

TITLE: Business Case Study – TARGET SQL

Description: 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 analysing 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.

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.

Ans :

```
SELECT COLUMN_NAME, DATA_TYPE
FROM casestudy-430607.target.INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='customers'
```

Output:

The screenshot shows the BigQuery web interface. On the left, the sidebar displays project resources under 'casetudy-430607'. Under the 'target' dataset, the 'customers' table is selected. In the main area, an 'Untitled query' is running, displaying the following SQL code:

```
1 SELECT COLUMN_NAME, DATA_TYPE
2 FROM casestudy-430607.target.INFORMATION_SCHEMA.COLUMNS
3 WHERE TABLE_NAME='customers'
```

The 'RESULTS' tab is selected, showing the query results in a table format:

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:

In the above output screenshot the first column is column names and the second column is corresponding data types of customers table. All columns in this example are saved as strings, except customer_zip_code_prefix which is integer.

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

Ans:

```
WITH cte AS
(SELECT *,
EXTRACT (DATE FROM order_purchase_timestamp) AS order_date
FROM casestudy-430607.target.orders
)

SELECT MIN(order_purchase_timestamp) AS first_order,
MAX(order_purchase_timestamp) AS last_order,
DATE_DIFF(MAX(order_date),MIN(order_date),year) AS range_in_years
FROM cte;
```

Output:

The screenshot shows the Google BigQuery web interface. On the left, there's a sidebar with a search bar and sections for 'Viewing resources.', 'SHOW STARRED ONLY', and a list of datasets and tables under 'target'. The 'target' dataset contains tables like customers, geolocation, order_items, etc. On the right, the main area has tabs for 'Untitled query', 'RUN', 'SAVE', 'DOWNLOAD', 'SHARE', 'SCHEDULE', and 'MO'. Below these are buttons for 'Press Alt+F', 'SAVE RESULTS', and 'EXPL'. A 'Query results' section has tabs for 'JOB INFORMATION', 'RESULTS' (which is selected), 'CHART', 'JSON', 'EXECUTION DETAILS', and 'EXECUTION GRAPH'. The results table has columns 'Row', 'first_order', 'last_order', and 'range_in_years'. There is one row with values: first_order is 2016-09-04 21:15:19 UTC, last_order is 2018-10-17 17:30:18 UTC, and range_in_years is 2.

Row	first_order	last_order	range_in_years
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC	2

Insights:

The time range between which the orders were placed is 2 years. First column is timestamp of first order placed and second column is timestamp of last order placed in the given range.

3. Count the Cities & States of customers who ordered during the given period.

Ans :

```
SELECT COUNT(DISTINCT customer_city) AS number_of_cities,
       COUNT(DISTINCT customer_state) AS number_of_states
  FROM casestudy-430607.target.customers
 WHERE customer_id IN (SELECT customer_id
                        FROM casestudy-430607.target.orders
                       GROUP BY customer_id);
```

Output:

The screenshot shows the BigQuery web interface. On the left, there's a sidebar with a search bar and a 'Viewing resources' section. Under 'target', several datasets are listed: customers, geolocation, order_items, order_reviews, orders, payments, products, and sellers. The main area is titled 'Untitled query' with a 'RUN' button. The query code is as follows:

```
1 SELECT COUNT(DISTINCT customer_city) AS number_of_cities,
2 COUNT(DISTINCT customer_state) AS number_of_states
3 FROM casestudy-430607.target.customers
4 WHERE customer_id IN (SELECT customer_id
5                         FROM casestudy-430607.target.orders
6                         GROUP BY customer_id);
```

Below the query is a 'Query results' table with one row:

Row	number_of_cities	number_of_states
1	4119	27

Insights:

Total 4119 number of cities and 27 states of customers ordered in the given period of time. In the above query we have used the sub query for finding the customer_id's of customers who ordered during the given period.

Q2) In-depth Exploration:

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

Ans:

```
SELECT EXTRACT(YEAR FROM order_purchase_timestamp)
      AS year,
      COUNT(*) AS number_of_Orders
  FROM casestudy-430607.target.orders
 GROUP BY year
 ORDER BY year;
```

Output:

The screenshot shows the Google BigQuery web interface. On the left, there's a sidebar with a search bar and a tree view of resources under 'Viewing resources'. Under the 'target' dataset, several tables are listed: customers, geolocation, order_items, order_reviews, orders, payments, products, and sellers. The main area is titled 'Untitled query' with a 'RUN' button. The query code is:

```
1 SELECT EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
2           COUNT(*) AS number_of_Orders
3   FROM casestudy-430607.target.orders
4 GROUP BY year
5 ORDER BY year;
```

The results section shows a table with three rows of data:

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

Insights:

In the above output we can see the number of orders placed every year. After analysing the output there is a growing trend in the number of orders placed over the past year.

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

Ans:

```
SELECT EXTRACT(month from order_purchase_timestamp) AS month_number,
       FORMAT_TIMESTAMP("%B", order_purchase_timestamp) AS month,
       COUNT(*) AS number_of_Orders
  FROM casestudy-430607.target.orders
 GROUP BY month,month_number
 ORDER BY month_number;
```

Output:

The screenshot shows the BigQuery web interface. On the left is the Explorer sidebar with resources like 'casestudy-430607' and 'target'. The main area is titled 'Untitled query' with a 'RUN' button. Below it is a table titled 'Query results' showing monthly order counts from January to December.

Row	month_number	month	number_of_Orders
1	1	January	8069
2	2	February	8508
3	3	March	9893
4	4	April	9343
5	5	May	10573
6	6	June	9412
7	7	July	10318
8	8	August	10843
9	9	September	4305
10	10	October	4959
11	11	November	7544
12	12	December	5674

Insights:

In the above output screenshot we can see some kind of monthly seasonality in terms of the number of orders being placed. In the month of august highest number of orders were placed. And in the month of september lowest number of orders were placed.

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

Ans:

```

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-430607.target.orders
GROUP BY order_time_interval
ORDER BY order_count DESC;

```

Output:

The screenshot shows the BigQuery web interface. On the left, there's a sidebar for 'Viewing resources' with sections for 'casestudy-430607' (Queries, Notebooks, Data canvases, External connections) and 'target' (customers, geolocation, order_items, order_reviews, orders, payments, products). The main area is titled 'Untitled query' with a 'RUN' button. The query code is pasted above. Below it, the 'Query results' section shows a table with four rows. The table has columns for 'Row', 'order_time_interval', and 'order_count'. The data is as follows:

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

Insights:

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. Brazilian customers frequently order more in the afternoons and customers placed least orders in Dawn time.

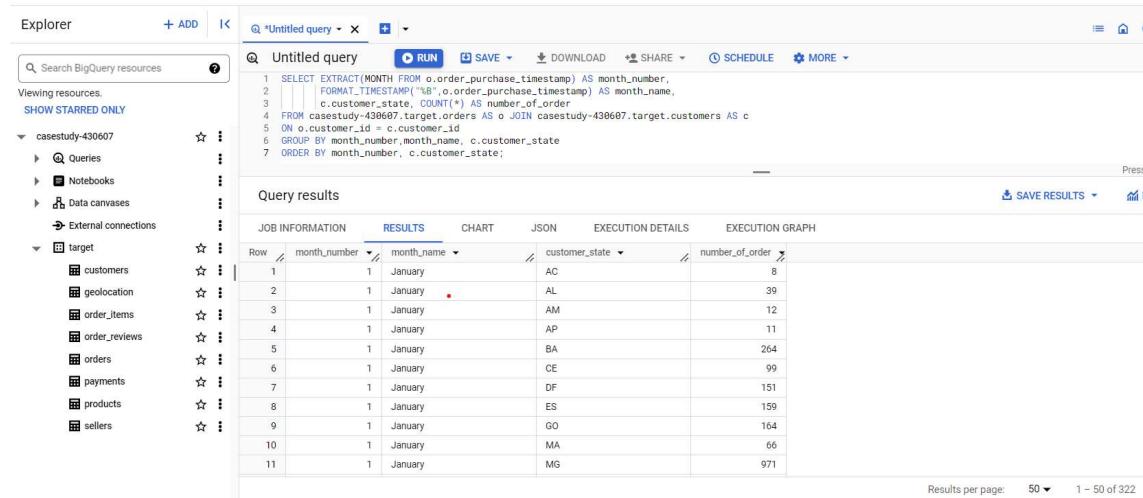
Q3) Evolution of E-commerce orders in the Brazil region:

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

Ans:

```
SELECT EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month_number,
       FORMAT_TIMESTAMP("%B",o.order_purchase_timestamp) AS month_name,
       c.customer_state, COUNT(*) AS number_of_order
  FROM casestudy-430607.target.orders AS o JOIN
casestudy-430607.target.customers AS c
    ON o.customer_id = c.customer_id
 GROUP BY month_number,month_name, c.customer_state
ORDER BY month_number, c.customer_state;
```

Output:



The screenshot shows the Google BigQuery interface. On the left is the Explorer sidebar with project and dataset navigation. The main area contains an 'Untitled query' tab with the SQL code. Below it is the 'Query results' section, which displays a table of data. The table has columns: Row, month_number, month_name, customer_state, and number_of_order. The data shows monthly order counts for various states, with SP having the highest count of 971 in January.

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

Insights:

In the above output we can find the month on month number of orders placed in each state. After analysing the query result we can find that for every month the state called "SP" has the highest number of orders.

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

Ans:

```
SELECT customer_state,
       COUNT(DISTINCT customer_id) AS total_custmers
  FROM casestudy-430607.target.customers
 GROUP BY customer_state
 ORDER BY total_custmers DESC;
```

Output:

The screenshot shows the BigQuery interface. On the left is the Explorer sidebar with resources like 'casestudy-430607' and 'target'. The main area is titled 'Untitled query' with the following SQL code:

```
1 SELECT customer_state,
2        COUNT(DISTINCT customer_id) AS total_custmers
3   FROM casestudy-430607.target.customers
4  GROUP BY customer_state
5 ORDER BY total_custmers DESC;
```

Below the code is the 'Query results' section. It has tabs for 'JOB INFORMATION', 'RESULTS' (which is selected), 'CHART', 'JSON', 'EXECUTION DETAILS', and 'EXECUTION GRAPH'. The results table has columns 'Row', 'customer_state', and 'total_custmers'. The data is as follows:

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

Insights:

In the above query result we can see the total customers of each state along with their states. After analysing the query result we can find that the state called "SP" has the highest number of customers and the state called "RR" has the fewest number of customers.

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.

Ans:

```
With cte as(
    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_cost_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_cost_2018
    FROM casestudy-430607.target.payments AS p JOIN
    casestudy-430607.target.orders AS o
    ON p.order_id = o.order_id
)

SELECT
    ROUND(((total_cost_2018 - total_cost_2017) / total_cost_2017)
    * 100), 2) AS percentage_increase
FROM cte;
```

Output:

The screenshot shows a database interface with two tabs: 'Untitled query' and 'Untitled query'. The first tab contains the SQL code provided above. The second tab, titled 'Query results', displays the output of the query. The results table has one row with the value 136.98 under the 'percentage_increase' column.

Row	percentage_increase
1	136.98

Insights:

In the above query first we have written cte for finding the cost of orders placed from January to August in the year 2017 and 2018 individually. After analysing the result we can find the growth rate approximately 137% from 2017 to 2018.

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

Ans:

```
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-430607.target.payments AS p JOIN
casestudy-430607.target.orders AS o
  ON p.order_id = o.order_id JOIN
casestudy-430607.target.customers AS c
  ON o.customer_id = c.customer_id
 GROUP BY customer_state
 ORDER BY total_order_price DESC;
```

Output:

The screenshot shows the BigQuery web interface. On the left, the 'Explorer' sidebar lists datasets like 'casestudy-430607' and 'target', and tables such as 'customers', 'payments', etc. In the center, an 'Untitled query' tab is open with the SQL code provided above. Below the code, the 'RESULTS' tab is selected, showing a table titled 'Query results'. The table has columns: Row, customer_state, total_order_price, and average_order_price. The data is as follows:

Row	customer_state	total_order_price	average_order_price
1	SP	599826.96	137.5
2	RJ	2144379.69	158.53
3	MG	1872257.26	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

Insights:

In the above result the sum of all order prices for each state is displayed in the "total_order_price" column, which represents the total revenue in each state. The "average_order_price" column shows the average order price of customers in each state.

After analysing the query result we can find that the state called "SP" has the highest revenue and the state called "RR" has the lowest revenue. In the state of "RR" the number of customers are less compared to other states. while increasing the customers we can get more revenue because "RR" state has good average_order_price.

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

Ans:

```
SELECT customer_state,
       ROUND(SUM(it.freight_value),2) AS total_freight,
       ROUND(AVG(it.freight_value),2) AS average_freight
  FROM casestudy-430607.target.orders AS o
  JOIN casestudy-430607.target.order_items AS it ON o.order_id = it.order_id
  JOIN casestudy-430607.target.customers AS c ON c.customer_id = o.customer_id
 GROUP BY customer_state
 ORDER BY total_freight DESC;
```

Output:

The screenshot shows the BigQuery interface with the following details:

- Explorer:** Shows resources under "casestudy-430607" and "target".
- Untitled query:** The SQL query is pasted here:

```
1 SELECT customer_state,
       ROUND(SUM(it.freight_value),2) AS total_freight,
       ROUND(AVG(it.freight_value),2) AS average_freight
  FROM casestudy-430607.target.orders AS o
  JOIN casestudy-430607.target.order_items AS it ON o.order_id = it.order_id
  JOIN casestudy-430607.target.customers AS c ON c.customer_id = o.customer_id
 GROUP BY customer_state
 ORDER BY total_freight DESC;
```
- Query results:** The results table has columns: Row, customer_state, total_freight, average_freight. The data is as follows:

Row	customer_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

Insights:

By analysing the Query result We can find the state called "SP" has the highest total freight costs which could point to regions with higher shipping prices.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:

- time_to_deliver = order_delivered_customer_date - order_purchase_timestamp
- diff_estimated_delivery = order_delivered_customer_date - order_estimated_delivery_date

Ans:

```
SELECT order_id,
DATE_DIFF(DATE(order_delivered_customer_date),
          DATE(order_purchase_timestamp), DAY) AS time_to_delivar,
DATE_DIFF(DATE(order_estimated_delivery_date),
          DATE(order_delivered_customer_date), DAY) AS diff_estimated_delivery
FROM casestudy-430607.target.orders;
```

Output:

The screenshot shows a data analysis interface with a query editor at the top and a results table below. The query editor contains the following SQL code:

```
1 SELECT order_id,
2        DATE_DIFF(DATE(order_delivered_customer_date),
3                   DATE(order_purchase_timestamp), DAY) AS time_to_deliver,
4        DATE_DIFF(DATE(order_estimated_delivery_date),
5                   DATE(order_delivered_customer_date), DAY) AS diff_estimated_delivery
6   FROM casestudy-430607.target.orders;
7
```

The results table has columns: Row, order_id, time_to_deliver, and diff_estimated_delivery. The data is as follows:

Row	order_id	time_to_deliver	diff_estimated_delivery
1	1950d777989f6a877539f5379...	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	31	29
3	65d1e226dfaeb8cdc42f66542...	36	17
4	695c894d068ac37e6e03dc54e...	31	2
5	3b97562c3aee8bdedcb5c2e45...	33	1
6	68f47f50f04c4cb6774570cfde...	30	2
7	276e9ec344d3bf029ff83a161c...	44	-4

Results per page: 50 ▾ 1 – 50 of 99441

Insights:

By analysing the above query result we can find out the effectiveness of the delivery process, including any delays or early deliveries. It can be applied to manage customer expectations, enhance customer satisfaction, optimize the delivery process.

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

Ans:

```
with cte as(
SELECT customer_state,
       ROUND(AVG(it.freight_value),2) AS average_freight,
       rank() over(order by ROUND(AVG(it.freight_value),2) DESC) as rank
  FROM casestudy-430607.target.orders AS o
 JOIN casestudy-430607.target.order_items AS it ON o.order_id = it.order_id
```

```

JOIN casestudy-430607.target.customers AS c ON c.customer_id = o.customer_id
GROUP BY customer_state
ORDER BY average_freight DESC
limit 5),
cte1 as (
SELECT customer_state,
       ROUND(AVG(it.freight_value),2) AS average_freight,
       rank() over(order by ROUND(AVG(it.freight_value),2) asc) as rank
FROM casestudy-430607.target.orders AS o
JOIN casestudy-430607.target.order_items AS it
ON o.order_id = it.order_id
JOIN casestudy-430607.target.customers AS c ON c.customer_id = o.customer_id
GROUP BY customer_state
ORDER BY average_freight asc
limit 5)

select cte.customer_state as high_avg_states,
       cte.average_freight,
       cte1.customer_state as low_avg_states,
       cte1.average_freight
from cte join cte1
on cte.rank=cte1.rank

```

Output:

The screenshot shows the Google BigQuery interface. On the left, the 'Explorer' sidebar lists datasets like 'casestudy-430607' and 'target', and tables such as 'customers', 'order_items', etc. The main area is titled 'Untitled query' and contains the SQL code provided above. Below the code, the 'Query results' section displays a table with five rows of data.

Row	high_avg_states	average_freight	low_avg_states	average_freight_1
1	RR	42.98	SP	15.15
2	PB	42.72	PR	20.53
3	RO	41.07	MG	20.63
4	AC	40.07	RJ	20.96
5	PI	39.15	DF	21.04

Insights:

In the above query result we can find that The states with the highest average freight values like states called RR and PB may experience greater shipping prices due to various reasons. And the states with the lowest average freight values like states such as SP and PR relatively reduced shipping prices by looking at the average freight values.

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

Ans:

```
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-430607.target.orders
WHERE order_status = 'delivered'
AND order_delivered_customer_date IS NOT NULL
ORDER BY order_purchase_timestamp) AS t1
JOIN casestudy-430607.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 avg_delivery_time,
c2.customer_state AS high_state,
c2.avg_delivery_time AS avg_delivery_time
FROM
(SELECT *,ROW_NUMBER() OVER (ORDER BY cte.avg_delivery_time DESC) AS rnk
FROM cte
ORDER BY rnk LIMIT 5) AS c2
JOIN
(SELECT *,ROW_NUMBER() OVER (ORDER BY cte.avg_delivery_time) AS rnk
FROM cte
ORDER BY rnk LIMIT 5) AS c1
ON c1.rnk = c2.rnk;
```

Output:

The screenshot shows the Google BigQuery interface. On the left, there's an 'Explorer' sidebar with a search bar and a tree view of resources under 'casestudy-430607'. Under 'target', several tables are listed: customers, geolocation, order_items, order_reviews, orders, payments, products, and sellers. The main area is titled 'Untitled query' with a 'RUN' button. The query code is as follows:

```
1 WITH cte AS (SELECT c.customer_state,ROUND(AVG(t1.delivery_time),2) AS avg_delivery_time
2   FROM
3   (SELECT *, TIMESTAMP_DIFF(order_delivered_customer_date,
4   | order_purchase_timestamp, day) AS delivery_time,
5   | FROM casestudy-430607.target.orders
6   WHERE order_status = 'delivered' AND order_delivered_customer_date IS NOT NULL
7   ORDER BY order_purchase_timestamp) AS t1
8 JOIN
9 casestudy-430607.target.customers AS c
10 ON t1.customer_id = c.customer_id
11 GROUP BY c.customer_state
12 ORDER BY avg_delivery_time)
13
14 SELECT c1.customer_state AS low_state, c1.avg_delivery_time AS avg_delivery_time,
15 c2.customer_state AS high_state, c2.avg_delivery_time AS avg_delivery_time
16 FROM (SELECT * ,ROW_NUMBER() OVER (ORDER BY cte.avg_delivery_time DESC) AS rnk
17   FROM cte
18   ORDER BY rnk
19   limit 5) AS c2
20 JOIN (SELECT * ,ROW_NUMBER() OVER (ORDER BY cte.avg_delivery_time) AS rnk
21   FROM cte
22   ORDER BY rnk
23   limit 5) AS c1
24 ON c1.rnk = c2.rnk]
```

Below the code, there's a 'Query results' section with tabs for 'RESULTS', 'CHART', 'JSON', 'EXECUTION DETAILS', and 'EXECUTION GRAPH'. The 'RESULTS' tab shows a table with the following data:

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

Insights:

In the above query result we can find that the states with the highest average delivery time like states called RR and AP and the states with the lowest average delivery time like states called SP and PR. 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.

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.

Ans:

```

with cte 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 rnk
FROM casestudy-430607.target.orders AS o
JOIN casestudy-430607.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
ORDER BY rnk
limit 5)

select customer_state,avg_delivery_speed
from cte;

```

Output:

The screenshot shows a data processing interface with a query editor at the top and a results table below it.

Query Editor (Untitled query):

```

1 with cte as
2 (SELECT c.customer_state,
3     AVG(DATE_DIFF(o.order_delivered_customer_date,o.order_estimated_delivery_date, DAY)) AS avg_delivery_speed,
4     ROW_NUMBER() OVER (ORDER BY AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date, DAY))) AS
5     rnk
6 FROM casestudy-430607.target.orders AS o JOIN casestudy-430607.target.customers AS c
7 ON o.customer_id = c.customer_id
8 WHERE o.order_delivered_customer_date IS NOT NULL AND o.order_estimated_delivery_date IS NOT NULL
9 GROUP BY c.customer_state
10 ORDER BY rnk
11 limit 5)
12 select customer_state,avg_delivery_speed
13 from cte;

```

Results Table:

customer_state	avg_delivery_speed
AC	-19.762500000...
RO	-19.1316872427...
AP	-18.7313432835...
AM	-18.6068965517...
RR	-16.4146341463...

Insights:

In the above query result we can find that AC, RO, AP, AM and RR are the top 5 states where the order delivery is really fast as compared to the estimated date of delivery. We can use this data to get more revenue from the customers of these states through increasing their orders.

Q6) Analysis based on the payments:

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

Ans:

```
SELECT FORMAT_TIMESTAMP('%Y-%m', o.order_purchase_timestamp) AS year_month,
       p.payment_type,
       COUNT(o.order_id) AS total_order
  FROM casestudy-430607.target.orders AS o JOIN
       casestudy-430607.target.payments AS p ON o.order_id = p.order_id
 GROUP BY year_month, p.payment_type
 ORDER BY year_month;
```

Output:

The screenshot shows the BigQuery web interface. On the left, the 'Explorer' sidebar lists datasets like 'casestudy-430607' and 'target', and specific tables such as 'customers', 'geolocation', 'order_items', etc. The main area is titled 'Untitled query' and contains the SQL code for the analysis. Below the code, the 'Query results' section displays the output in a table format. The table has columns for 'year_month' (Date), 'payment_type' (Type of payment), and 'total_order' (Number of orders). The data shows monthly order counts for credit_card, UPI, and voucher payments across different months from 2016-10 to 2017-02.

year_month	payment_type	total_order
2016-10	credit_card	254
2016-10	UPI	63
2016-10	voucher	23
2016-10	debit_card	2
2016-12	credit_card	1
2017-01	credit_card	583
2017-01	UPI	197
2017-01	voucher	61
2017-01	debit_card	9
2017-02	credit_card	1356
2017-02	UPI	398

Insights:

In the above query result we can find that credit card as a payment method was most used in November 2017. Because highest number of orders placed in the november 2017 through credit card.

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

Ans :

```
SELECT payment_installments,
       COUNT(DISTINCT order_id) AS total_orders
  FROM casestudy-430607.target.payments
 GROUP BY payment_installments
```

Output :

The screenshot shows the Google BigQuery web interface. On the left is the 'Explorer' sidebar with project and dataset navigation. The main area is titled 'Untitled query' with a 'RUN' button. Below it is the 'Query results' section with tabs for 'RESULTS' (selected), 'CHART', 'JSON', 'EXECUTION DETAILS', and 'EXECUTION GRAPH'. The results table has columns 'Row', 'payment_installment', and 'total_orders'. The data is as follows:

Row	payment_installment	total_orders
1	0	2
2	1	49050
3	2	12389
4	3	10443
5	4	7088
6	5	5234
7	6	3916
8	7	1623
9	8	4253
10	9	644
11	10	5315
12	11	22

Insights:

In the above query result we can find that 49060 orders were placed where payment instalment was 1. According to this analysis we can say that most of the customers are prefer Or interested in the one payment instalment process.