Problem Statement:

Analyzing the given dataset to extract valuable insights and provide actionable recommendations.

---

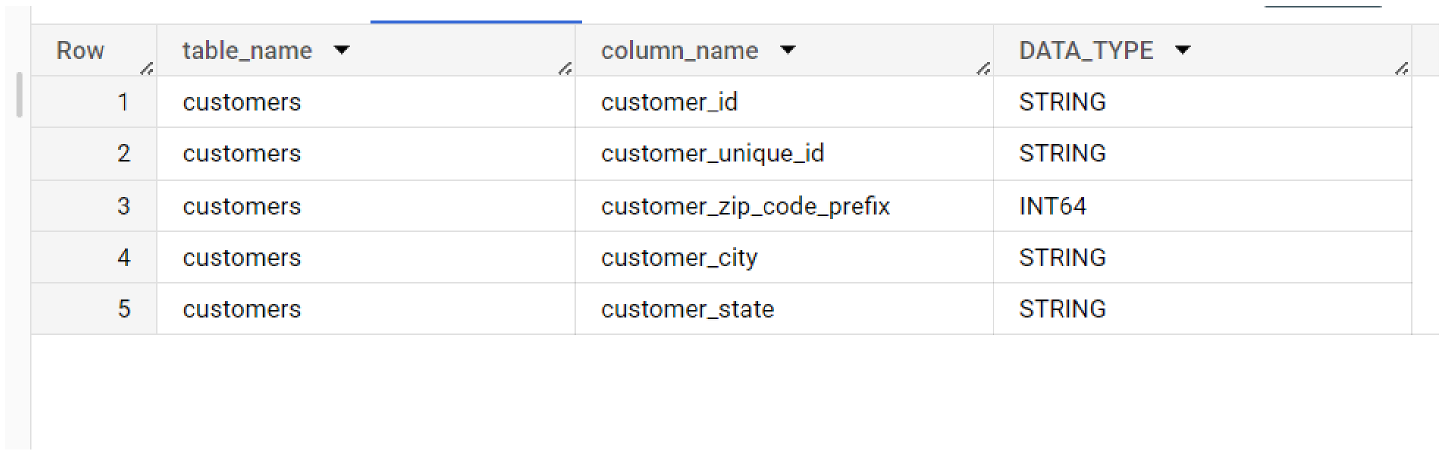
1. Importing the dataset and performing typical exploratory analysis steps, such as checking the structure and characteristics of the dataset:

1.1) Data type of all columns in the "customers" table:

Query:

```
SELECT
  table_name,
  column_name,
  DATA_TYPE
FROM
  `ecommerce-394512`.target.INFORMATION_SCHEMA.COLUMNS
WHERE
  table_name = 'customers';
```

Output:



Row	table_name	column_name	DATA_TYPE
1	customers	customer_id	STRING
2	customers	customer_unique_id	STRING
3	customers	customer_zip_code_prefix	INT64
4	customers	customer_city	STRING
5	customers	customer_state	STRING

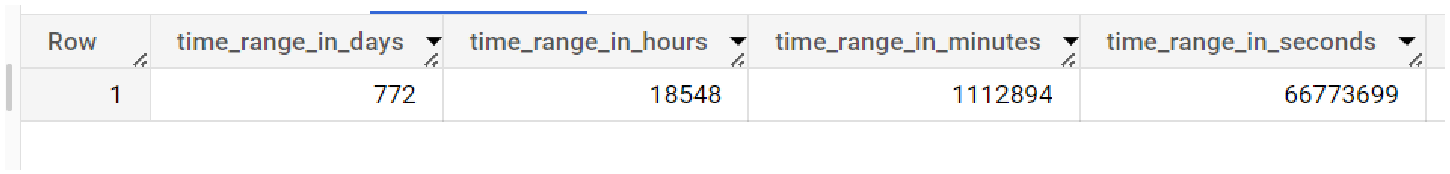
1.2) Time range during which the orders were placed:

Query:

```
SELECT
  TIMESTAMP_DIFF(MAX(order_purchase_timestamp), MIN(order_purchase_timestamp),
DAY) time_range_in_days,
  TIMESTAMP_DIFF(MAX(order_purchase_timestamp), MIN(order_purchase_timestamp),
HOUR) time_range_in_hours,
  TIMESTAMP_DIFF(MAX(order_purchase_timestamp), MIN(order_purchase_timestamp),
MINUTE) time_range_in_minutes,
  TIMESTAMP_DIFF(MAX(order_purchase_timestamp), MIN(order_purchase_timestamp),
SECOND) time_range_in_seconds
```

```
    FROM
      `target.orders`;
```

Output:



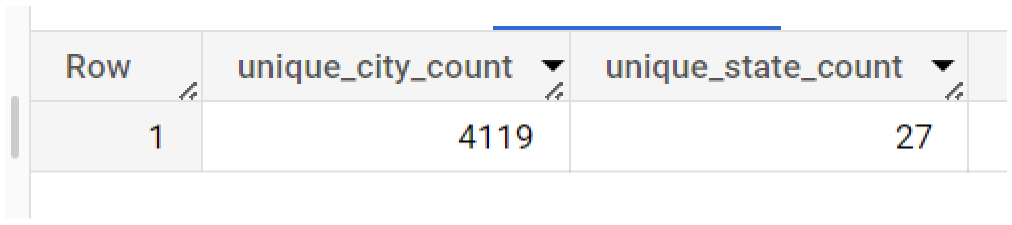
Row	time_range_in_days	time_range_in_hours	time_range_in_minutes	time_range_in_seconds
1	772	18548	1112894	66773699

1.3) Count of cities and states from which customers placed orders during the specified period.

Query:

```
    SELECT
      COUNT(DISTINCT customer_city) AS unique_city_count,
      COUNT(DISTINCT customer_state) AS unique_state_count
    FROM
      `target.customers`;
```

Output:



Row	unique_city_count	unique_state_count
1	4119	27

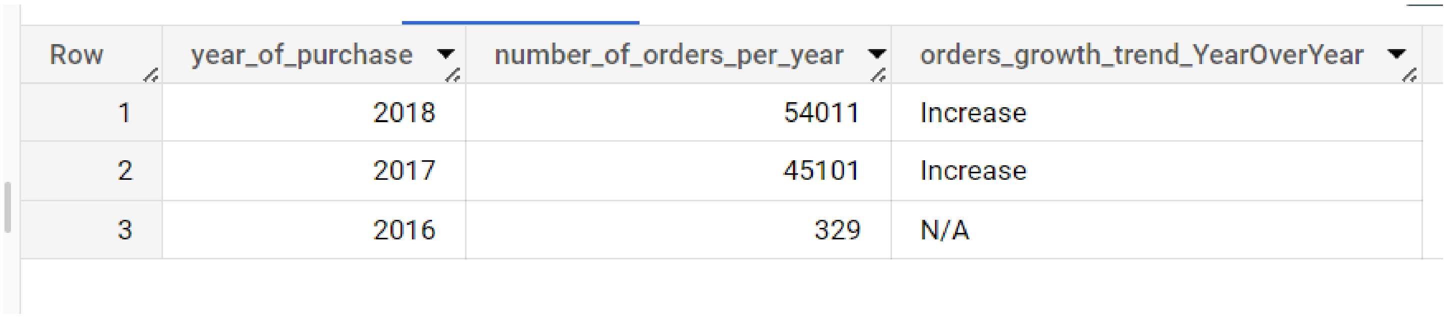
---

2) In-depth Exploration:

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

Query:

```
    WITH
      yearly_order_counts AS (
      SELECT
        EXTRACT(YEAR FROM order_purchase_timestamp) AS year_of_purchase,
        COUNT(*) AS number_of_orders_per_year
      FROM
        `target.orders`
      GROUP BY
        year_of_purchase
      ORDER BY
        year_of_purchase DESC)
    SELECT
      year_of_purchase,
      number_of_orders_per_year,
      CASE
        WHEN number_of_orders_per_year > LEAD(number_of_orders_per_year) OVER
    (ORDER BY year_of_purchase DESC) THEN 'Increase'
        WHEN LEAD(number_of_orders_per_year) OVER (ORDER BY year_of_purchase DESC)
    IS NULL THEN 'N/A'
        WHEN number_of_orders_per_year < LEAD(number_of_orders_per_year) OVER
    (ORDER BY year_of_purchase DESC) THEN 'Decrease'
      ELSE
      'No Change'
    END
      AS orders_growth_trend_YearOverYear
    FROM
      yearly_order_counts
    ORDER BY
      year_of_purchase DESC;
```



Row	year_of_purchase	number_of_orders_per_year	orders_growth_trend_YearOverYear
1	2018	54011	Increase
2	2017	45101	Increase
3	2016	329	N/A

Insights:
Based on the analysis, it is evident that there is a growing trend in the number of orders placed over the past years. The data reveals that the current year's order count is higher compared to the previous year, indicating a positive growth trajectory for the company.

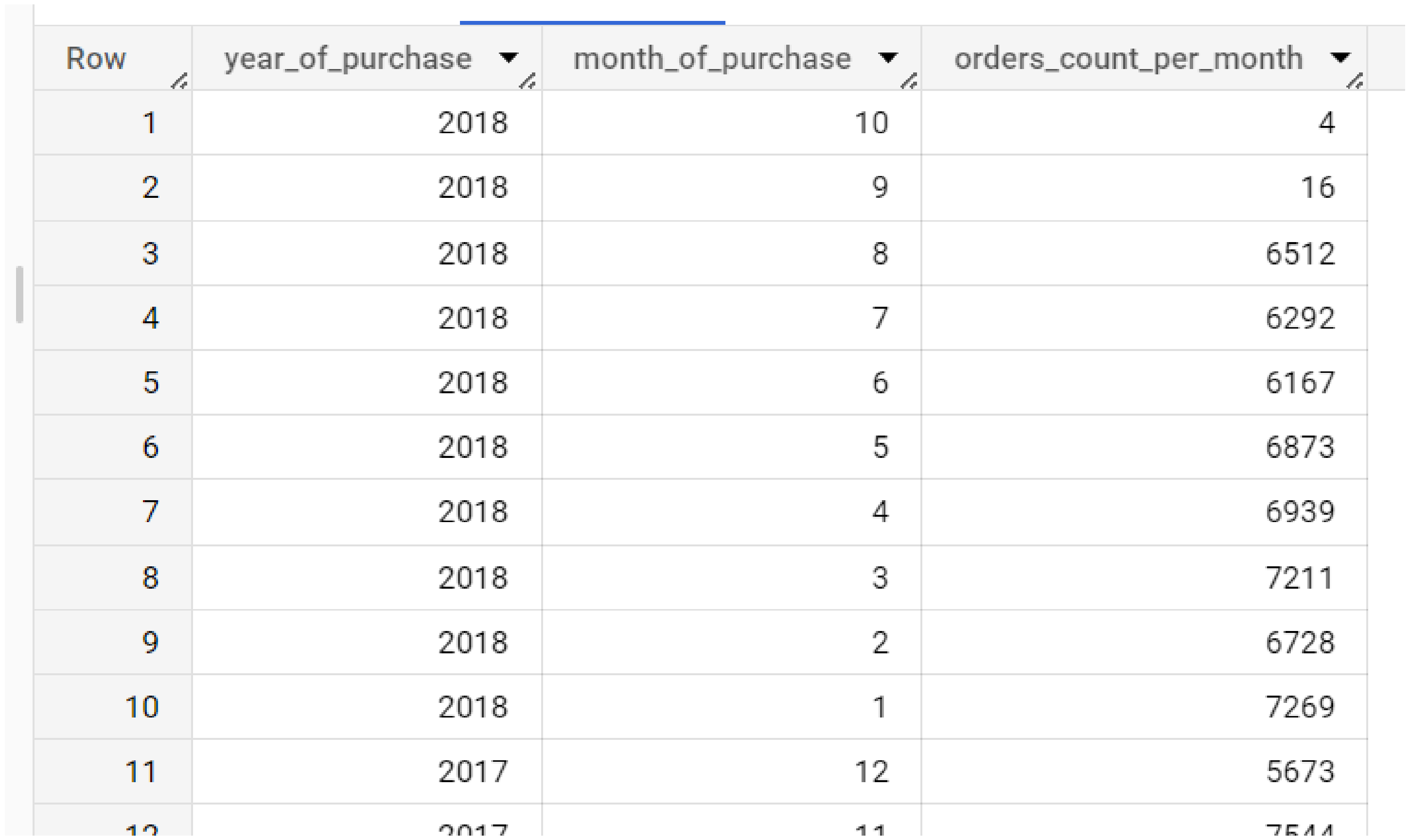
Recommendations:
- Optimizing inventory management to enhance efficiency and reduce costs.
- Implementing customer retention strategies to foster long-term loyalty and repeat business.

2.2) Analyzing the monthly seasonality in terms of the number of orders being placed:

Query1:

```
SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) AS year_of_purchase,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS month_of_purchase,
  COUNT(*) AS orders_count_per_month
FROM
  `target.orders`
GROUP BY
  year_of_purchase,
  month_of_purchase
ORDER BY
  year_of_purchase DESC,
  month_of_purchase DESC;
```

Output:

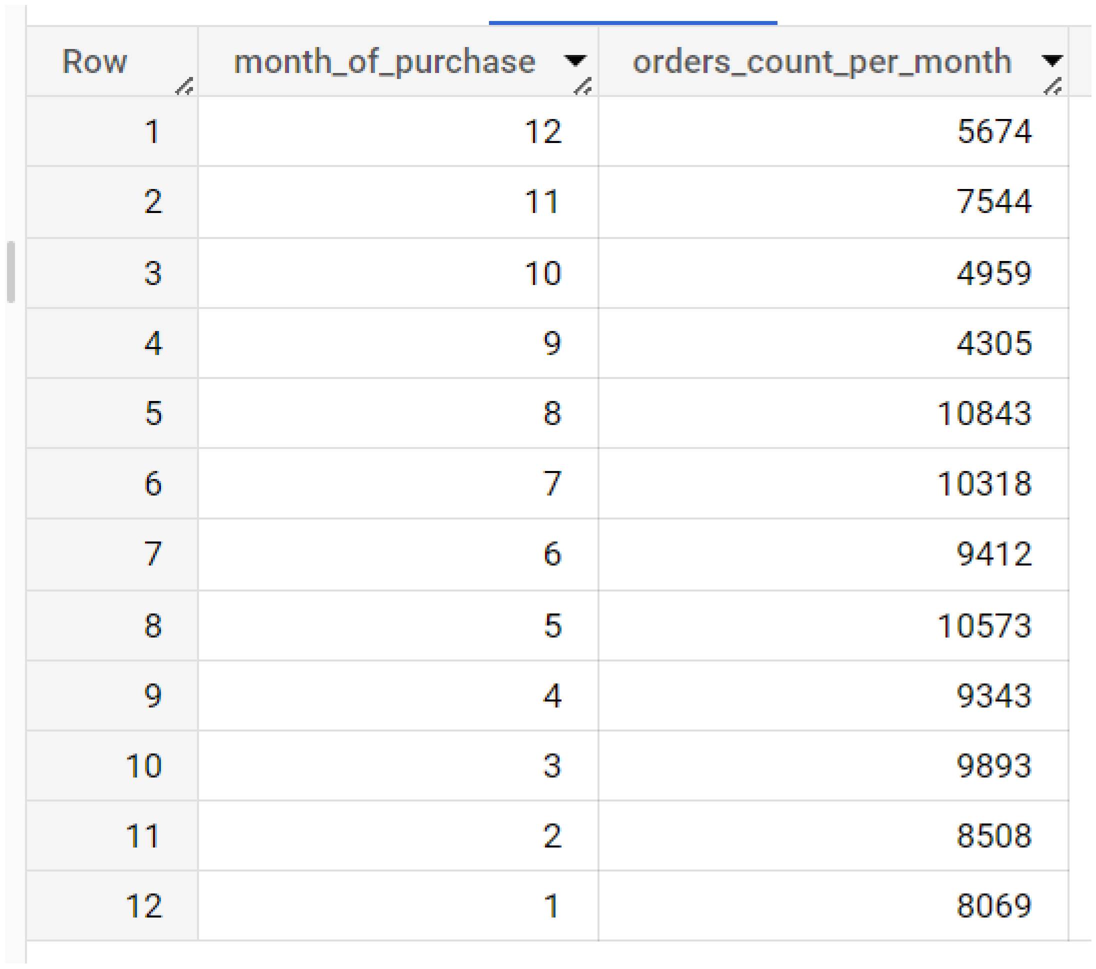


Row	year_of_purchase	month_of_purchase	orders_count_per_month
1	2018	10	4
2	2018	9	16
3	2018	8	6512
4	2018	7	6292
5	2018	6	6167
6	2018	5	6873
7	2018	4	6939
8	2018	3	7211
9	2018	2	6728
10	2018	1	7269
11	2017	12	5673
12	2017	11	7544

Query2:

```
SELECT
  EXTRACT(MONTH FROM order_purchase_timestamp) AS month_of_purchase,
  COUNT(*) AS orders_count_per_month
FROM
  `target.orders`
GROUP BY
  month_of_purchase
ORDER BY
  month_of_purchase DESC;
```

Output:



Row	month_of_purchase	orders_count_per_month
1	12	5674
2	11	7544
3	10	4959
4	9	4305
5	8	10843
6	7	10318
7	6	9412
8	5	10573
9	4	9343
10	3	9893
11	2	8508
12	1	8069

Insights:

The analysis of monthly seasonality in terms of the number of orders reveals that the dataset contains gaps, with missing data for some months in 2016 and 2018. As a result, a comprehensive assessment of seasonality is inconclusive due to incomplete data. However, focusing solely on the available data from the year 2017, it is apparent that there is a higher trend in customer orders during the month of November.

Recommendations:

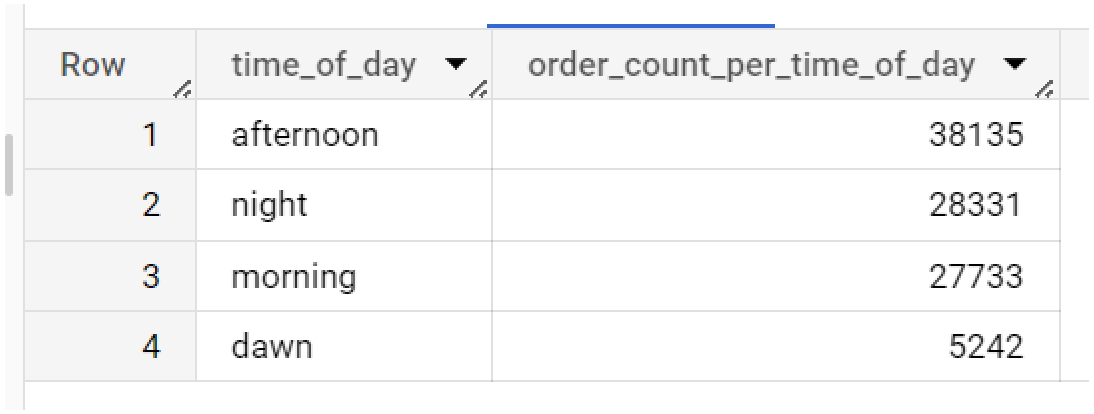
- Further analysis and data completeness for other years are necessary to draw more conclusive insights on seasonal patterns.
- Post identifying monthly seasonality, the company could consider implementing targeted marketing campaigns, promotions, and special offers during the month of seasonality to attract more customers and boost sales.

2.3) Analysis to determine the most common time of day when Brazilian customers place their orders, categorized as Dawn, Morning, Afternoon, or Night.

Time Categories:
0-6 hrs: Dawn
7-12 hrs: Morning
13-18 hrs: Afternoon
19-23 hrs: Night

Query:

```
WITH hourly_order_count AS
(
  SELECT
    EXTRACT(HOUR FROM order_purchase_timestamp) AS order_hour,
    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
'morning'
      WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN
'afternoon'
      ELSE 'night'
    END AS time_of_day,
    COUNT(*) AS order_count_per_order_hour
  FROM
    `target.orders`
  GROUP BY
    order_hour, time_of_day
)
SELECT
    time_of_day,
    SUM(order_count_per_order_hour) AS order_count_per_time_of_day
FROM
    hourly_order_count
GROUP BY
    time_of_day
ORDER BY
    order_count_per_time_of_day DESC;
```



Row	time_of_day	order_count_per_time_of_day
1	afternoon	38135
2	night	28331
3	morning	27733
4	dawn	5242

Insights:
Based on the data analysis, it is evident that a significant portion of Brazilian customers tend to place their orders during the afternoon hours.
Recommendation:
Since customers appear to be placing more orders during the afternoon, enhancing customer support and service availability during peak ordering hours can help improve the overall shopping experience, leading to higher customer satisfaction and retention.

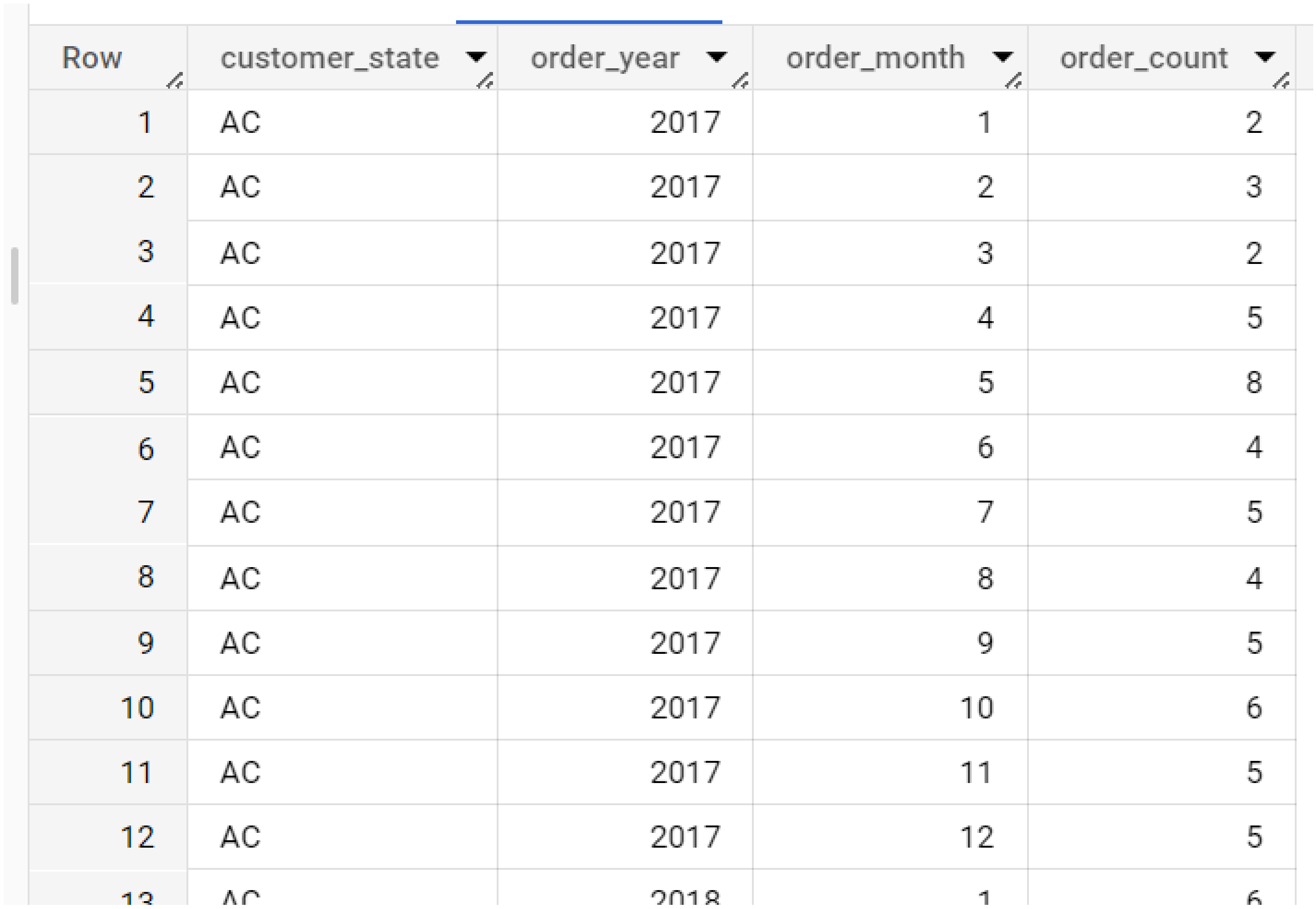
3. The evolution of E-commerce orders in the Brazil region:

3.1) The Month-on-Month (MoM) number of orders placed in each state.

Query:

```
SELECT
  c.customer_state,
  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
  COUNT(*) AS order_count
FROM
  `target.orders` o
JOIN `target.customers` c
  ON c.customer_id=o.customer_id
GROUP BY
  order_year, order_month, customer_state
ORDER BY
  customer_state, order_year, order_month;
```

Output:

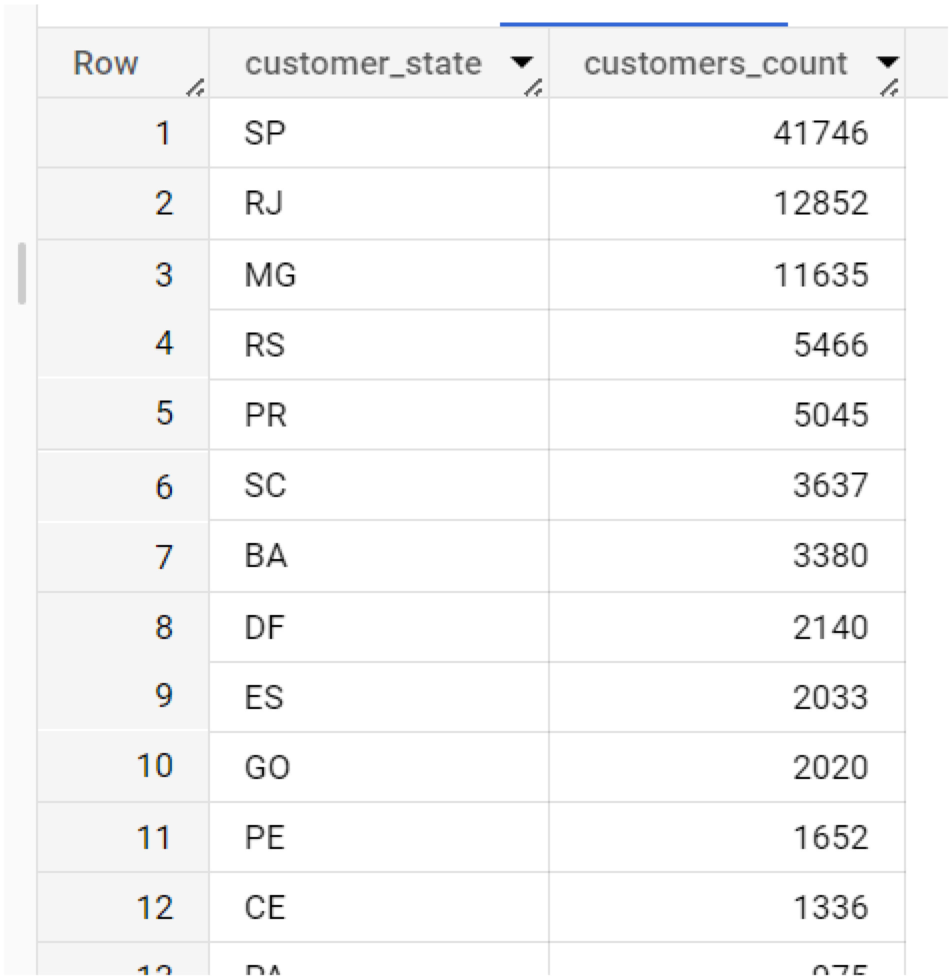


Row	customer_state	order_year	order_month	order_count
1	AC	2017	1	2
2	AC	2017	2	3
3	AC	2017	3	2
4	AC	2017	4	5
5	AC	2017	5	8
6	AC	2017	6	4
7	AC	2017	7	5
8	AC	2017	8	4
9	AC	2017	9	5
10	AC	2017	10	6
11	AC	2017	11	5
12	AC	2017	12	5
13	AC	2018	1	6

3.2) The distribution of customers across all states:

```
SELECT
  customer_state,
  COUNT(customer_id) AS customers_count
FROM
  `target.customers`
GROUP BY
  customer_state
ORDER BY
  customers_count DESC;
```

Output:



Row	customer_state	customers_count
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

Insights:

The distribution of customers across all states in Brazil and MoM ordering pattern shows that a majority of customers/orders are concentrated in three states. (i.e., SP, RJ and MG).

Recommendation:

Implementing localized strategies, such as offering region-specific product assortments or leveraging partnerships with local influencers, can further strengthen Target's presence in these states.

4) Impact on the Economy:

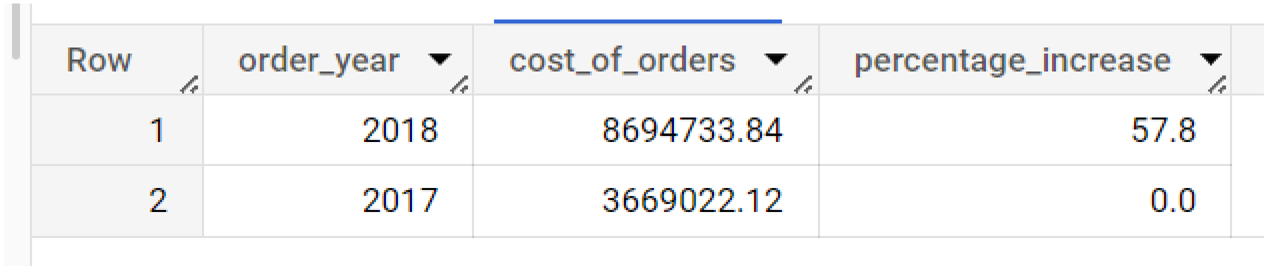
Analyzing the money movement in e-commerce by examining order prices, freight costs, and other financial factors:

4.1) Calculating the percentage increase in the cost of orders from the year 2017 to 2018 (considering only the months between January to August):

Query:

```
WITH cost AS (
  SELECT
    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
    ROUND(SUM(p.payment_value),2) AS cost_of_orders
  FROM
    `target.payments` p
  JOIN
    `target.orders` o ON o.order_id = p.order_id
  WHERE
    EXTRACT(YEAR FROM order_purchase_timestamp) BETWEEN 2017 AND 2018
    AND EXTRACT(MONTH FROM order_purchase_timestamp) BETWEEN 1 AND 8
  GROUP BY
    order_year
)
SELECT
  order_year,
  cost_of_orders,
  ROUND(IFNULL((((cost_of_orders - LEAD(cost_of_orders) OVER (ORDER BY
order_year DESC)) / cost_of_orders) * 100),0),2) AS percentage_increase
FROM
  cost
ORDER BY
  order_year DESC;
```

Output:



Row	order_year	cost_of_orders	percentage_increase
1	2018	8694733.84	57.8
2	2017	3669022.12	0.0

Insight & Recommendation:

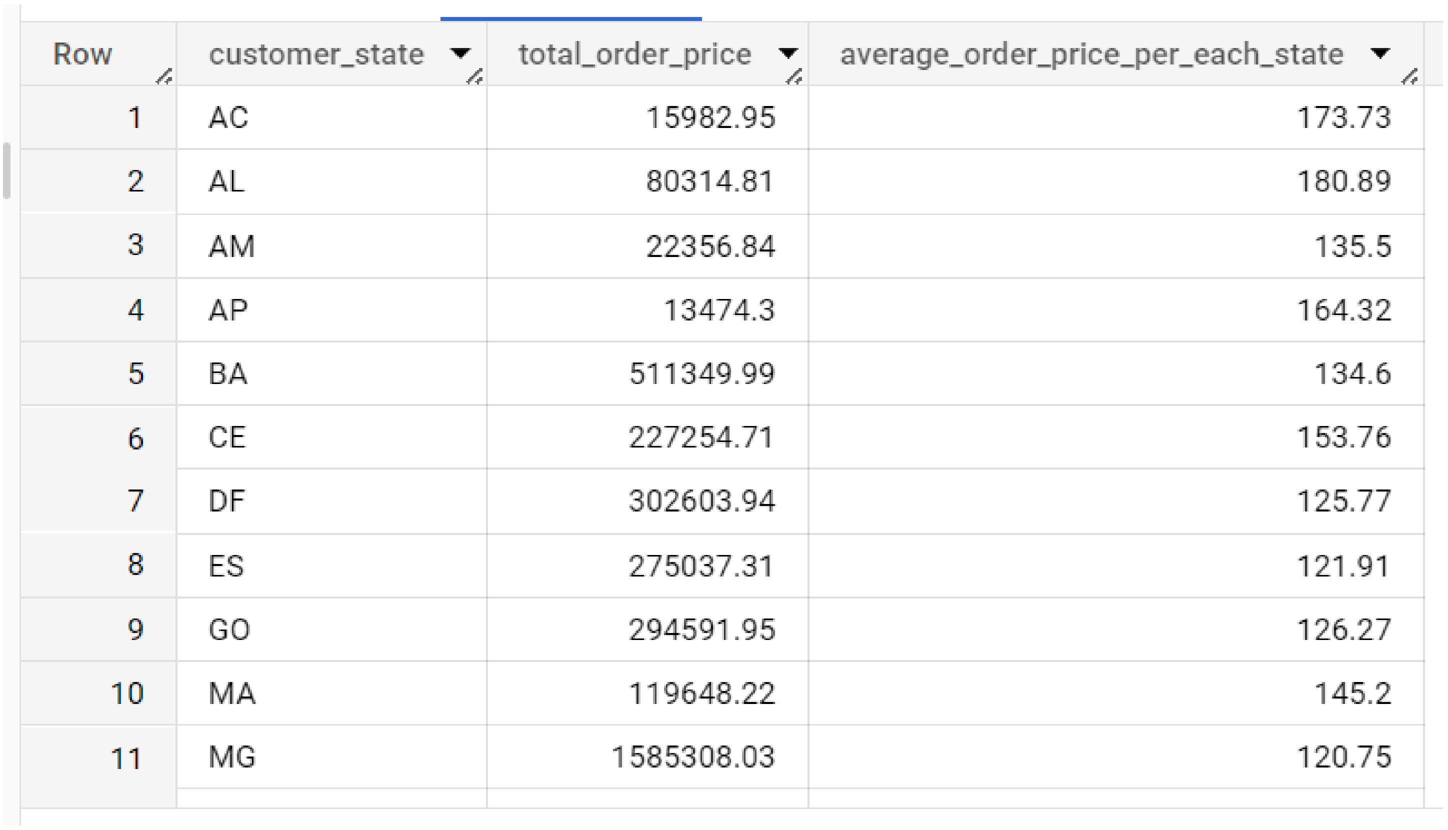
There has been a significant increase in the cost of orders from 2017 to 2018, with a percentage rise of more than 50%. This rise could be attributed to various factors, such as inflation, changes in product pricing, or alterations in customer preferences. It is essential for Target to closely analyze the contributing factors behind this surge to make informed pricing decisions and maintain competitive pricing in the market.

4.2) Calculating the total and average value of the order price for each state:

Query:

```
SELECT
  c.customer_state,
  ROUND(SUM(oi.price),2) AS total_order_price,
  ROUND(AVG(oi.price),2) AS average_order_price_per_each_state
FROM
  `target.order_items` oi
JOIN
  `target.orders` o ON o.order_id=oi.order_id
JOIN
  `target.customers` c ON o.customer_id = c.customer_id
GROUP BY
  c.customer_state
ORDER BY
  c.customer_state;
```

Output:



Row	customer_state	total_order_price	average_order_price_per_each_state
1	AC	15982.95	173.73
2	AL	80314.81	180.89
3	AM	22356.84	135.5
4	AP	13474.3	164.32
5	BA	511349.99	134.6
6	CE	227254.71	153.76
7	DF	302603.94	125.77
8	ES	275037.31	121.91
9	GO	294591.95	126.27
10	MA	119648.22	145.2
11	MG	1585308.03	120.75

Insights & Recommendation:

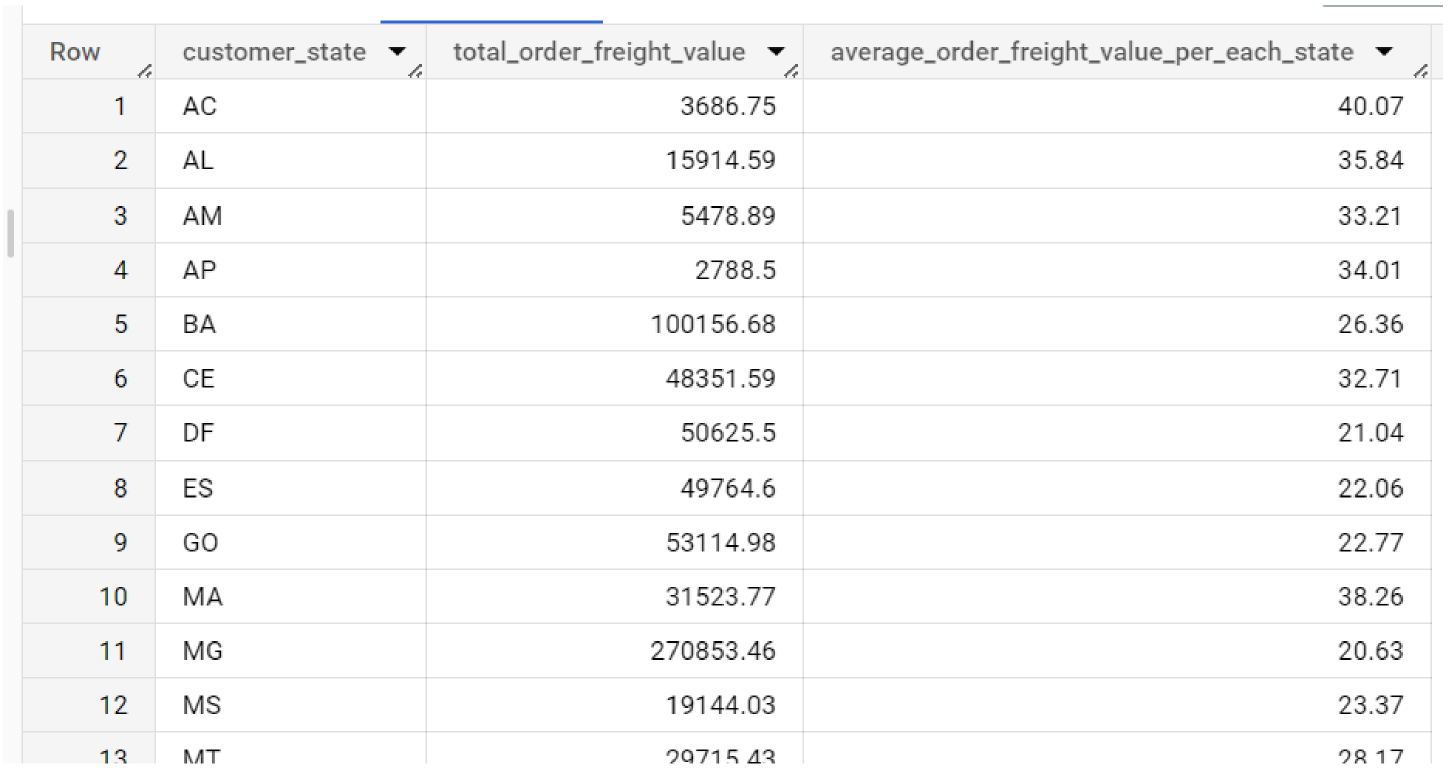
States with higher average order values suggest that customers in those regions tend to spend more per transaction. For states with high total order values, it may be beneficial to allocate more resources to marketing and expanding the customer base.

4.3) Calculating the total and average value of order freight for each state.

Query:

```
SELECT
  c.customer_state,
  ROUND(SUM(oi.freight_value),2) AS total_order_freight_value,
  ROUND(AVG(oi.freight_value),2) AS average_order_freight_value_per_each_state
FROM
  `target.order_items` oi
JOIN
  `target.orders` o ON o.order_id=oi.order_id
JOIN
  `target.customers` c ON o.customer_id = c.customer_id
GROUP BY
  c.customer_state
ORDER BY
  c.customer_state;
```

Output:



Row	customer_state	total_order_freight_value	average_order_freight_value_per_each_state
1	AC	3686.75	40.07
2	AL	15914.59	35.84
3	AM	5478.89	33.21
4	AP	2788.5	34.01
5	BA	100156.68	26.36
6	CE	48351.59	32.71
7	DF	50625.5	21.04
8	ES	49764.6	22.06
9	GO	53114.98	22.77
10	MA	31523.77	38.26
11	MG	270853.46	20.63
12	MS	19144.03	23.37
13	MT	29715.43	28.17

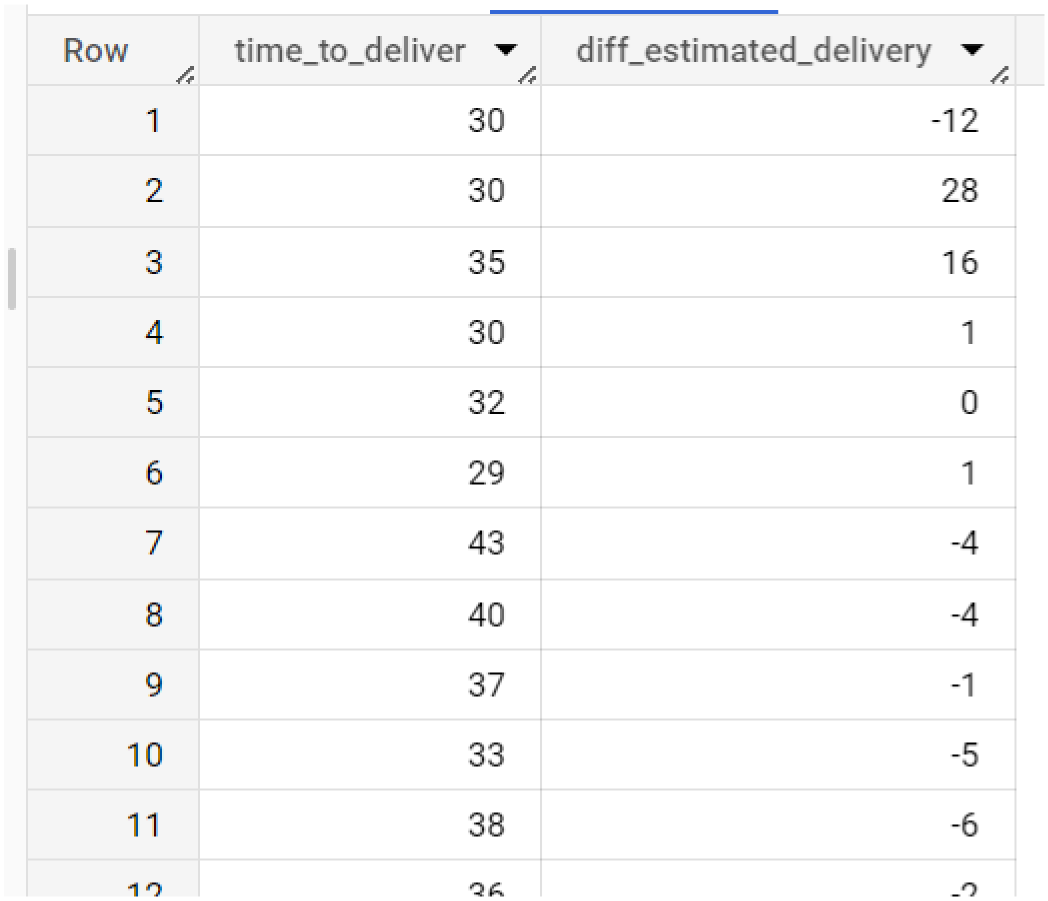
5) Analysis based on sales, freight and delivery time:

5.1) Calculating the number of days taken to deliver each order from the order's purchase date. Additionally, determining the difference (in days) between the estimated and actual delivery date of an order:

Query:

```
SELECT
  TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)
AS time_to_deliver,
  TIMESTAMP_DIFF(order_estimated_delivery_date, order_delivered_customer_date,
DAY) AS diff_estimated_delivery
FROM `target.orders`;
```

Output:



Row	time_to_deliver	diff_estimated_delivery
1	30	-12
2	30	28
3	35	16
4	30	1
5	32	0
6	29	1
7	43	-4
8	40	-4
9	37	-1
10	33	-5
11	38	-6
12	36	-2

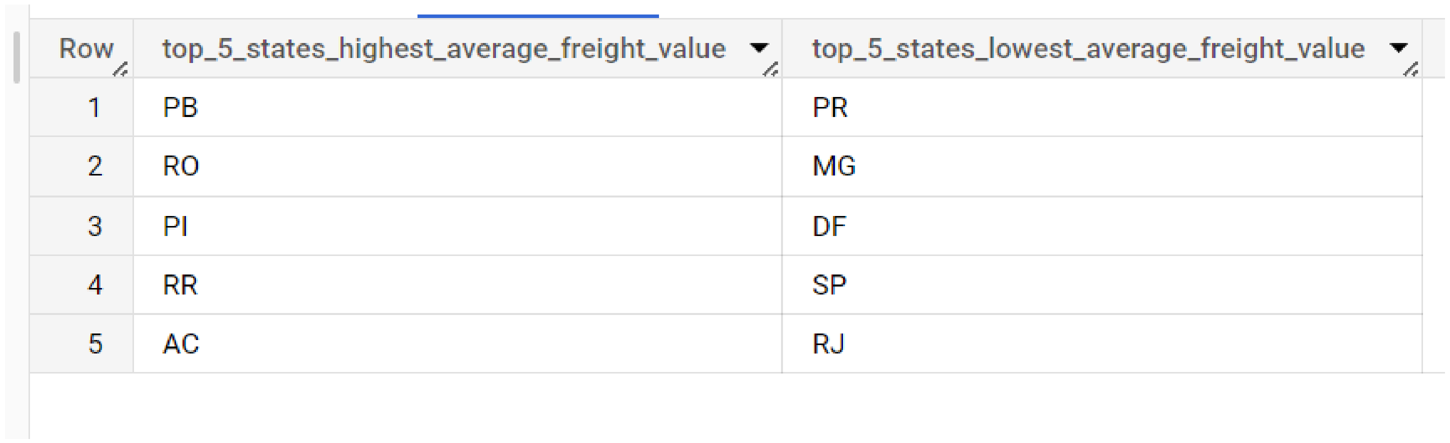
5.2) Finding the top 5 states with the highest and lowest average freight value:

Query:

```
WITH ranking AS (SELECT
  c.customer_state,
  AVG(oi.freight_value) AS average_order_freight_value_per_each_state,
  rank() over(order by AVG(oi.freight_value) desc) rank_high,
  rank() over(order by AVG(oi.freight_value) asc) rank_low
FROM
  `target.order_items` oi
JOIN
  `target.orders` o ON o.order_id=oi.order_id
JOIN
  `target.customers` c ON o.customer_id = c.customer_id
GROUP BY
  c.customer_state), top5_high AS
(SELECT customer_state,rank_high rnk from ranking
where rank_high <= 5), top5_low AS
(SELECT customer_state,rank_low rnk from ranking
where rank_low <= 5)
SELECT h.customer_state AS top_5_states_highest_average_freight_value,
l.customer_state AS top_5_states_lowest_average_freight_value from top5_high h
JOIN top5_low l
```

```
using(rnk);
```

Output:



Row	top_5_states_highest_average_freight_value	top_5_states_lowest_average_freight_value
1	PB	PR
2	RO	MG
3	PI	DF
4	RR	SP
5	AC	RJ

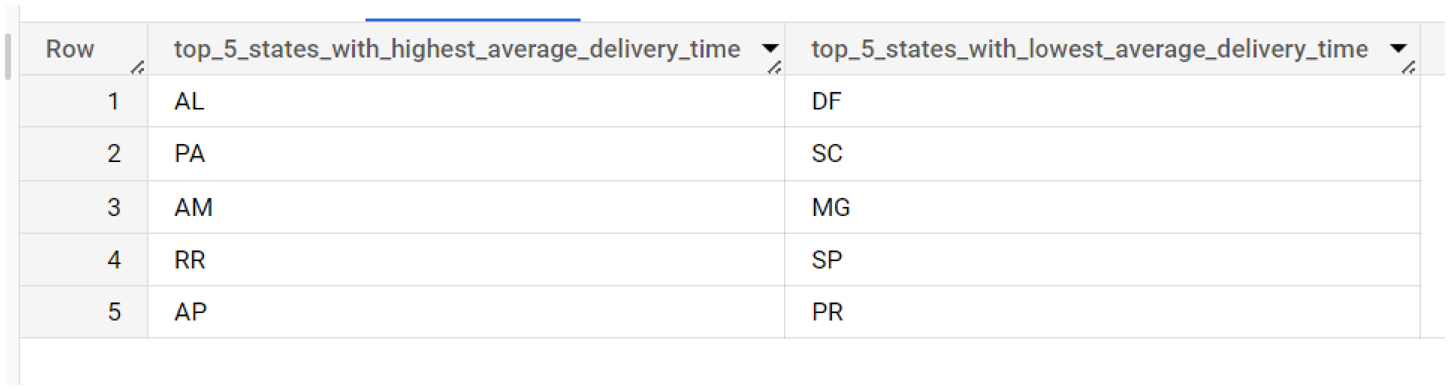
Insights & Recommendation:

States with higher total freight costs may require optimization strategies to reduce shipping expenses and improve supply chain management. Target can focus on potential cost-saving opportunities and enhance delivery services to meet customer expectations.

5.3) Finding the top 5 states with the highest and lowest average delivery time:

```
WITH ranking AS (select
  c.customer_state,
  AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date,
o.order_purchase_timestamp, SECOND)) AS average_time_to_deliver,
  rank() over(order by AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date,
o.order_purchase_timestamp, SECOND)) desc) AS rank_high,
  rank() over(order by AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date,
o.order_purchase_timestamp, SECOND))) AS rank_low
from `target.customers` c
JOIN `target.orders` o
on o.customer_id=c.customer_id
  WHERE
    order_delivered_customer_date IS NOT NULL
    AND order_purchase_timestamp IS NOT NULL
group by c.customer_state),
      states_highest AS
    (SELECT customer_state,rank_high AS rnk from ranking where rank_high <=5),
      states_lowest AS
    (SELECT customer_state,rank_low AS rnk from ranking where rank_low <=5)
SELECT h.customer_state AS top_5_states_with_highest_average_delivery_time,
      l.customer_state AS top_5_states_with_lowest_average_delivery_time
      FROM states_highest h JOIN states_lowest l using(rnk);
```

Output:



Row	top_5_states_with_highest_average_delivery_time	top_5_states_with_lowest_average_delivery_time
1	AL	DF
2	PA	SC
3	AM	MG
4	RR	SP
5	AP	PR

Insights & Recommendation:

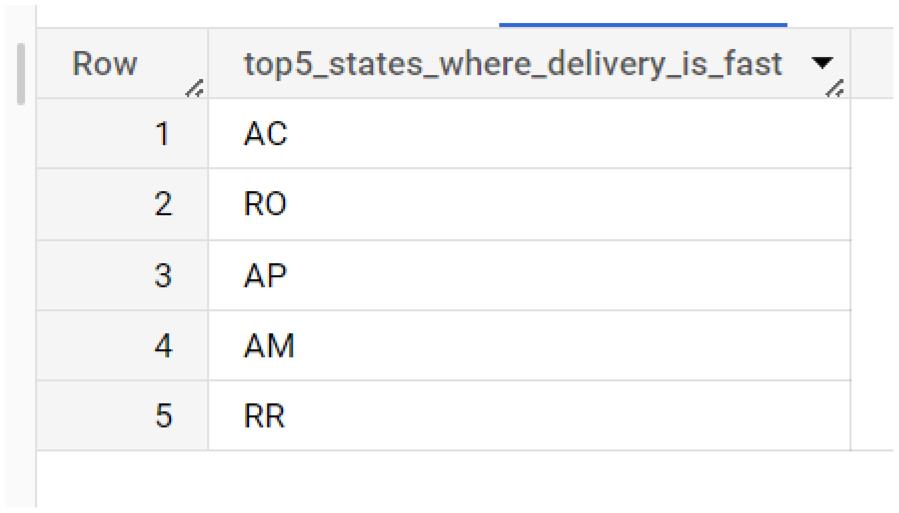
Shorter delivery times positively impact customer experience, leading to higher customer retention rates and increased loyalty. For the states where average delivery time is higher, implementing faster and more reliable shipping options, optimizing inventory management, and collaborating with efficient carriers can lead to improved delivery performance.

5.4) Finding the top 5 states where the order delivery is significantly faster compared to the estimated date of delivery:

Query:

```
SELECT
  customer_state AS top5_states_where_delivery_is_fast
FROM (
  SELECT
    c.customer_state,
    AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date,
o.order_estimated_delivery_date, SECOND)) AS average_time_diff,
  FROM
    `target.customers` c
  JOIN
    `target.orders` o
  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
    average_time_diff
  LIMIT
    5) t;
```

Output:



Row	top5_states_where_delivery_is_fast
1	AC
2	RO
3	AP
4	AM
5	RR

---

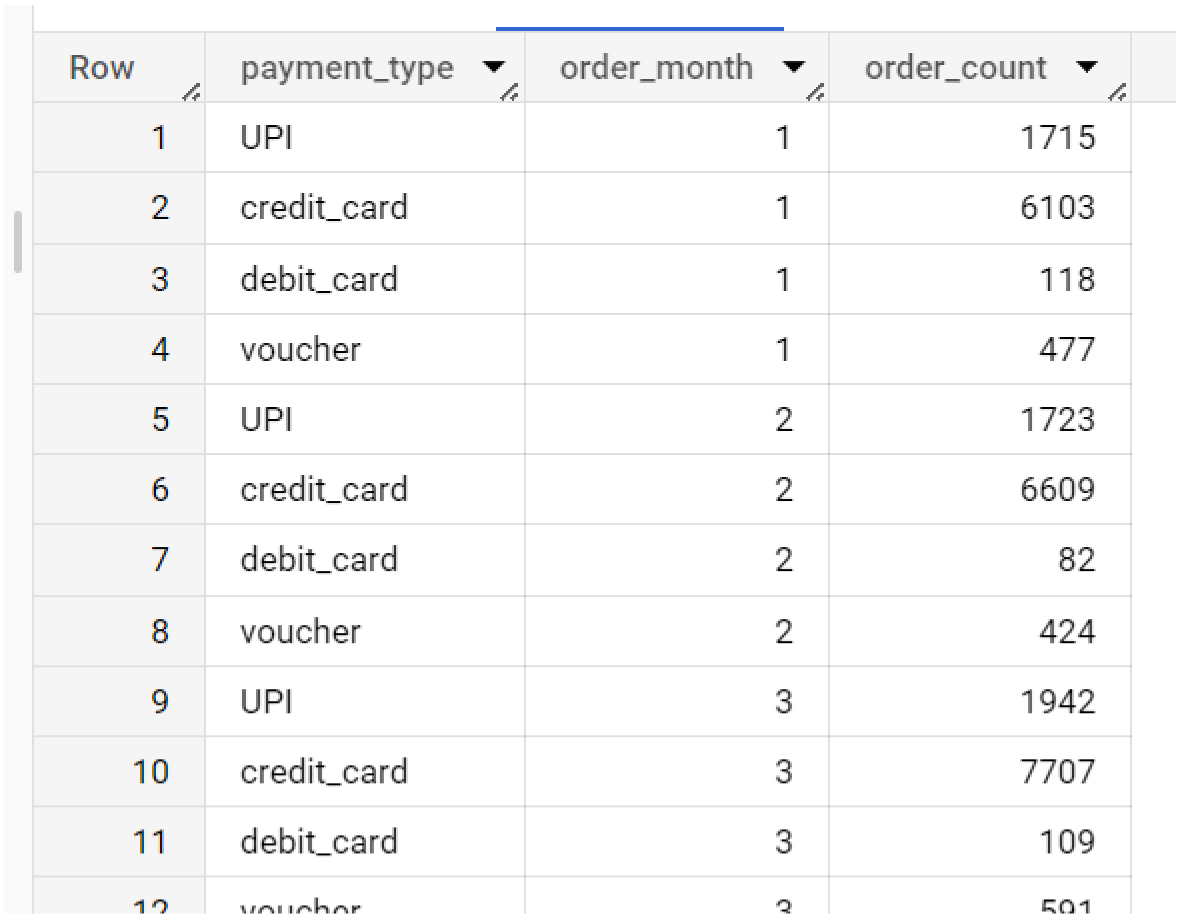
6) Analysis based on the payments:

6.1) Analyzing the month-on-month number of orders placed using different payment types:

Query:

```
SELECT
  p.payment_type,
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
  COUNT(*) AS order_count
FROM
  `target.payments` p
JOIN
  `target.orders` o ON p.order_id = o.order_id
GROUP BY
  payment_type,
  order_month
ORDER BY
  order_month,
  payment_type;
```

Output:



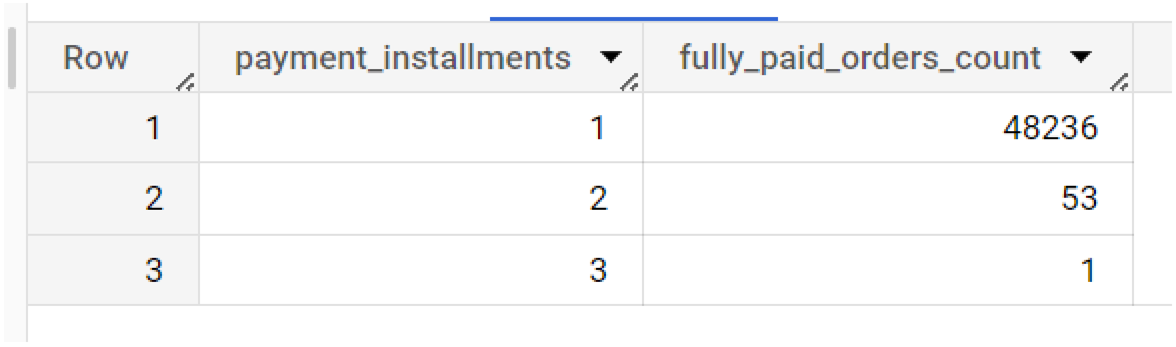
Row	payment_type	order_month	order_count
1	UPI	1	1715
2	credit_card	1	6103
3	debit_card	1	118
4	voucher	1	477
5	UPI	2	1723
6	credit_card	2	6609
7	debit_card	2	82
8	voucher	2	424
9	UPI	3	1942
10	credit_card	3	7707
11	debit_card	3	109
12	voucher	3	591

6.2) The number of orders placed based on the payment installments that have been paid:

Query:

```
SELECT
  payment_installments,
  COUNT(order_id) AS fully_paid_orders_count
FROM
  `target.payments`
WHERE payment_installments = payment_sequential
GROUP BY
  payment_installments
```

Output:

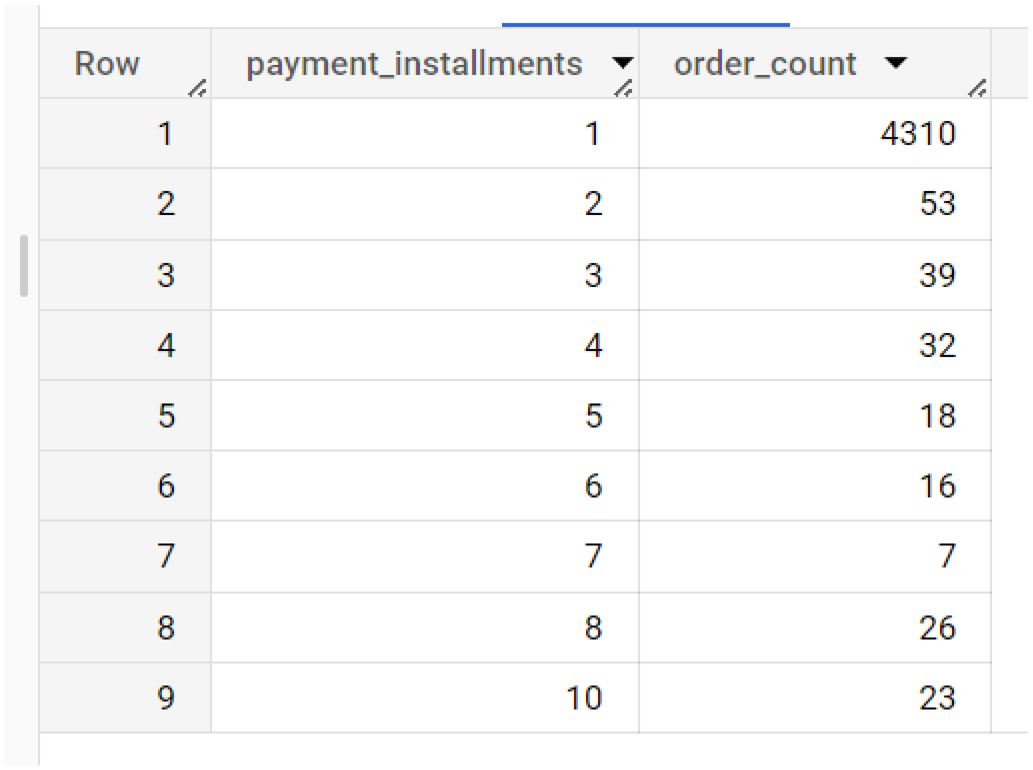


Row	payment_installments	fully_paid_orders_count
1	1	48236
2	2	53
3	3	1

Query:

```
SELECT
  payment_installments,
  COUNT(order_id) AS order_count
FROM
  `target.payments`
WHERE
  payment_sequential > 1 and payment_installments >=1
GROUP BY
  payment_installments
ORDER BY
  payment_installments;
```

Output:



Row	payment_installments	order_count
1	1	4310
2	2	53
3	3	39
4	4	32
5	5	18
6	6	16
7	7	7
8	8	26
9	10	23