

Business Case: Target United States.

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 `Target_Sales_Data.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'customers'
```

The screenshot shows the Google Cloud BigQuery console interface. The left sidebar displays the project hierarchy: 'target-brazil-390016' > 'Target_Sales_Data'. The main panel shows a query titled 'Untitled 11' with the following SQL code:

```
1 SELECT column_name, data_type
2 FROM `Target_Sales_Data.INFORMATION_SCHEMA.COLUMNS`
3 WHERE table_name = 'customers'
```

The query results are displayed in a table with columns 'column_name' and 'data_type'.

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

- 1.2. Get the time range between which the orders were placed.

```
SELECT MIN(order_purchase_timestamp) as first_order_time,
MAX(order_purchase_timestamp) as last_order_time
FROM `Target_Sales_Data.orders`
```

The screenshot shows the Google Cloud BigQuery console interface. The left sidebar displays the project hierarchy: 'target-brazil-390016' > 'Target_Sales_Data'. The main panel shows a query titled 'Untitled 3' with the following SQL code:

```
1 SELECT MIN(order_purchase_timestamp) as first_order_time, MAX(order_purchase_timestamp) as last_order_time
2 FROM `Target_Sales_Data.orders`
```

The query results are displayed in a table with columns 'first_order_time' and 'last_order_time'.

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

Business Case: Target United States.

1.3. Count the number of Cities and States in our dataset.

```
select
  count(distinct (geolocation_city)) city_count,
  count(distinct (geolocation_state)) as state_count
from `Target_Sales_Data.geolocation`
```

The screenshot shows the Google Cloud BigQuery console interface. The Explorer pane on the left displays the project 'target-brazil-390016' and the dataset 'Target_Sales_Data' with various tables like 'customers', 'geolocation', 'order_items', etc. The main editor shows a query in 'Untitled 4' that counts distinct cities and states. The 'Query results' pane at the bottom shows a single row with 'city_count' as 8011 and 'state_count' as 27.

Row	city_count	state_count
1	8011	27

2. In-depth Exploration:

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

```
SELECT
  Count(order_id) as month_over_month ,
  extract(MONTH FROM order_purchase_timestamp) as months,
  EXTRACT(YEAR FROM order_purchase_timestamp) as Year,
from `Target_Sales_Data.orders`
group by months,Year
order by months
```

The screenshot shows the Google Cloud BigQuery console with a query in 'Untitled 5' that groups orders by month and year. The 'Query results' pane displays a table with columns for 'month_over_month', 'months', and 'Year'. The results show data for the year 2017 across months 5 to 13.

Row	month_over_month	months	Year
8	3700	5	2017
9	3245	6	2017
10	4026	7	2017
11	4331	8	2017
12	4285	9	2017
13	4631	10	2017

Business Case: Target United States.

Observation:

- No of Orders decline every year from march to April.
- There is continues decline in no of orders from march to June in year 2018

2.2 Can we see some kind of monthly seasonality in terms of the no. of orders being placed? (from above code)

Observation- No of orders increase in jan in new-year

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

0-6 hrs : Dawn

7-12 hrs : Mornings

13-18 hrs : Afternoon

19-23 hrs : Night

```
Select days_count, time_of_day
from( select
      Count(order_id) as days_count,
      CASE
        WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 0 AND EXTRACT(HOUR FROM
order_purchase_timestamp) <= 6 THEN 'Dawn'
        WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 7 AND EXTRACT(HOUR FROM
order_purchase_timestamp) <= 12 THEN 'Morning'
        WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 13 AND EXTRACT(HOUR
FROM order_purchase_timestamp) <= 18 THEN 'Afternoon'
        WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 19 AND EXTRACT(HOUR
FROM order_purchase_timestamp) <= 23 THEN 'Night'
      END AS time_of_day,
      extract(MONTH FROM order_purchase_timestamp) as months,
      EXTRACT(YEAR FROM order_purchase_timestamp) as Year,
      from `Target_Sales_Data.orders`
    GROUP BY time_of_day, months, Year ) AS A
order by A.days_count desc
```

The screenshot shows the Google Cloud BigQuery console interface. The query editor on the left contains the following SQL code:

```
1 select days_count, time_of_day
2 from( select
3       Count(order_id) as days_count,
4       CASE
```

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

Row	days_count	time_of_day
1	2857	Afternoon
2	2830	Afternoon
3	2786	Afternoon
4	2651	Afternoon
5	2648	Afternoon
6	2629	Afternoon
7	2482	Afternoon
8	2464	Afternoon
9	2295	Afternoon

The interface also shows a sidebar with a file explorer, a search bar, and a bottom status bar with system information like temperature and time.

Observation – most orders are placed in after noon

Business Case: Target United States.

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 total, customer_state, months, month_Nme
from( SELECT count(order_id) as total, customer_state, FORMAT_DATE('%B',
order_purchase_timestamp) AS month_Nme,
EXTRACT(MONTH FROM order_purchase_timestamp) as months,
EXTRACT(YEAR FROM order_purchase_timestamp) as Year
from `Target_Sales_Data.orders` as o
Join `Target_Sales_Data.customers` as c
on o.customer_id = c.customer_id
group by months, Year, customer_state, month_Nme) as a
order by total desc, customer_state asc
```

The screenshot shows the Google Cloud BigQuery console interface. The query editor on the left contains the SQL query from the previous block. The 'Query results' section on the right displays the output of the query. The results are organized into columns: Row, total, customer_state, months, and month_Nme. The data shows the number of orders (total) for each customer state (customer_state) across different months (months) and month names (month_Nme). The results are sorted by total orders in descending order.

Row	total	customer_state	months	month_Nme
1	3253	SP	8	August
2	3207	SP	5	May
3	3059	SP	4	April
4	3052	SP	1	January
5	3037	SP	3	March
6	3012	SP	11	November
7	2777	SP	7	July

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

```
SELECT Count(customer_id) as customer_no_per_aera, customer_state
from `Target_Sales_Data.customers`
GROUP BY customer_state
order by customer_no_per_aera desc
```

Business Case: Target United States.

The screenshot shows the Google Cloud BigQuery console interface. The Explorer panel on the left lists workspace resources under 'Target_Sales_Data', including customers, geolocation, order_items, order_reviews, orders, payments, products, and sellers. The main editor shows a SQL query in 'Untitled 8' that counts customers by state. The query results are displayed in a table with columns 'customer_no_per' and 'customer_state'.

```
1 SELECT Count(customer_id) as customer_no_per_aera, customer_state
2 from `Target_Sales_Data.customers`
3 GROUP BY customer_state
4 order by customer_no_per_aera desc
```

Row	customer_no_per	customer_state
1	41746	SP
2	12852	RJ
3	11635	MG
4	5466	RS
5	5045	PR
6	3637	SC
7	3380	BA
8	2140	DF

Additional - If we want Distribution of Order across states -

```
SELECT total, customer_state
  from( SELECT count(order_id) as total, customer_state,
  from `Target_Sales_Data.orders` as o
  Join `Target_Sales_Data.customers` as c
  on o.customer_id = c.customer_id
  group by customer_state) as a
order by total desc
```

Observation - Most Customers are from SP

The screenshot shows the Google Cloud BigQuery console interface. The Explorer panel on the left lists workspace resources under 'Target_Sales_Data'. The main editor shows a SQL query in 'Untitled 7' that counts orders by state. The query results are displayed in a table with columns 'total' and 'customer_state'.

```
1 SELECT total, customer_state
2 from( SELECT count(order_id) as total, customer_state,
3 from `Target_Sales_Data.orders` as o
4 Join `Target_Sales_Data.customers` as c
5 on o.customer_id = c.customer_id
6 group by customer_state) as a
7 order by total desc
```

Row	total	customer_state
1	41746	SP
2	12852	RJ
3	11635	MG
4	5466	RS
5	5045	PR
6	3637	SC
7	3380	BA

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

Business Case: Target United States.

4.1. 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 order_purchase_timestamp) = 2018 THEN
payment_value ELSE 0 END)
  - SUM(CASE WHEN EXTRACT (YEAR from order_purchase_timestamp) = 2017 THEN
payment_value ELSE 0 END))
  / SUM(CASE WHEN EXTRACT (YEAR from order_purchase_timestamp) = 2017 THEN
payment_value ELSE 0 END)) * 100 AS percentage_increase
FROM `Target_Sales_Data.orders` as o
join `Target_Sales_Data.payments` as p
on o.order_id=p.order_id
WHERE
  EXTRACT (MONTH FROM order_purchase_timestamp) BETWEEN 1 AND 8;
```

The screenshot displays the Google Cloud BigQuery console interface. On the left, the 'Explorer' pane shows a project named 'Target_Brazil' with a folder 'Target_Sales_Data' containing tables like 'customers', 'geolocation', 'order_items', 'orders', 'payments', 'products', and 'sellers'. The main editor shows a SQL query (Untitled 4) that calculates the percentage increase in payment values from 2017 to 2018 for orders placed between January and August. The query is executed, and the 'Query results' pane shows a single row with the value 136.9768716466... under the column 'percentage_increase'.

Row	percentage_increase
1	136.9768716466...

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

```
SELECT customer_state, Round (Sum(payment_value),2) as Total_order_value_per_sate,
Round(AVG(payment_value),2) as avg_order_val_per_state
from `Target_Sales_Data.payments` as p
join `Target_Sales_Data.orders` as o
on p.order_id=o.order_id
join `Target_Sales_Data.customers` as c
on o.customer_id=c.customer_id
group by customer_state
order by customer_state
```

Business Case: Target United States.

The screenshot shows the Google Cloud BigQuery console interface. The left sidebar displays the Explorer view with a search bar and a list of workspace resources including 'Target_Sales_Data', 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', and 'sellers'. The main panel shows a SQL query titled 'Untitled 2' with the following code:

```
1 SELECT customer_state, Round (Sum(payment_value),2) as Total_order_value_per_sate, Round(AVG(payment_value),2) as  
2 avg_order_val_per_state  
3 from `Target_Sales_Data.payments` as p  
4 join `Target_Sales_Data.orders` as o  
5 on p.order_id=o.order_id
```

The query results are displayed in a table with columns: Row, customer_state, Total_order_value_per_sate, and avg_order_val_per_state. The results are as follows:

Row	customer_state	Total_order_value_per_sate	avg_order_val_per_state
1	AC	19680.62	234.29
2	AL	96962.06	227.08
3	AM	27966.93	181.6
4	AP	16262.8	232.33
5	BA	616645.82	170.82
6	CE	279464.03	199.9
7	DF	355141.08	161.13
8	ES	325967.55	154.71

The bottom of the console shows the Windows taskbar with the system clock at 22:49 on 19-06-2023.

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

```
SELECT customer_state,  
Round (Sum(freight_value),2) as Total_freight_value_per_sate,  
Round(AVG(freight_value),2) as avg_frieght_val_per_state  
from `Target_Sales_Data.order_items` as p  
join `Target_Sales_Data.orders` as o  
on p.order_id=o.order_id  
join `Target_Sales_Data.customers` as c  
on o.customer_id=c.customer_id  
group by customer_state  
order by customer_state
```

The screenshot shows the Google Cloud BigQuery console interface. The left sidebar displays the Explorer view with a search bar and a list of workspace resources including 'target-brazil-390016', 'Target_Sales_Data', 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', and 'sellers'. The main panel shows a SQL query titled 'Untitled 2' with the following code:

```
1 SELECT customer_state,  
2 Round (Sum(freight_value),2) as Total_freight_value_per_sate,  
3 Round(AVG(freight_value),2) as avg_frieght_val_per_state  
4 from `Target_Sales_Data.order_items` as p  
5 join `Target_Sales_Data.orders` as o  
6 on p.order_id=o.order_id
```

The query results are displayed in a table with columns: Row, customer_state, Total_freight_value_per_sate, and avg_frieght_val_per_state. The results are as follows:

Row	customer_state	Total_freight_value_per_sate	avg_frieght_val_per_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

The bottom of the console shows the Windows taskbar with the system clock at 22:56 on 19-06-2023.

Business Case: Target United States.

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.

Below is gives actual delivery time and difference in estimated & actual delivery date only where order were delayed with sate of customer and seller.

```
SELECT customer_state,seller_state,
TIMESTAMP_DIFF (order_delivered_carrier_date, order_purchase_timestamp, DAY) as
time_to_deliver,
TIMESTAMP_DIFF (order_estimated_delivery_date, order_delivered_carrier_date,
DAY) as diff_estimated_delivery
from `Target_Sales_Data.orders` as o
join `Target_Sales_Data.customers` as c
on o.customer_id=c.customer_id
join `Target_Sales_Data.order_items` as oi
on o.order_id=oi.order_id
join `Target_Sales_Data.sellers` as s
on oi.seller_id=s.seller_id
where LOWER(order_status) = "delivered" and
TIMESTAMP_DIFF (order_estimated_delivery_date, order_delivered_carrier_date,
DAY) < 0
Order by time_to_deliver desc
```

The screenshot shows the Google Cloud BigQuery console interface. On the left, the 'Explorer' pane displays the project 'target-brazil-390016' and a folder 'Target_Sales_Data' containing tables like 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', and 'sellers'. The main pane shows a SQL query in 'Untitled 5' with the following text:

```
2 TIMESTAMP_DIFF (order_delivered_carrier_date, order_purchase_timestamp, DAY) as time_to_deliver,
3 TIMESTAMP_DIFF (order_estimated_delivery_date, order_delivered_carrier_date, DAY) as diff_estimated_delivery
4 from `Target_Sales_Data.orders` as o
5 join `Target_Sales_Data.customers` as c
6 on o.customer_id=c.customer_id
7 join `Target_Sales_Data.order_items` as oi
8 on o.order_id=oi.order_id
```

The query results are displayed in a table with the following columns: 'customer_state', 'seller_state', 'time_to_deliver', and 'diff_estimated_delivery'. The results are sorted by 'time_to_deliver' in descending order.

Row	customer_state	seller_state	time_to_deliver	diff_estimated_delivery
1	RJ	SP	125	-99
2	SP	SP	107	-80
3	SP	SP	104	-90
4	SP	SP	66	-39
5	SP	SP	66	-39
6	SP	SP	62	-12
7	PA	MG	61	-10

The bottom of the console shows the 'PERSONAL HISTORY' and 'PROJECT HISTORY' tabs, and a 'REFRESH' button.

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

```
Select
customer_state, avg_frieght_val_per_state
from (SELECT
customer_state,
Round(AVG(freight_value),2) as avg_frieght_val_per_state
from `Target_Sales_Data.order_items` as p
join `Target_Sales_Data.orders` as o
on p.order_id=o.order_id
```


Business Case: Target United States.

```
join `Target_Sales_Data.customers` as c
on o.customer_id=c.customer_id
group by customer_state
order by avg_frieght_val_per_state
LIMIT 5) as lower_freight
UNION ALL
Select
customer_state, avg_frieght_val_per_state
from (SELECT
customer_state,
Round(AVG(freight_value),2) as avg_frieght_val_per_state
from `Target_Sales_Data.order_items` as p
join `Target_Sales_Data.orders` as o
on p.order_id=o.order_id
join `Target_Sales_Data.customers` as c
on o.customer_id=c.customer_id
group by customer_state
order by avg_frieght_val_per_state DESC
LIMIT 5) as highest_freight
```

The screenshot shows the Google Cloud BigQuery console interface. On the left, the Explorer pane displays the project structure for 'target-brazil-390016', including external connections and a dataset named 'Target_Sales_Data' with tables like 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', and 'sellers'. The main area shows a query titled 'Untitled 6' with the following SQL:

```
1 Select
2 customer_state, avg_frieght_val_per_state
```

The query results are displayed in a table with 10 rows, sorted by average freight value in descending order. The table has two columns: 'customer_state' and 'avg_frieght_val_per_state'.

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

The bottom of the console shows the Windows taskbar with the system clock indicating 20:50 on 21-06-2023.

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

```
SELECT customer_state,
time_to_deliver,
diff_estimated_delivery
FROM
(SELECT customer_state,
TIMESTAMP_DIFF(order_delivered_carrier_date, order_purchase_timestamp,
DAY) AS time_to_deliver,
TIMESTAMP_DIFF(order_estimated_delivery_date,
order_delivered_carrier_date, DAY) AS diff_estimated_delivery
FROM `Target_Sales_Data.orders` AS o
JOIN `Target_Sales_Data.customers` AS c
ON o.customer_id = c.customer_id
```

Business Case: Target United States.

```
WHERE LOWER(order_status) = "delivered" AND
TIMESTAMP_DIFF(order_delivered_carrier_date, order_purchase_timestamp, DAY) = 0
ORDER BY time_to_deliver
LIMIT 5) AS top_deliveries
UNION ALL
SELECT customer_state,
       time_to_deliver,
       diff_estimated_delivery
FROM
  (SELECT customer_state,
          TIMESTAMP_DIFF(order_delivered_carrier_date, order_purchase_timestamp,
DAY) AS time_to_deliver,
          TIMESTAMP_DIFF(order_estimated_delivery_date,
order_delivered_carrier_date, DAY) AS diff_estimated_delivery
FROM `Target_Sales_Data.orders` AS o
JOIN `Target_Sales_Data.customers` AS c
ON o.customer_id = c.customer_id
WHERE LOWER(order_status) = "delivered"
ORDER BY time_to_deliver DESC
LIMIT 5) AS bottom_deliveries;
```

The screenshot shows the Google Cloud BigQuery console interface. On the left, the 'Explorer' pane displays the project 'target-brazil-390016' and a dataset 'Target_Sales_Data' with tables like 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', and 'sellers'. The main area shows a query in 'Untitled 5' with the following SQL:

```
1 SELECT customer_state,
2       time_to_deliver,
3       diff_estimated_delivery
```

The query results are displayed in a table with columns: 'customer_state', 'time_to_deliver', and 'diff_estimated_delivery'. The results are sorted by 'time_to_deliver' in descending order.

Row	customer_state	time_to_deliver	diff_estimated_delivery
1	SP	0	35
2	RS	0	47
3	CE	0	44
4	SP	0	19
5	SP	0	18
6	RJ	125	-99
7	SP	107	-80
8	SP	104	-90
9	SP	66	-39
10	SP	62	-12

5.4. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

```
SELECT customer_state,
       Round (AVG(TIMESTAMP_DIFF(order_estimated_delivery_date,
order_delivered_carrier_date, DAY)),2) AS AVG_diff_estimated_delivery
FROM `Target_Sales_Data.orders` AS o
JOIN `Target_Sales_Data.customers` AS c
ON o.customer_id = c.customer_id
WHERE LOWER(order_status) = "delivered"
GROUP BY customer_state
ORDER BY AVG_diff_estimated_delivery asc
limit 5
```

Business Case: Target United States.

The screenshot shows the Google Cloud BigQuery console interface. The Explorer panel on the left displays the project structure for 'target-brazil-390016', including 'External connections' and 'Target_Sales_Data' with tables like 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', and 'sellers'. The main editor shows a SQL query in 'Untitled 8' that calculates the average difference in delivery time (in days) between estimated and actual delivery dates, grouped by customer state. The query results are displayed in a table with columns 'customer_state' and 'AVG_diff_estimated_delivery'. The results show that the 'SP' state has the fastest delivery with an average difference of 15.64 days.

```
1 SELECT customer_state,
2       Round (AVG(TIMESTAMP_DIFF(order_estimated_delivery_date, order_delivered_carrier_date, DAY)),2) AS
3       AVG_diff_estimated_delivery
4 FROM `Target_Sales_Data.orders` AS o
5 JOIN `Target_Sales_Data.customers` AS c
6 ON o.customer_id = c.customer_id
7 WHERE LOWER(order_status) = "delivered"
8 GROUP BY customer_state
9 ORDER BY AVG_diff_estimated_delivery asc
10 limit 5
```

customer_state	AVG_diff_estimated_delivery
SP	15.64
DF	20.79
MG	20.96
PR	21.05
ES	21.85

Observation: SP has fastest delivery

6. Analysis based on the payments:

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

```
SELECT distinct payment_type as mode_of_pay,
       COUNT(o.order_id) AS NO_OF_ORDER,
       EXTRACT(MONTH FROM order_purchase_timestamp) as months,
       EXTRACT(YEAR FROM order_purchase_timestamp) as Year
from `Target_Sales_Data.orders` as o
join `Target_Sales_Data.payments` as p
on o.order_id=p.order_id
GROUP BY months,Year,payment_type
order by payment_type
```

The screenshot shows the Google Cloud BigQuery console interface. The Explorer panel on the left displays the project structure for 'target-brazil-390016'. The main editor shows a SQL query in 'Untitled 9' that calculates the number of orders (NO_OF_ORDER) for each payment type (mode_of_pay) grouped by month and year. The query results are displayed in a table with columns 'mode_of_pay', 'NO_OF_ORDER', 'months', and 'Year'. The results show that UPI is the most common payment type, with 1518 orders in January 2018.

```
1 SELECT distinct payment_type as mode_of_pay,
2       COUNT(o.order_id) AS NO_OF_ORDER,
3       EXTRACT(MONTH FROM order_purchase_timestamp) as months,
4       EXTRACT(YEAR FROM order_purchase_timestamp) as Year
5 from `Target_Sales_Data.orders` as o
6 join `Target_Sales_Data.payments` as p
7 on o.order_id=p.order_id
8 GROUP BY months,Year,payment_type
9 order by payment_type
```

mode_of_pay	NO_OF_ORDER	months	Year
UPI	1518	1	2018
UPI	1287	4	2018
UPI	993	10	2017
UPI	590	3	2017
UPI	398	2	2017
UPI	1139	8	2018
UPI	63	10	2016
UPI	1100	6	2018
UPI	772	5	2017

Business Case: Target United States.

6.2. Find the no. of orders placed on the basis of the payment instalments that have been paid

```
SELECT payment_installments, COUNT(*) AS order_count
FROM `Target_Sales_Data.payments`
GROUP BY payment_installments
ORDER BY order_count desc
```

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

```
1 SELECT payment_installments, COUNT(*) AS order_count
2 FROM `Target_Sales_Data.payments`
3 GROUP BY payment_installments
4 ORDER BY order_count desc
```

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

Row	payment_installment	order_count
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	10	5328
6	5	5239
7	8	4268
8	6	3920
9	7	1626

The interface also shows a sidebar with the Explorer view, displaying the project structure for 'target-brazil-390016' and the 'Target_Sales_Data' dataset. The bottom status bar indicates the results per page (50) and the total number of results (1 - 24 of 24).

Observation – More than 9 instalment is least preferred.