Case Study with SQL : Target (e-commerce) Platform

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

- I have imported all the 8 tables from the target data sets in my BigQuery platform and started analysing all the tables and its characteristics generally.

a) Data type of all columns in the "customers" table

```
select
column_name,
data_type
from `scaler-dsml-sql-gcp.target_case_study.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'customers';
```

Press Option+F1 for accessibility options.

Query results — SAVE RESULTS — EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

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

- I just verified all the data types of each column from the customers table by fetching information from the table.
- Except customer_zip_code_prefix column all the other columns are string data type.

b)Get the time range between which the orders were placed.

```
SELECT
MIN(order_purchase_timestamp) AS earliest_order_timestamp,
MAX(order_purchase_timestamp) AS latest_order_timestamp
FROM `scaler-dsml-sql-gcp.target_case_study.orders`;
```

Press Option+F1 for accessibility optio

Query results — SAVE RESULTS — EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

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

As per the requirement, I have analysed the range of orders placed during a given interval of time period.
I used max(), min() functions to fetch range.
Minimum timestamp of order were placed on 2016-09-04 21:15:19 UTC
Maximum timestamp of order were placed on 2018-10-17 17:30:18 UTC

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

```
#Count the Cities & States of customers who ordered during the given period.
select
count(distinct C.customer_city) as city,
count(distinct C.customer_state) as state
from `scaler-dsml-sql-gcp.target_case_study.customers` C
JOIN `scaler-dsml-sql-gcp.target_case_study.orders` O
on C.customer_id = O.customer_id;
```

Press Option+F1 for accessibility option

Query results — SAVE RESULTS — EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	city	state
1	4119	27

- It results totally 4119 cities and 27 states of customers were placed order during the given period

Case Study with SQL : Target (e-commerce) Platform

2) In-depth Exploration:

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

```sql
#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 num_of_orders
FROM `scaler-dsml-sql-gcp.target_case_study.orders`
GROUP BY year
ORDER BY year;
```

Press Option+F1 for accessib

Query results — SAVE RESULTS — EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

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

- There are drastic changes in the number of orders placed between 2016 and 2017,18.

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

```sql
#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 monthly_seasonality
FROM `scaler-dsml-sql-gcp.target_case_study.orders`
GROUP BY month
ORDER BY month
```

Query results — SAVE RESULTS — EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	month	monthly_seasonality
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

- It results number of orders placed month wise

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

i) 0-6 hrs : Dawn
ii) 7-12 hrs : Mornings
iii) 13-18 hrs : Afternoon
iv) 19-23 hrs : Night

```sql
SELECT
  CASE
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 6 THEN 'Dawn'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN 'Morning'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN 'Afternoon'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 19 AND 23 THEN 'Night'
  END AS time_of_the_day,
  COUNT(*) AS num_of_orders
FROM `scaler-dsml-sql-gcp.target_case_study.orders`
GROUP BY time_of_the_day
ORDER BY time_of_the_day
```

Press Option+F1 for accessibility options.

Query results — SAVE RESULTS — EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	time_of_the_day	num_of_orders
1	Afternoon	38135
2	Dawn	5242
3	Morning	27733
4	Night	28331

- It results orders been placed different times in a day

Case Study with SQL : Target (e-commerce) Platform

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

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

```sql
#Get the month on month no. of orders placed in each state.
select
c.customer_state,
extract(month from order_purchase_timestamp) as month,
count(1) as num_orders
from `scaler-dsml-sql-gcp.target_case_study.orders` o
INNER JOIN `scaler-dsml-sql-gcp.target_case_study.customers` c
ON o.customer_id = c.customer_id
group by 1,2
order by 3 desc;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON

Row	customer_state	month	num_orders
1	SP	8	4982
2	SP	5	4632
3	SP	7	4381
4	SP	6	4104
5	SP	3	4047
6	SP	4	3967
7	SP	2	3357
8	SP	1	3351
9	SP	11	3012
10	SP	12	2357

b)How are the customers distributed across all the states?

```sql
#How are the customers distributed across all the states?
select
customer_state as state,
count(customer_id) as num_customers,
from `scaler-dsml-sql-gcp.target_case_study.customers`
group by 1
order by 2 desc;
```

Query results

JOB INFORMATION | RESULTS | CHART | JS

Row	state	num_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

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

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

```sql
#a)Get the % increase in the cost of orders from year 2017 to 2018
#(include months between Jan to Aug only)
SELECT
    year,
    cost,
    LEAD(cost) OVER (ORDER BY year) AS next_year_cost,
    ROUND(((LEAD(cost) OVER (ORDER BY year) - cost) / cost) * 100, 2) AS percent_increase
FROM (
    SELECT
        EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
        SUM(payment_value) AS cost
    FROM
        `scaler-dsml-sql-gcp.target_case_study.orders` o
        INNER JOIN `scaler-dsml-sql-gcp.target_case_study.payments` p ON o.order_id = p.order_id
    WHERE
        EXTRACT(YEAR FROM order_purchase_timestamp) BETWEEN 2017 AND 2018
        AND EXTRACT(MONTH FROM order_purchase_timestamp) BETWEEN 1 AND 8
    GROUP BY
        EXTRACT(YEAR FROM order_purchase_timestamp)
) AS yearly_costs;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	year	cost	next_year_cost	percent_increase
1	2018	8694733.839999...	null	null
2	2017	3669022.119999...	8694733.839999...	136.98

Case Study with SQL : Target (e-commerce) Platform

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

```sql
#Calculate the Total & Average value of order price for each state.
select
c.customer_state,
count(distinct o.order_id) as num_orders,
ROUND(sum(oi.price),2) as total_value,
ROUND(avg(oi.price),2) as avg_price
from `scaler-dsml-sql-gcp.target_case_study.order_items` oi
INNER JOIN `scaler-dsml-sql-gcp.target_case_study.orders` o
ON oi.order_id = o.order_id
INNER JOIN `scaler-dsml-sql-gcp.target_case_study.customers` c
ON c.customer_id = o.customer_id
group by customer_state
order by 2 desc;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	customer_state	num_orders	total_value	avg_price
1	SP	41375	5202955.05	109.65
2	RJ	12762	1824092.67	125.12
3	MG	11544	1585308.03	120.75
4	RS	5432	750304.02	120.34
5	PR	4998	683083.76	119.0
6	SC	3612	520553.34	124.65
7	BA	3358	511349.99	134.6
8	DF	2125	302603.94	125.77
9	ES	2025	275037.31	121.91
10	GO	2007	294591.95	126.27

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

```sql
#Calculate the Total & Average value of order freight for each

select
c.customer_state,
count(distinct o.order_id) as num_orders,
ROUND(sum(oi.
freight_value),2) as total_value,
ROUND(avg(oi.
freight_value),2) as avg_freight_price
from `scaler-dsml-sql-gcp.target_case_study.order_items` oi
INNER JOIN `scaler-dsml-sql-gcp.target_case_study.orders` o
ON oi.order_id = o.order_id
INNER JOIN `scaler-dsml-sql-gcp.target_case_study.customers` c
ON c.customer_id = o.customer_id
group by customer_state
order by 2 desc;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	customer_state	num_orders	total_value	avg_freight_price
1	SP	41375	718723.07	15.15
2	RJ	12762	305589.31	20.96
3	MG	11544	270853.46	20.63
4	RS	5432	135522.74	21.74
5	PR	4998	117851.68	20.53
6	SC	3612	89660.26	21.47
7	BA	3358	100156.68	26.36
8	DF	2125	50625.5	21.04
9	ES	2025	49764.6	22.06
10	GO	2007	53114.98	22.77

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

a)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.

```sql
#5.a Find the no. of days taken to deliver each order from the order's purchase date as delivery time.
#calculate the difference (in days) between the estimated & actual delivery date of an order.

SELECT
timestamp_diff(order_delivered_customer_date, order_purchase_timestamp, DAY)
AS time_to_deliver,
timestamp_diff(order_delivered_customer_date, order_estimated_delivery_date, DAY)
AS diff_estimated_deliver
FROM `scaler-dsml-sql-gcp.target_case_study.orders`
WHERE order_status = 'delivered';
```

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	time_to_deliver	diff_estimated_deliver
1	30	-1
2	32	0
3	29	-1
4	43	4
5	40	4
6	37	1
7	33	5
8	38	6
9	36	2
10	34	0

Case Study with SQL : Target (e-commerce) Platform

b)(i) Find out the top 5 states with the highest average freight value.

```sql
#5.b Find out top 5 states with the highest freight value

SELECT s.seller_state,
AVG(oi.freight_value) AS avg_freight_value
FROM `scaler-dsml-sql-gcp.target_case_study.sellers` s
JOIN `scaler-dsml-sql-gcp.target_case_study.order_items` oi
ON s.seller_id = oi.seller_id
GROUP BY s.seller_state
ORDER BY avg_freight_value DESC
LIMIT 5;
```

Press Option+F1 for accessibility o

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	seller_state	avg_freight_value
1	RO	50.91285714285...
2	CE	46.38117021276...
3	PB	39.18815789473...
4	PI	36.94333333333...
5	AC	32.84

b)(ii) Find out the top 5 states with the lowest average freight value.

```sql
# Find out top 5 states with the lowest freight value

SELECT s.seller_state,
AVG(oi.freight_value) AS avg_freight_value
FROM `scaler-dsml-sql-gcp.target_case_study.sellers` s
JOIN `scaler-dsml-sql-gcp.target_case_study.order_items` oi
ON s.seller_id = oi.seller_id
GROUP BY s.seller_state
ORDER BY avg_freight_value ASC
LIMIT 5;
```

Press Option+F1 for accessibility

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	seller_state	avg_freight_value
1	SP	18.45221266585...
2	PA	19.38874999999...
3	RJ	19.47486508924...
4	DF	20.57181312569...
5	PR	22.72096874639...

Case Study with SQL : Target (e-commerce) Platform

c)(i) Find out the top 5 states with the highest average delivery time.

```sql
# 5.C Find out the top 5 states with the highest average delivery time.
SELECT
c.customer_state,
AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY)) AS avg_delivery_time_days
FROM `scaler-dsml-sql-gcp.target_case_study.customers` c
JOIN `scaler-dsml-sql-gcp.target_case_study.orders` o
ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY avg_delivery_time_days DESC
LIMIT 5;
```

Press Option+F1 for accessibility opt

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_state	avg_delivery_time_da
1	RR	28.97560975609…
2	AP	26.73134328358…
3	AM	25.98620689655…
4	AL	24.04030226700…
5	PA	23.31606765327…

c)(ii) Find out the top 5 states with the lowest average delivery time.

```sql
# 5.C Find out the top 5 states with the lowest average delivery time.
SELECT
c.customer_state,
AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY)) AS avg_delivery_time_days
FROM `scaler-dsml-sql-gcp.target_case_study.customers` c
JOIN `scaler-dsml-sql-gcp.target_case_study.orders` o
ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY avg_delivery_time_days
LIMIT 5;
```

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION

Row	customer_state	avg_delivery_time_days
1	SP	8.2980614890726656
2	PR	11.526711354864963
3	MG	11.543813298106565
4	DF	12.509134615384614
5	SC	14.479560191711288

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

```sql
#5.d. Find out the top 5 states where the order delivery is really fast as compared
# to the estimated date of delivery.

SELECT
customer_state AS state,
ROUND(SUM(TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp,
DAY))/COUNT(ORDER_ID), 2) AS average_time_for_delivery,
ROUND(SUM(TIMESTAMP_DIFF(order_estimated_delivery_date, order_purchase_timestamp,
DAY))/COUNT(ORDER_ID), 2) AS average_est_del_time,
FROM `scaler-dsml-sql-gcp.target_case_study.orders` o
INNER JOIN `scaler-dsml-sql-gcp.target_case_study.customers` c
ON o.customer_id=c.customer_id
WHERE order_status='delivered'
GROUP BY customer_state
ORDER BY (average_time_for_delivery-average_est_del_time)
limit 5;
```

Case Study with SQL : Target (e-commerce) Platform

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	year	month	payment_type	num_orders
1	2016	9	credit_card	3
2	2016	10	UPI	63
3	2016	10	credit_card	254
4	2016	10	debit_card	2
5	2016	10	voucher	23
6	2016	12	credit_card	1
7	2017	1	UPI	197
8	2017	1	credit_card	583
9	2017	1	debit_card	9
10	2017	1	voucher	61

6).Analysisbased on the payments:

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

```
#6.a. Find the month on month no. of orders placed using different payment types.

SELECT
EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
p.payment_type,
COUNT(*) AS num_orders
FROM `scaler-dsml-sql-gcp.target_case_study.orders` o
JOIN `scaler-dsml-sql-gcp.target_case_study.payments` p
ON o.order_id = p.order_id
GROUP BY year, month, p.payment_type
ORDER BY year, month, p.payment_type;

```

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	year	month	payment_type	num_orders
1	2016	9	credit_card	3
2	2016	10	UPI	63
3	2016	10	credit_card	254
4	2016	10	debit_card	2
5	2016	10	voucher	23
6	2016	12	credit_card	1
7	2017	1	UPI	197
8	2017	1	credit_card	583
9	2017	1	debit_card	9
10	2017	1	voucher	61

Case Study with SQL : Target (e-commerce) Platform

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

```sql
#6.b. Find the no. of orders placed on the basis of the payment installments that
#have been paid.
select
payment_installments as installmenets,
count(order_id) as num_of_orders
from `scaler-dsml-sql-gcp.target_case_study.payments`
where payment_installments >= 1
group by payment_installments
order by 2 desc;
```

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	installmenets	num_of_orders
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
10	9	644

7) Actionable insights and recommendations:

- In conclusion, the analysis of the target dataset revealed several key insights and actionable recommendations to drive growth and enhance operational efficiency.
- The total & average value of order price for state SP is comparatively higher than the other states. And the difference between SP and RJ is 64%. So it is better to focus on regional purchases for overall growth.
- I would suggest Integrate augmented reality technology into the target (e-commerce) platform to offer immersive shopping experiences, allowing customers to visualise products in their real-world environment which will enhance the purchase rate.
- Also observed that most of the customers were ordered using credit cards.Providing extra offers for those who make payments using credit cards would definitely increase the number of purchases for ex : 20% off, up to $ 350.
- Finally Customer satisfaction directly correlates with efficient delivery times and accurate order fulfilment.