

Business Case: Target SQL Project

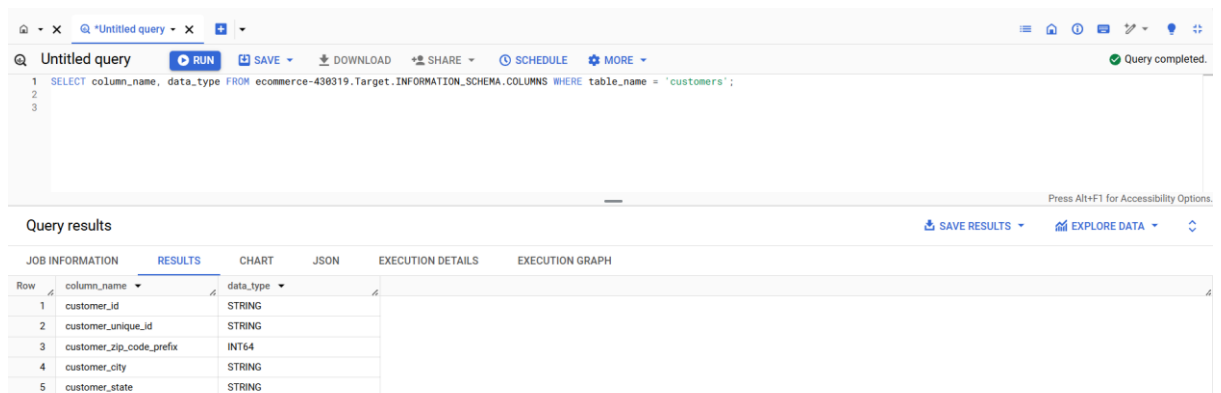
Context: -

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.

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

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

```
SELECT column_name, data_type
FROM ecommerce-
430319.Target.INFORMATION_SCHEMA.COLUMNS
WHERE table_name = 'customers';
```



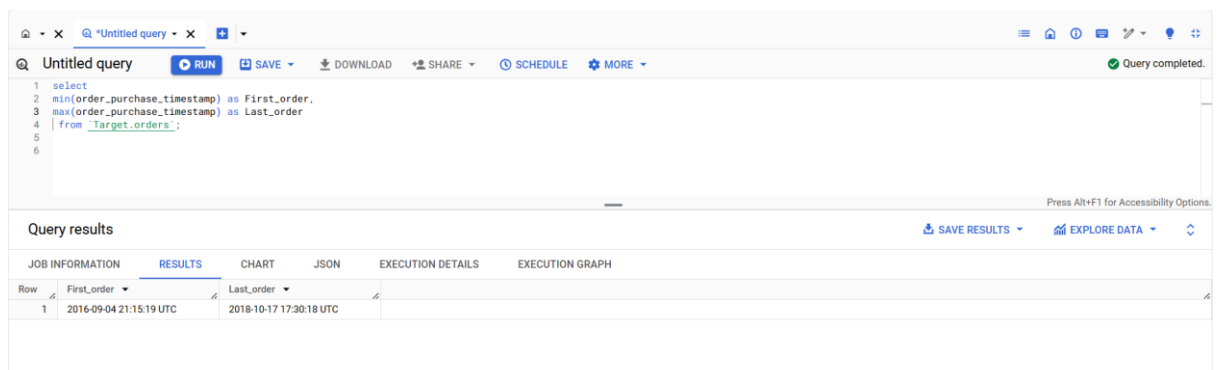
Query results

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:** - The data type of all the column in the customers tables talks about the type of data which will be stored in that column.

➤ 1.2) Get the time range between which the orders were placed.

```
select
min(order_purchase_timestamp) as First_order,
max(order_purchase_timestamp) as Last_order
from `Target.orders`
```



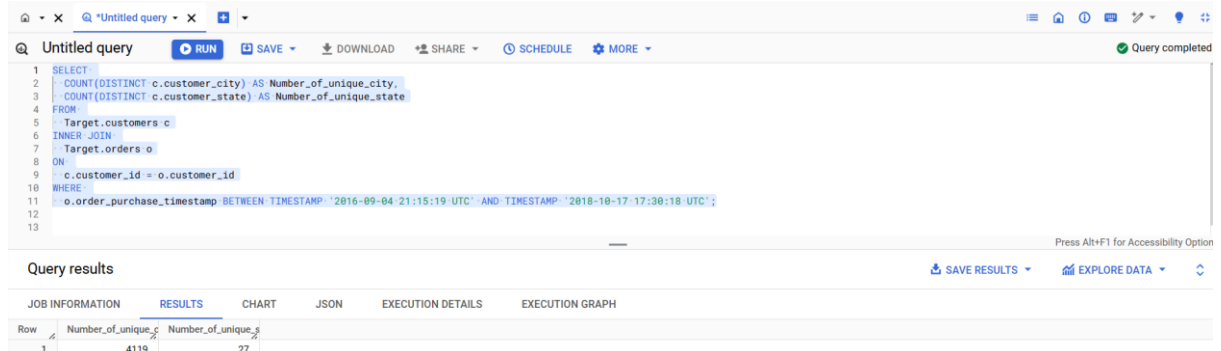
Query results

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

- ❖ **Insights:** - The range between the first order and last order indicates when the first order was placed and when the last order was placed in the given dataset.

➤ **1.3) Count the Cities & States of customers who ordered during the given period.**

```
SELECT
COUNT(DISTINCT c.customer_city) AS number_of_unique_city,
COUNT(DISTINCT c.customer_state) AS
number_of_unique_state
FROM
Target.customers c
INNER JOIN
Target.orders o
ON
c.customer_id = o.customer_id
WHERE
o.order_purchase_timestamp BETWEEN TIMESTAMP '2016-09-
04 21:15:19 UTC' AND TIMESTAMP '2018-10-17 17:30:18
UTC';
```



The screenshot shows a SQL query editor with a query that counts unique cities and states for orders placed between September 4, 2016, and October 17, 2018. The query is executed, and the results are displayed in a table with two columns: 'Number_of_unique_c' and 'Number_of_unique_s'. The results show 4119 unique cities and 27 unique states.

Job Information	Results	Chart	JSON	Execution Details	Execution Graph
Row	Number_of_unique_c	Number_of_unique_s			
1	4119	27			

- ❖ **Insights:** - The number of unique cities and states indicates the customers who made purchases during the period from '2016-09-04 21:15:19 UTC' to '2018-10-17 17:30:18 UTC'.

2. In-depth Exploration.

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

```
WITH buying_trend AS (
SELECT
```

```

        order_id,
        EXTRACT(YEAR FROM order_purchase_timestamp) AS
buying_trend_year,
        EXTRACT(MONTH FROM order_purchase_timestamp) AS
buying_trend_MONTH
    FROM
        Target.orders
    ORDER BY
        buying_trend_year
)
SELECT
    buying_trend_year, buying_trend_MONTH, count(*) as
no_of_orders
FROM
    buying_trend
group by buying_trend_year, buying_trend_MONTH
order by buying_trend_year, buying_trend_MONTH asc;

```

Query results

Row	buying_trend_year	buying_trend_MONTH	no_of_orders
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

- Yearly trend: -

```

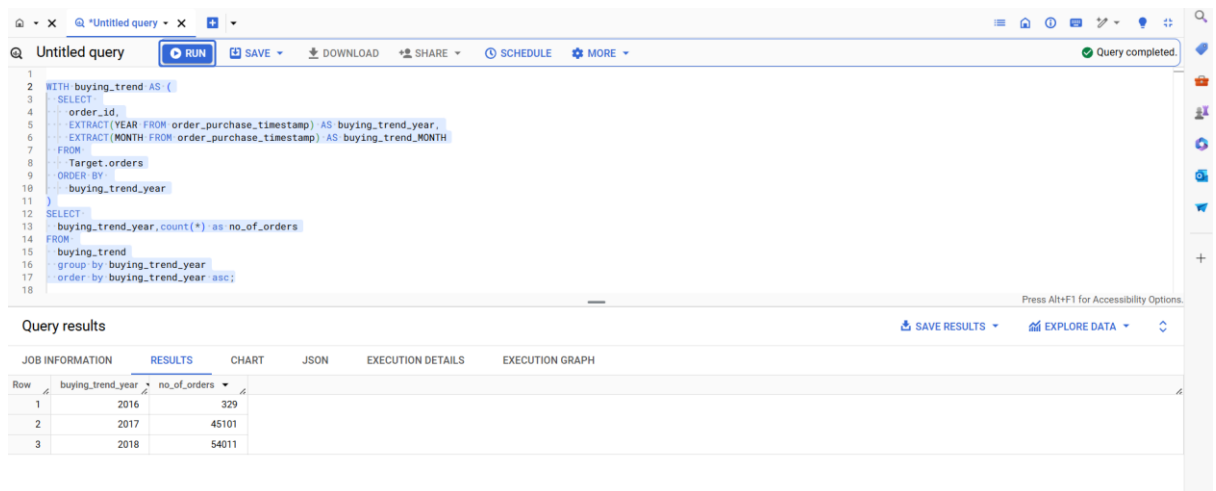
WITH buying_trend AS (
    SELECT
        order_id,
        EXTRACT(YEAR FROM order_purchase_timestamp) AS
buying_trend_year,
        EXTRACT(MONTH FROM order_purchase_timestamp) AS
buying_trend_MONTH
    FROM

```

```

        Target.orders
    ORDER BY
        buying_trend_year
)
SELECT
    buying_trend_year, count(*) as no_of_orders
FROM
    buying_trend
group by buying_trend_year
order by buying_trend_year asc;

```



The screenshot shows a SQL query editor with a query that creates a CTE named 'buying_trend' and then selects the number of orders for each year. The query is executed, and the results are displayed in a table.

Query results

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

- Monthly Trend: -

```

WITH buying_trend AS (
    SELECT order_id,
        EXTRACT(YEAR FROM order_purchase_timestamp) AS buying_trend_year,
        EXTRACT(MONTH FROM order_purchase_timestamp) AS buying_trend_MONTH
    FROM
        Target.orders
    ORDER BY
        buying_trend_year
)
SELECT
    buying_trend_MONTH, count(*) as no_of_orders
FROM
    buying_trend

```

group by buying_trend_MONTH
order by buying_trend_MONTH asc;

The screenshot shows a SQL query in a web-based editor. The query is as follows:

```

1 WITH buying_trend AS (
2   SELECT order_id,
3         EXTRACT(YEAR FROM order_purchase_timestamp) AS buying_trend_year,
4         EXTRACT(MONTH FROM order_purchase_timestamp) AS buying_trend_MONTH
5   FROM Target.orders
6   ORDER BY buying_trend_year
7 )
8 SELECT
9   buying_trend_MONTH, count(*) as no_of_orders
10 FROM buying_trend
11 group by buying_trend_MONTH
12 order by buying_trend_MONTH asc;

```

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

Row	buying_trend_MONTH	no_of_orders
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318
8	8	10843
9	9	4305
10	10	4959

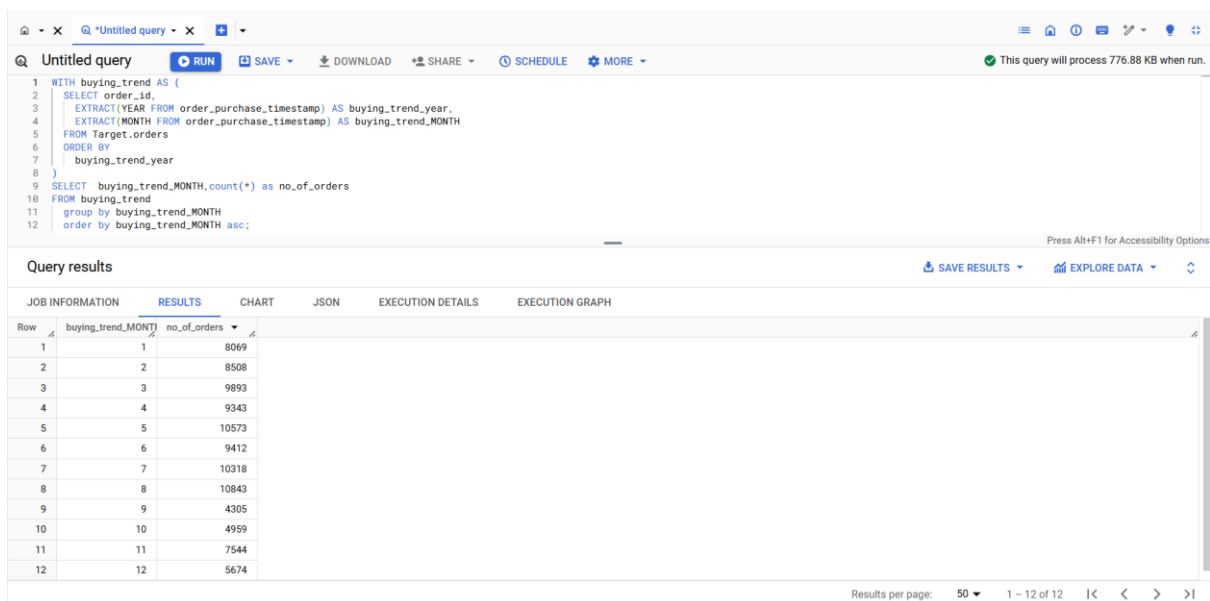
- ❖ **Insights:** - Between 2016 and 2018, we have observed a substantial increase in the volume of orders. This upward trend is a strong positive indicator of business growth and market demand. The consistent rise in orders over this period reflects the effectiveness of our strategies, the increasing recognition of our brand, and the successful engagement with our target market. This growth not only enhances our revenue potential but also strengthens our position in the industry.
- ❖ **Recommendations:** - We can review the strategies and initiatives implemented between 2016 and 2018 to identify what contributed to the growth. This might include marketing campaigns, product offerings, or improvements in customer experience.
- ❖ We can focus on retaining existing customers by enhancing loyalty programs, personalized offers, and exceptional customer service.
- **2.2) Can we see some kind of monthly seasonality in terms of the no. of orders being placed?**

WITH buying_trend AS (
 SELECT order_id,
 EXTRACT(YEAR FROM order_purchase_timestamp) AS
 buying_trend_year,
 EXTRACT(MONTH FROM order_purchase_timestamp) AS
 buying_trend_MONTH

```

FROM
    Target.orders
ORDER BY
    buying_trend_year
)
SELECT
    buying_trend_MONTH, count(*) as no_of_orders
FROM
    buying_trend
group by buying_trend_MONTH
order by buying_trend_MONTH asc;

```



Query results

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

- ❖ **Insights:** - It is evident from the data that the months from March to August have a relatively higher order count compared to the rest of the year. As we know, March to May constitutes the autumn season and June to August is the winter season in Brazil. During these periods, people prefer shopping online due to favourable weather conditions, reflecting a clear seasonality pattern.
- ❖ **Recommendations:** - we should Offer special discounts or promotions during these peak months to attract more customers and increase sales. Consider bundling products or offering limited time deals to create urgency.
- ❖ We can work on increase inventory levels for products that are in high demand during these months. Monitor sales trends and adjust stock levels accordingly to avoid stockouts and meet customer demand.

- ❖ We should use targeted advertising to reach customers during these peak months. Utilize digital advertising platforms to run seasonal ads that drive traffic to your website.

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

```
with buying_time as (  
    select order_id, order_purchase_timestamp, extract(hour from  
order_purchase_timestamp) as 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 hourly_purchasing  
from Target.orders  
order by order_purchase_timestamp  
)
```

```
SELECT  
    hourly_purchasing, count(*) as count_of_purchasing  
FROM  
    buying_time  
group by hourly_purchasing  
order by count_of_purchasing;
```

The screenshot shows a SQL query editor with a toolbar at the top containing buttons for RUN, SAVE, DOWNLOAD, SHARE, SCHEDULE, and MORE. The query text is as follows:

```
1  
2 with buying_time as (  
3     select order_id, order_purchase_timestamp, extract(hour from order_purchase_timestamp) as hour ,  
4     case  
5         when (extract(hour from order_purchase_timestamp)) between 0 and 6 then 'Dawn'  
6         when (extract(hour from order_purchase_timestamp)) between 7 and 12 then 'Morning'  
7         when (extract(hour from order_purchase_timestamp)) between 13 and 18 then 'Afternoon'  
8         else 'Night' end as hourly_purchasing  
9     from Target.orders  
10    order by order_purchase_timestamp  
11 )  
12  
13 SELECT  
14     hourly_purchasing, count(*) as count_of_purchasing  
15 FROM  
16     buying_time  
17    group by hourly_purchasing  
18    order by count_of_purchasing;  
19  
20  
21
```

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

Row	hourly_purchasing	count_of_purchasing
1	Dawn	5242
2	Morning	27733
3	Night	28331
4	Afternoon	38135

- ❖ **Insights:** - Based on the data we can clearly see that there is high buying activity due during afternoon during lunch break or post work hours. Second most buying activity can be seen during night. This suggests that many customers prefer shopping online after their daily activities are completed, possibly during leisure time at home. We can witness less buying activity during dawn. This suggests that very few customers shop during the early morning hours, likely due to sleep patterns and the general inactivity of this time.
- ❖ **Recommendations:** - To boost order volume during the morning and early dawn hours, we can organize flash sales that are active for one to two hours. This strategic approach is designed to drive higher purchase activity and increase order counts during these specific time periods.

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

- **3.1) Get the month-on-month no. of orders placed in each state.**

```
select t.geolocation_state,t.month_name, count(distinct t.order_id)
as no_of_orders
from
(select o.order_id,g.geolocation_state,o.order_purchase_timestamp,
extract(month from o.order_purchase_timestamp) as month,
FORMAT_TIMESTAMP('%B', o.order_purchase_timestamp) AS
month_name,
count(o.order_id) over (partition by g.geolocation_state order by
o.order_purchase_timestamp asc ) as no_of_order_statewise
from Target.orders o
left join Target.customers c using(customer_id)
left join Target.geolocation g on c.customer_zip_code_prefix =
g.geolocation_zip_code_prefix
order by o.order_purchase_timestamp)t
where t.geolocation_state is not null
group by 1,2
order by 1,2;
```

Untitled query

DownloadSaveShareScheduleMore

```
1 select t.geolocation_state,t.month_name,count(distinct t.order_id) as no_of_orders from
2 (select o.order_id,g.geolocation_state,o.order_purchase_timestamp,
3  extract(month from o.order_purchase_timestamp) as month,
4  format(timestamp 'dd', o.order_purchase_timestamp) AS month_name,
5  count(o.order_id) over (partition by g.geolocation_state order by o.order_purchase_timestamp asc) as no_of_order_statewise
6  from Target.orders o
7  left join Target.customers c using(customer_id)
8  left join Target.geolocation g on c.customer_zip_code_prefix = g.geolocation_zip_code_prefix
9  order by o.order_purchase_timestamp) t
10 where t.geolocation_state is not null
11 group by 1,2
12 order by 1,2;
```

Query results

Save ResultsExplore Data

Job InformationResultsChartJSONExecution DetailsExecution Graph

Row	geolocation_state	month_name	no_of_orders
1	AC	April	13
2	AC	August	11
3	AC	December	6
4	AC	February	9
5	AC	January	10
6	AC	July	16
7	AC	June	8
8	AC	March	6
9	AC	May	17
10	AC	November	7
11	AC	October	9

Results per page: 501 - 50 of 322

❖ **Insights:** - The order count varies significantly across different states, with SP, RJ, MG, and RS consistently demonstrating a high volume of orders. This pattern highlights a strong e-commerce presence and penetration in these regions. The consistent and substantial order counts in these states suggest a well-established online shopping culture, robust infrastructure, and possibly higher disposable incomes, making them key markets for e-commerce businesses.

❖ **Recommendations:** - We should invest in customer service and support to enhance satisfaction in high-sales states mentioned in the insights. This could involve improving response times, offering personalized services, and addressing any pain points.

❖ We can Create campaigns tailored to the interests and preferences of customers in high-sales states. We can use data insights to craft messages that resonate with the local audience.

❖ We should highlight products that are in high demand or have been popular in similar markets. We can create compelling offers and promotions based on customer insights.

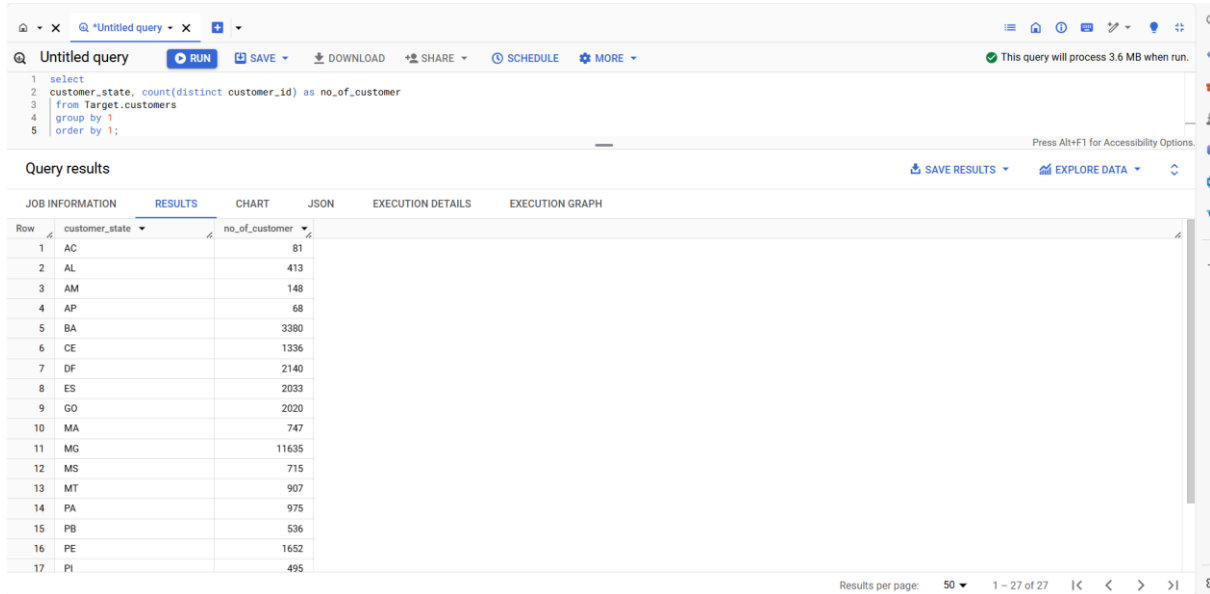
❖ We can also check the seasonality trends and then we can prepare ahead of time for seasonal trends and events. We should analyze past data to identify other peak periods throughout the year and plan our marketing and sales strategies accordingly.

➤ 3.2) How are the customers distributed across all the states?

```

select
customer_state, count(distinct customer_id) as no_of_customer
from Target.customers
group by 1
order by 1;

```



Query results

Row	customer_state	no_of_customer
1	AC	81
2	AL	413
3	AM	148
4	AP	68
5	BA	3380
6	CE	1336
7	DF	2140
8	ES	2033
9	GO	2020
10	MA	747
11	MG	11635
12	MS	715
13	MT	907
14	PA	975
15	PB	536
16	PE	1652
17	PI	495

- ❖ **Insights:** - The data clearly indicates that states such as SP, RJ, and MG have the highest number of customers, reflecting a robust e-commerce presence in these regions. The strong market penetration and consistent customer engagement in these areas suggest well-established online shopping behaviours and significant e-commerce infrastructure.
- ❖ **Recommendations:** - We can use data analytics to identify consumer preferences and behaviours in RR, AP, and AL. we can create tailored advertisements that resonate with local tastes and trends. Leverage social media and local influencers to amplify your message.
- ❖ We can offer products that cater to local needs and preferences. This could include regional food items, clothing styles, or home goods. Research what products are popular or needed in each state.
- ❖ We can conduct market research to get insights into consumer behaviour, preferences, and economic conditions in RR, AP, and AL. This can guide your marketing strategies and product offerings more effectively.

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 total_cost_2017 as
(select round(sum(p.payment_value),2) as total_value_2017 from
Target.orders o join Target.payments p using(order_id)
where extract(year from o.order_purchase_timestamp) =2017 and
extract(month from o.order_purchase_timestamp) between 1 and
8),
total_cost_2018 as
(select round(sum(p.payment_value),2) as total_value_2018 from
Target.orders o join Target.payments p using(order_id)
where extract(year from o.order_purchase_timestamp) =2018 and
extract(month from o.order_purchase_timestamp) between 1 and 8)

select total_value_2017,total_value_2018,
round((((total_cost_2018.total_value_2018 -
total_cost_2017.total_value_2017) /
total_cost_2017.total_value_2017) * 100,2) AS
percentage_increase
FROM total_cost_2017, total_cost_2018;
```

The screenshot displays the Google Cloud BigQuery interface. The top navigation bar includes the Google Cloud logo, a dropdown menu for 'Ecommerce', and a search bar. Below the navigation bar, the 'Explorer' panel on the left shows a search bar and a list of resources, including 'ecommerce-430319'. The main panel displays a SQL query titled 'Untitled query' with the following code:

```
1 with total_cost_2017 as
2 (select round(sum(p.payment_value),2) as total_value_2017 from Target.orders o join Target.payments p using(order_id)
3 where extract(year from o.order_purchase_timestamp) =2017 and extract(month from o.order_purchase_timestamp) between 1 and 8),
4 total_cost_2018 as
5 (select round(sum(p.payment_value),2) as total_value_2018 from Target.orders o join Target.payments p using(order_id)
6 where extract(year from o.order_purchase_timestamp) =2018 and extract(month from o.order_purchase_timestamp) between 1 and 8)
7
8 select total_value_2017,total_value_2018, round((((total_cost_2018.total_value_2018 - total_cost_2017.total_value_2017) /
9 total_value_2017) * 100,2) AS percentage_increase
10 FROM total_cost_2017, total_cost_2018;
```

The query results are displayed in a table with the following columns: 'total_value_2017', 'total_value_2018', and 'percentage_increase'. The results show a total value of 3669022.12 for 2017, 8694733.84 for 2018, and a percentage increase of 136.98.

Row	total_value_2017	total_value_2018	percentage_increase
1	3669022.12	8694733.84	136.98

The bottom of the interface shows the 'Job history' section with a 'REFRESH' button.

- ❖ **Insights:** - The data indicates a 136% increase in the cost of orders compared to the previous year, suggesting rapid growth in the e-commerce sector. This substantial rise underscores the expanding market and heightened consumer engagement with online shopping platforms.
- ❖ **Recommendations:** - We should invest in improving the online shopping experience, such as enhancing website navigation, providing personalized recommendations, and ensuring a seamless checkout process. A better customer experience can lead to increased sales and higher average order values.
- ❖ We can also pay attention on increase marketing efforts to capitalize on the growing market. Focus on targeted advertising campaigns, leveraging data insights to reach potential customers and boost engagement.
- ❖ Expand product assortment to cater to a wider range of customer preferences.
- ❖ Invest in logistics, data analytics, and customer retention strategies to support growth and maintain competitiveness.

➤ **4.2) Calculate the Total & Average value of order price for each state.**

```
select c.customer_state, round(sum(oi.price),2) as Total_val_order,
round(avg(oi.price),2) as Total_avg_val_order from
Target.customers c join Target.orders o using(customer_id)
join Target.order_items oi using(order_id)
group by c.customer_state
order by 1,Total_val_order,Total_avg_val_order;
```

The screenshot shows the Google Cloud BigQuery console interface. At the top, there's a search bar and navigation tabs. The main area displays a query titled 'Untitled query' with the following SQL code:

```

1 select c.customer_state, round(sum(o.price),2) as Total_val_order, round(avg(o.price),2) as Total_avg_val_order from Target.customers c join
2 Target.orders o using(customer_id)
3 join Target.order_items oi using(order_id)
4 group by c.customer_state
5 order by 1,Total_val_order,Total_avg_val_order;

```

Below the query, the 'Query results' section shows a table with 10 rows of data. The table has columns for 'customer_state', 'Total_val_order', and 'Total_avg_val_order'. The results are as follows:

Row	customer_state	Total_val_order	Total_avg_val_order
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

At the bottom of the results table, it indicates 'Results per page: 50' and '1 - 27 of 27'. There are also buttons for 'SAVE RESULTS', 'EXPLORE DATA', and 'REFRESH'.

- ❖ **Insights:** - To calculate the total and average order values by state, I performed a detailed query of the Target Business Case Study database. This query involved joining the customers, orders, and order_items tables to derive the total and average order prices for each state.
- ❖ **Recommendations:** - Concentrate marketing efforts and promotions in states with high total and average order values to maximize revenue potential.
- ❖ Implement targeted discounts and promotions in states with lower average order values to stimulate sales and improve market penetration.
- ❖ Optimize inventory allocation by prioritizing high-value regions, ensuring that demand is met efficiently in areas with greater sales potential.
- ❖ Conduct in-depth analysis in both high and low-performing states to uncover the factors influencing differences in order values, enabling more informed decision-making.
- ❖ Develop and implement region-specific marketing strategies tailored to the unique preferences and purchasing power of customers in each state, leveraging the insights gained from regional order value data.

➤ **4.3) Calculate the Total & Average value of order freight for each state.**

```

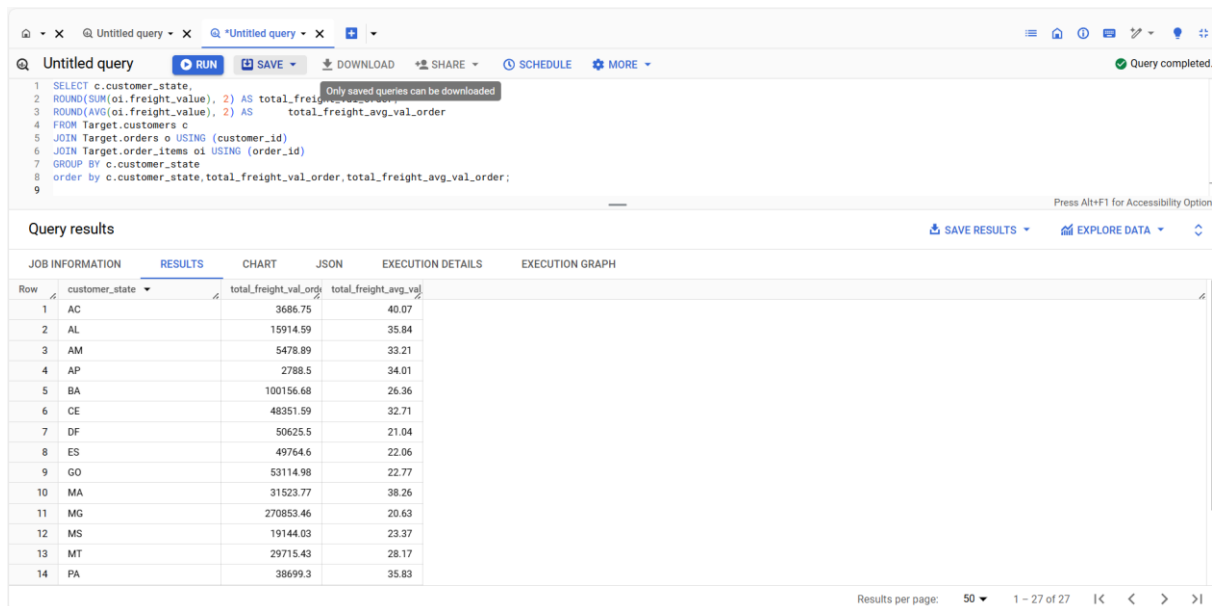
SELECT c.customer_state,
ROUND(SUM(oi.freight_value), 2) AS total_freight_val_order,

```

```

ROUND(AVG(oi.freight_value), 2) AS
total_freight_avg_val_order
FROM Target.customers c
JOIN Target.orders o USING (customer_id)
JOIN Target.order_items oi USING (order_id)
GROUP BY c.customer_state
order by
c.customer_state,total_freight_val_order,total_freight_avg_val_order;

```



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

```

1 SELECT c.customer_state,
2 ROUND(SUM(oi.freight_value), 2) AS total_freight_val_order,
3 ROUND(AVG(oi.freight_value), 2) AS total_freight_avg_val_order
4 FROM Target.customers c
5 JOIN Target.orders o USING (customer_id)
6 JOIN Target.order_items oi USING (order_id)
7 GROUP BY c.customer_state
8 order by c.customer_state,total_freight_val_order,total_freight_avg_val_order;
9

```

The query results are displayed in a table with the following columns: Row, customer_state, total_freight_val_order, and total_freight_avg_val_order. The results are sorted by customer_state.

Row	customer_state	total_freight_val_order	total_freight_avg_val_order
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
14	PA	38699.3	35.83

- ❖ **Insights:** - To analyze the distribution of freight costs across different states in Brazil, I conducted a comprehensive query of the customers, orders, and order_items tables from the Target Business Case Study database. This analysis involved calculating both the total and average freight values for each state, offering valuable insights into the overall logistics expenses and the average freight cost per order across the region.
- ❖ **Recommendations:** - Analyze states with high average freight costs to identify opportunities for optimizing shipping routes and reducing logistics expenses.
- ❖ Utilize the high shipping volumes in states like SP as leverage to negotiate more favourable shipping rates with carriers.
- ❖ Explore the possibility of establishing regional warehouses in states with high order volumes to shorten shipping distances and reduce costs.

- ❖ Implement dynamic pricing strategies that reflect the varying freight costs across different states, ensuring that pricing remains competitive while maintaining profitability.
- ❖ Offer targeted shipping incentives or discounts in states with higher average freight costs to stimulate order volume and offset the impact of higher logistics expenses.

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

- **5.1) 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
distinct order_id, date_diff(order_delivered_customer_date,
order_purchase_timestamp, day) as time_to_deliver,
date_diff(order_delivered_customer_date, order_estimated_delivery_date, day) as diff_estimated_delivery
from Target.orders
where order_delivered_customer_date is not null
order by order_id asc;
```

The screenshot shows the Google Cloud BigQuery interface. The query editor contains the following SQL query:

```
1 select
2 distinct order_id, date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as time_to_deliver,
3 date_diff(order_delivered_customer_date, order_estimated_delivery_date, day) as diff_estimated_delivery
4 from Target.orders
5 where order_delivered_customer_date is not null
6 order by order_id asc;
```

The query results are displayed in a table with the following columns: Row, order_id, time_to_deliver, and diff_estimated_delivery. The results show 10 rows of data.

Row	order_id	time_to_deliver	diff_estimated_delivery
1	00010242fe8c5a6d1ba2dd792...	7	-8
2	00018f772f0320c557190d7a1...	16	-2
3	000229ec398224efca0657da...	7	-13
4	00024acbcd0a6daa1e931b03...	6	-5
5	00042b26cf59d7ce69dfabb4e...	25	-15
6	00048cc3ae777c65dbb7d2a06...	6	-14
7	00054e8431b9d767580bcb8...	8	-16
8	000576fe39319847cbb9d288c...	5	-15
9	0005a1a1728c9d785b8e2b08...	9	0
10	0005f50442ch953dcd1d71e1f...	2	-18

- ❖ **Insights:** -The maximum delivery time recorded is 209 days, while the estimated delivery date is significantly shorter, at 181 days. This discrepancy is observed in multiple cases and could potentially harm the

company's reputation. Prolonged delivery times are likely to result in low customer satisfaction and may adversely affect the company's image.

- ❖ **Recommendations:** - Implement measures to improve the accuracy of delivery time estimates and reduce actual delivery times. This could involve optimizing supply chain processes, enhancing inventory management, and improving logistical operations.
- ❖ Keep customers informed about any potential delays and provide regular updates on their order status to manage expectations and reduce dissatisfaction.
- ❖ Regularly track delivery performance metrics to identify and address any recurring issues promptly, ensuring continuous improvement in delivery times and customer satisfaction.

➤ **5.2) Find out the top 5 states with the highest & lowest average freight value.**

- highest average freight value: -

```
with cte as (select c.customer_state, round(avg(oi.freight_value),2)
average_freight_val
from Target.customers c join Target.orders o using(customer_id)
join Target.order_items oi using(order_id)
where o.order_delivered_customer_date is not null
group by customer_state)

select * from cte order by cte.average_freight_val desc limit 5;
```

The screenshot shows the Google BigQuery interface. On the left is the Explorer panel with a search bar and a list of resources, including 'ecommerce-430319'. The main panel displays a SQL query titled 'Untitled query' with the following code:

```

1 with cte as (select c.customer_state, round(avg(oi.freight_value),2) average_freight_val
2 from Target.customers c join Target.orders o using(customer_id)
3 join Target.order_items oi using(order_id)
4 where o.order_delivered_customer_date is not null
5 group by customer_state)
6
7 select * from cte order by cte.average_freight_val desc limit 5;
8
9
10
11

```

Below the query editor, the 'Query results' tab is active, showing a table with 5 rows and 2 columns: 'customer_state' and 'average_freight_val'.

Row	customer_state	average_freight_val
1	PB	43.09
2	RR	43.09
3	RO	41.33
4	AC	40.05
5	PI	39.12

At the bottom, the 'Job history' tab is visible.

- Lowest average freight value: -

```

with cte as (select c.customer_state, round(avg(oi.freight_value),2)
average_freight_val
from Target.customers c join Target.orders o using(customer_id)
join Target.order_items oi using(order_id)
where o.order_delivered_customer_date is not null
group by customer_state)
select * from cte order by cte.average_freight_val asc limit 5;

```

The screenshot shows the Google BigQuery interface with the same SQL query as the first image, but the results are ordered by ascending average freight value.

The 'Query results' tab shows a table with 5 rows and 2 columns: 'customer_state' and 'average_freight_val'.

Row	customer_state	average_freight_val
1	SP	15.11
2	PR	20.47
3	MG	20.63
4	RJ	20.91
5	DF	21.07

The 'Job history' tab is also visible at the bottom.

❖ **Insights:** - The analysis aims to identify the top 5 states with the highest and lowest average freight values. This information helps

in understanding the shipping costs associated with different regions, which is crucial for logistics and pricing strategies.

- ❖ **Recommendations:** - Explore more cost-effective shipping methods or negotiate better rates with carriers in high-cost states.
 - ❖ Consider establishing regional distribution centers to reduce shipping distances and costs.
 - ❖ Use the lower shipping costs to offer more competitive pricing or free shipping promotions in low-cost states.
 - ❖ Target these regions with aggressive marketing campaigns to boost sales further.
 - ❖ Adjust product pricing to account for higher freight costs in specific regions, ensuring profitability without deterring customers.
 - ❖ Clearly communicate shipping policies and potential costs to customers in high-cost regions to manage expectations and reduce cart abandonment.
- **5.3) Find out the top 5 states with the highest & lowest average delivery time.**

```
with cust_info as (select c.customer_state,
round(avg(date_diff(o.order_delivered_customer_date,
o.order_purchase_timestamp, day)),2)as time_to_deliver
from Target.customers c left join Target.orders o
using (customer_id)
where o.order_delivered_customer_date is not null
group by c.customer_state
), cust_ranking as (select customer_state, time_to_deliver,
dense_rank() over(order by time_to_deliver desc
) as rnk from cust_info
)
select customer_state, time_to_deliver as average_delivery_time
from cust_ranking
where rnk >22 or rnk < 6
order by time_to_deliver desc;
```

Query results

Row	customer_state	average_delivery_time
1	RR	28.98
2	AP	26.73
3	AM	25.99
4	AL	24.04
5	PA	23.32
6	SC	14.48
7	DF	12.51
8	MG	11.54
9	PR	11.53
10	SP	8.3

- **5.4) Find out the top 5 states where the order delivery is 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.

```
with cte as (select c.customer_state,
round(avg(date_diff(
order_delivered_customer_date,order_purchase_timestamp,
day))),2) as actual_delivery_time,
round(avg(date_diff(order_estimated_delivery_date,order_purchase_timestamp,
day))),2) as estimated_delivery_time
from Target.customers c join Target.orders o using(customer_id)
where o.order_delivered_customer_date is not null
group by customer_state)
```

```
select * from cte where cte.actual_delivery_time <
cte.estimated_delivery_time order by actual_delivery_time limit 5;
```

The screenshot shows the BigQuery interface. The query editor contains the following SQL code:

```

1 with cte as (select c.customer_state,
2 round(avg(date_diff(order_delivered_customer_date, order_purchase_timestamp, day)), 2) as actual_delivery_time,
3 round(avg(date_diff(order_estimated_delivery_date, order_purchase_timestamp, day)), 2) as estimated_delivery_time
4 from Target.customers c join Target.orders o using(customer_id)
5 where o.order_delivered_customer_date is not null
6 group by customer_state)
7
8 select * from cte where cte.actual_delivery_time < cte.estimated_delivery_time order by actual_delivery_time limit 5;
9

```

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

Row	customer_state	actual_delivery_time	estimated_delivery_time
1	SP	8.3	18.78
2	PR	11.53	24.25
3	MG	11.54	24.19
4	DF	12.51	23.95
5	SC	14.48	25.42

- ❖ **Insights:** - To determine which states, have the fastest order delivery times compared to the estimated dates, I queried the orders and customers tables. The query calculates the average actual delivery time and the average estimated delivery time for each state. The difference between these two averages indicates how much faster (or slower) the deliveries are compared to the estimated dates.
- ❖ **Recommendations:** - Investigate states with slower delivery times or positive differences to identify bottlenecks and improve delivery processes.
- ❖ Highlight faster delivery times in marketing materials for the top-performing states to enhance customer satisfaction and attract more customers.
- ❖ Allocate resources to optimize logistics in regions where delivery times significantly exceed the estimates.

6. Analysis based on the payments.

- **6.1) Find the month-on-month no. of orders placed using different payment types.**

with order_info as (select p.payment_type,
extract(month from o.order_purchase_timestamp) as
Purchasing_Month,
extract(year from o.order_purchase_timestamp) as
Purchasing_Year,
o.order_id as orders

```

from Target.orders o join Target.payments p using(order_id))

select Purchasing_Year ,Purchasing_Month,payment_type,
count(distinct orders) as Orders from order_info
group by 1,2,3
order by 1,2 asc;

```

The screenshot shows the Google Cloud BigQuery interface. On the left is the Explorer panel with a search bar and a list of resources, including 'ecommerce-430319'. The main area displays an 'Untitled query' with the following SQL code:

```

1 with order_info as (select p.payment_type,
2   extract(month from o.order_purchase_timestamp ) as Purchasing_Month,
3   extract(year from o.order_purchase_timestamp ) as Purchasing_Year,
4   o.order_id as orders
5   from Target.orders o join Target.payments p using(order_id))
6
7 select Purchasing_Year ,Purchasing_Month,payment_type, count(distinct orders) as Orders from order_info
8 group by 1,2,3
9 order by 1,2 asc;

```

Below the query editor, the 'Query results' section is visible, showing a table with the following data:

Row	Purchasing_Year	Purchasing_Month	payment_type	Orders
1	2016	9	credit_card	3
2	2016	10	credit_card	253
3	2016	10	UPI	63
4	2016	10	voucher	11
5	2016	10	debit_card	2
6	2016	12	credit_card	1
7	2017	1	credit_card	582
8	2017	1	UPI	197
9	2017	1	credit_card	1

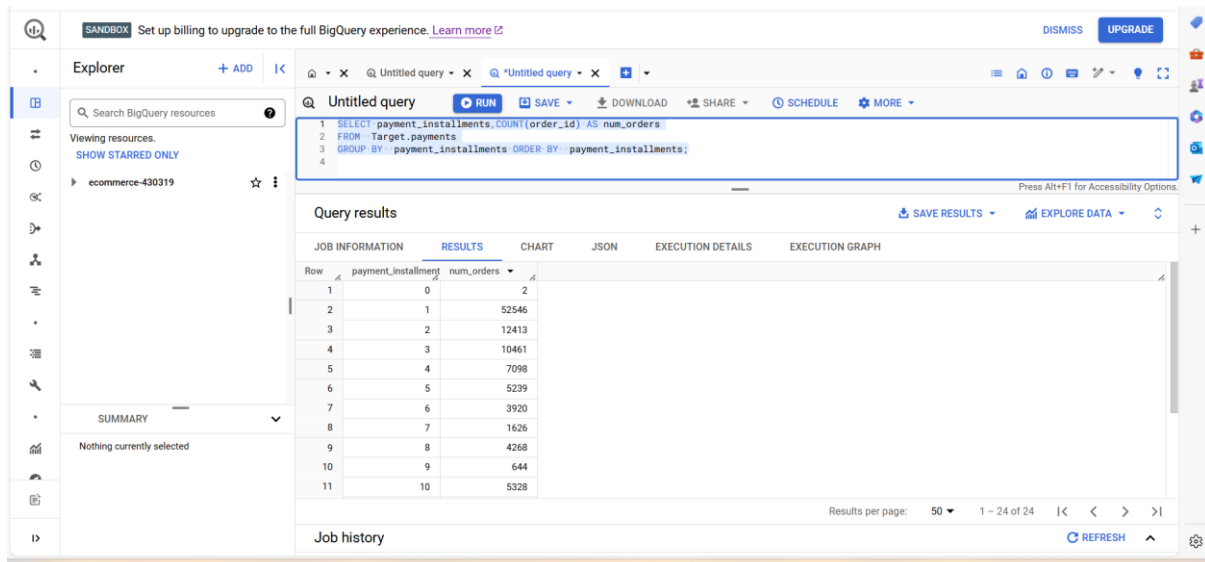
The interface also includes a 'Job history' section at the bottom and a 'SUMMARY' tab on the left sidebar.

- ❖ **Insights:** - The data clearly indicates that credit cards are the predominant payment method, significantly surpassing the usage of alternative payment options such as UPI, debit cards, and vouchers.
- ❖ The second most used payment method is UPI, followed by debit cards, with vouchers being the least utilized option.
- ❖ **Recommendations:** - Since a significant portion of our customers prefers using credit cards, we should implement tailored offers and rewards for these users. By providing exclusive promotions, discounts, or loyalty benefits for credit card transactions, we can incentivize continued use of credit cards as the preferred payment method, reinforcing customer loyalty and driving repeat business.
- ❖ We can also think about establishing partnerships with banks to introduce co-branded credit cards that offer targeted cashback incentives. These credit cards could provide additional benefits for users who make purchases through our platform, further promoting credit card usage and enhancing customer loyalty.

- ❖ To boost the usage of debit cards and UPI as payment methods, we should actively promote their benefits. Offering attractive cashbacks or discounts for transactions made using debit cards or UPI can incentivize customers to choose these payment options. This strategy could enhance the adoption of these methods and balance the payment method distribution.

➤ **6.2) 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_orders
FROM Target.payments
GROUP BY payment_installments ORDER
BY payment_installments;
```



The screenshot shows the BigQuery interface with a query editor and results table. The query is: `SELECT payment_installments,COUNT(order_id) AS num_orders FROM Target.payments GROUP BY payment_installments ORDER BY payment_installments;` The results table has two columns: `payment_installment` and `num_orders`. The data shows a decreasing trend in the number of orders as the number of installments increases.

Row	payment_installment	num_orders
1	0	2
2	1	52546
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

- ❖ **Insights:** - The data clearly demonstrates that the majority of customers have made their purchases using one to four payment instalments. People do not want long term EMI.
- ❖ **Recommendations:** - It is advisable to offer customers a range of payment options, including both one-time payments and instalment plans. This flexibility will cater to diverse customer preferences and financial situations, potentially increasing customer satisfaction and sales.
- ❖ Implement exclusive promotions and discounts for customers who choose to pay via instalment plans. By providing financial incentives, we can encourage more customers to select this payment method, which may lead to increased adoption and potentially higher overall transaction values.

