

BUSINESS CASE – TARGET SQL

Ananya Jayaprakash | December 2023

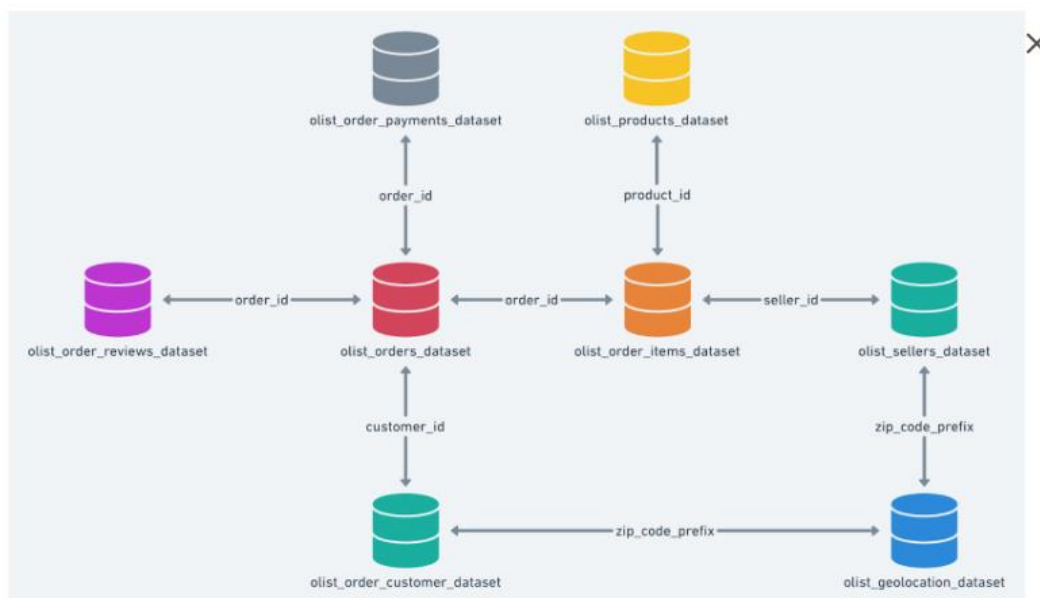
This project focuses on analyzing the e-commerce operations of Target in Brazil, utilizing a comprehensive dataset FROM 2016 to 2018. The goal is to extract valuable insights into order trends, customer behavior, and economic impacts.

About company

Target is a globally recognized retailer in the United States, known for its exceptional value and guest experience. 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 analysis leverages data FROM 100,000 orders to explore various aspects such AS order status, pricing strategies, payment performance, and customer satisfaction in Brazil.

Tools and Techniques Used

- SQL queries for data extraction and analysis.
- Data cleaning and preparation using Excel.
- MySQL Workbench for managing and querying the database.



1) Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:

1. Data type of all the columns in the 'customers' table

```
SELECT column_name, data_type
FROM `target.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'customers';
```

The screenshot shows a SQL query editor interface with a toolbar at the top containing icons for home, search, and file management. The editor has tabs for 'Untitled', '*Untitled 2', 'target', and 'Untitled 3'. The 'Untitled 2' tab is active, showing a SQL query. Below the editor is a 'Query results' section with a toolbar for 'SAVE RESULTS', 'EXPLORE DATA', and a refresh icon. The results are displayed in a table with columns 'column_name' and 'data_type'. The table has 5 rows of data. Below the table is a 'Job history' section with a 'REFRESH' button.

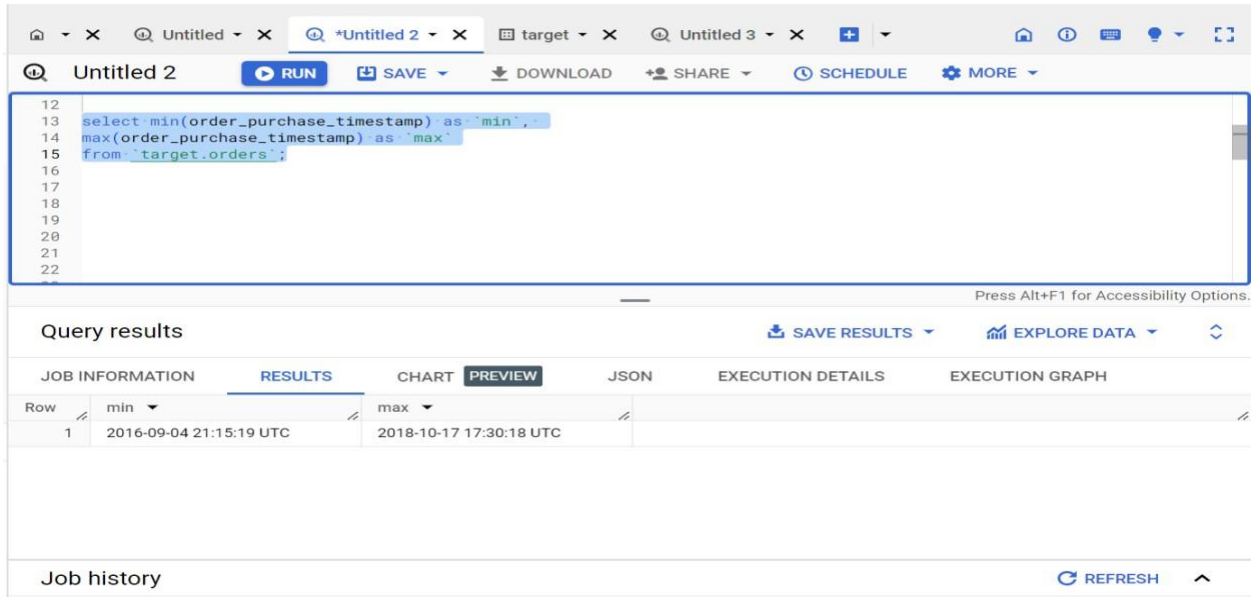
Query results

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

Job history

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

```
SELECT MIN (order_purchase_timestamp) AS `min`,  
MAX (order_purchase_timestamp) AS `max`  
FROM `target.orders`;
```



The screenshot shows a web-based SQL interface. At the top, there's a toolbar with buttons for RUN, SAVE, DOWNLOAD, SHARE, SCHEDULE, and MORE. Below the toolbar is a text area containing the SQL query: `select min(order_purchase_timestamp) as `min`, max(order_purchase_timestamp) as `max` from `target.orders`;`. The query is highlighted in blue. Below the text area is a section titled "Query results" with tabs for JOB INFORMATION, RESULTS, CHART, PREVIEW, JSON, EXECUTION DETAILS, and EXECUTION GRAPH. The RESULTS tab is active, showing a table with two columns: "min" and "max". The "min" column contains the value "2016-09-04 21:15:19 UTC" and the "max" column contains the value "2018-10-17 17:30:18 UTC". At the bottom of the interface is a "Job history" section with a REFRESH button.

Row	min	max
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

INSIGHTS

- Order Temporal Span:** The orders span FROM September 4, 2016, to October 17, 2018, indicating a historical perspective of customer transactions.
- Historical Order Patterns:** Analyze order patterns over time to identify trends, seasonality, or any significant events that might have influenced customer behavior.

RECOMMENDATIONS

- Seasonal Promotions:** Plan marketing promotions or campaigns based on historical order patterns. Identify seasons or specific time periods with increased order activity for targeted promotions.
- Inventory Planning:** Optimize inventory management based on historical order data. Anticipate high-demand periods and ensure appropriate stock levels to meet customer needs.

3. COUNT the cities and names of the customers who ordered during the given period.

```
SELECT COUNT (DISTINCT c.customer_city) AS `cities`, COUNT(DISTINCT c.customer_state) AS `states`  
FROM `target.customers` AS `c`  
INNER JOIN `target.orders` AS `o`  
ON c.customer_id = o.customer_id;
```

The screenshot shows a web-based SQL interface. At the top, there's a toolbar with buttons for RUN, SAVE, DOWNLOAD, SHARE, SCHEDULE, and MORE. Below the toolbar is a text area containing the SQL query. The query is: `select count(distinct c.customer_city) as `cities`, count(distinct c.customer_state) as `states` from `target.customers` as `c` inner join `target.orders` as `o` on c.customer_id = o.customer_id;`. Below the query editor, there's a section titled "Query results" with tabs for JOB INFORMATION, RESULTS, CHART, PREVIEW, JSON, EXECUTION DETAILS, and EXECUTION GRAPH. The RESULTS tab is active, showing a table with two columns: "cities" and "states". The first row shows 4119 for cities and 27 for states. At the bottom, there's a "Job history" section with a REFRESH button.

Row	cities	states
1	4119	27

INSIGHTS

- a. **City count:** There are 4,119 DISTINCT cities represented in the dataset where customers have placed orders. This indicates a broad geographical distribution of customers.
- b. **State count:** Customers who placed orders are spread across 27 DISTINCT states. The presence of orders in multiple states suggests a diverse customer base.

RECOMMENDATIONS

- a. **Geographic Targeting:** Leverage the insights on city and state Counts to inform geographic targeting strategies. Consider tailoring marketing campaigns or promotions to specific cities or states based on order distribution.
- b. **Localized Marketing:** Explore opportunities for localized marketing efforts. Understand the preferences and needs of customers in specific cities or states to create more personalized and effective marketing messages.
- c. **Expansion Opportunities:** Identify areas with lower order activity or untapped markets. Consider strategies to expand and increase market share in those regions.

2) In-depth Exploration:

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

```
SELECT EXTRACT(year FROM order_purchase_timestamp)`year`,  
COUNT(order_id) `no_of_orders`  
FROM `target.orders` GROUP BY 1 ORDER BY 1;
```

The screenshot shows a SQL query editor interface. The query is as follows:

```
select extract(year from order_purchase_timestamp)`year`, count(order_id) `no_of_orders`  
from `target.orders`  
group by 1  
order by 1;
```

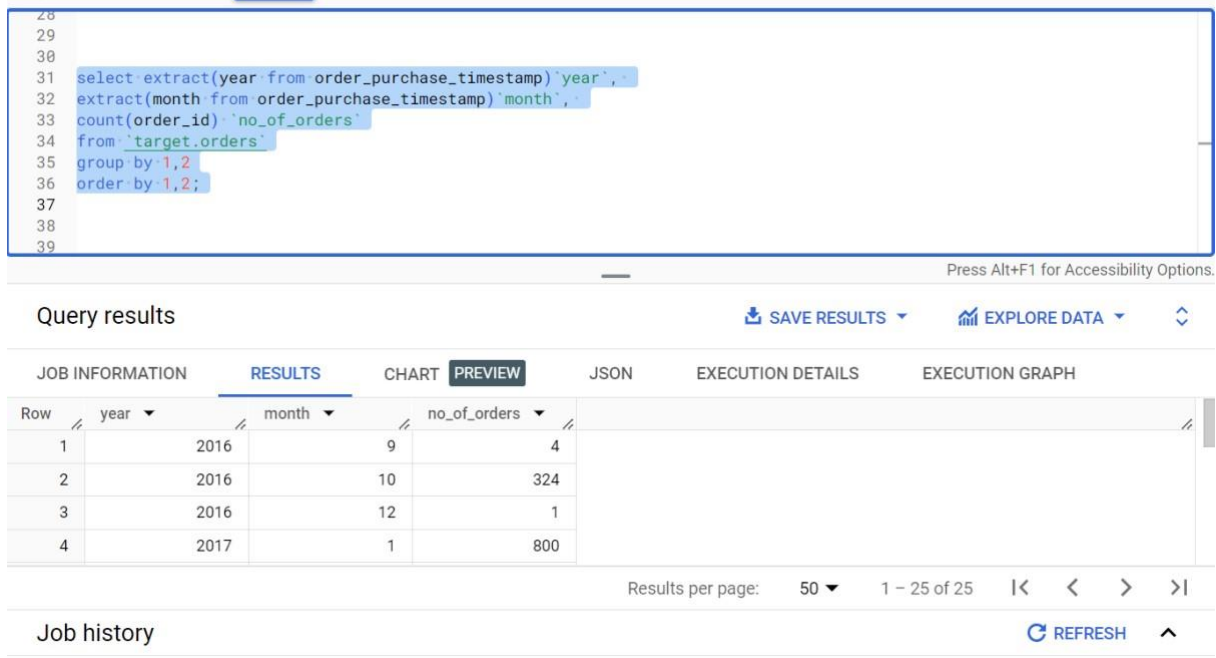
Below the query editor, the "Query results" section is displayed. It includes tabs for "JOB INFORMATION", "RESULTS", "CHART", "PREVIEW", "JSON", "EXECUTION DETAILS", and "EXECUTION GRAPH". The "RESULTS" tab is active, showing a table with the following data:

Row	year	no_of_orders
1	2016	329
2	2017	45101
3	2018	54011

At the bottom of the interface, there is a "Job history" section with a "REFRESH" button.

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

```
SELECT EXTRACT(year FROM order_purchase_timestamp)`year`,  
EXTRACT(month FROM order_purchase_timestamp)`month`,  
COUNT(order_id) `no_of_orders`  
FROM `target.orders`  
GROUP BY 1, 2  
ORDER BY 1, 2;
```



Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS CHART PREVIEW JSON EXECUTION DETAILS EXECUTION GRAPH

Row	year	month	no_of_orders
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800

Results per page: 50 1 - 25 of 25

Job history REFRESH

INSIGHTS

- Temporal distribution:** The query provides a breakdown of the number of orders for each month across different years. This information helps visualize the temporal distribution of order activity over time.
- Seasonal Trends:** Analyze the monthly order Counts to identify any seasonal trends. Higher order Counts during specific months may indicate seasonality or special events that influence customer purchasing behavior.

RECOMMENDATIONS

- Seasonal Campaigns:** Plan marketing campaigns or promotions based on seasonal trends. Tailor promotions to align with months that historically have higher order activity.
- Inventory Planning:** Use insights FROM monthly order Counts to optimize inventory planning. Anticipate peaks in demand during certain months and adjust stock levels accordingly.
- Customer Engagement:** Implement targeted customer engagement strategies during months with higher order activity. This could include special promotions, loyalty programs, or personalized communication.

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 'morning'  
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_of_day, COUNT(*) AS order_count
```

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

```
GROUP BY time_of_day
```

```
ORDER BY time_of_day;
```

The screenshot shows a SQL query editor with a query that categorizes orders by time of day and counts them. Below the editor, the query results are displayed in a table with columns 'time_of_day' and 'order_count'.

Query results

Row	time_of_day	order_count
1	Afternoon	38135
2	Dawn	5242
3	Morning	27733
4	Night	28331

INSIGHTS

- a. **Order Distribution Across Time of The Day:** The query categorizes orders into different times of the day Afternoon, Dawn, Morning, and Night.
- b. **Order count Breakdown:** The COUNT of orders is provided for each time-of-day category, offering insights into when customers tend to make purchases.

RECOMMENDATIONS

- a. **Time-based Marketing Strategies:** Leverage the insights on order Counts across different times of the day to implement time-based marketing strategies. For example, consider targeted promotions or advertisements during periods with higher order Counts.
- b. **Personalized Communication:** Tailor communication strategies based on the time of day. Different times may attract different customer segments, and personalized messaging during those times can enhance customer engagement.
- c. **Optimize AD Campaigns:** Optimize online advertising campaigns by targeting specific times of the day when order activity is high. This can maximize the impact of ad spending and improve conversion rates.

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

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

```
SELECT EXTRACT(month FROM order_purchase_timestamp) AS order_month,
extract (year FROM order_purchase_timestamp) AS order_year, customer_state,
COUNT(*) AS order_count FROM `target.orders` AS `o` join `target.customers` AS
`c` on o.customer_id = c.customer_id
GROUP BY order_year, order_month, customer_state ORDER BY
order_year, order_month, customer_state;
```

The screenshot shows a SQL query editor with a query that extracts month and year from the purchase timestamp, joins orders with customers, and counts orders by month, year, and state. Below the query, the 'Query results' section is displayed with a table of results.

Query results					
JOB INFORMATION RESULTS CHART PREVIEW JSON EXECUTION DETAILS EXECUTION GRAPH					
Row	order_month	order_year	customer_state	order_count	
1	9	2016	RR	1	
2	9	2016	RS	1	
3	9	2016	SP	2	
4	10	2016	AL	2	
5	10	2016	BA	4	
6	10	2016	CE	8	
7	10	2016	DF	6	
8	10	2016	ES	4	
9	10	2016	GO	9	
10	10	2016	MA	1	

INSIGHTS

- Monthly and Yearly Order Breakdown:** The query breaks down order counts by month and year, providing insights into the monthly and yearly distribution of orders.
- Geographic Order Distribution:** Order counts are further categorized by customer state, offering insights into the geographic distribution of orders across different regions.

RECOMMENDATIONS

- Seasonal Marketing Campaigns:** Leverage the monthly and yearly order breakdown to plan seasonal marketing campaigns. Identify peak months and design promotions or advertisements tailored to customer behavior during those periods.
- Regional Targeting:** Utilize the geographic order distribution to target specific regions for marketing initiatives. Tailor promotional strategies based on the order patterns observed in different customer states.

2. How are the customers distributed across all the states?

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

The screenshot shows a SQL query editor with a query that counts distinct customer IDs by state. Below the editor, the query results are displayed in a table. The table has two columns: 'customer_state' and 'customer_count'. The results are ordered by customer count in descending order. The states and their corresponding counts are: SP (41746), RJ (12852), MG (11635), RS (5466), PR (5045), SC (3637), BA (3380), DF (2140), ES (2033), GO (2020), and PE (1652).

Row	customer_state	customer_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

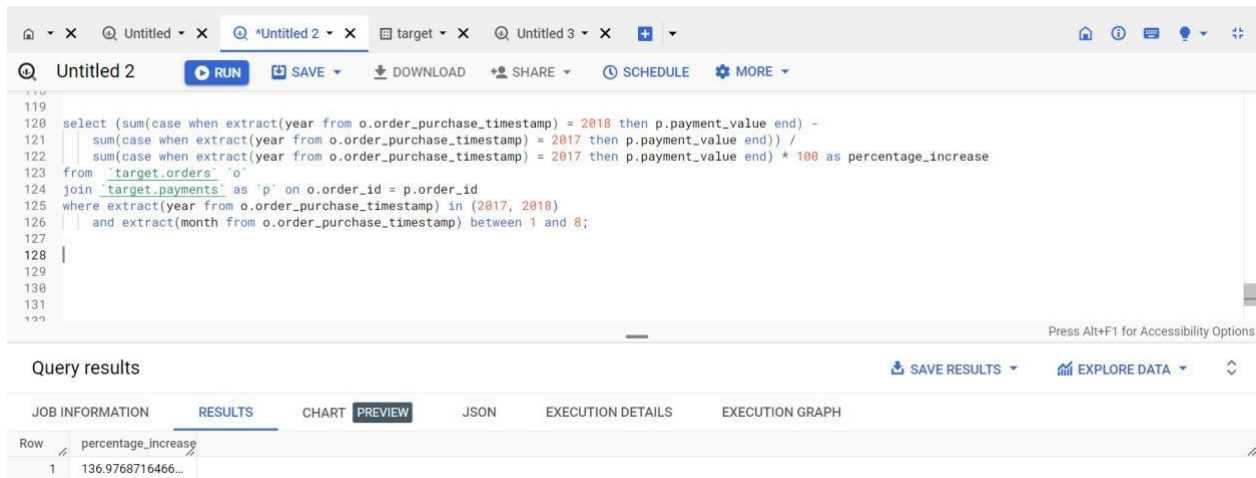
INSIGHTS

- Customer Distribution by State:** The query provides insights into the distribution of customers across different states, showcasing the COUNT of unique customer IDs for each state.
- Regional Variation in Customer Count:** There is significant variation in customer Counts among different states, with some states having a higher concentration of customers compared to others.

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

1. Get the % increase in the cost of orders FROM year 2017 to 2018 (include months between Jan to Aug only).

```
SELECT (SUM(CASE WHEN EXTRACT(year FROM o.order_purchase_timestamp) = 2018 THEN p.payment_value END) -  
SUM(CASE WHEN EXTRACT(year FROM o.order_purchase_timestamp) = 2017 THEN p.payment_value END)) /  
SUM(CASE WHEN EXTRACT(year FROM o.order_purchase_timestamp) = 2017 THEN p.payment_value END) * 100 AS  
percentage_increase FROM `target.orders` `o`  
JOIN `target.payments` AS `p` ON o.order_id = p.order_id  
WHERE EXTRACT(year FROM o.order_purchase_timestamp) IN (2017, 2018)  
AND EXTRACT(month FROM o.order_purchase_timestamp) BETWEEN 1 AND 8;
```



The screenshot shows a SQL query editor with the following query:

```
select (sum(case when extract(year from o.order_purchase_timestamp) = 2018 then p.payment_value end) -  
sum(case when extract(year from o.order_purchase_timestamp) = 2017 then p.payment_value end)) /  
sum(case when extract(year from o.order_purchase_timestamp) = 2017 then p.payment_value end) * 100 as percentage_increase  
from `target.orders` `o`  
join `target.payments` as `p` on o.order_id = p.order_id  
where extract(year from o.order_purchase_timestamp) in (2017, 2018)  
and extract(month from o.order_purchase_timestamp) between 1 and 8;
```

The query results are displayed in a table with the following data:

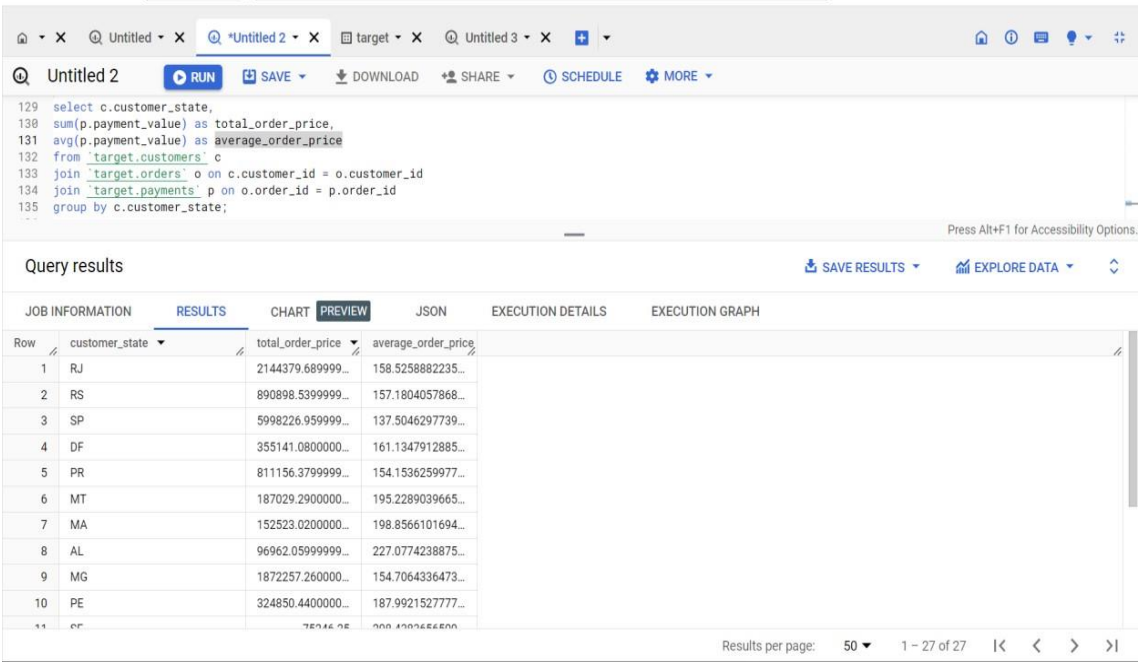
Row	percentage_increase
1	136.9768716466...

INSIGHTS

- a. Year-Over-Year Payment Increase: The query calculates the percentage increase in payments FROM 2017 to 2018 for orders placed between January and August. The result indicates a substantial increase of approximately 136.98%.

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

```
SELECT c.customer_state, SUM(p.payment_value) AS total_order_price,
AVG(p.payment_value) AS average_order_price
FROM `target.customers` c
JOIN `target.orders` o on c.customer_id = o.customer_id
JOIN `target.payments` p on o.order_id = p.order_id
GROUP BY c.customer_state;
```



INSIGHTS

- a. **Total Order Value by State:** The query provides the total order value and average order price for each customer state.
- b. **Variation in Average Order Price:** There is variation in average order prices among different states, indicating potential differences in customer spending habits or preferences.

RECOMMENDATIONS

- a. **Regional Marketing strategies:** Tailor marketing strategies based on the observed differences in average order prices across states. Consider region-specific promotions or campaigns to appeal to diverse customer preferences.
- b. **Customer Segmentations:** Segment customers based on their state and analyze their purchasing behavior. This can help in creating targeted marketing messages and offers for specific customer segments.
- c. **Localized Product Offerings:** Assess whether the product offerings align with the preferences of customers in each state. Consider introducing or highlighting products that resonate well with the customer base in a particular region.

- d. **Customer Engagement Program:** Implement customer engagement programs that encourage repeat purchases. This can include loyalty programs, personalized recommendations, or exclusive offers to enhance customer retention.
- e. **Price Optimization:** Evaluate the pricing strategy for products in different states. Adjust prices based on local market conditions, competitor pricing, and customer sensitivity to enhance competitiveness.

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

```
SELECT c.customer_state, sum(oi.freight_value) AS total_order_freight,  
avg(oi.freight_value) AS average_order_freight FROM `target.customers` c  
join `target.orders` o on c.customer_id = o.customer_id join  
`target.order_items` oi on o.order_id = oi.order_id  
GROUP BY c.customer_state;
```

The screenshot shows a SQL query editor with the following query:

```
select c.customer_state, sum(oi.freight_value) as total_order_freight,  
avg(oi.freight_value) as average_order_freight  
from `target.customers` c  
join `target.orders` o on c.customer_id = o.customer_id  
join `target.order_items` oi on o.order_id = oi.order_id  
group by c.customer_state;
```

Below the query editor, the 'Query results' section displays a table with the following data:

Row	customer_state	total_order_freight	average_order_freight
1	MT	29715.43000000...	28.16628436018...
2	MA	31523.77000000...	38.25700242718...
3	AL	15914.58999999...	35.84367117117...
4	SP	718723.0699999...	15.14727539041...
5	MG	270853.4600000...	20.63016680630...
6	PE	59449.65999999...	32.91786267995...
7	RJ	305589.3100000...	20.96092393168...
8	DF	50625.49999999...	21.04135494596...
9	RS	135522.7400000...	21.73580433039...
10	SE	14111.46999999...	36.65316883116...

INSIGHTS

- Total Order Freight by State:** freight and average order freight for each customer state.
- Variation in Average Order Freight:** There is variation in average order freight among different states, suggesting differences in shipping costs or order fulfillment methods.

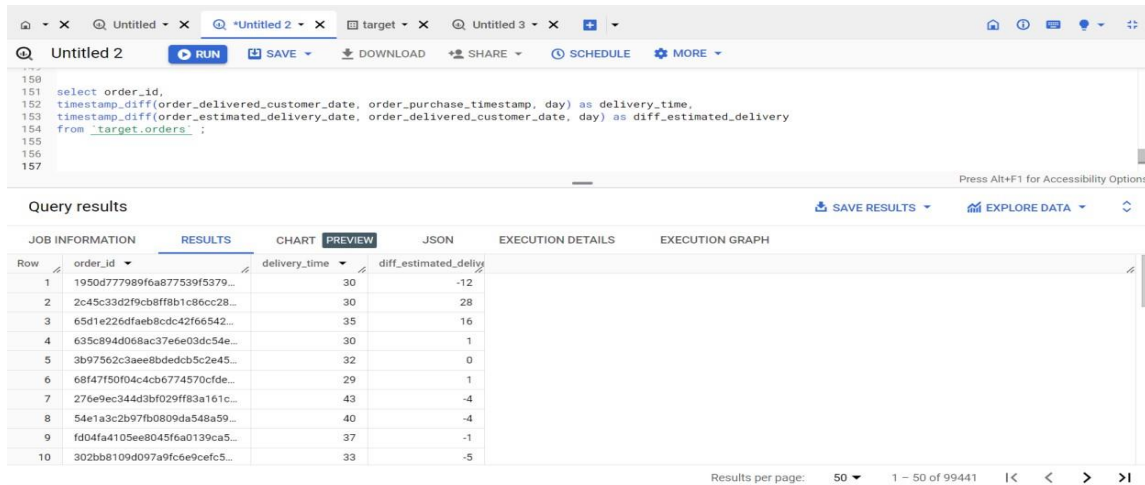
RECOMMENDATIONS

- Shipping Cost Optimization:** Evaluate shipping costs and logistics strategies in states with higher average order freight. Explore opportunities to optimize shipping routes, negotiate better freight rates, or introduce cost-effective shipping solutions.
- Regional Warehouse Planning:** Assess the Feasibility of regional warehouses to minimize shipping distances and reduce freight costs. Regional fulfillment centers can improve delivery times and enhance the overall customer experience.
- Free Shipping Thresholds:** Consider implementing free shipping thresholds for customers in states with higher average order freight. This can incentivize larger purchases and potentially offset shipping costs.
- Shipping Partnerships:** Explore partnerships with shipping carriers to negotiate favorable rates or explore discounted shipping programs. Collaborate with reliable carriers to optimize shipping processes.
- Packaging Efficiency:** Optimize packaging processes to minimize size and weight without compromising product safety. Efficient packaging can contribute to reduced freight costs and environmental sustainability.

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

1. 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.

```
SELECT order_id,  
timestamp_diff(order_delivered_customer_date, order_purchase_timestamp,  
day) AS delivery_time,  
timestamp_diff(order_estimated_delivery_date, order_delivered_customer_date,  
day) AS diff_estimated_delivery  
FROM `target.orders` ;
```



The screenshot shows a SQL query editor with a query window and a results window. The query window contains the following SQL code:

```
select order_id,  
timestamp_diff(order_delivered_customer_date, order_purchase_timestamp, day) as delivery_time,  
timestamp_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as diff_estimated_delivery  
from `target.orders` ;
```

The results window displays the following data:

Row	order_id	delivery_time	diff_estimated_delivery
1	1950d777989f6a877539f5379...	30	-12
2	2c45c33d2f9cb8f8b1c86cc28...	30	28
3	65d1e226d6aeb8cdc42f66542...	35	16
4	635c894d068ac37e6e03dc54e...	30	1
5	3b97562c3aee8bdedcb5c2e45...	32	0
6	68f47150f04c4cb6774570cfd...	29	-1
7	276e9ec344d3bf029ff83a161c...	43	-4
8	54e1a3c2b97fb0809da548a59...	40	-4
9	fd04fa4105ee8045f6a0139ca5...	37	-1
10	302bb8109d097a9fcae9cfc5...	33	-5

INSIGHTS

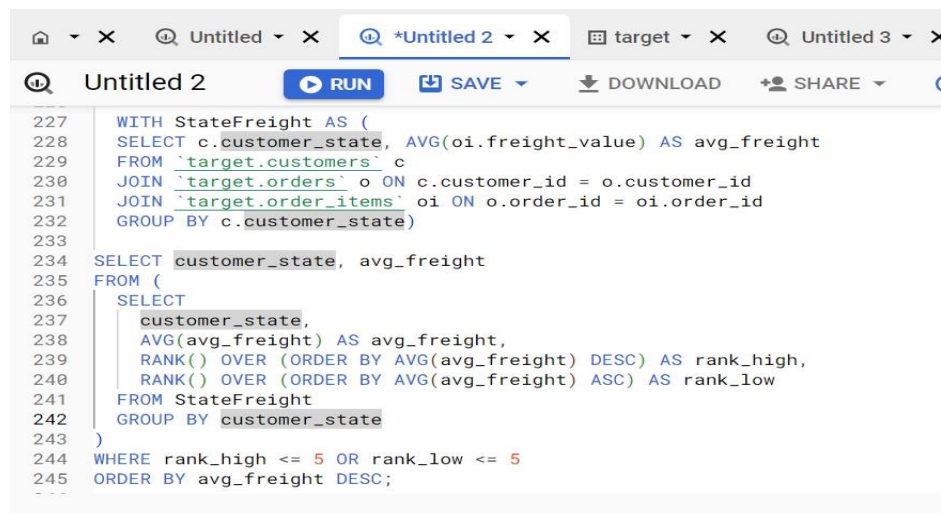
- Delivery Time Analysis:** The delivery_time column represents the number of days it took for an order to be delivered to the customer after the purchase. Positive values indicate delayed deliveries, while negative values indicate deliveries made before the estimated date.
- Difference From Estimated Delivery:** The diff_estimated_delivery column calculates the difference, in days, between the estimated delivery date and the actual delivery date. Positive values indicate deliveries made later than estimated, while negative values indicate deliveries made earlier.

RECOMMENDATIONS

- Improving Timeliness:** Focus on reducing the average delivery_time to enhance customer satisfaction. Implement strategies to streamline order processing, optimize logistics, and minimize delays in delivery.
- Estimated Delivery Accuracy:** Assess and refine the accuracy of estimated delivery dates. Enhance the precision of estimated delivery times to provide customers with reliable expectations, reducing the gap between estimated and actual delivery.
- Communication with Customers:** Implement proactive communication with customers regarding the status of their orders. Notify customers about any potential delays and provide real-time updates on the delivery process to manage expectations effectively.
- Identify Bottlenecks in Fulfillment:** Identify and address bottlenecks in the order fulfillment process. This may involve optimizing warehouse operations, improving inventory management, or collaborating with logistics partners to expedite shipments.

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

```
WITH statefreight AS (  
  SELECT c.customer_state, AVG(oi.freight_value) AS avg_freight  
  FROM `target.customers` c  
  JOIN `target.orders` o ON c.customer_id = o.customer_id  
  JOIN `target.order_items` oi ON o.order_id = oi.order_id  
  GROUP BY c.customer_state)  
  
  SELECT customer_state, avg_freight  
  FROM (SELECT customer_state, avg(avg_freight) AS avg_freight, RANK() OVER(ORDER BY  
    AVG(avg_freight) DESC) AS rank_high,  
    RANK() OVER(ORDER BY avg(avg_freight) ASC) AS rank_low  
  FROM statefreight  
  GROUP BY customer_state )  
  WHERE rank_high <= 5 or rank_low <= 5  
  ORDER BY avg_freight DESC;
```



The screenshot shows a SQL editor with a toolbar at the top containing icons for file operations and buttons for 'RUN', 'SAVE', 'DOWNLOAD', and 'SHARE'. The query code is displayed in a text area, with line numbers 227 through 245 visible on the left margin. The code defines a CTE named 'statefreight' and then uses it to calculate the average freight and rank for each state, filtering for the top 5 states by both highest and lowest average freight.

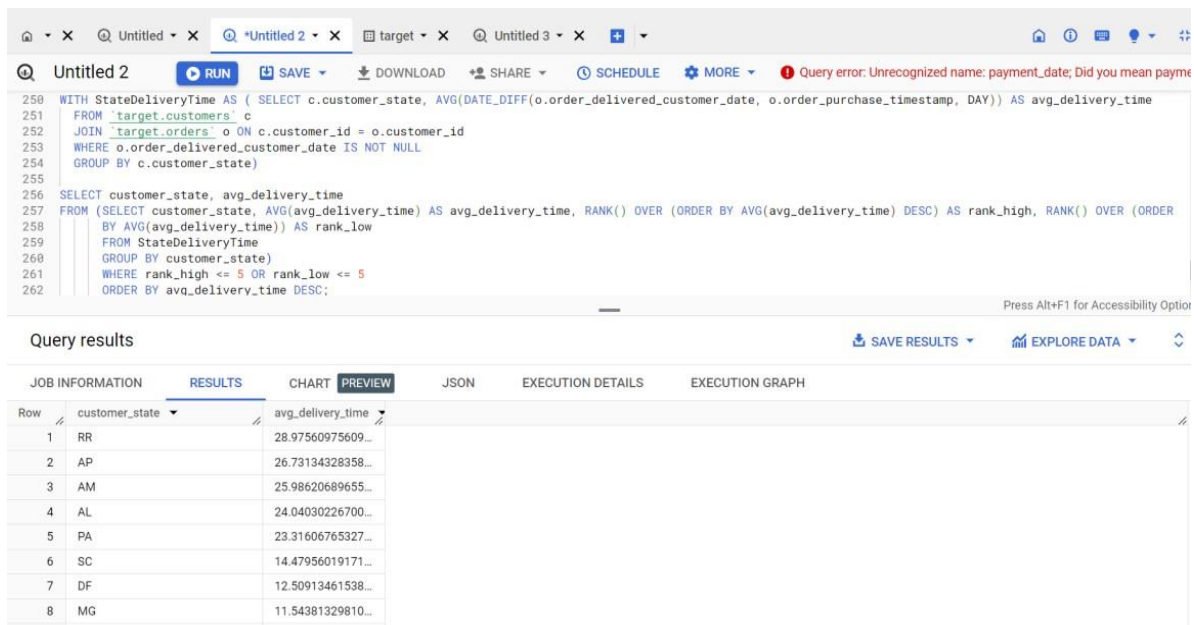
Query results

JOB INFORMATION		RESULTS	CHART	PREVIEW
Row	customer_state	avg_delivery_time		
1	RR	28.97560975609...		
2	AP	26.73134328358...		
3	AM	25.98620689655...		
4	AL	24.04030226700...		
5	PA	23.31606765327...		
6	SC	14.47956019171...		
7	DF	12.50913461538...		
8	MG	11.54381329810...		
9	PR	11.52671135486...		
10	SP	8.298061489072...		

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

```
WITH StateDeliveryTime AS ( SELECT c.customer_state,
AVG(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, day)) AS
avg_delivery_time
FROM `target.customers` c
JOIN `target.orders` o ON c.customer_id = o.customer_id
WHERE o.order_delivered_customer_date is not null
GROUP BY c.customer_state)

SELECT customer_state, avg_delivery_time
FROM (SELECT customer_state, AVG(avg_delivery_time) AS avg_delivery_time,
RANK() OVER(ORDER BY AVG(avg_delivery_time) DESC) AS rank_high,
rank() OVER(ORDER BY AVG(avg_delivery_time)) AS rank_low
FROM statedeliverytime
GROUP BY customer_state)
WHERE rank_high <= 5 or rank_low <= 5
ORDER BY avg_delivery_time DESC;
```



Query results

Row	customer_state	avg_delivery_time
1	RR	28.97560975609...
2	AP	26.73134328358...
3	AM	25.98620689655...
4	AL	24.04030226700...
5	PA	23.31606765327...
6	SC	14.47956019171...
7	DF	12.50913461538...
8	MG	11.54381329810...

INSIGHTS:

1. **Average Delivery Time by State:** The query calculates the average delivery time for each customer state, considering the time taken between order placement and actual delivery.
2. **Top and Bottom Rankings:** The results are filtered to display only the top 5 and bottom 5 states based on their average delivery times. These rankings are determined using the RANK() function.
3. **Variability in Delivery Performance:** There is notable variability in average delivery times among different states, ranging from shorter times in some states to longer times in others.

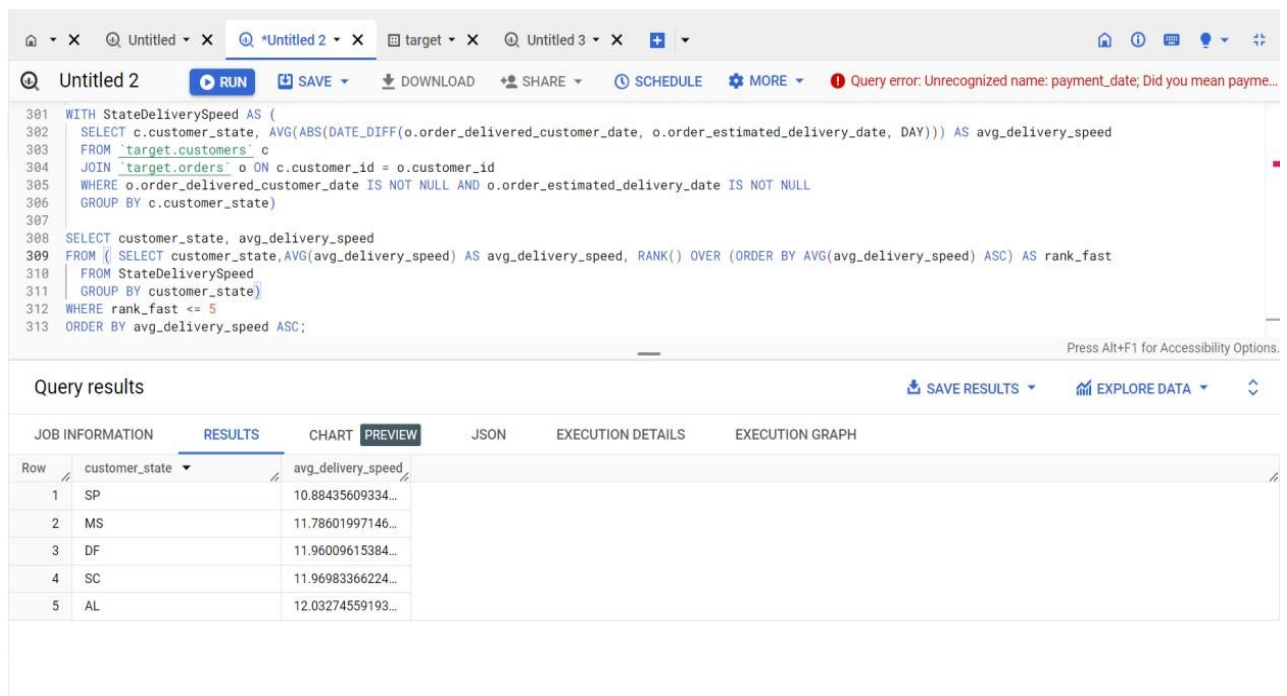
RECOMMENDATIONS

High-Performing States:

1. **Recognize and Reward Excellence:** Acknowledge and reward states with consistently high delivery performance. Recognition can boost morale and motivate teams to maintain or further improve their efficiency.
2. **Share Best Practices:** Facilitate knowledge-sharing sessions between high-performing states to disseminate best practices. Encourage collaboration and the adoption of successful strategies to optimize delivery times.
3. **Enhance Capacity as Needed:** Evaluate the demand and order volume in high-performing states. If needed, enhance operational capacity to sustain the efficient delivery times and meet potential increases in demand.

4. Find out the top 5 states where the order delivery is really fASt AS compared to the estimated date of delivery.

```
WITH StateDeliverySpeed AS (  
SELECT c.customer_state, AVG(ABS(date_diff(o.order_delivered_customer_date,  
o.order_estimated_delivery_date, day))) AS avg_delivery_speed  
FROM `target.customers` c  
JOIN `target.orders` o ON c.customer_id = o.customer_id  
WHERE o.order_delivered_customer_date is not null and  
o.order_estimated_delivery_date is not null  
GROUP BY c.customer_state)  
  
SELECT customer_state, avg_delivery_speed  
FROM ( SELECT customer_state,AVG(avg_delivery_speed) AS avg_delivery_speed,  
RANK() OVER(ORDER BY AVG(avg_delivery_speed) ASC) AS rank_fASt  
FROM StateDeliverySpeed GROUP BY customer_state)  
WHERE rank_fASt <= 5  
ORDER BY avg_delivery_speed ASC;
```



Query error: Unrecognized name: payment_date; Did you mean payme...

Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	customer_state	avg_delivery_speed					
1	SP	10.88435609334...					
2	MS	11.78601997146...					
3	DF	11.96009615384...					
4	SC	11.96983366224...					
5	AL	12.03274559193...					

INSIGHTS

1. **Fastest Delivery States:** SP has the best average delivery speed, which might indicate strong logistics or proximity to warehouses.
2. **Close Performance:** The difference in delivery speed among the top 5 states is relatively small (about 1.15 days between SP and AL). This suggests consistent delivery performance in these areas.
3. **Potential Delays:** Even the fastest delivery state, SP, averages over 10 days, which might indicate room for improvement, depending on customer expectations or industry standards.

RECOMMENDATIONS:

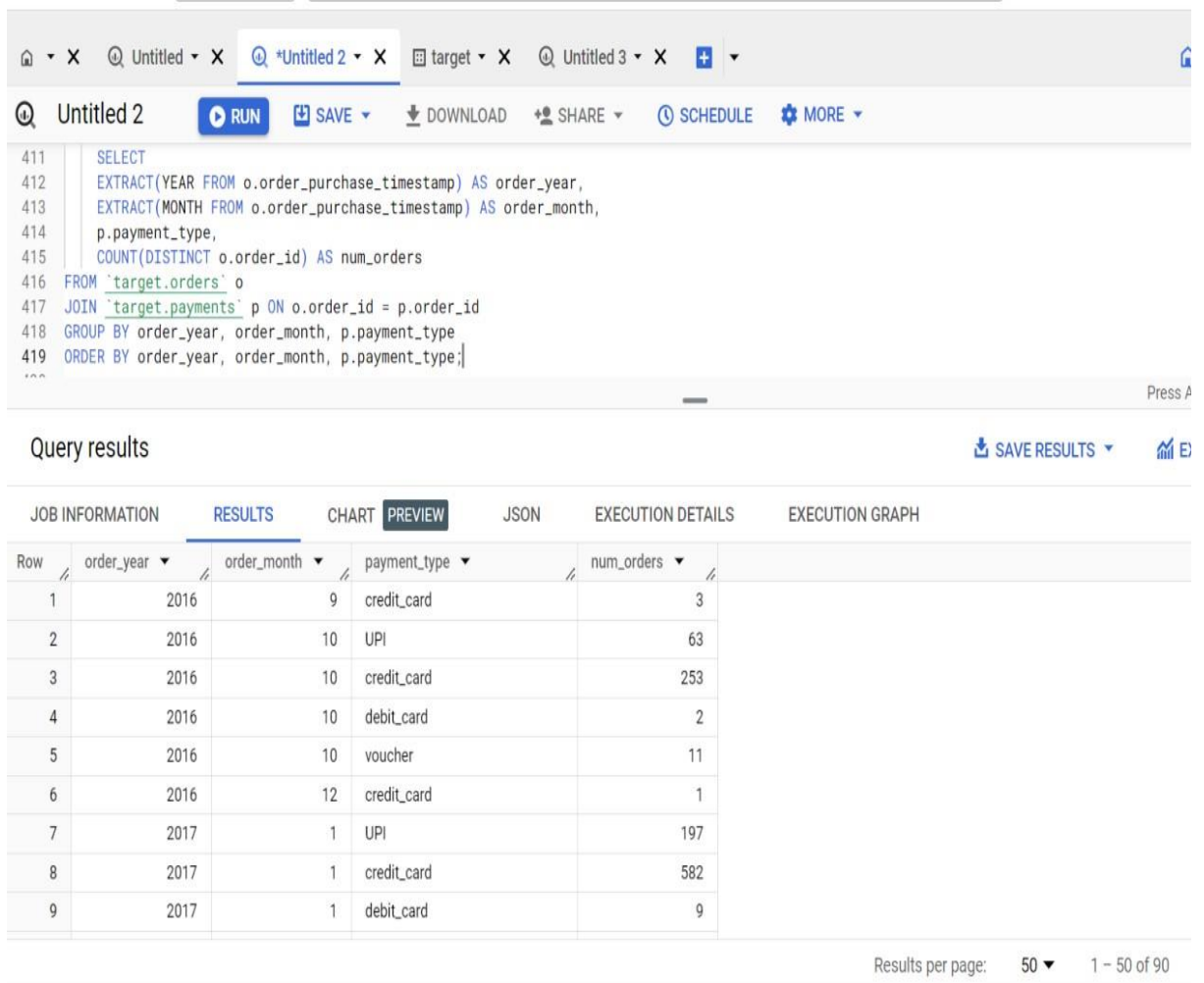
1. **Optimize Delivery in Other States:** While the top 5 states show fairly efficient delivery, the focus should be on improving delivery speeds in states outside of the top performers. Investigate whether delays are caused by logistical challenges, distance from warehouses, or other factors.
2. **Benchmarking Best Performers:** Use SP as a benchmark to analyze why it performs the best. It could be due to proximity to distribution centers, better transportation networks, or more efficient logistics. The strategies used in SP can be replicated in other regions to improve delivery speeds.
3. **Set Delivery Expectations:** Since the fastest average delivery time is still over 10 days, it might be helpful to adjust customer delivery expectations or explore ways to expedite shipping. Offering express delivery options or providing more accurate estimated delivery windows could improve customer satisfaction.
4. **Improve Estimated Delivery Accuracy:** There is likely a gap between estimated and actual delivery dates, as your query accounts for this comparison. Reassessing how delivery estimates are calculated (perhaps using machine learning for better predictions) could narrow this gap and lead to more accurate promises to customers.

6) Analysis based on the payments:

1. Find the month on month no. of orders placed using different payment types.

SELECT

```
EXTRACT(year FROM o.order_purchase_timestamp) AS order_year,  
EXTRACT(month FROM o.order_purchase_timestamp) AS order_month,  
p.payment_type,  
COUNT(DISTINCT o.order_id) AS num_orders  
FROM `target.orders` o  
JOIN `target.payments` p ON o.order_id = p.order_id  
GROUP BY order_year, order_month, p.payment_type  
ORDER BY order_year, order_month, p.payment_type;
```



Query results

SAVE RESULTS

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	order_year	order_month	payment_type	num_orders			
1	2016	9	credit_card	3			
2	2016	10	UPI	63			
3	2016	10	credit_card	253			
4	2016	10	debit_card	2			
5	2016	10	voucher	11			
6	2016	12	credit_card	1			
7	2017	1	UPI	197			
8	2017	1	credit_card	582			
9	2017	1	debit_card	9			

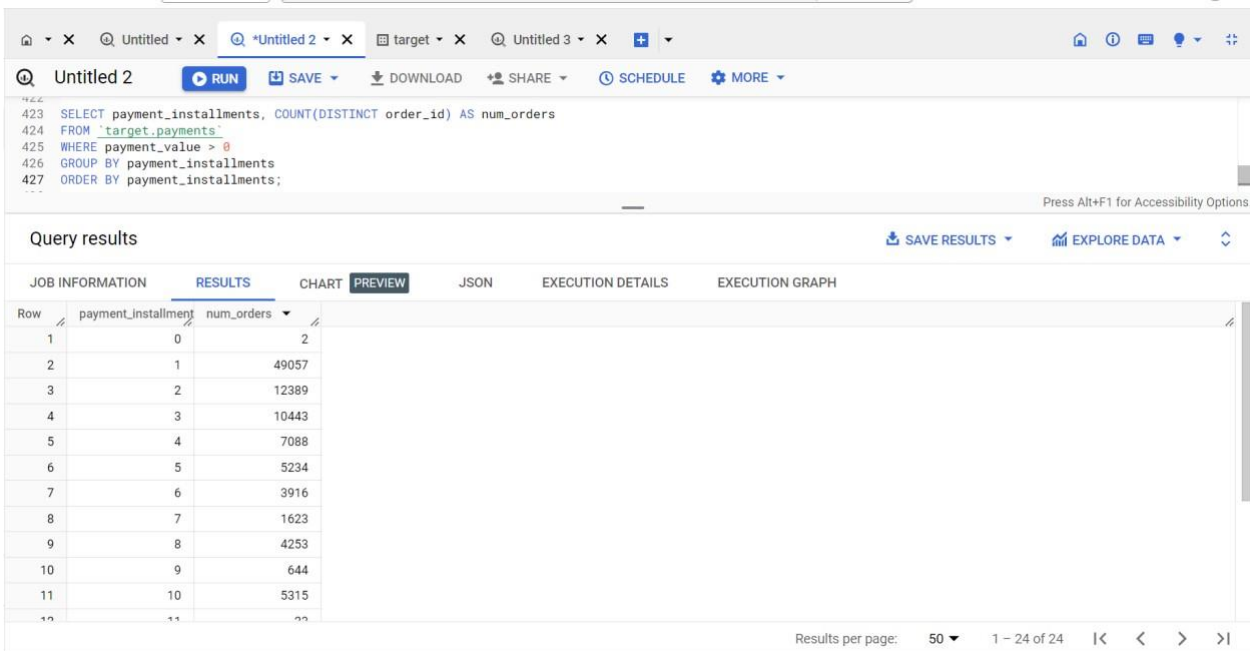
Results per page: 50 1 - 50 of 90

INSIGHTS

- a. **Temporal Distribution:** The results provide a breakdown of the number of orders (num_orders) for each combination of order_year and order_month. This allows you to observe how order volumes fluctuate over different months and years.
- b. **Payment Type Analysis:** The results further break down the order COUNT by payment_type, indicating the distribution of payment methods used by customers. This insight can be valuable for understanding the popularity of different payment types over time.
- c. **Trends and Patterns:** By examining the data over various months and years, you can identify trends and patterns in customer purchasing behavior. For example, certain payment types may become more prevalent during specific seasons or promotional periods.
- d. **Seasonal Variations:** Explore whether there are seasonal variations in the usage of different payment types. Some payment methods may see increased adoption during holiday seasons, sales events, or other specific times of the year.
- e. **Payment Preference Shifts:** Look for any shifts in payment preferences over the years. Changes in the popularity of specific payment types could be influenced by factors such as changes in consumer preferences, marketing strategies, or the introduction of new payment options.

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

```
SELECT payment_installments, COUNT(DISTINCT order_id) AS num_orders
FROM `target.payments`
WHERE payment_value > 0
GROUP BY payment_installments
ORDER BY payment_installments;
```



Query results

Row	payment_installments	num_orders
1	0	2
2	1	49057
3	2	12389
4	3	10443
5	4	7088
6	5	5234
7	6	3916
8	7	1623
9	8	4253
10	9	644
11	10	5315
12	11	22

INSIGHTS

The query results show the distribution of the number of orders (num_orders) based on the number of payment installments (payment_installments).

- Installment Preferences:** Customers appear to prefer a smaller number of payment installments, with the majority of orders having 4 or 5 installments. This suggests that a significant portion of customers may choose payment plans that offer a moderate number of installments.
- Common Installment Plans:** The most common installment plans are 4, 5, and 6 installments, which have the highest order Counts. These installment options are likely popular among customers, possibly due to their simplicity and flexibility.
- Larger Installments Less Common:** As the number of installments increases beyond 6, the order Counts decrease. This indicates that customers are less inclined to choose larger installment plans, possibly due to the associated interest or fees.
- Infrequent Usage of Higher Installments:** Installment plans with 11 and 12 installments have very low order Counts, suggesting that customers rarely opt for these longer-term payment options.

RECOMMENDATIONS

- a. **Promote Moderate Installment Plans:** Consider promoting installment plans with 4, 5, and 6 installments, AS they are popular among customers. This could be done through marketing campaigns, promotions, or highlighting the benefits of these plans during the checkout process.
 - b. **Flexible Payment Options:** Provide a range of installment options to cater to different customer preferences. Offering flexibility in the number of installments allows customers to choose a plan that aligns with their financial preferences.
 - c. **Educate on Cost Implication:** Clearly communicate any interest rates or fees Associated with installment plans, especially for larger installment options. Educate customers on the cost implications of choosing longer-term payment plans to make informed decisions.
 - d. **Promotional Strategies:** Consider implementing promotional strategies, such AS discounts or incentives, for specific installment plans. Encouraging the use of certain installment options through promotions can influence customer choices.
-