

TITLE: Business Case Study – TARGET SQL

Description: Target is a well-known brand on a global scale and a significant retailer in the US. By providing unmatched value, creativity, innovation, and an extraordinary customer experience that no other retailer can match, Target establishes itself as a preferred shopping destination.

The operations of Target in Brazil are the subject of this business case, which offers analytical data on 100,000 orders placed between 2016 and 2018. The dataset provides a detailed look at several variables, such as order status, pricing, payment and freight performance, customer geography, product features, and customer reviews.

This large dataset can be analyzed to reveal important information on Target's Brazilian operations. The data can provide insight into several business-related topics, including order fulfilment, pricing tactics, the effectiveness of payments and shipping, client demographics, product features, and customer satisfaction levels.

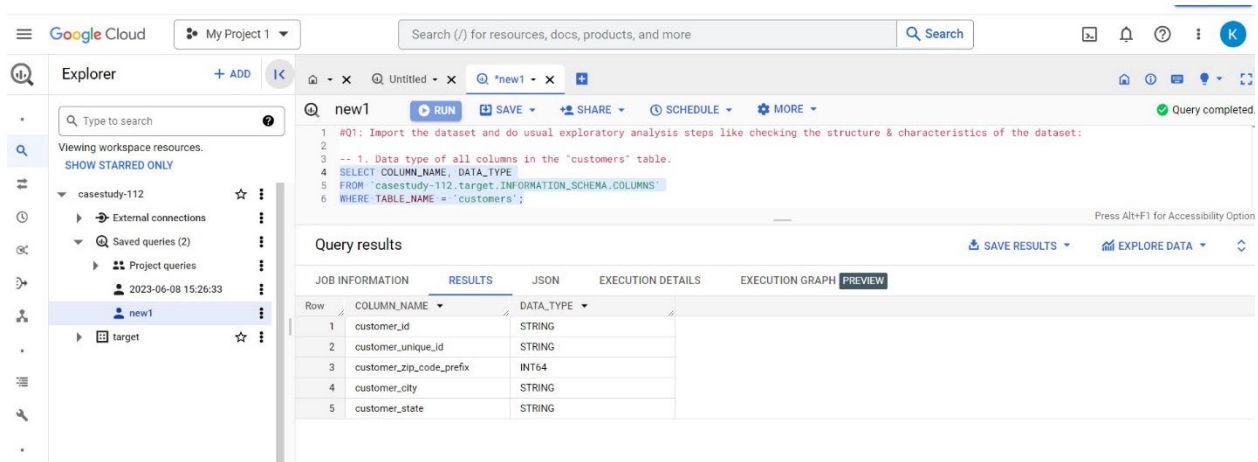
#Q1: Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:

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

Answer Query:

```
SELECT COLUMN_NAME, DATA_TYPE
FROM `casestudy-112.target.INFORMATION_SCHEMA.COLUMNS`
WHERE TABLE_NAME = 'customers';
```

Output screenshot:



The screenshot shows the Google Cloud BigQuery interface. On the left, the Explorer pane shows a project named 'casestudy-112' with a table named 'target'. The main pane displays a query named 'new1' with the following SQL code:

```
1 #Q1: Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:
2
3 -- 1. Data type of all columns in the 'customers' table.
4 SELECT COLUMN_NAME, DATA_TYPE
5 FROM `casestudy-112.target.INFORMATION_SCHEMA.COLUMNS`
6 WHERE TABLE_NAME = 'customers';
```

The query results are displayed in a table with the following columns: JOB INFORMATION, RESULTS, JSON, EXECUTION DETAILS, EXECUTION GRAPH, and PREVIEW. The RESULTS tab is selected, showing a table with 5 rows and 2 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

Insights:

1. All columns in this example are saved as strings (VARCHAR), except customer_zip_code_prefix which is integer. This implies that the data might mostly be textual in nature, and that any values relating to dates or numbers are probably saved as strings.

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

Answer Query:

```
WITH cte AS
(SELECT *,
EXTRACT (DATE FROM order_purchase_timestamp) AS order_date
FROM `casestudy-112.target.orders`
)
```

```
SELECT
DATE_DIFF(MAX(order_date), MIN(order_date), year) AS
range_in_years,
DATE_DIFF(MAX(order_date), MIN(order_date), month) AS
range_in_month,
DATE_DIFF(MAX(order_date), MIN(order_date), day) AS
range_in_days
FROM cte;
```

Output screenshot:

The screenshot shows the Google Cloud BigQuery interface. The Explorer panel on the left shows the project structure with 'casestudy-112' and its tables. The main panel displays a query named 'new1' with the following SQL code:

```
-- 2. Get the time range between which the orders were placed.
WITH cte AS
(
  SELECT *,
  EXTRACT (DATE FROM order_purchase_timestamp) AS order_date
  FROM `casestudy-112.target.orders`
)
SELECT
DATE_DIFF(MAX(order_date), MIN(order_date), year) AS range_in_years,
DATE_DIFF(MAX(order_date), MIN(order_date), month) AS range_in_month,
DATE_DIFF(MAX(order_date), MIN(order_date), day) AS range_in_days
FROM cte;
```

The query results are displayed in a table with the following columns: range_in_years, range_in_month, range_in_days. The results show 2 years, 25 months, and 773 days.

Row	range_in_years	range_in_month	range_in_days
1	2	25	773

Insights:

1. The orders in this case were placed between 2 years, 25 months, or 773 days.
2. To analyze trends, seasonality, and overall order patterns over a certain time, it can be helpful to know the time range of the orders.

--3. Count the number of Cities and States in our dataset.

Answer Query:

```
SELECT COUNT(DISTINCT geolocation_city) AS number_of_cities,
COUNT(DISTINCT geolocation_state) AS number_of_states
FROM `casestudy-112.target.geolocation`;
```

Output screenshot:

The screenshot shows the Google Cloud BigQuery interface. The Explorer panel on the left shows the project structure with 'casestudy-112' and its tables. The main panel displays a query named 'new1' with the following SQL code:

```
-- 3. Count the number of Cities and States in our dataset.
SELECT COUNT(DISTINCT geolocation_city) AS number_of_cities, COUNT(DISTINCT geolocation_state) AS number_of_states
FROM `casestudy-112.target.geolocation`;
```

The query results are displayed in a table with the following columns: number_of_cities, number_of_states. The results show 8011 cities and 27 states.

Row	number_of_cities	number_of_states
1	8011	27

Insights:

1. The dataset in this example has **27 distinct states and 8011 distinct cities.** These figures can aid in our comprehension of the geographic distribution of our clientele or the scope of our dataset.
2. Analyzing the distribution of cities and states might reveal information about the diversity or concentration of our clientele in various geographic areas. It might be helpful for regional analysis, figuring out hotspots, or figuring out how far our company has spread across the nation or the globe.

#Q2: In-depth Exploration:

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

Answer Query:

```
SELECT EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
COUNT(*) AS number_of_years
FROM `casestudy-112.target.orders`
GROUP BY year
ORDER BY year;
```

Output screenshot:

The screenshot displays the Google Cloud BigQuery console. On the left, the 'Explorer' pane shows the project 'casestudy-112' and the dataset 'target'. The main editor shows a SQL query titled 'new1' with the following text:

```
#Q2: 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) AS year, COUNT(*) AS number_of_orders
FROM `casestudy-112.target.orders`
GROUP BY year
ORDER BY year;
```

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

Row	year	number_of_orders
1	2015	929
2	2017	45101
3	2018	54011

Insights:

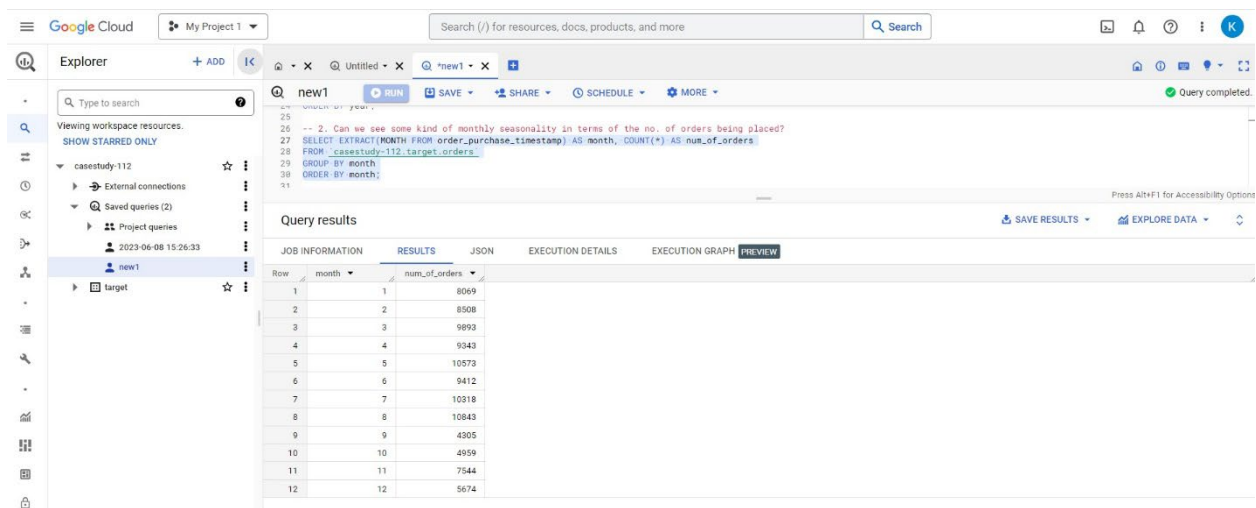
1. There has been an upward trend in the number of orders over the past few years after examining the results. A favorable trend can be seen if the order number regularly rises year over year. If there are variations or a negative tendency, however, it points to a different pattern.

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

Answer Query:

```
WITH cte AS
(
SELECT
    *,
    EXTRACT ( DATE FROM order_purchase_timestamp) AS order_date,
    EXTRACT ( YEAR FROM order_purchase_timestamp) AS order_year,
    EXTRACT ( MONTH FROM order_purchase_timestamp) AS order_month,
FROM `casestudy-112.target.orders`
)
SELECT
    order_month,
    order_year,
    COUNT(order_id) AS total_orders
FROM cte
GROUP BY
    order_month,
    order_year
ORDER BY
    order_year,
    order_month
```

Output screenshot:



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

```
-- 2. Can we see some kind of monthly seasonality in terms of the no. of orders being placed?
SELECT EXTRACT(MONTH FROM order_purchase_timestamp) AS month, COUNT(*) AS num_of_orders
FROM `casestudy-112.target.orders`
GROUP BY month
ORDER BY month;
```

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

Row	month	num_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:

1. We see a seasonal trend for Nov 2017 where there was Black Friday and there is a huge increase in the orders placed.
2. There is also a growth trend in Jan 2017 and Jan 2018 where New Years is experienced, and people may have preordered for the Carnival in Feb
3. There is also an increase in orders in Q1 of 2018 during which FIFA World Cup was scheduled.
4. Understanding monthly seasonality can help with operational planning, marketing tactics, and consumer behavior. It can aid in better planning of promotional activities, inventory management optimization, and peak period identification and resource allocation.

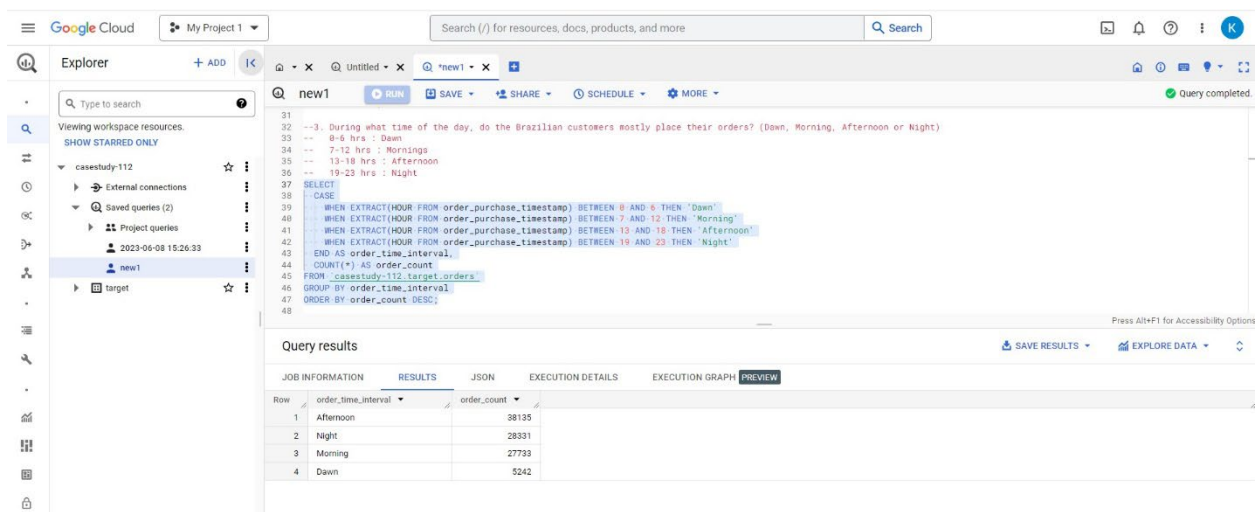
--3. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

- 0-6 hrs : Dawn
- 7-12 hrs : Mornings
- 13-18 hrs : Afternoon
- 19-23 hrs : Night

Answer Query:

```
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 order_time_interval,
COUNT(*) AS order_count
FROM `casestudy-112.target.orders`
GROUP BY order_time_interval
ORDER BY order_count DESC;
```

Output screenshot:



The screenshot displays the Google Cloud BigQuery console. The left sidebar shows the Explorer view with a project named 'casestudy-112' and a dataset named 'target'. The main editor shows a SQL query that categorizes order timestamps into four time intervals: Dawn, Morning, Afternoon, and Night, and counts the number of orders in each interval. The query is executed, and the results are displayed in a table below the editor.

Query results

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

Insights:

1. Based on the hour component of the timestamp, the query divides the order timestamps into various time groups (Dawn, Morning, Afternoon, Night). The results are then sorted based on how many orders fell inside each time frame.

2. We can ascertain the time of day when Brazilian clients often place their orders by analyzing the data. We will learn more about their ordering habits and preferences. For instance, we discover that **Brazilian clients frequently order more in the afternoons**, indicating that this is a period when people want to shop online. By scheduling customer assistance personnel or launching focused marketing efforts during the busiest ordering periods, for example, we may operate more efficient operations with the aid of this information. Also, **customers are buying least during dawn**.

#Q3. Evolution of E-commerce orders in the Brazil region:

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

Answer Query:

```
SELECT EXTRACT(MONTH FROM o.order_purchase_timestamp) AS
order_month, c.customer_state, COUNT(*) AS number_of_order
FROM `casestudy-112.target.orders` AS o
JOIN `casestudy-112.target.customers` AS c
ON o.customer_id = c.customer_id
GROUP BY order_month, c.customer_state
ORDER BY order_month, c.customer_state;
```


Output screenshot:

The screenshot shows the Google Cloud BigQuery console. On the left is the Explorer pane with a tree view containing 'casestudy-112', 'External connections', 'Saved queries (2)', 'Project queries', and 'new1'. The 'new1' query is selected. The main pane displays the SQL query and its results. The query is:
-- 1. Get the month on month no. of orders placed in each state
SELECT EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month, c.customer_state, COUNT(*) AS number_of_order
FROM `casestudy-112.target.orders` AS o
JOIN `casestudy-112.target.customers` AS c
ON o.customer_id = c.customer_id
GROUP BY order_month, c.customer_state
ORDER BY order_month, c.customer_state;
The 'Query results' pane shows a table with 11 rows. The columns are 'order_month', 'customer_state', and 'order_count'. The data shows that for every month, the state 'SP' has the highest number of orders, with counts ranging from 971 to 151.

Row	order_month	customer_state	order_count
1	1	AC	8
2	1	AL	39
3	1	AM	12
4	1	AP	11
5	1	BA	264
6	1	CE	99
7	1	DF	151
8	1	ES	159
9	1	GO	164
10	1	MA	66
11	1	MG	971

Insights:

1. We can learn more about the monthly order count for each state by examining the query's results. Over time, we can spot trends, patterns, or seasonality in the order volume for various states. We can use it to determine which states have consistently high order volumes and to pinpoint any months or states where order counts have significantly changed. Here in our data, we can find that for every month the **state called SP has the highest number of orders**.
2. We can target marketing efforts in states with rising order volumes, spot potential operational issues in states with falling order volumes or optimize inventory management based on order trends across different states by analyzing these insights.

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

Answer Query:

```
SELECT customer_state, COUNT(DISTINCT customer_id) AS  
total_custmers  
FROM `casestudy-112.target.customers`  
GROUP BY customer_state  
ORDER BY total_custmers DESC;
```

Output screenshot:

The screenshot shows the Google Cloud BigQuery console. On the left, the Explorer pane shows a project named 'My Project 1' with a dataset 'casesstudy-112' containing a table 'target'. The main editor displays a SQL query named 'new1' with the following text:

```
-- 2. How are the customers distributed across all the states?  
SELECT customer_state, COUNT(DISTINCT customer_id) AS total_customers  
FROM `casesstudy-112.target.customers`  
GROUP BY customer_state  
ORDER BY total_customers DESC;
```

Below the query, the 'Query results' section is active, showing a table with two columns: 'customer_state' and 'total_customers'. The results are sorted in descending order of 'total_customers'.

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

Insights:

1. The distribution of clients across states will be shown by analyzing the query's results. Which states have the most customers and which states have comparatively fewer consumers can be determined. Here the state called SP has the highest clients and the state called RR has the fewest clients. There are several uses for this information, including: Market targeting, Expansion opportunities and Customer service.
2. We can learn more about the geographic distribution of our client base, spot prospective growth areas, and make wise decisions to optimize our company strategy by looking at the customer distribution between states.

#Q4: 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). You can use the "payment_value" column in the payments table to get the cost of orders.

Answer Query:

```
SELECT
  ROUND((((total_payment_2018 - total_payment_2017) /
total_payment_2017) * 100), 2) AS percentage_increase
FROM (
  SELECT
    SUM(CASE
      WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017
AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND
8 THEN p.payment_value
      ELSE 0
    END) AS total_payment_2017,
    SUM(CASE
      WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2018
AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND
8 THEN p.payment_value
      ELSE 0
    END) AS total_payment_2018
  FROM
    `casestudy-112.target.payments` AS p
  JOIN
    `casestudy-112.target.orders` AS o
  ON
    p.order_id = o.order_id
);
```

Output screenshot:

The screenshot shows the Google Cloud BigQuery console. The query editor displays a SQL query that calculates the percentage increase in the total payment value from 2017 to 2018, considering only orders placed between January and August. The query is as follows:

```
--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.
SELECT
  ROUND((total_payment_2018 - total_payment_2017) / total_payment_2017 * 100, 2) AS percentage_increase
FROM (
  SELECT
    SUM(CASE
      WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017 AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8 THEN p.payment_value
      ELSE 0
    END) AS total_payment_2017,
    SUM(CASE
      WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2018 AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8 THEN p.payment_value
      ELSE 0
    END) AS total_payment_2018
  FROM
    `casestudy-112.target.payments` AS p
  JOIN
    `casestudy-112.target.orders` AS o
  ON
    p.order_id = o.order_id
);
```

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

Row	percentage_increase
1	136.98

Insights:

1. For both 2017 and 2018, only orders placed from January to August are considered.
2. To get the % increase, the query analyses the monthly prices between 2017 and 2018.
3. The findings tell us a **growth rate of approximately 137% from 2017 to 2018.**

--2. Calculate the Total & Average value of order price for each state.

Answer Query:

```
SELECT customer_state,
       ROUND(SUM(p.payment_value),2) AS total_order_price,
       ROUND(AVG(p.payment_value),2) AS average_order_price
FROM `casestudy-112.target.payments` AS p
JOIN `casestudy-112.target.orders` AS o
ON p.order_id = o.order_id
JOIN `casestudy-112.target.customers` AS c
ON o.customer_id = c.customer_id
GROUP BY customer_state
ORDER BY total_order_price DESC;
```

Output screenshot:

The screenshot displays the Google Cloud BigQuery console. On the left, the 'Explorer' pane shows a project named 'My Project 1' with a dataset 'casestudy-112' and a table 'new1'. The main editor shows a SQL query labeled 'new1' that calculates the total and average order price for each state. The query results are displayed in a table with columns 'customer_state', 'total_order_price', and 'average_order_price'. The results show data for 10 states: SP, RJ, MG, RS, PR, SC, BA, DF, GO, and ES. The 'total_order_price' column shows values ranging from approximately 165,76 to 998,226.96, and the 'average_order_price' column shows values ranging from approximately 137.5 to 170.82.

```
--3. Calculate the Total & Average value of order price for each state.
85
86
87
88 SELECT customer_state,
89         ROUND(SUM(p.payment_value), 2) AS total_order_price,
90         ROUND(AVG(p.payment_value), 2) AS average_order_price
91 FROM `casestudy-112.target.payments` AS p
92 JOIN `casestudy-112.target.orders` AS o
93 ON p.order_id = o.order_id
94 JOIN `casestudy-112.target.customers` AS c
95 ON o.customer_id = c.customer_id
96 GROUP BY customer_state
97 ORDER BY total_order_price DESC;
```

Row	customer_state	total_order_price	average_order_price
1	SP	998226.96	137.5
2	RJ	2144379.69	158.53
3	MG	1872257.25	154.71
4	RS	890898.54	157.18
5	PR	811156.38	154.15
6	SC	623086.43	165.98
7	BA	616645.82	170.82
8	DF	355141.08	161.13
9	GO	350092.31	165.76
10	ES	325967.55	154.71

Insights:

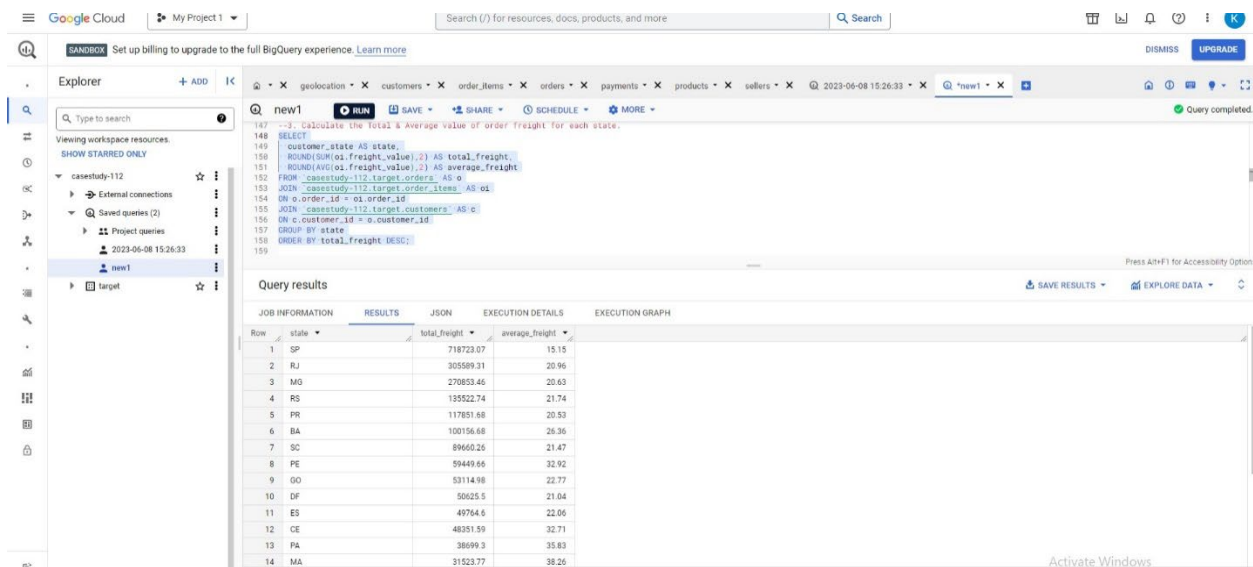
1. The sum of all order prices for each state is displayed in the "total_order_price" column, which represents the total amount of orders placed.
2. The "average_order_price" column shows the normal order value for each state together with the average order price for that state.
3. We can find states with large total order values, which point to potentially profitable marketplaces, by analyzing the results.
4. To develop focused marketing or pricing strategies, it can be helpful to compare the average order prices across states to find areas with higher or lower average spending.
5. To obtain more understanding and make wise judgements based on the data, it's critical to consider the context of each state, such as population, economic variables, or customer behavior.

--3. Calculate the Total & Average value of order freight for each state.

Answer Query:

```
SELECT
  customer_state AS state,
  ROUND(SUM(oi.freight_value),2) AS total_freight,
  ROUND(AVG(oi.freight_value),2) AS average_freight
FROM `casestudy-112.target.orders` AS o
JOIN `casestudy-112.target.order_items` AS oi
ON o.order_id = oi.order_id
JOIN `casestudy-112.target.customers` AS c
ON c.customer_id = o.customer_id
GROUP BY state
ORDER BY total_freight DESC;
```

Output screenshot:



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

```
--3. Calculate the Total & Average value of order freight for each state.
SELECT
  customer_state AS state,
  ROUND(SUM(oi.freight_value),2) AS total_freight,
  ROUND(AVG(oi.freight_value),2) AS average_freight
FROM `casestudy-112.target.orders` AS o
JOIN `casestudy-112.target.order_items` AS oi
ON o.order_id = oi.order_id
JOIN `casestudy-112.target.customers` AS c
ON c.customer_id = o.customer_id
GROUP BY state
ORDER BY total_freight DESC;
```

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

Row	state	total_freight	average_freight
1	SP	718723.07	15.15
2	RJ	305589.31	20.96
3	MG	270853.46	20.63
4	RS	135522.74	21.74
5	PR	117851.68	20.53
6	BA	100156.68	26.36
7	SC	89660.26	21.47
8	PE	59449.66	32.92
9	GO	53114.98	22.77
10	DF	50635.5	21.04
11	ES	49764.8	22.06
12	CE	48351.59	32.71
13	PA	38699.3	35.83
14	MA	31523.77	36.26

Insights:

1. We can find states with high total freight costs, here in our case a **state called SP**, by analyzing the results, which could point to regions with higher shipping prices or logistical difficulties.

2. When optimizing logistics operations or pricing strategies, it might be helpful to discover regions with higher or lower average shipping prices by comparing the average order freight costs across states.
3. Understanding the differences in order freight rates between states can offer information about local shipping habits, supplier locations, or client preferences that can be used to optimize processes and cut costs.

#Q5. 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. You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula: $\text{time_to_deliver} = \text{order_delivered_customer_date} - \text{order_purchase_timestamp}$
 $\text{diff_estimated_delivery} = \text{order_estimated_delivery_date} - \text{order_delivered_customer_date}$

Answer Query:

```
SELECT
    order_id,
    DATE_DIFF(DATE(order_delivered_customer_date),
DATE(order_purchase_timestamp), DAY) AS delivery_time,
    DATE_DIFF(DATE(order_estimated_delivery_date),
DATE(order_delivered_customer_date), DAY) AS
diff_estimated_delivery
FROM
    `casestudy-112.target.orders`;
```

Output screenshot:

The screenshot shows the Google Cloud BigQuery console. A SQL query is executed, and the results are displayed in a table. The query calculates the difference between the actual delivery time and the estimated delivery time for various orders.

```
SELECT
  order_id,
  DATE_DIFF(DATE(order_delivered_customer_date), DATE(order_purchase_timestamp), DAY) AS delivery_time,
  DATE_DIFF(DATE(order_estimated_delivery_date), DATE(order_delivered_customer_date), DAY) AS diff_estimated_delivery
FROM
  `casestudy-112.target.orders`;
```

Row	order_id	delivery_time	diff_estimated_delivery
1	1950d777989fea877539f5379...	30	-12
2	2c45c3d279cb8ffeb1c86cc28...	31	29
3	65d1e226dfa8b8dc42f65542...	36	17
4	635c994d068ac37e6e03dc54e...	31	2
5	3b97562c3aee8bdec5c3e45...	33	1
6	6814750f04c4cb6774570cde...	30	2
7	276e9ec344d3b029f83a161c...	44	-4
8	54e1a3c2b97b0809da548a59...	41	-4
9	f604fa4105ee045f6a0139ca5...	37	-1
10	302bb109d097a9fc6b9cfc5...	34	-5
11	66057d37308e787052a32828...	39	-6
12	19135c945c554eebf67576c73...	36	-2
13	4493a45e7ca1084efcd38ddeb...	34	0

Insights:

1. Insights into the effectiveness of the delivery process, including any delays or early deliveries compared to the projected timeframe, can be gained by analyzing the `delivery_time` and `diff_estimated_delivery` columns.
2. These columns can be further examined to find trends, outliers, or elements that affect delivery times or discrepancies between estimated and actual delivery dates.
3. These insights can be applied to manage customer expectations, enhance customer satisfaction, optimize the delivery process, and improve logistics operations.

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

Answer Query:

SELECT

```
high.customer_state AS high_state,
high.average_freight_value AS high_avg_freight,
low.customer_state AS low_state,
```



```

    low.average_freight_value AS low_avg_freight
FROM
(
    SELECT
        c.customer_state,
        ROUND(AVG(p.freight_value),2) AS average_freight_value,
        ROW_NUMBER() OVER(ORDER BY
(ROUND(AVG(p.freight_value),2))DESC) AS rowval1
    FROM `casestudy-112.target.orders` AS o
    JOIN `casestudy-112.target.order_items` AS p
    ON o.order_id = p.order_id
    JOIN `casestudy-112.target.customers` AS c
    ON o.customer_id = c.customer_id
    GROUP BY
        c.customer_state
    ORDER BY
        average_freight_value DESC
    LIMIT
        5
) AS high
JOIN
(
    SELECT
        c.customer_state,
        ROUND(AVG(p.freight_value),2) AS average_freight_value,
        ROW_NUMBER() OVER(ORDER BY (ROUND(AVG(p.freight_value),2)))
AS rowval2
    FROM `casestudy-112.target.orders` AS o
    JOIN `casestudy-112.target.order_items` AS p
    ON o.order_id = p.order_id
    JOIN `casestudy-112.target.customers` AS c
    ON o.customer_id = c.customer_id
    GROUP BY
        c.customer_state
    ORDER BY
        average_freight_value
    LIMIT
        5
) AS low
ON high.rowval1 = low.rowval2;

```

Output screenshot:

The screenshot displays a data analytics interface. On the left, a sidebar shows a project tree with folders like 'External connections', '2023-08-08 15:28:33', and 'new1'. The 'new1' folder is expanded, showing tables: 'customers', 'geo_location', 'order_items', 'order_reviews', 'orders', 'payments', 'products', and 'users'. The main area shows a SQL query with line numbers 159 to 243. The query finds the top 5 states with the highest and lowest average freight values. Below the query, the 'Query results' section shows a table with 5 rows and 4 columns: 'high_state', 'high_avg_freight', 'low_state', and 'low_avg_freight'. The results are as follows:

Row	high_state	high_avg_freight	low_state	low_avg_freight
1	RR	42.94	DP	12.15
2	PB	42.72	PR	20.53
3	RD	41.07	MD	20.53
4	AC	40.07	RJ	20.99
5	PI	39.15	DF	21.04

Insights:

1. The states with the **highest average freight values like states called RR and PB** may experience greater shipping prices due to reasons like remote locations, higher transportation costs, or supply chain difficulties.
2. It might be useful for our company to try to optimize logistics operations or save costs to locate places with relatively reduced shipping prices by looking at the states with the **lowest average freight values like states such as SP and PR.**
3. This data can help us develop focused initiatives, bargain freight costs, or spot possible opportunities to reduce costs in our supply chain operations.
4. When assessing the data and drawing conclusions from these insights, it is crucial to consider additional elements like distance, transportation infrastructure, carrier availability, or regional economic variations.

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

Answer Query:

```
WITH cte AS
(
    SELECT
        c.customer_state,
        ROUND(AVG(t1.delivery_time),2) AS avg_delivery_time
    FROM
    (
        SELECT
            *,
            TIMESTAMP_DIFF(order_delivered_customer_date,
            order_purchase_timestamp, day) AS delivery_time,
        FROM
            `casestudy-112.target.orders`
        WHERE
            order_status = 'delivered' AND
            order_delivered_customer_date IS NOT NULL
        ORDER BY
            order_purchase_timestamp
    ) AS t1
    JOIN
        `casestudy-112.target.customers` AS c
        ON t1.customer_id = c.customer_id
    GROUP BY
        c.customer_state
    ORDER BY
        avg_delivery_time
)

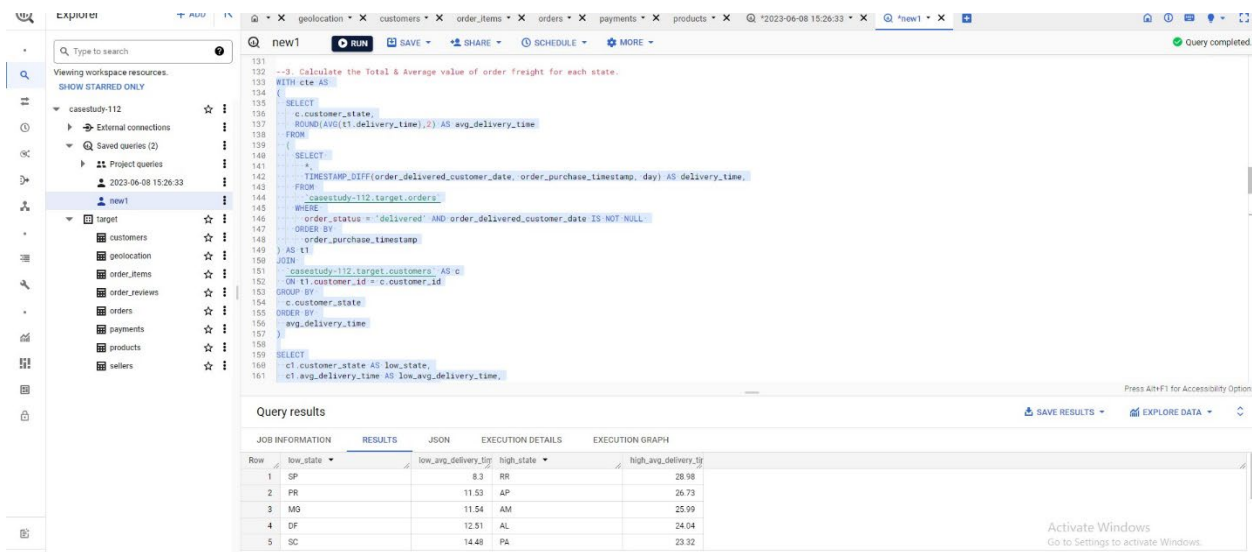
SELECT
    c1.customer_state AS low_state,
    c1.avg_delivery_time AS low_avg_delivery_time,
```

```

    c2.customer_state AS high_state,
    c2.avg_delivery_time AS high_avg_delivery_time
FROM
(
    SELECT
        *,
        ROW_NUMBER() OVER (ORDER BY cte.avg_delivery_time DESC) AS
rowval2
    FROM
        cte
    ORDER BY
        rowval2
) AS c2
JOIN
(
    SELECT
        *,
        ROW_NUMBER() OVER (ORDER BY cte.avg_delivery_time) AS
rowval1
    FROM
        cte
    ORDER BY
        rowval1
) AS c1
ON
    c1.rowval1 = c2.rowval2
LIMIT
5;

```

Output screenshot:



```
131 --3 Calculate the Total & Average value of order freight for each state.
132 WITH cte AS
133 (
134     SELECT
135         c.customer_state,
136         ROUND(AVG(t1.delivery_time),2) AS avg_delivery_time
137     FROM
138     (
139         SELECT
140             *
141             ,TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, day) AS delivery_time
142         FROM
143             casestudy112.target.orders
144         WHERE
145             order_status = 'delivered' AND order_delivered_customer_date IS NOT NULL
146     ) ORDER BY
147         order_purchase_timestamp
148     AS t1
149 JOIN
150     casestudy112.target.customers AS c
151     ON t1.customer_id = c.customer_id
152 GROUP BY
153     c.customer_state
154 )
155 ORDER BY
156     avg_delivery_time
157 )
158 SELECT
159     c1.customer_state AS low_state,
160     c1.avg_delivery_time AS low_avg_delivery_time,
```

Row	low_state	low_avg_delivery_time	high_state	high_avg_delivery_time
1	SP	8.3	RR	28.98
2	PR	11.53	AP	26.73
3	MO	11.54	AM	25.99
4	DF	12.51	AL	24.04
5	SC	14.48	PA	23.32

Insights:

1. Finding areas with effective delivery operations, quicker transit times, or solid logistics networks can be done by looking at the states like SP and PR with the lowest average delivery times and states called RR and AP with highest average delivery times.
2. These insights can be helpful for our company looking to improve customer satisfaction, operational efficiency, delivery process optimization, and setting reasonable expectations for customers based on regional delivery time patterns.
3. When evaluating the data and drawing conclusions from these insights, it's crucial to take additional elements into account, such as population density, the distinction between urban and rural locations, customer expectations, or unique logistical restrictions.
4. Utilizing this information, our company can concentrate on areas where delivery efficiency improvements can be made, thereby improving customer experiences and operational efficiencies.

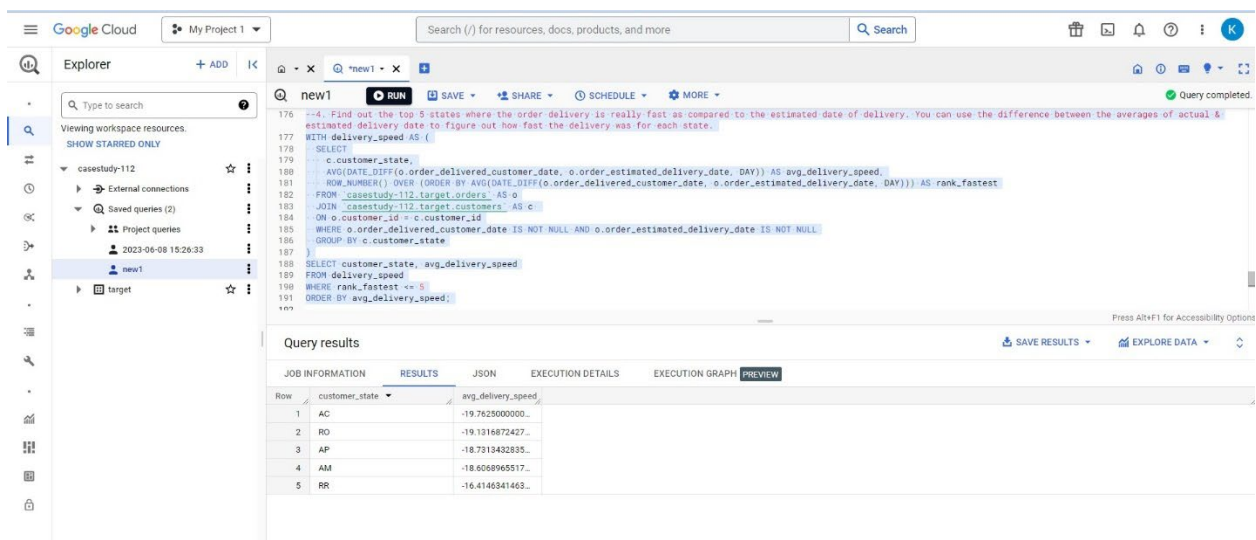
--4. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery. You

can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

Answer Query:

```
WITH delivery_speed AS (  
  SELECT  
    c.customer_state,  
    AVG(DATE_DIFF(o.order_delivered_customer_date,  
o.order_estimated_delivery_date, DAY)) AS avg_delivery_speed,  
    ROW_NUMBER() OVER (ORDER BY  
AVG(DATE_DIFF(o.order_delivered_customer_date,  
o.order_estimated_delivery_date, DAY))) AS rank_fastest  
  FROM `casestudy-112.target.orders` AS o  
  JOIN `casestudy-112.target.customers` AS c  
  ON o.customer_id = c.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 delivery_speed  
WHERE rank_fastest <= 5  
ORDER BY avg_delivery_speed;
```

Output screenshot:



The screenshot shows the Google Cloud BigQuery interface. The query editor displays the SQL query from the previous block. The query results are shown in a table with 5 rows, representing the top 5 states by delivery speed. The table has columns for Row, customer_state, and avg_delivery_speed.

Row	customer_state	avg_delivery_speed
1	AC	-19.7625000000...
2	RO	-19.1316872427...
3	AP	-18.7313432835...
4	AM	-18.6008965517...
5	RR	-16.4146341463...

Insights:

1. Our company operating in these states called AC, RO, AP, and AM where average delivery speed is highest can take advantage of the quicker delivery times by highlighting their rapid and dependable service, thereby drawing more clients, and boosting client satisfaction.
2. These data can help us improve our operations, enhance customer experience, optimize logistics, or look for expansion prospects in areas with a track record of quick order delivery.

#Q6: Analysis based on the payments:

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

Answer Query:

```
SELECT
    FORMAT_TIMESTAMP('%Y-%m', o.order_purchase_timestamp) AS
month,
    p.payment_type,
    COUNT(DISTINCT o.order_id) AS order_count
FROM `casestudy-112.target.orders` AS o
JOIN `casestudy-112.target.payments` AS p
ON o.order_id = p.order_id
GROUP BY month, p.payment_type
ORDER BY month;
```

Output screenshot:

The screenshot shows the Google Cloud BigQuery console. On the left is the Explorer pane with a tree view containing 'casestudy-112', 'External connections', 'Saved queries (2)', 'Project queries', and 'new1'. The main editor displays a SQL query:
--2. Find the month on month no. of orders placed using different payment types.
SELECT
FORMAT_TIMESTAMP('%Y-%m', o.order_purchase_timestamp) AS month,
p.payment_type,
COUNT(DISTINCT o.order_id) AS order_count
FROM `casestudy-112.target.orders` AS o
JOIN `casestudy-112.target.payments` AS p
ON o.order_id = p.order_id
GROUP BY month, p.payment_type
ORDER BY month;
The 'Query results' pane at the bottom shows a table with 10 rows and 4 columns: Row, month, payment_type, and order_count. The data is as follows:

Row	month	payment_type	order_count
1	2016-09	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	2017-01	credit_card	582
8	2017-01	UPI	197
9	2017-01	voucher	33
10	2017-01	debit_card	9

Insights:

1. We identify that credit card as a payment method was most used in November 2017.
2. To analyze seasonality, identify peak months, or evaluate the effects of marketing efforts or outside variables on consumer behavior, tracking the month-to-month trends in order counts can be helpful.
3. Based on the payment preferences noticed during various months, these insights might help firms optimize their payment procedures, customize marketing campaigns, or enhance customer experiences.

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

Answer Query:

```
SELECT payment_installments, COUNT(DISTINCT order_id) AS  
order_count  
FROM `casestudy-112.target.payments`  
GROUP BY payment_installments
```


ORDER BY payment_installments;

Output screenshot:

The screenshot shows the Google Cloud BigQuery console. The query editor displays a SQL query to find the number of orders placed based on payment installments. The query results are shown in a table with columns 'payment_installments' and 'order_count'.

```
178 --2. Find the no. of orders placed on the basis of the payment installments that have been paid.
179
180 SELECT payment_installments, COUNT(DISTINCT order_id) AS order_count
181 FROM `casestudy-112.target.payments`
182 GROUP BY payment_installments
183 ORDER BY payment_installments;
```

Row	payment_installments	order_count
1	0	2
2	1	49060
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	23

Insights:

- 1. 49060 orders were placed where payment installment was 1.**
- This analysis can help determine whether payment installment alternatives are popular or preferred by clients.
- Customers' preferences for budgeting or financing may be discerned by whether they tend to select a particular number of payment installments.
- Monitoring the distribution of orders according to payment installments might reveal information about the buying habits of clients and their preference for flexible payment methods.

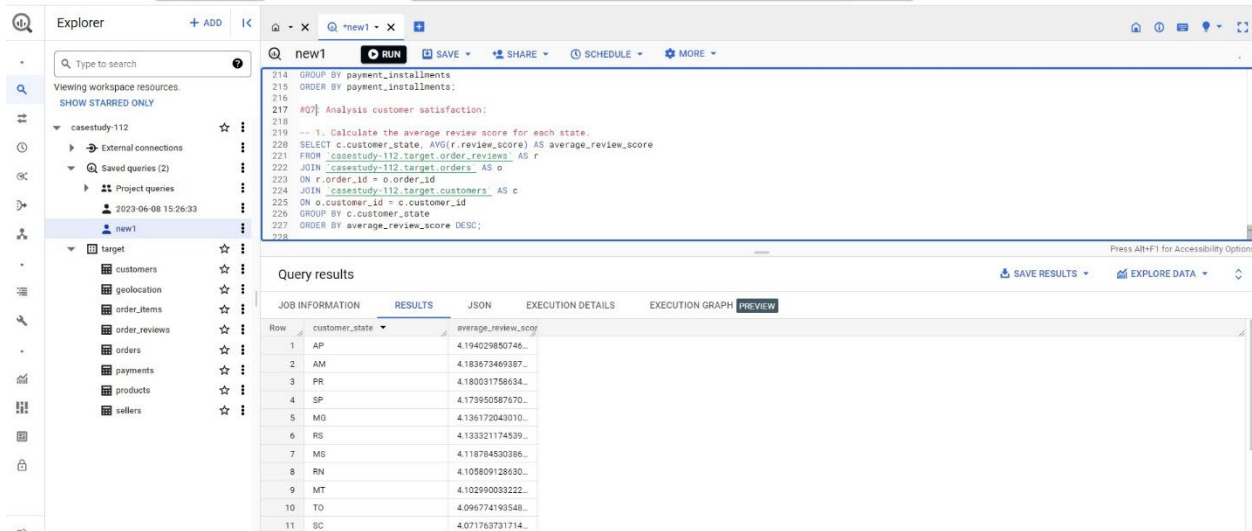
#Q7: Analysis customer satisfaction: (my own question)

-- 1. Calculate the average review score for each state.

Answer Query:

```
SELECT c.customer_state, AVG(r.review_score) AS
average_review_score
FROM `casestudy-112.target.order_reviews` AS r
JOIN `casestudy-112.target.orders` AS o
ON r.order_id = o.order_id
JOIN `casestudy-112.target.customers` AS c
ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY average_review_score DESC;
```

Output screenshot:



The screenshot shows a SQL IDE interface. On the left is an Explorer pane with a tree view of database resources including 'casestudy-112', 'target', 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', and 'vellers'. The main editor displays a SQL query with line numbers 214 to 228. Below the editor, the 'Query results' section is active, showing a table with 11 rows and 2 columns: 'customer_state' and 'average_review_score'. The results are ordered by the average review score in descending order.

Row	customer_state	average_review_score
1	AP	4.194029850746...
2	AM	4.183679469387...
3	PR	4.180031758634...
4	SP	4.179950587670...
5	MG	4.136172043010...
6	RS	4.133321174539...
7	MS	4.118784530386...
8	RN	4.105809128630...
9	MT	4.102990033222...
10	TD	4.096774193548...
11	SC	4.071763731714...

Insights:

1. The average review score for each state gives a general idea of how satisfied customers are in various areas.

-- 2. Identify the top 5 states with the highest average review scores.

Answer Query:

```
SELECT c.customer_state, AVG(r.review_score) AS
average_review_score
FROM `casestudy-112.target.order_reviews` AS r
JOIN `casestudy-112.target.orders` AS o
```

```

ON r.order_id = o.order_id
JOIN `casestudy-112.target.customers` AS c
ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY average_review_score DESC
LIMIT 5;

```

Output screenshot:

The screenshot shows the Google Cloud BigQuery console. On the left, the Explorer pane displays the project structure, including a dataset named 'casestudy-112' with tables like 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', and 'sellers'. The main editor shows a SQL query that identifies the top 5 states with the highest average review scores. The query results are displayed in a table with 5 rows, showing the top states: AP, AM, PR, SP, and MI.

Row	customer_state	average_review_score
1	AP	4.194029850746...
2	AM	4.183673469987...
3	PR	4.180031758634...
4	SP	4.173950587670...
5	MI	4.136172043010...

Insights:

1. Customers have shown greater levels of satisfaction in the top 5 states with the highest average review scores.
2. Our company may concentrate on enhancing the client experience and resolving any issues by using these insights to pinpoint areas where customer satisfaction is high or low.

#Q8: Analysis on product popularity: (my own question)

--1. Calculate the total number of products sold for each product category.

Answer Query:

```

SELECT
  p.product_category,
  COUNT(*) AS total_products_sold

```

```

FROM `casestudy-112.target.order_items` AS oi
JOIN `casestudy-112.target.products` AS p
ON oi.product_id = p.product_id
GROUP BY p.product_category
ORDER BY total_products_sold DESC
LIMIT 10;

```

Output screenshot:

The screenshot shows the Google Cloud BigQuery interface. On the left is the Explorer pane with a tree view of the 'casestudy-112' dataset, including tables like 'customers', 'order_items', 'products', and 'sellers'. The main editor shows a SQL query (lines 247-257) that calculates the total number of products sold for each product category. Below the query editor, the 'Query results' tab is active, displaying a table with 10 rows of data. The table has two columns: 'product_category' and 'total_products_sold'. The results are ordered by 'total_products_sold' in descending order.

Row	product_category	total_products_sold
1	bed table bath	11115
2	HEALTH BEAUTY	9670
3	sport leisure	8641
4	Furniture Decoration	8334
5	computer accessories	7827
6	housewares	6964
7	Watches present	5991
8	telephony	4545
9	Garden tools	4347
10	automotive	4235

Insights:

1. The popularity of various product categories can be determined by calculating the total number of products sold for each product category.
2. The analysis aids in identifying the product categories with larger sales volume, a sign of consumer preference.

--2. Analyze the correlation between the number of product photos and product sales.

Answer Query:

```

SELECT p.product_photos_qty, COUNT(*) AS total_products_sold
FROM `casestudy-112.target.products` AS p
JOIN `casestudy-112.target.order_items` AS oi
ON p.product_id = oi.product_id
GROUP BY p.product_photos_qty
ORDER BY p.product_photos_qty;

```

Output screenshot:

The screenshot displays the Google Cloud BigQuery console. At the top, there's a search bar and navigation tabs. The left sidebar shows the 'Explorer' panel with a tree view of workspace resources including 'casestudy-112', 'External connections', 'Saved queries (2)', and 'Project queries'. The main area shows a query editor with a SQL query and a 'Query results' section. The query is:
--2. Analyze the correlation between the number of product photos and product sales.
SELECT p.product_photos_qty, COUNT(*) AS total_products_sold
FROM `casestudy-112.target.products` AS p
JOIN `casestudy-112.target.order_items` AS oi
ON p.product_id = oi.product_id
GROUP BY p.product_photos_qty
ORDER BY p.product_photos_qty;
The 'Query results' section shows a table with 13 rows and 2 columns: 'product_photos_qty' and 'total_products_sold'. The results are as follows:

Row	product_photos_qty	total_products_sold
1	null	1603
2	1	56028
3	2	21963
4	3	12392
5	4	8437
6	5	5368
7	6	3786
8	7	1501
9	8	727
10	9	313
11	10	342
12	11	71
13	12	53

Insights:

1. Our company can ascertain the influence of visual material on customers' purchase selections by comprehending the relationship between the quantity of product photographs and product sales.
2. By concentrating on top-performing product categories and improving the visual presentation of products with more images, this information can be used to optimize product listings.