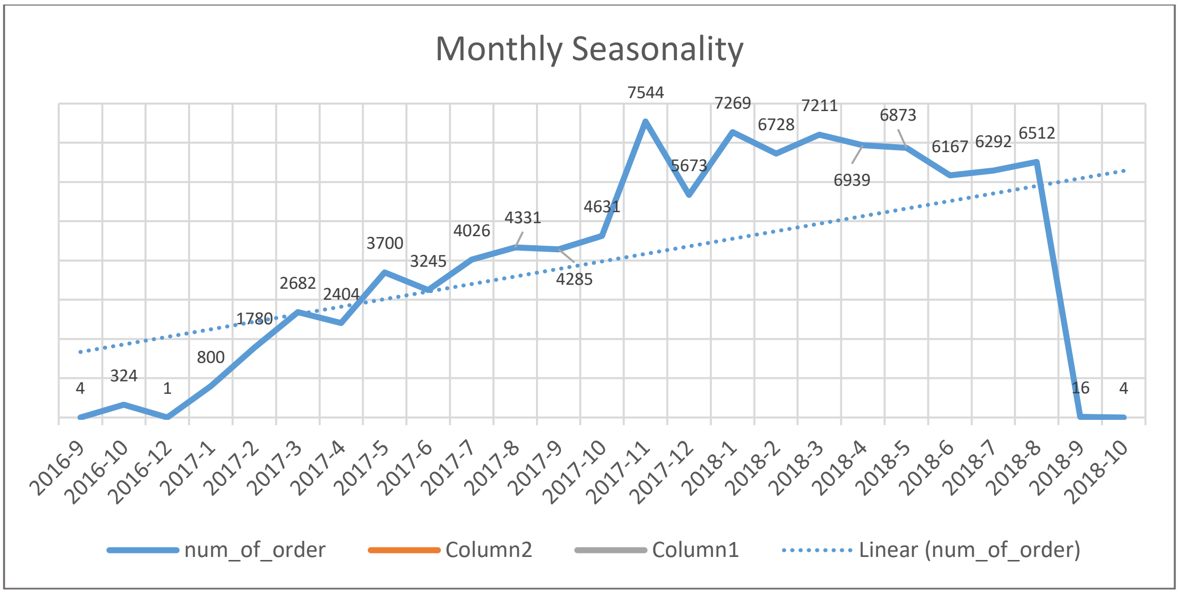


```
--2.b Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

SELECT
      EXTRACT(YEAR FROM order_purchase_timestamp ) AS `YEAR`,
      EXTRACT(MONTH FROM order_purchase_timestamp ) AS `MONTH`,
      COUNT(order_id) as `num_of_order`
FROM
      `Target_SQL_1.order`
GROUP BY
        YEAR,
        MONTH
order by
     YEAR ASC,
     MONTH ASC;
```

Query results

Row	YEAR	MONTH	num_of_order
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026
11	2017	8	4331
12	2017	9	4285
13	2017	10	4631



Insights:-

1. The query aimed to calculate the total count of orders for each month over the years to understand and manage business order patterns.
2. In November 2017, there was a big spike in orders due to Black Friday.
3. There was also growth in orders in January 2017 and January 2018, likely because of New Year's and pre-orders for Carnival in February.
4. Orders increased in the first quarter of 2018, during the FIFA World Cup.
5. Understanding these monthly trends helps with planning operations, marketing strategies, and understanding consumer behave. This can improve promotional planning, inventory management, and resource allocation during peak times.

c) During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

0-6 hrs : Dawn

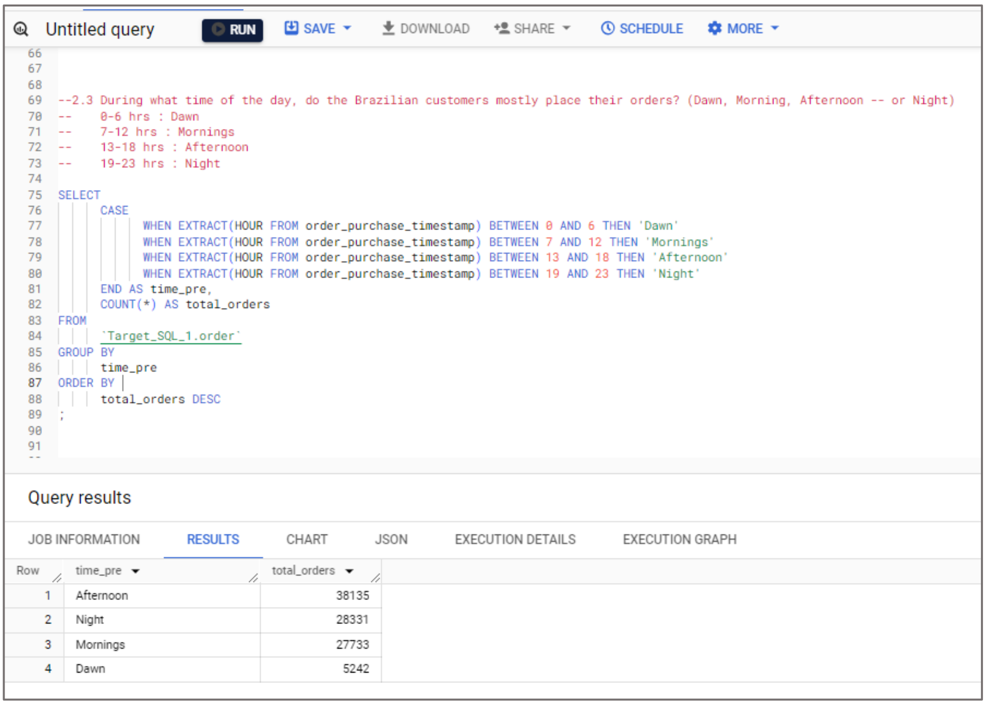
7-12 hrs : Mornings

13-18 hrs : Afternoon

19-23 hrs : Night

Answer:-

```
SELECT
    CASE
        WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 6 THEN 'Dawn'
        WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN 'Mornings'
        WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN 'Afternoon'
        WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 19 AND 23 THEN 'Night'
    END AS time_pre,
    COUNT(*) AS total_orders
FROM
    `Target_SQL_1.order`
GROUP BY
    time_pre
ORDER BY
    time_pre ASC;
```

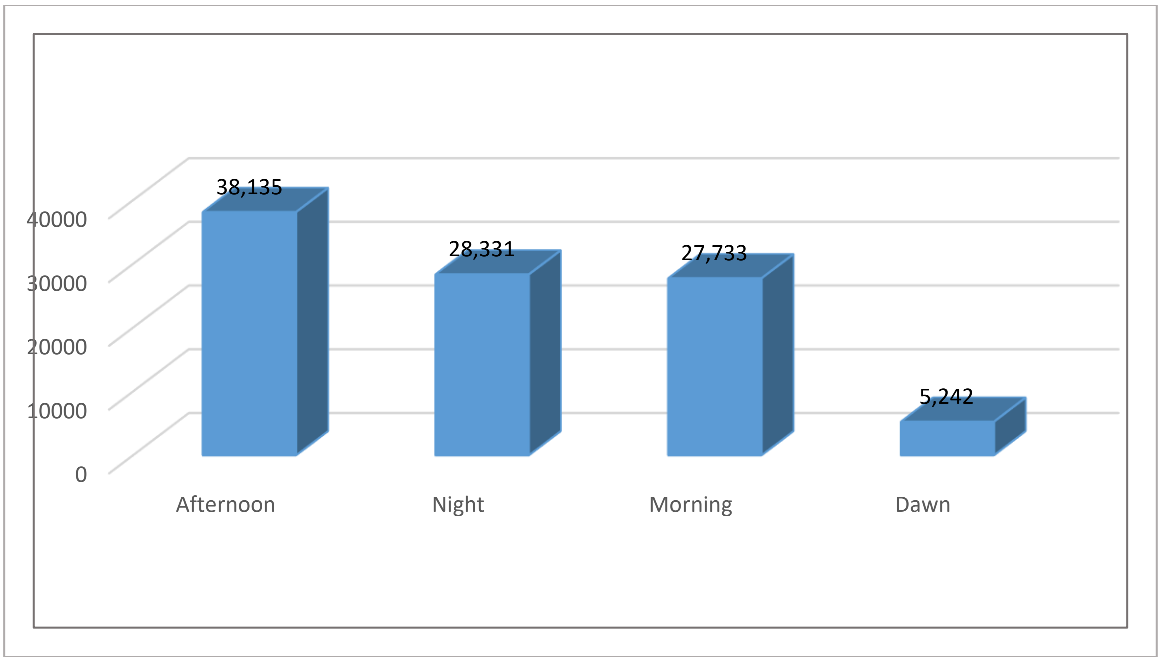


```
--2.3 During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon -- or Night)
--     0-6 hrs : Dawn
--     7-12 hrs : Mornings
--     13-18 hrs : Afternoon
--     19-23 hrs : Night

SELECT
      CASE
            WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 6 THEN 'Dawn'
            WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN 'Mornings'
            WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN 'Afternoon'
            WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 19 AND 23 THEN 'Night'
      END AS time_pre,
      COUNT(*) AS total_orders
FROM
      `Target_SQL_1.order`
GROUP BY
      time_pre
ORDER BY
      total_orders DESC
;
```

Query results

Row	time_pre	total_orders
1	Afternoon	38135
2	Night	28331
3	Mornings	27733
4	Dawn	5242



Insights:-

Objective:

- Our objective is to find the best time for order placement and analyzed the results using charts.

Order Timing Insights:

- Brazilian customers place the most orders in the afternoon, followed by night.
- Mornings also see a substantial number of orders.
- Dawn has the fewest orders.

Customer Behaviour:

- Brazilians are most active in online shopping during the afternoon and evening, likely during leisure time.
- Mornings also show significant shopping activity.

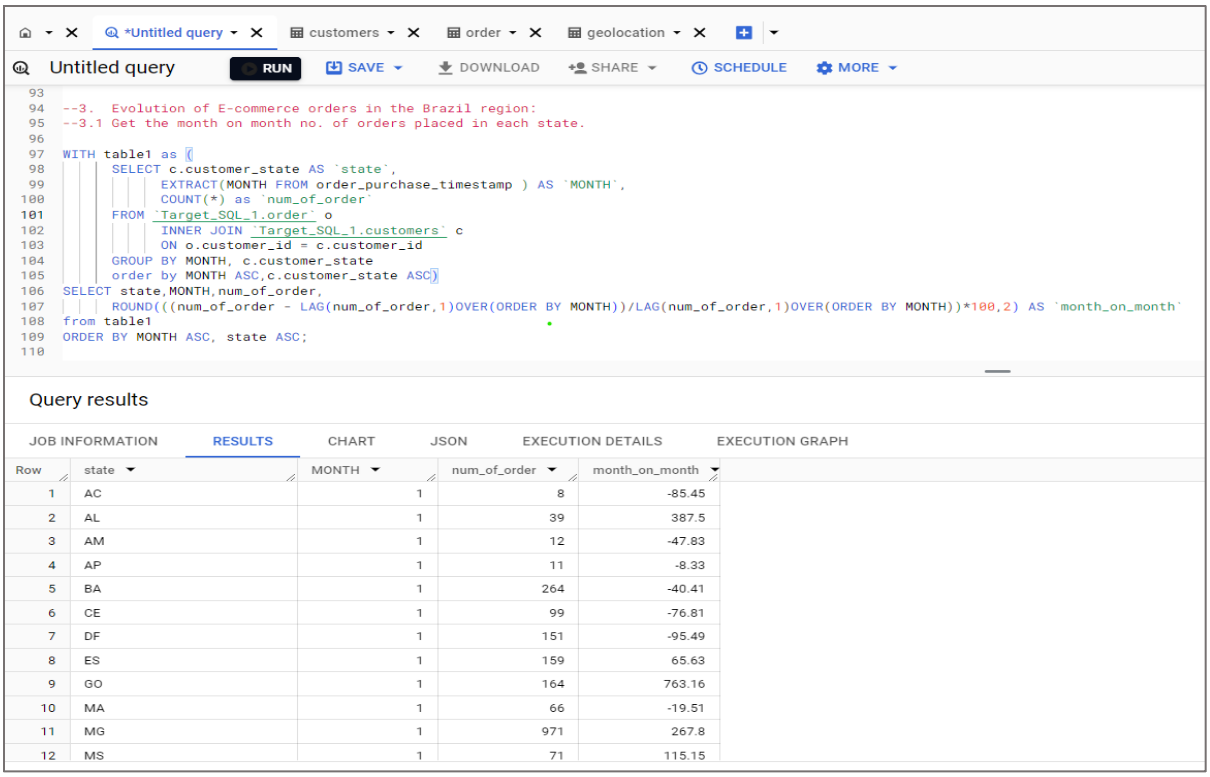
3. Evolution of E-commerce orders in the Brazil region:

a) Get the month on month no. of orders placed in each state.

Answer:-

```
WITH table1 as (
    SELECT c.customer_state AS `state`,
        EXTRACT(MONTH FROM order_purchase_timestamp ) AS `MONTH`,
        COUNT(*) as `num_of_order`
    FROM `Target_SQL_1.order` o
        INNER JOIN `Target_SQL_1.customers` c
        ON o.customer_id = c.customer_id
    GROUP BY MONTH, c.customer_state
    order by MONTH ASC,c.customer_state ASC)

SELECT state,MONTH,num_of_order,
    ROUND(((num_of_order - LAG(num_of_order,1)OVER(ORDER BY
MONTH))/LAG(num_of_order,1)OVER(ORDER BY MONTH))*100,2) AS `month_on_month`
from table1
ORDER BY MONTH ASC, state ASC;
```

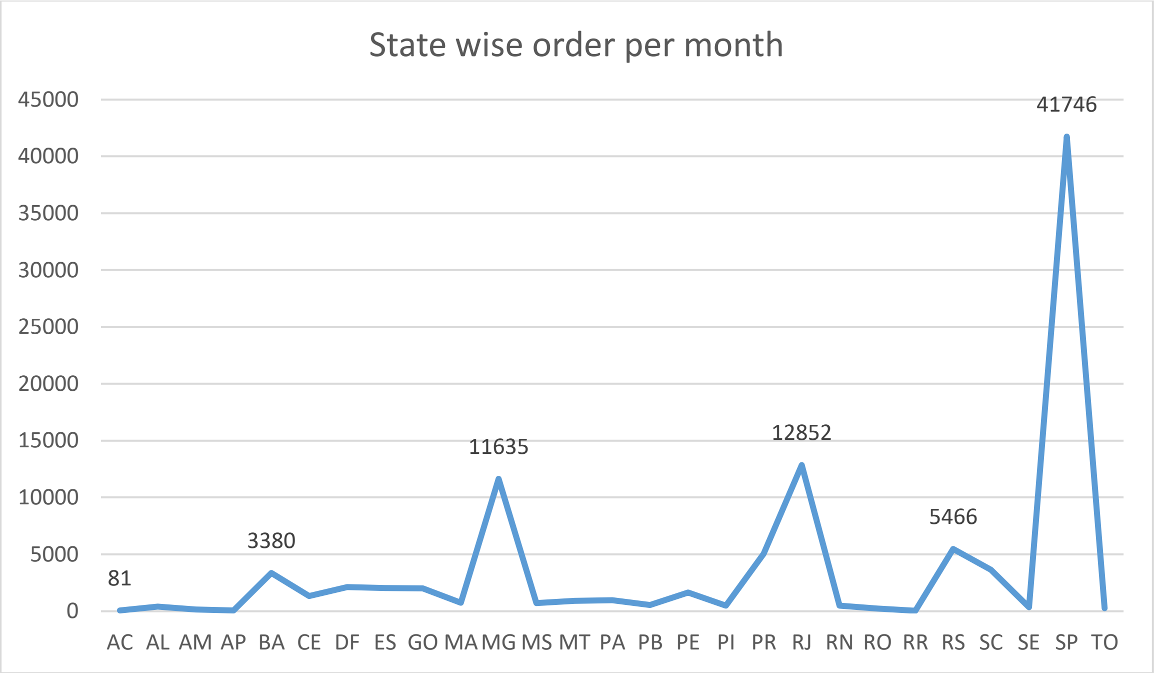


```
--3.  Evolution of E-commerce orders in the Brazil region:
--3.1 Get the month on month no. of orders placed in each state.

WITH table1 as (
    SELECT c.customer_state AS `state`,
        EXTRACT(MONTH FROM order_purchase_timestamp ) AS `MONTH`,
        COUNT(*) as `num_of_order`
    FROM `Target_SQL_1.order` o
        INNER JOIN `Target_SQL_1.customers` c
        ON o.customer_id = c.customer_id
    GROUP BY MONTH, c.customer_state
    order by MONTH ASC,c.customer_state ASC)
SELECT state,MONTH,num_of_order,
    ROUND(((num_of_order - LAG(num_of_order,1)OVER(ORDER BY MONTH))/LAG(num_of_order,1)OVER(ORDER BY MONTH))*100,2) AS `month_on_month`
from table1
ORDER BY MONTH ASC, state ASC;
```

Query results

Row	state	MONTH	num_of_order	month_on_month
1	AC	1	8	-85.45
2	AL	1	39	387.5
3	AM	1	12	-47.83
4	AP	1	11	-8.33
5	BA	1	264	-40.41
6	CE	1	99	-76.81
7	DF	1	151	-95.49
8	ES	1	159	65.63
9	GO	1	164	763.16
10	MA	1	66	-19.51
11	MG	1	971	267.8
12	MS	1	71	115.15



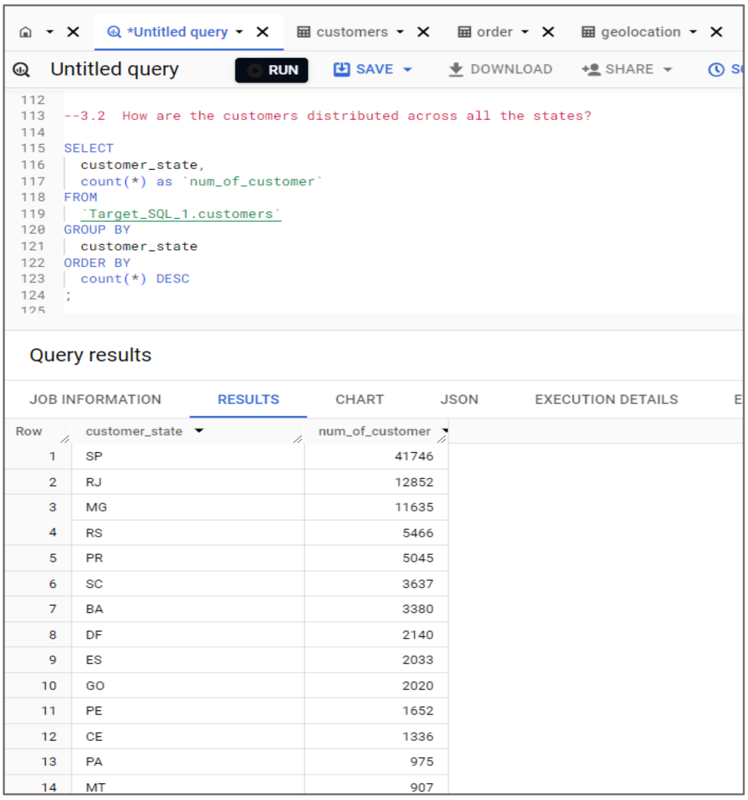
Insights:-

- The query provides information on the monthly order count for each state.
- Comparing order counts with minimum and maximum values helps identify states with consistent high or low order numbers. This helps prioritize areas for targeted strategies to improve orders. For example, in our data, the state "SP" consistently has the highest number of orders every month.
- State-wise average order counts show overall trends, highlighting states with steady or increasing order numbers. This information identifies opportunities for growth and effective marketing strategies in different regions.
- Analyzing this data enables data-driven decision-making. It helps in production planning, considering seasonality, state-wise performance, and growth trends. This optimization helps in marketing, operations, and resource allocation strategies across different states in Brazil.

b) How are the customers distributed across all the states?

Answer:-

```
SELECT
   customer_state,
   count(*) as `num_of_customer`
FROM
  `Target_SQL_1.customers`
GROUP BY
   customer_state
ORDER BY
   count(*) DESC;
```



```
--3.2  How are the customers distributed across all the states?

SELECT
  customer_state,
  count(*) as `num_of_customer`
FROM
  `Target_SQL_1.customers`
GROUP BY
  customer_state
ORDER BY
  count(*) DESC
;
```

Query results

Row	customer_state	num_of_customer
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020
11	PE	1652
12	CE	1336
13	PA	975
14	MT	907

Insights:-

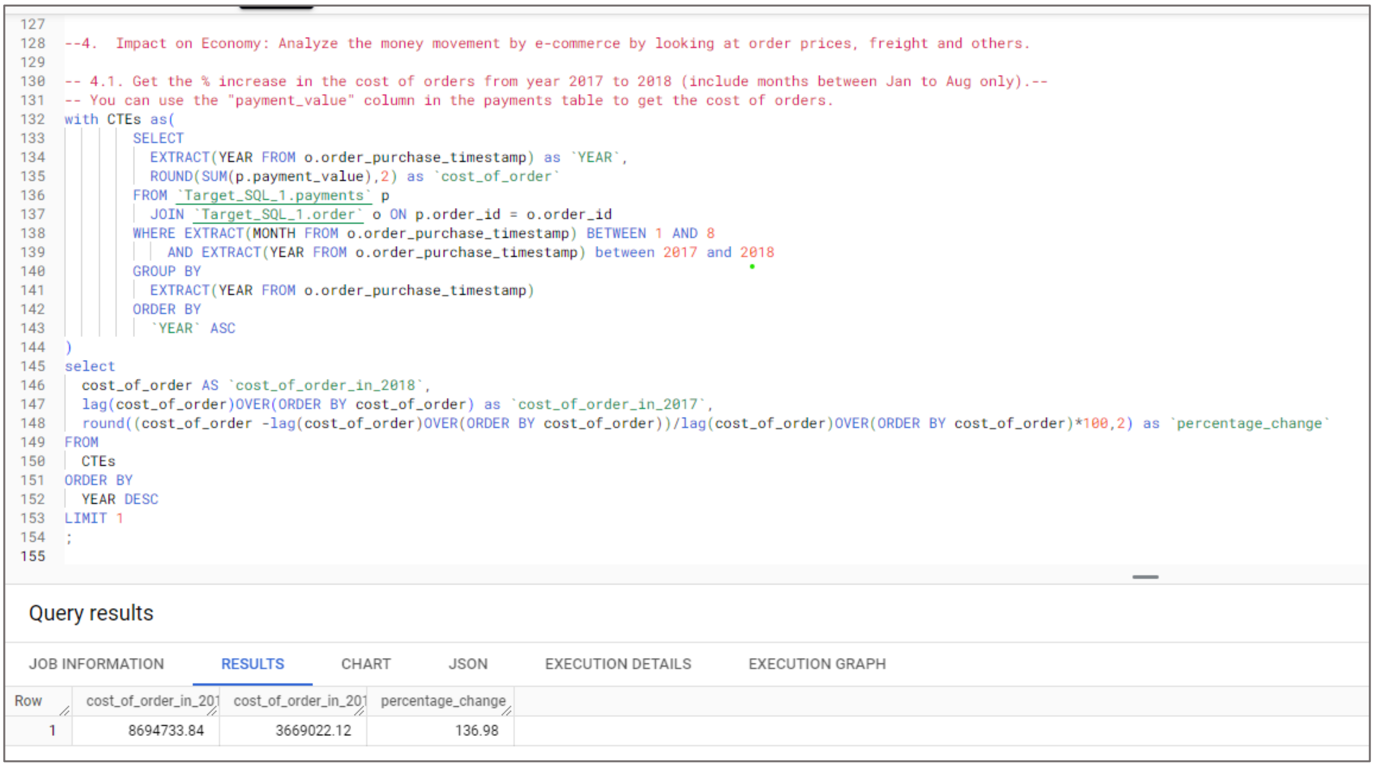
Analyzing the query results will show how customers are distributed across states, indicating which states have the most and fewest customers. For example, the state "SP" has the highest number of customers, while "RR" has the fewest. This information is useful for market targeting, identifying expansion opportunities, and improving customer service. By understanding the distribution of our client base, we can identify areas for potential growth and make strategic decisions to optimize our company strategy.

4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

a) Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).
You can use the "payment_value" column in the payments table to get the cost of orders.

Answer:-

```
with CTEs as(
	SELECT
	  EXTRACT(YEAR FROM o.order_purchase_timestamp) as `YEAR`,
	  ROUND(SUM(p.payment_value),2) as `cost_of_order`
	FROM `Target_SQL_1.payments` p
	  JOIN `Target_SQL_1.order` o ON p.order_id = o.order_id
	WHERE EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
	    AND EXTRACT(YEAR FROM o.order_purchase_timestamp) between 2017 and 2018
	GROUP BY
	  EXTRACT(YEAR FROM o.order_purchase_timestamp)
	ORDER BY
	  `YEAR` ASC
)
select
  cost_of_order AS `cost_of_order_in_2018`,
  lag(cost_of_order)OVER(ORDER BY cost_of_order) as `cost_of_order_in_2017`,
  round((cost_of_order -lag(cost_of_order)OVER(ORDER BY
cost_of_order))/lag(cost_of_order)OVER(ORDER BY cost_of_order)*100,2) as `percentage_change`
FROM
  CTEs
ORDER BY
  YEAR DESC
LIMIT 1
;
```



```
--4.  Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

-- 4.1. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).--
-- You can use the "payment_value" column in the payments table to get the cost of orders.
with CTEs as(
        SELECT
          EXTRACT(YEAR FROM o.order_purchase_timestamp) as `YEAR`,
          ROUND(SUM(p.payment_value),2) as `cost_of_order`
        FROM `Target_SQL_1.payments` p
          JOIN `Target_SQL_1.order` o ON p.order_id = o.order_id
        WHERE EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
            AND EXTRACT(YEAR FROM o.order_purchase_timestamp) between 2017 and 2018
        GROUP BY
          EXTRACT(YEAR FROM o.order_purchase_timestamp)
        ORDER BY
          `YEAR` ASC
)
select
  cost_of_order AS `cost_of_order_in_2018`,
  lag(cost_of_order)OVER(ORDER BY cost_of_order) as `cost_of_order_in_2017`,
  round((cost_of_order -lag(cost_of_order)OVER(ORDER BY cost_of_order))/lag(cost_of_order)OVER(ORDER BY cost_of_order)*100,2) as `percentage_change`
FROM
  CTEs
ORDER BY
  YEAR DESC
LIMIT 1
;
```

Query results

Row	cost_of_order_in_201	cost_of_order_in_201	percentage_change
1	8694733.84	3669022.12	136.98

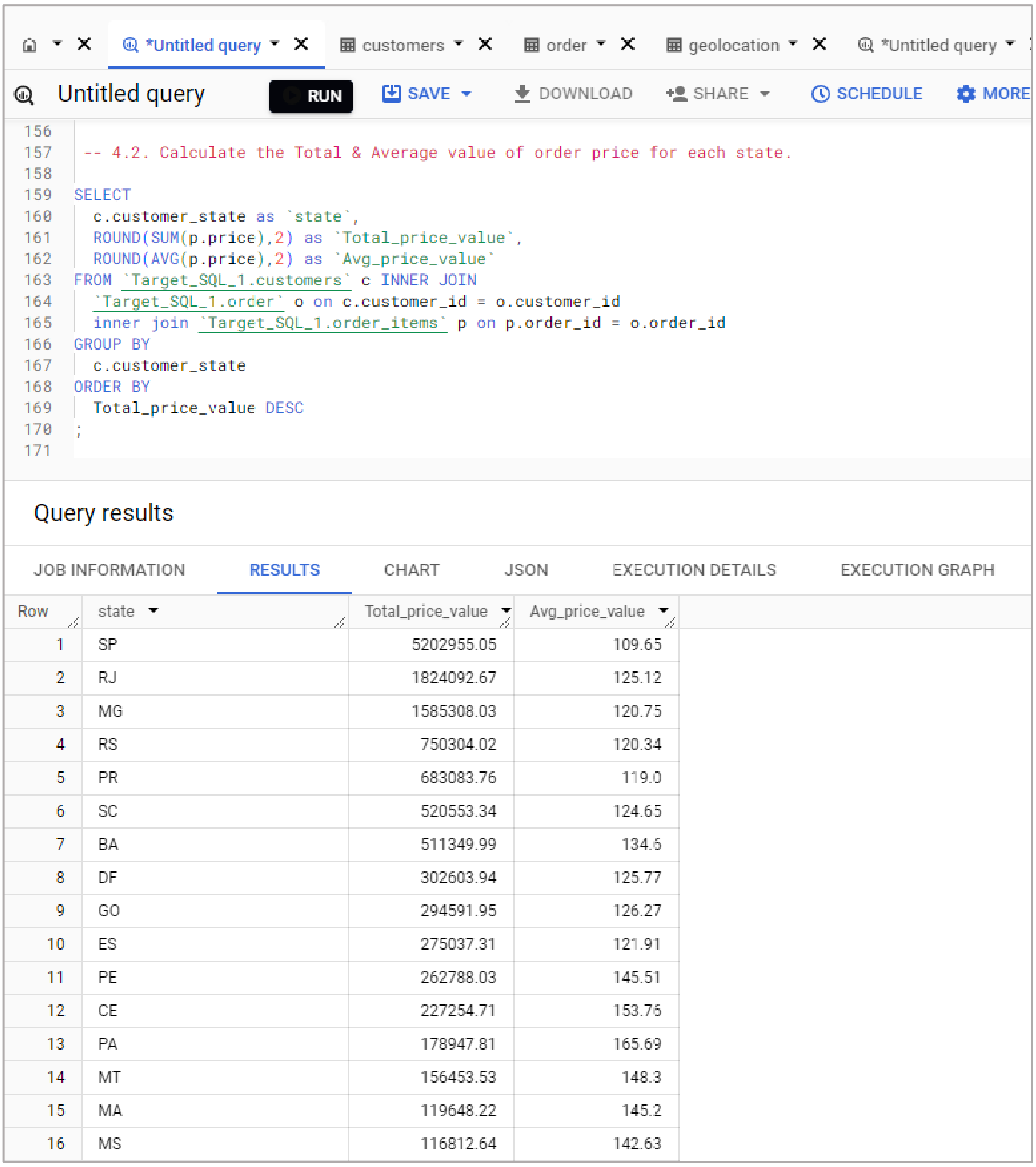
Insights:-

- Only orders made from January to August in both 2017 and 2018 are included.
- The query calculates the percentage increase by analyzing monthly prices from 2017 to 2018.
- cost_of_order_in_2018 is 86,94,733.84 and cost_of_order_in_2017 is 3669022.12
- The results show a growth rate of around 136.98% from 2017 to 2018.

b) Calculate the Total & Average value of order price for each state.

Answer:-

```
SELECT
  c.customer_state as `state`,
  ROUND(SUM(p.price),2) as `Total_price_value`,
  ROUND(AVG(p.price),2) as `Avg_price_value`
FROM `Target_SQL_1.customers` c INNER JOIN
  `Target_SQL_1.order` o on c.customer_id = o.customer_id
  inner join `Target_SQL_1.order_items` p on p.order_id = o.order_id
GROUP BY
  c.customer_state
ORDER BY
  Total_price_value DESC;
```



```

 -- 4.2. Calculate the Total & Average value of order price for each state.

SELECT
  c.customer_state as `state`,
  ROUND(SUM(p.price),2) as `Total_price_value`,
  ROUND(AVG(p.price),2) as `Avg_price_value`
FROM `Target_SQL_1.customers` c INNER JOIN
  `Target_SQL_1.order` o on c.customer_id = o.customer_id
  inner join `Target_SQL_1.order_items` p on p.order_id = o.order_id
GROUP BY
  c.customer_state
ORDER BY
  Total_price_value DESC
;

```

Row	state	Total_price_value	Avg_price_value
1	SP	5202955.05	109.65
2	RJ	1824092.67	125.12
3	MG	1585308.03	120.75
4	RS	750304.02	120.34
5	PR	683083.76	119.0
6	SC	520553.34	124.65
7	BA	511349.99	134.6
8	DF	302603.94	125.77
9	GO	294591.95	126.27
10	ES	275037.31	121.91
11	PE	262788.03	145.51
12	CE	227254.71	153.76
13	PA	178947.81	165.69
14	MT	156453.53	148.3
15	MA	119648.22	145.2
16	MS	116812.64	142.63

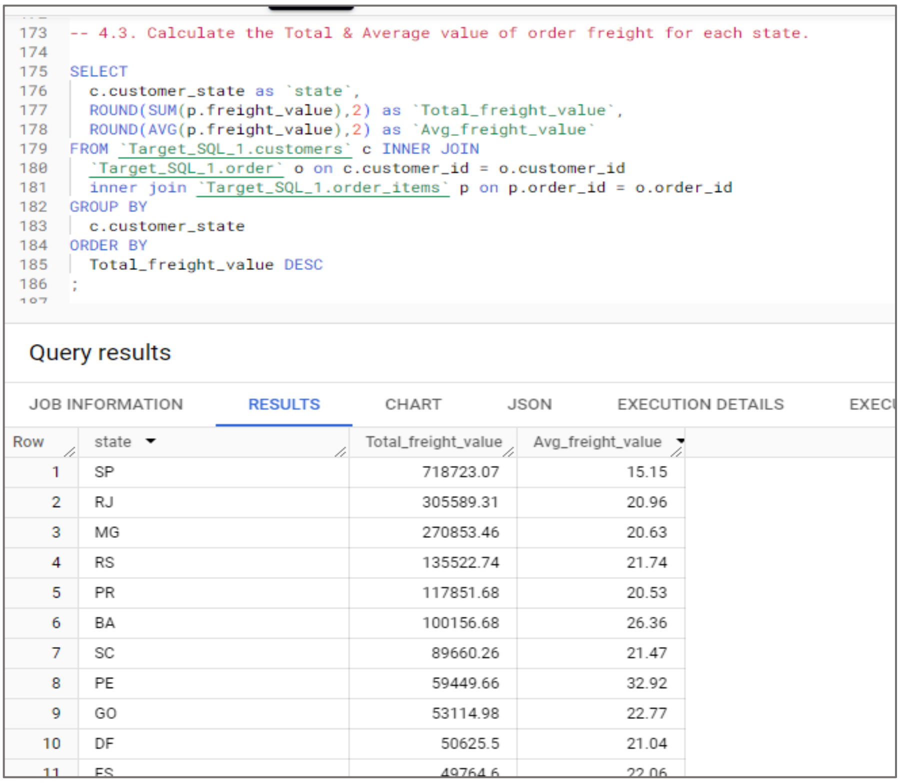
Insights:-

- The "Total_price_value" column shows the total amount of orders placed in each state.
- The "avg_price_value" column shows the average order value for each state.
- Analyzing these results can identify states with large Total_price_value, indicating profitable markets.
- Comparing avg_price_value across states can help develop targeted marketing or pricing strategies by highlighting areas with higher or lower spending.
- To gain deeper insights and make informed decisions, consider each state's context, such as population, economic factors, or customer behaviour

c) Calculate the Total & Average value of order freight for each state.

Answer:-

```
SELECT
  c.customer_state as `state`,
  ROUND(SUM(p.freight_value),2) as `Total_freight_value`,
  ROUND(AVG(p.freight_value),2) as `Avg_freight_value`
FROM `Target_SQL_1.customers` c INNER JOIN
  `Target_SQL_1.order` o on c.customer_id = o.customer_id
  inner join `Target_SQL_1.order_items` p on p.order_id = o.order_id
GROUP BY
  c.customer_state
ORDER BY
  Total_freight_value DESC;
```



```
-- 4.3. Calculate the Total & Average value of order freight for each state.

SELECT
  c.customer_state as `state`,
  ROUND(SUM(p.freight_value),2) as `Total_freight_value`,
  ROUND(AVG(p.freight_value),2) as `Avg_freight_value`
FROM `Target_SQL_1.customers` c INNER JOIN
  `Target_SQL_1.order` o on c.customer_id = o.customer_id
  inner join `Target_SQL_1.order_items` p on p.order_id = o.order_id
GROUP BY
  c.customer_state
ORDER BY
  Total_freight_value DESC
;
```

Query results

Row	state	Total_freight_value	Avg_freight_value
1	SP	718723.07	15.15
2	RJ	305589.31	20.96
3	MG	270853.46	20.63
4	RS	135522.74	21.74
5	PR	117851.68	20.53
6	BA	100156.68	26.36
7	SC	89660.26	21.47
8	PE	59449.66	32.92
9	GO	53114.98	22.77
10	DF	50625.5	21.04
11	ES	49764.6	22.06

Insights:-

- By analyzing the results, we can find states with high total freight costs, such as the state "SP," indicating regions with higher shipping prices or logistical challenges.
- Comparing average order freight costs across states can help identify regions with higher or lower shipping prices, which is useful for optimizing logistics operations or pricing strategies.
- Understanding the differences in freight rates between states provides insights into local shipping habits, supplier locations, or client preferences, helping to optimize processes and reduce costs.

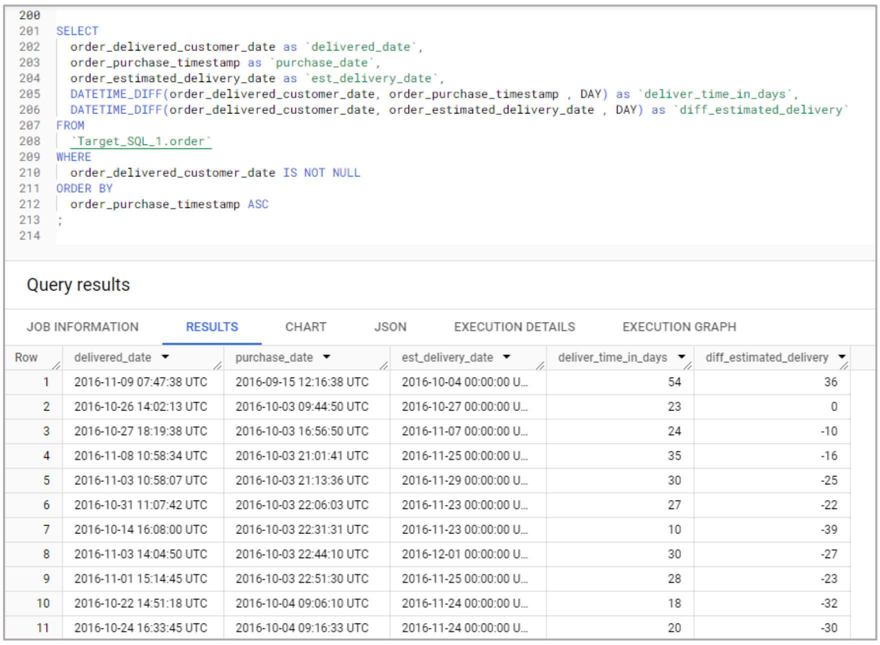
5. **Analysis based on sales, freight and delivery time.**

    a) **Find the no. of days taken to deliver each order from the order's purchase date as delivery time.**
    Also, calculate the difference (in days) between the estimated & actual delivery date of an order.
    Do this in a single query.
    You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

- **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
- **diff_estimated_delivery** = order_delivered_customer_date - order_estimated_delivery_date

Answer:-

```
SELECT
  order_delivered_customer_date as `delivered_date`,
  order_purchase_timestamp as `purchase_date`,
  order_estimated_delivery_date as `est_delivery_date`,
  DATETIME_DIFF(order_delivered_customer_date, order_purchase_timestamp , DAY) as
`deliver_time_in_days`,
  DATETIME_DIFF(order_delivered_customer_date, order_estimated_delivery_date , DAY) as
`diff_estimated_delivery`
FROM
  `Target_SQL_1.order`
WHERE
  order_delivered_customer_date IS NOT NULL
ORDER BY
  order_purchase_timestamp ASC;
```



```

SELECT
  order_delivered_customer_date as `delivered_date`,
  order_purchase_timestamp as `purchase_date`,
  order_estimated_delivery_date as `est_delivery_date`,
  DATETIME_DIFF(order_delivered_customer_date, order_purchase_timestamp , DAY) as `deliver_time_in_days`,
  DATETIME_DIFF(order_delivered_customer_date, order_estimated_delivery_date , DAY) as `diff_estimated_delivery`
FROM
  `Target_SQL_1.order`
WHERE
  order_delivered_customer_date IS NOT NULL
ORDER BY
  order_purchase_timestamp ASC
;

```

Query results

Row	delivered_date	purchase_date	est_delivery_date	deliver_time_in_days	diff_estimated_delivery
1	2016-11-09 07:47:38 UTC	2016-09-15 12:16:38 UTC	2016-10-04 00:00:00 U...	54	36
2	2016-10-26 14:02:13 UTC	2016-10-03 09:44:50 UTC	2016-10-27 00:00:00 U...	23	0
3	2016-10-27 18:19:38 UTC	2016-10-03 16:56:50 UTC	2016-11-07 00:00:00 U...	24	-10
4	2016-11-08 10:58:34 UTC	2016-10-03 21:01:41 UTC	2016-11-25 00:00:00 U...	35	-16
5	2016-11-03 10:58:07 UTC	2016-10-03 21:13:36 UTC	2016-11-29 00:00:00 U...	30	-25
6	2016-10-31 11:07:42 UTC	2016-10-03 22:06:03 UTC	2016-11-23 00:00:00 U...	27	-22
7	2016-10-14 16:08:00 UTC	2016-10-03 22:31:31 UTC	2016-11-23 00:00:00 U...	10	-39
8	2016-11-03 14:04:50 UTC	2016-10-03 22:44:10 UTC	2016-12-01 00:00:00 U...	30	-27
9	2016-11-01 15:14:45 UTC	2016-10-03 22:51:30 UTC	2016-11-25 00:00:00 U...	28	-23
10	2016-10-22 14:51:18 UTC	2016-10-04 09:06:10 UTC	2016-11-24 00:00:00 U...	18	-32
11	2016-10-24 16:33:45 UTC	2016-10-04 09:16:33 UTC	2016-11-24 00:00:00 U...	20	-30

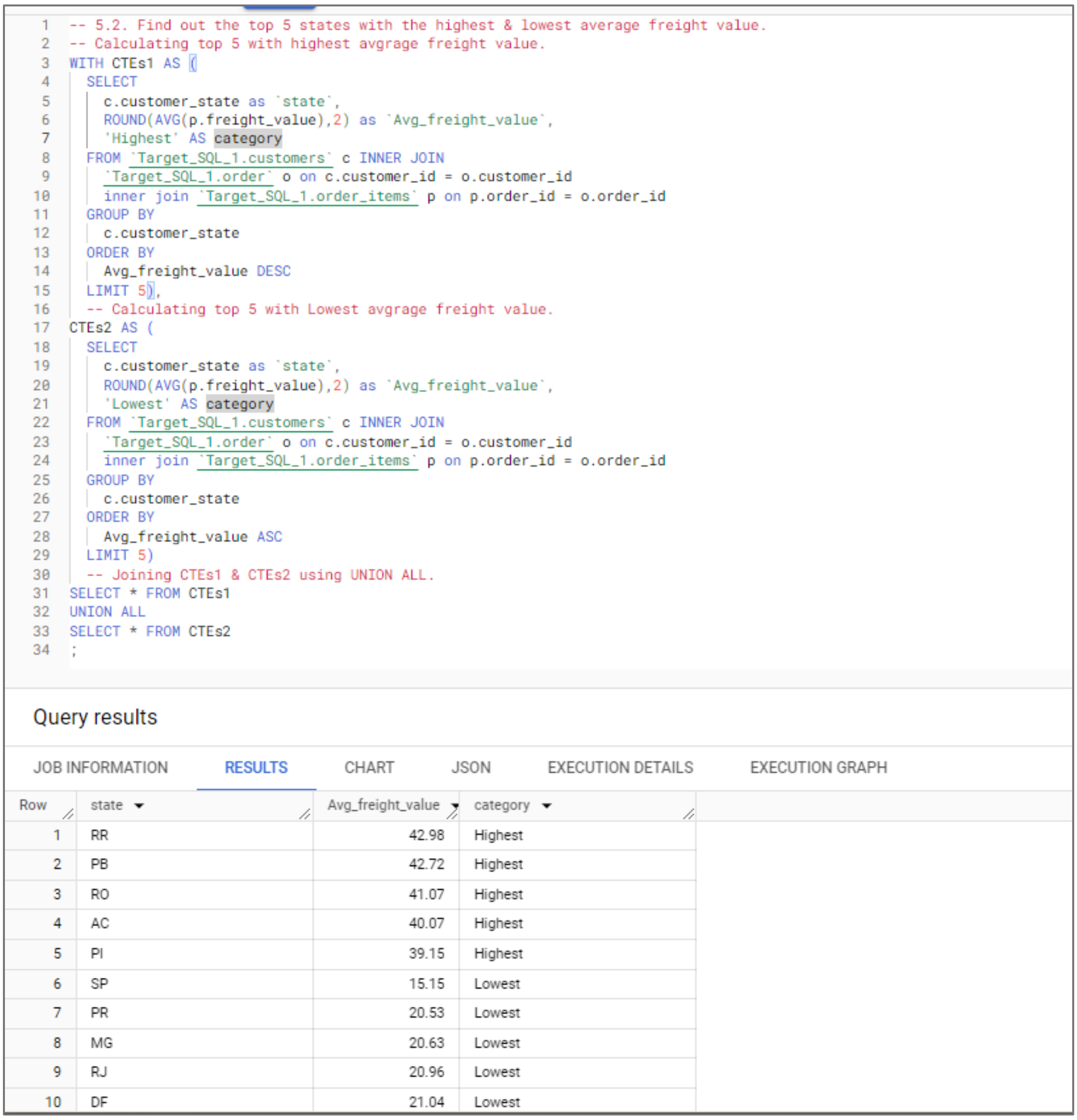
Insights:-

- The " delivery_time " column represents the number of days taken to deliver an order to the customer from the purchase date, while the "diff_estimated_delivery" column indicates the difference between the estimated delivery date and the actual delivery date.
- By looking at the "delivery_time" and "diff_estimated_delivery" columns, we can see how well the delivery process is working, including any delays or early deliveries compared to the expected timeframe.
- We can examine these columns further to find patterns, unusual data points, or factors that affect delivery times or differences between estimated and actual delivery dates.
- These insights can help manage customer expectations, improve customer satisfaction, make the delivery process better, and enhance logistics operations.

b. Find out the top 5 states with the highest & lowest average freight value.

Answer:-

```
WITH CTEs1 AS (
 SELECT
  c.customer_state as `state`,
  ROUND(AVG(p.freight_value),2) as `Avg_freight_value`,
  'Highest' AS category
 FROM `Target_SQL_1.customers` c INNER JOIN
  `Target_SQL_1.order` o on c.customer_id = o.customer_id
  inner join `Target_SQL_1.order_items` p on p.order_id = o.order_id
 GROUP BY
  c.customer_state
 ORDER BY
  Avg_freight_value DESC
 LIMIT 5),
 -- Calculating top 5 with Lowest avgrage freight value.
CTEs2 AS (
 SELECT
  c.customer_state as `state`,
  ROUND(AVG(p.freight_value),2) as `Avg_freight_value`,
  'Lowest' AS category
 FROM `Target_SQL_1.customers` c INNER JOIN
  `Target_SQL_1.order` o on c.customer_id = o.customer_id
  inner join `Target_SQL_1.order_items` p on p.order_id = o.order_id
 GROUP BY
  c.customer_state
 ORDER BY
  Avg_freight_value ASC
 LIMIT 5)
 -- Joining CTEs1 & CTEs2 using UNION ALL.
SELECT * FROM CTEs1
UNION ALL
SELECT * FROM CTEs2;
```



```
-- 5.2. Find out the top 5 states with the highest & lowest average freight value.
-- Calculating top 5 with highest avgrage freight value.
WITH CTEs1 AS (
  SELECT
    c.customer_state as `state`,
    ROUND(AVG(p.freight_value),2) as `Avg_freight_value`,
    'Highest' AS category
  FROM `Target_SQL_1.customers` c INNER JOIN
    `Target_SQL_1.order` o on c.customer_id = o.customer_id
    inner join `Target_SQL_1.order_items` p on p.order_id = o.order_id
  GROUP BY
    c.customer_state
  ORDER BY
    Avg_freight_value DESC
  LIMIT 5),
  -- Calculating top 5 with Lowest avgrage freight value.
CTEs2 AS (
  SELECT
    c.customer_state as `state`,
    ROUND(AVG(p.freight_value),2) as `Avg_freight_value`,
    'Lowest' AS category
  FROM `Target_SQL_1.customers` c INNER JOIN
    `Target_SQL_1.order` o on c.customer_id = o.customer_id
    inner join `Target_SQL_1.order_items` p on p.order_id = o.order_id
  GROUP BY
    c.customer_state
  ORDER BY
    Avg_freight_value ASC
  LIMIT 5)
  -- Joining CTEs1 & CTEs2 using UNION ALL.
SELECT * FROM CTEs1
UNION ALL
SELECT * FROM CTEs2
;
```

Query results

Row	state	Avg_freight_value	category
1	RR	42.98	Highest
2	PB	42.72	Highest
3	RO	41.07	Highest
4	AC	40.07	Highest
5	PI	39.15	Highest
6	SP	15.15	Lowest
7	PR	20.53	Lowest
8	MG	20.63	Lowest
9	RJ	20.96	Lowest
10	DF	21.04	Lowest

Insight:-

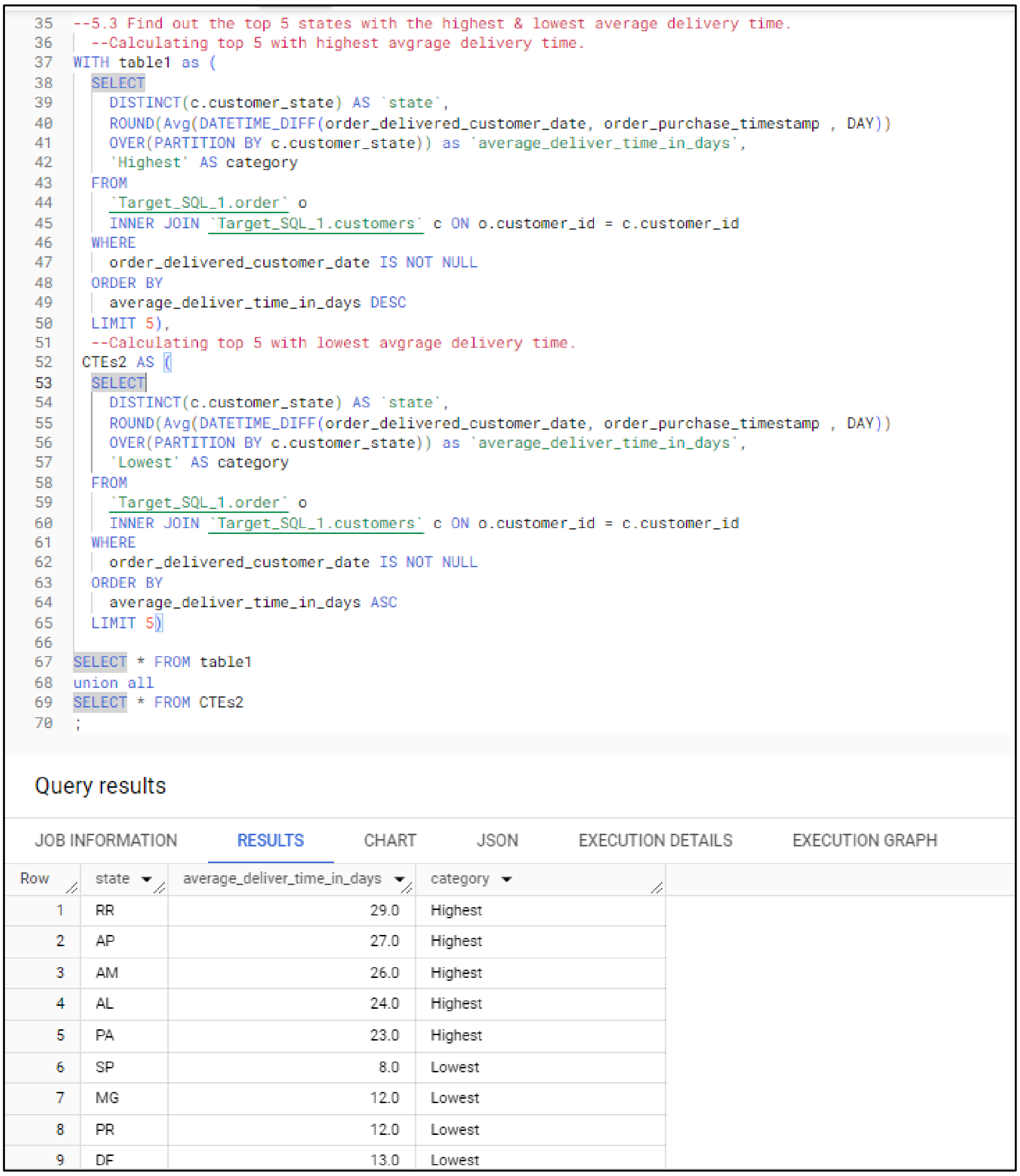
- States with high average freight costs, like RR and PB, might have higher shipping prices due to being remote, having higher transportation costs, or facing supply chain issues.
- To save on shipping costs, we should look at states with low average freight costs, like SP and PR, to find places with cheaper shipping.
- This information can help us create targeted plans, negotiate better freight rates, or find ways to cut costs in our supply chain.
- When analysing the data and making conclusions, we should also consider factors like distance, transportation infrastructure, carrier availability, and regional economic differences.

c. Find out the top 5 states with the highest & lowest average delivery time.

Answer:-

```
 --Calculating top 5 with highest avgrage delivery time.
WITH table1 as (
 SELECT
  DISTINCT(c.customer_state) AS `state`,
  ROUND(Avg(DATETIME_DIFF(order_delivered_customer_date, order_purchase_timestamp , DAY))
  OVER(PARTITION BY c.customer_state)) as `average_deliver_time_in_days`,
  'Highest' AS category
 FROM
  `Target_SQL_1.order` o
  INNER JOIN `Target_SQL_1.customers` c ON o.customer_id = c.customer_id
 WHERE
  order_delivered_customer_date IS NOT NULL
 ORDER BY
  average_deliver_time_in_days DESC
 LIMIT 5),
 --Calculating top 5 with lowest avgrage delivery time.
 CTEs2 AS (
 SELECT
  DISTINCT(c.customer_state) AS `state`,
  ROUND(Avg(DATETIME_DIFF(order_delivered_customer_date, order_purchase_timestamp , DAY))
  OVER(PARTITION BY c.customer_state)) as `average_deliver_time_in_days`,
  'Lowest' AS category
 FROM
  `Target_SQL_1.order` o
  INNER JOIN `Target_SQL_1.customers` c ON o.customer_id = c.customer_id
 WHERE
  order_delivered_customer_date IS NOT NULL
 ORDER BY
  average_deliver_time_in_days ASC
 LIMIT 5)

SELECT * FROM table1
union all
SELECT * FROM CTEs2
;
```



```
--5.3 Find out the top 5 states with the highest & lowest average delivery time.
  --Calculating top 5 with highest avgrage delivery time.
WITH table1 as (
  SELECT
    DISTINCT(c.customer_state) AS `state`,
    ROUND(Avg(DATETIME_DIFF(order_delivered_customer_date, order_purchase_timestamp , DAY))
    OVER(PARTITION BY c.customer_state)) as `average_deliver_time_in_days`,
    `Highest` AS category
  FROM
    `Target_SQL_1.order` o
    INNER JOIN `Target_SQL_1.customers` c ON o.customer_id = c.customer_id
  WHERE
    order_delivered_customer_date IS NOT NULL
  ORDER BY
    average_deliver_time_in_days DESC
  LIMIT 5),
  --Calculating top 5 with lowest avgrage delivery time.
 CTEs2 AS (
  SELECT
    DISTINCT(c.customer_state) AS `state`,
    ROUND(Avg(DATETIME_DIFF(order_delivered_customer_date, order_purchase_timestamp , DAY))
    OVER(PARTITION BY c.customer_state)) as `average_deliver_time_in_days`,
    `Lowest` AS category
  FROM
    `Target_SQL_1.order` o
    INNER JOIN `Target_SQL_1.customers` c ON o.customer_id = c.customer_id
  WHERE
    order_delivered_customer_date IS NOT NULL
  ORDER BY
    average_deliver_time_in_days ASC
  LIMIT 5)

SELECT * FROM table1
union all
SELECT * FROM CTEs2
;
```

Query results

Row	state	average_deliver_time_in_days	category
1	RR	29.0	Highest
2	AP	27.0	Highest
3	AM	26.0	Highest
4	AL	24.0	Highest
5	PA	23.0	Highest
6	SP	8.0	Lowest
7	MG	12.0	Lowest
8	PR	12.0	Lowest
9	DF	13.0	Lowest

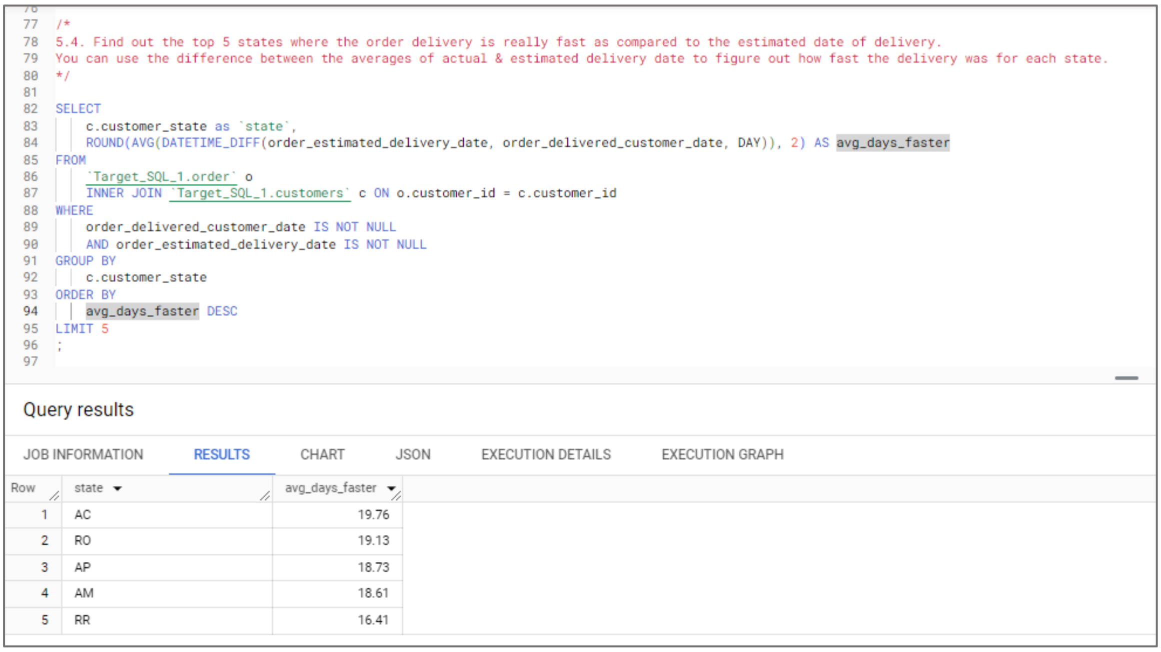
Insight:-

- By examining states like SP and PR with the lowest average delivery times and states like RR and AP with the highest average delivery times, we can identify areas with efficient delivery operations and strong logistics networks.
- These insights can help our company improve customer satisfaction, operational efficiency, and optimize the delivery process. We can also set realistic expectations for customers based on regional delivery time patterns.
- When analyzing the data and drawing conclusions, it’s important to consider factors such as population density, differences between urban and rural areas, customer expectations, and specific logistical challenges.
- Using this information, our company can focus on areas where delivery efficiency can be improved, enhancing both customer experiences and operational efficiency.

d. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.
   You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

Answer:-

```
SELECT
    c.customer_state as `state`,
    ROUND(AVG(DATETIME_DIFF(order_estimated_delivery_date, order_delivered_customer_date,
DAY)), 2) AS avg_days_faster
FROM
    `Target_SQL_1.order` o
    INNER JOIN `Target_SQL_1.customers` c ON o.customer_id = c.customer_id
WHERE
    order_delivered_customer_date IS NOT NULL
    AND order_estimated_delivery_date IS NOT NULL
GROUP BY
    c.customer_state
ORDER BY
    avg_days_faster DESC
LIMIT 5;
```



```
/*
5.4. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.
You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.
*/

SELECT
    c.customer_state as `state`,
    ROUND(AVG(DATETIME_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY)), 2) AS avg_days_faster
FROM
    `Target_SQL_1.order` o
    INNER JOIN `Target_SQL_1.customers` c ON o.customer_id = c.customer_id
WHERE
    order_delivered_customer_date IS NOT NULL
    AND order_estimated_delivery_date IS NOT NULL
GROUP BY
    c.customer_state
ORDER BY
    avg_days_faster DESC
LIMIT 5
;
```

Query results

Row	state	avg_days_faster
1	AC	19.76
2	RO	19.13
3	AP	18.73
4	AM	18.61
5	RR	16.41

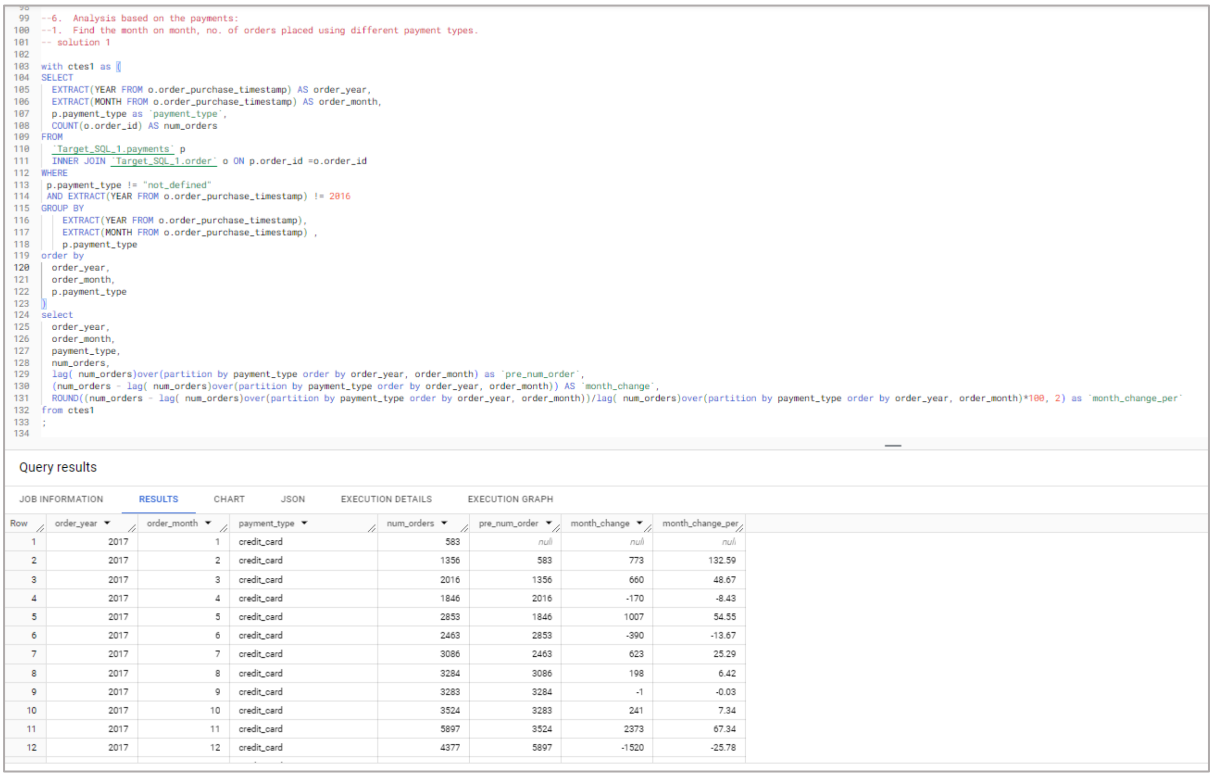
Insight:-

- In states like AC, RO, AP, and AM, where delivery is fastest, our company can highlight our quick and reliable service to attract more customers and increase satisfaction.
- This data can help us improve operations, enhance customer experience, optimize logistics, and find opportunities to expand in areas known for quick delivery

6. Analysis based on the payments:
    a. Find the month on month no. of orders placed using different payment types.

Answer:-

```
with ctes1 as (
SELECT
  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
  p.payment_type as `payment_type`,
  COUNT(o.order_id) AS num_orders
FROM
  `Target_SQL_1.payments` p
  INNER JOIN `Target_SQL_1.order` o ON p.order_id =o.order_id
WHERE
 p.payment_type != "not_defined"
 AND EXTRACT(YEAR FROM o.order_purchase_timestamp) != 2016
GROUP BY
    EXTRACT(YEAR FROM o.order_purchase_timestamp),
    EXTRACT(MONTH FROM o.order_purchase_timestamp) ,
    p.payment_type
order by
  order_year,
  order_month,
  p.payment_type
)
select
  order_year,
  order_month,
  payment_type,
  num_orders,
  lag( num_orders)over(partition by payment_type order by order_year, order_month) as
`pre_num_order`,
  (num_orders - lag( num_orders)over(partition by payment_type order by order_year,
order_month)) AS `month_change`,
  ROUND((num_orders - lag( num_orders)over(partition by payment_type order by order_year,
order_month))/lag( num_orders)over(partition by payment_type order by order_year,
order_month)*100, 2) as `month_change_per`
from ctes1
;
```



```
--6.  Analysis based on the payments:
--1.  Find the month on month, no. of orders placed using different payment types.
-- solution 1

with ctes1 as (
SELECT
  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
  p.payment_type as `payment_type`,
  COUNT(o.order_id) AS num_orders
FROM
  `Target_SQL_1.payments` p
  INNER JOIN `Target_SQL_1.order` o ON p.order_id =o.order_id
WHERE
 p.payment_type != "not_defined"
 AND EXTRACT(YEAR FROM o.order_purchase_timestamp) != 2016
GROUP BY
    EXTRACT(YEAR FROM o.order_purchase_timestamp),
    EXTRACT(MONTH FROM o.order_purchase_timestamp) ,
    p.payment_type
order by
  order_year,
  order_month,
  p.payment_type
)
select
  order_year,
  order_month,
  payment_type,
  num_orders,
  lag( num_orders)over(partition by payment_type order by order_year, order_month) as `pre_num_order`,
  (num_orders - lag( num_orders)over(partition by payment_type order by order_year, order_month)) AS `month_change`,
  ROUND((num_orders - lag( num_orders)over(partition by payment_type order by order_year, order_month))/lag( num_orders)over(partition by payment_type order by order_year, order_month)*100, 2) as `month_change_per`
from ctes1
;
```

Query results

Row	order_year	order_month	payment_type	num_orders	pre_num_order	month_change	month_change_per
1	2017	1	credit_card	583	null	null	null
2	2017	2	credit_card	1356	583	773	132.59
3	2017	3	credit_card	2016	1356	660	48.67
4	2017	4	credit_card	1846	2016	-170	-8.43
5	2017	5	credit_card	2853	1846	1007	54.55
6	2017	6	credit_card	2463	2853	-390	-13.67
7	2017	7	credit_card	3086	2463	623	25.29
8	2017	8	credit_card	3284	3086	198	6.42
9	2017	9	credit_card	3283	3284	-1	-0.03
10	2017	10	credit_card	3524	3283	241	7.34
11	2017	11	credit_card	5897	3524	2373	67.34
12	2017	12	credit_card	4377	5897	-1520	-25.78

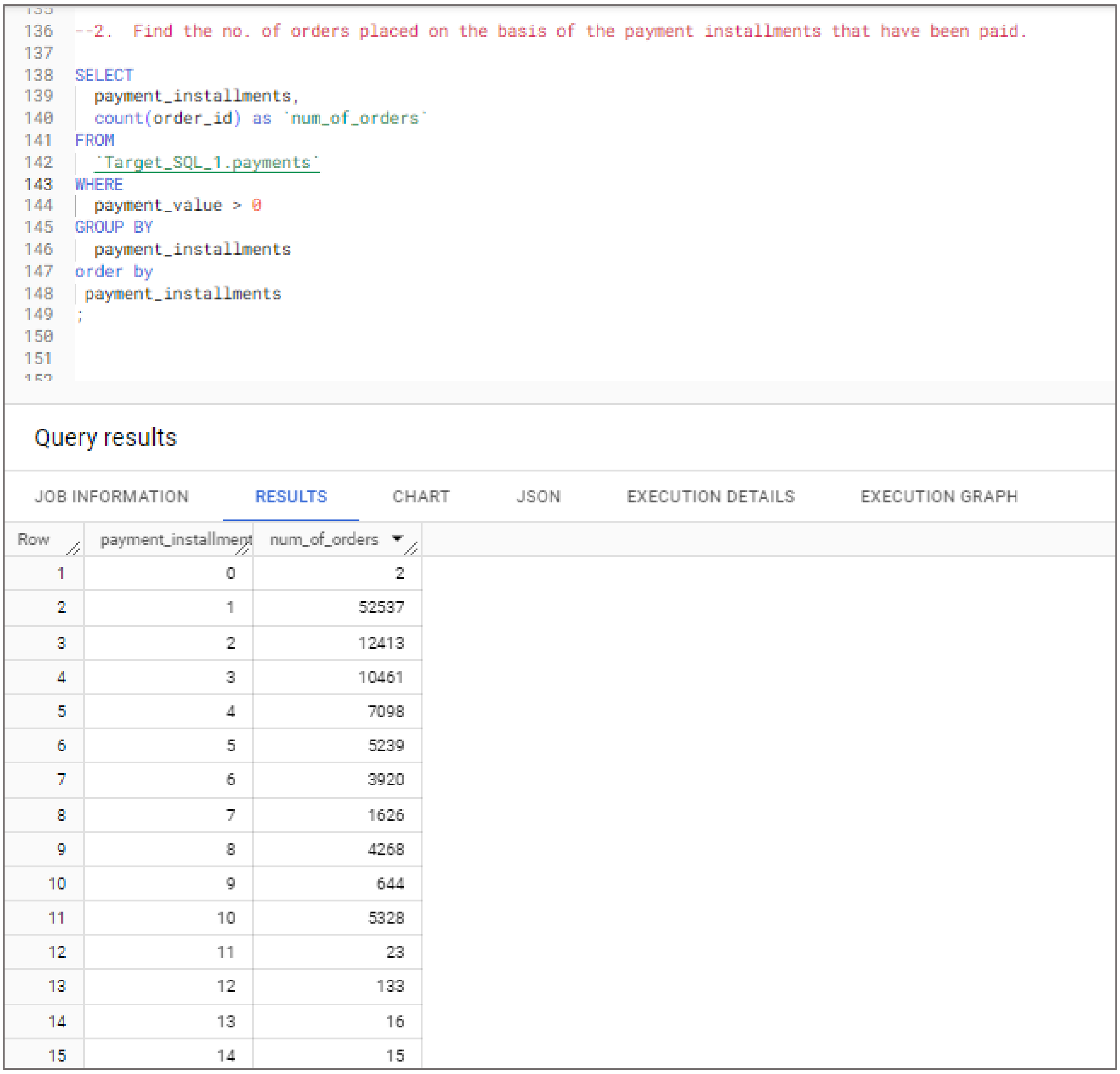
Insight:-

- Tracking monthly trends in order counts can help analyze seasonality, identify peak months, and evaluate the impact of marketing efforts or other factors on customer behaviour.
- Insights from monthly payment preferences can help companies optimize payment processes, tailor marketing campaigns, and improve customer experiences.

b. Find the no. of orders placed on the basis of the payment installments that have been paid.

Answer:-

```
SELECT
  payment_installments,
  count(order_id) as `num_of_orders`
FROM
  `Target_SQL_1.payments`
WHERE
  payment_value > 0
GROUP BY
  payment_installments
order by
 payment_installments ;
```



```sql
--2.  Find the no. of orders placed on the basis of the payment installments that have been paid.

SELECT
  payment_installments,
  count(order_id) as `num_of_orders`
FROM
  `Target_SQL_1.payments`
WHERE
  payment_value > 0
GROUP BY
  payment_installments
order by
 payment_installments
;
```

Row	payment_installmen	num_of_orders
1	0	2
2	1	52537
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626
9	8	4268
10	9	644
11	10	5328
12	11	23
13	12	133
14	13	16
15	14	15

Insight:-

- There were 52537 orders where the payment installment was 1.
- This analysis can show if customers prefer to pay in installments.
- We can understand customers' budgeting or financing preferences based on their choice of payment installments.
- Tracking the number of orders by payment installments can reveal customers' buying habits and their preference for flexible payment options.

4. Recommendations:

- Boost Sales During Low-Order Months: Conduct market research, partner with complementary businesses, offer promotions, and use targeted marketing. Tailor strategies for different times of the day to maximize sales. Focus on customer engagement in states with high customer bases and look for growth opportunities in states with lower customer bases. Leverage competitive advantages to stand out.
- Seasonal Campaigns: Identify popular product categories and market seasonal products related to Brazilian culture, such as New Year, Black Friday, Carnivals, FIFA World Cup, and Capoeira, to attract more customers.
- Optimize Operations: Improve shipping, pricing, and resource allocation. Reduce costs through negotiations and route improvements by partnering with local carriers and logistics partners. Use technology to enhance logistics and reduce delivery times, ensuring customer satisfaction.
- Proactive Communication: Keep customers informed about delivery expectations and provide timely updates on order status and potential delays.
- Secure Payment Methods: Ensure a secure payment system that supports various payment methods. Educate customers about alternative payment options and offer incentives to encourage their use. Promote the benefits of lower installment options through targeted marketing campaigns.
- Customer Feedback: Collect and analyze customer feedback to understand preferences and improve customer satisfaction. This helps build brand loyalty.

Based on this data analysis, businesses can gain insights to optimize operations, improve customer satisfaction, and drive sales growth.