Business Case: Target SQL

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**

Field name	Type	Mode	Ke
customer_id	STRING	NULLABLE	-
customer_unique_id	STRING	NULLABLE	-
customer_zip_code_prefix	INTEGER	NULLABLE	-
customer_city	STRING	NULLABLE	-
customer_state	STRING	NULLABLE	-

Inference : In BigQuery, you may double click any table, whose schema you want to see. Under the schema tab , you may see the desired result, i.e column name and its corresponding data type.

1.2 Get the time range between which the orders were placed

Query

```
SELECT
  MIN(order_purchase_timestamp) AS min_order_timestamp,
  MAX(order_purchase_timestamp) AS max_order_timestamp
FROM
  `SQL_CASE_STUDY.orders`
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXE

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

Inference : The time range when orders were placed is between 9:15 pm approx on 4/9/2016 and approx 5:30 pm on 17/10/2018.

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

Query:

```
select count(distinct c.customer_city) as city_count,count(distinct c.customer_state) as
state_count
from
`SQL_CASE_STUDY.orders` o inner join `SQL_CASE_STUDY.customers` c
on o.customer_id=c.customer_id
```

Query results

JOB INFORMATION | RESULTS | CHART

Row	city_count	state_count
1	4119	27

Inference : The customers who have ordered are in the orders table and details of city and state can be found in the customers table. So we need to join them and get the desired results.So we can say that customers who ordered were spread around 4119 cities and 27 states in total.

In-depth Exploration

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

Query:

```
Select year,count(year) as count_of_orders
from
(select Extract(year from order_purchase_timestamp) as year
from `SQL_CASE_STUDY.orders`)X
group by X.year
order by X.year
```

Query results

JOB INFORMATION | RESULTS | CHART

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

Inference : Clearly we can see that there is a growing trend in the number of orders. A drastic change from 2016 to 2017, and the trend maintains a positive slope even from 2017 to 2018. We have simply extracted years from the order's timestamp and hence analyzed counts as shown in the query.

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

Query:

```
Select month,count(month) as count_of_orders
from
(select FORMAT_TIMESTAMP('%B',order_purchase_timestamp) as month
from `SQL_CASE_STUDY.orders`)X
group by X.month
order by CASE
      WHEN month = 'January' THEN 1
      WHEN month = 'February' THEN 2
      WHEN month = 'March' THEN 3
      WHEN month = 'April' THEN 4
      WHEN month = 'May' THEN 5
      WHEN month = 'June' THEN 6
      WHEN month = 'July' THEN 7
      WHEN month = 'August' THEN 8
      WHEN month = 'September' THEN 9
      WHEN month = 'October' THEN 10
      WHEN month = 'November' THEN 11
      WHEN month = 'December' THEN 12
   END;
```

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

Query:

```
select max(count_of_orders) max_count_by_month,min(count_of_orders)
min_count_by_month,avg(count_of_orders) avg_count_by_month
from
(Select month,count(month) as count_of_orders
from
(select FORMAT_TIMESTAMP('%B',order_purchase_timestamp) as month
from `SQL_CASE_STUDY.orders`)X
group by X.month)Y
```

JOB INFORMATION | RESULTS | CHART | JSON | EXE

Row	max_count_by_mont	min_count_by_mont	avg_count_by_mont
1	10843	4305	8286.749999999...

Inference : We may deduce that sales between January and August are kind of above average sales (8286.74). But it drastically falls between September and December, where sales are way below the monthly average. So it means people are buying less in that particular time period of September to December in brazil. Moreover, maximum monthly sales are observed in August, i.e. 10843, after which sales begin to decline as September approaches.It is ironical to observe that September has the lowest sales i.e. 4305.

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**

Query:

```
select Time_of_Day,Count(Time_of_Day) as Count_of_orders,rank() over(order by
Count(Time_of_Day) desc) as rank_orders
from
(select
CASE
  WHEN hour between 0 and 6 THEN "Dawn"
  WHEN hour between 7 and 12 THEN "Mornings"
  WHEN hour between 13 and 18 THEN "Afternoon"
  WHEN hour between 19 and 23 THEN "Night"
END as Time_of_Day
from
(select Extract(hour from order_purchase_timestamp) as hour
from `SQL_CASE_STUDY.orders`) X)Y
group by Time_of_Day
order by Count_of_orders desc
```

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION

Row	Time_of_Day	Count_of_orders	rank_orders
1	Afternoon	38135	1
2	Night	28331	2
3	Mornings	27733	3
4	Dawn	5242	4

Inference : We may clearly observe that Brazilians are mostly ordering in the afternoon's time followed by night. The minimum orders are placed at the time of dawn, maybe they are fond of waking up late. In Fact the minimum slot orders i.e 5242 is nearly 1/8th of the maximum slot orders i.e 38135. Company needs to focus on load balancing and smooth functioning more in the afternoon slot in order to maintain a suitable user experience.

Evolution of E-commerce orders in the Brazil region

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

Query:

```
select customer_state,FORMAT_TIMESTAMP('%B',order_purchase_timestamp) as
month,count(customer_state)
from
`SQL_CASE_STUDY.orders` o inner join `SQL_CASE_STUDY.customers` c
on o.customer_id=c.customer_id
group by customer_state,month
order by customer_state,
CASE
      WHEN month = 'January' THEN 1
      WHEN month = 'February' THEN 2
      WHEN month = 'March' THEN 3
      WHEN month = 'April' THEN 4
      WHEN month = 'May' THEN 5
      WHEN month = 'June' THEN 6
      WHEN month = 'July' THEN 7
      WHEN month = 'August' THEN 8
      WHEN month = 'September' THEN 9
      WHEN month = 'October' THEN 10
      WHEN month = 'November' THEN 11
      WHEN month = 'December' THEN 12
  END;
```

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	customer_state	month	count_of_orders
1	AC	January	8
2	AC	February	6
3	AC	March	4
4	AC	April	9
5	AC	May	10
6	AC	June	7
7	AC	July	9
8	AC	August	7
9	AC	September	5
10	AC	October	6
11	AC	November	5

Inference : We have provided these details, after joining orders and customers table, then for clubbing state and month parameters , we have grouped them together and hence counted their corresponding number of orders.

3.2 How are the customers distributed across all the states?

Query:

```
select customer_state,count(customer_state) as count_of_customers
from `SQL_CASE_STUDY.customers`
group by customer_state
order by count_of_customers desc
```

JOB INFORMATION | RESULTS | CHART | JSON

Row	customer_state	count_of_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

Inference : The maximum number of customers are from state SP(41746) and minimum number of customers are from RR (46 customers). We have simply aggregated the count of states from the customers table after grouping it over state value.

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

4.1 Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).
You can use the "payment_value" column in the payments table to get the cost of orders.

Query:

```
Select
Z.Total_order_value_2017,
Z.Total_order_value_2018,
round(((Z.Total_order_value_2018-Z.Total_order_value_2017)/Z.Total_order_value_2017*100),
2) as Percentage_change
from
(Select round(SUM(if(year=2017,payment_value,0)),2) as Total_order_value_2017,
round(SUM(if(year=2018,payment_value,0)),2) as Total_order_value_2018
from
(Select payment_value,year,month
from
(Select *,
Extract(year from o.order_purchase_timestamp) as year,
Extract(month from o.order_purchase_timestamp) as month
 from
`SQL_CASE_STUDY.orders` o inner join `SQL_CASE_STUDY.payments` p
on o.order_id=p.order_id)X
where (year=2018 or year=2017) and (month between 1 and 8))Y)Z
```

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION

Row	Total_order_value_2017	Total_order_value_2018	Percentage_change
1	3669022.12	8694733.84	136.98

Inference : The query calculates the total order value for the years 2017 and 2018, considering only the months from January to August. It then computes the percentage change in the total order value between these two years. The results are rounded to two decimal places for clarity. Specifically, it provides the total order values for each year and the percentage increase from 2017 to 2018. Their is an approximate 136 % change in order value from 2017 to 2018.

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

```
select
distinct customer_state,
round(sum(payment_value) over(partition by customer_state),0) as Total_order_value,
round(avg(payment_value) over(partition by customer_state),0) as Avg_order_value
from
(select customer_state,payment_value
from
`SQL_CASE_STUDY.orders`o
inner join
 `SQL_CASE_STUDY.payments` p on o.order_id=p.order_id
inner join
`SQL_CASE_STUDY.customers`c on o.customer_id=c.customer_id)X
order by Total_order_value desc,Avg_order_value desc
```

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTIO

Row	customer_state	Total_order_value	Avg_order_value
1	SP	5998227.0	138.0
2	RJ	2144380.0	159.0
3	MG	1872257.0	155.0
4	RS	890899.0	157.0
5	PR	811156.0	154.0
6	SC	623086.0	166.0
7	BA	616646.0	171.0
8	DF	355141.0	161.0
9	GO	350092.0	166.0
10	ES	325968.0	155.0
11	PE	324850.0	188.0
12	CE	279464.0	200.0
13	PA	218296.0	216.0
14	MT	187029.0	195.0

Inference : The state with highest Total order value is SP(5998227.0) and the one with lowest is RR(10065.0) . The state with the highest average order value is PB (248.0) and the one with the lowest is SP (138.0). Interestingly, the state with the highest total has the lowest average , which shows that the number of orders in SP are quite high but individual cart value is usually low.

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

Query:

```
select
distinct customer_state,
round(sum(freight_value) over(partition by customer_state),0) as Total_freight_value,
round(avg(freight_value) over(partition by customer_state),0) as Avg_freight_value
from
(select customer_state,freight_value
from
`SQL_CASE_STUDY.orders`o
inner join
 `SQL_CASE_STUDY.order_items` oi on o.order_id=oi.order_id
inner join
`SQL_CASE_STUDY.customers`c on o.customer_id=c.customer_id)X
order by Total_freight_value desc,Avg_freight_value desc
```

Row	customer_state	Total_freight_value	Avg_freight_value
1	SP	718723.0	15.0
2	RJ	305589.0	21.0
3	MG	270853.0	21.0
4	RS	135523.0	22.0
5	PR	117852.0	21.0
6	BA	100157.0	26.0
7	SC	89660.0	21.0
8	PE	59450.0	33.0
9	GO	53115.0	23.0
10	DF	50626.0	21.0
11	ES	49765.0	22.0
12	CE	48352.0	33.0
13	PA	38699.0	36.0
14	MA	31524.0	38.0

Inference : The state with highest Total freight value is SP(718723.0) and the one with lowest is RR(2235.0) . The state with the highest average freight value is PB (43.0) and the one with the lowest is SP (15.0). Interestingly, the state with the highest total has the lowest average , which shows that the number of orders in SP are quite high but individual cart value is usually low.Also SP is ranked one in both Total order value and Total freight value, which seems obvious, as more the sales, more the transportation. Low average in SP shows that transportation is cheaper over there whereas high average in PB indicates expensive transportation over there.

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.

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**

Query:

```
select
order_id,order_purchase_timestamp,order_estimated_delivery_date,order_delivered_customer_date,
Date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as time_to_deliver,
Date_diff(order_delivered_customer_date,order_estimated_delivery_date,day) as diff_estimated_delivery
from
`SQL_CASE_STUDY.orders`
```

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

Row	order_id	order_purchase_timestamp	order_estimated_delivery_date	order_delivered_customer_date	time_to_deliver	diff_estimated_deliv
1	1950d777989f6a877539f5379...	2018-02-19 19:48:52 UTC	2018-03-09 00:00:00 UTC	2018-03-21 22:03:51 UTC	30	12
2	2c45c33d2f9cb8ff8b1c86cc28...	2016-10-09 15:39:56 UTC	2016-12-08 00:00:00 UTC	2016-11-09 14:53:50 UTC	30	-28
3	65d1e226dfaeb8cdc42f66542...	2016-10-03 21:01:41 UTC	2016-11-25 00:00:00 UTC	2016-11-08 10:58:34 UTC	35	-16
4	635c894d068ac37e6e03dc54e...	2017-04-15 15:37:38 UTC	2017-05-18 00:00:00 UTC	2017-05-16 14:49:55 UTC	30	-1
5	3b97562c3aee8bdedcb5c2e45...	2017-04-14 22:21:54 UTC	2017-05-18 00:00:00 UTC	2017-05-17 10:52:15 UTC	32	0
6	68f47f50f04c4cb6774570cfde...	2017-04-16 14:56:13 UTC	2017-05-18 00:00:00 UTC	2017-05-16 09:07:47 UTC	29	-1
7	276e9ec344d3bf029ff83a161c...	2017-04-08 21:20:24 UTC	2017-05-18 00:00:00 UTC	2017-05-22 14:11:31 UTC	43	4
8	54e1a3c2b97fb0809da548a59...	2017-04-11 19:49:45 UTC	2017-05-18 00:00:00 UTC	2017-05-22 16:18:42 UTC	40	4
9	fd04fa4105ee8045f6a0139ca5...	2017-04-12 12:17:08 UTC	2017-05-18 00:00:00 UTC	2017-05-19 13:44:52 UTC	37	1
10	302bb8109d097a9fc6e9cefc5...	2017-04-19 22:52:59 UTC	2017-05-18 00:00:00 UTC	2017-05-23 14:19:48 UTC	33	5
11	66057d37308e787052a32828...	2017-04-15 19:22:06 UTC	2017-05-18 00:00:00 UTC	2017-05-24 08:11:57 UTC	38	6

Inference : The query leverages the DATE_DIFF function to calculate two key metrics for each order: the delivery time and the accuracy of delivery estimates. The time_to_deliver is derived by finding the number of days between the purchase date and the actual delivery date, while diff_estimated_delivery measures the discrepancy between the estimated and actual delivery dates. These calculations help analyze the efficiency and reliability of the order delivery process.

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

Query (Top 5 for highest average freight value)

```
select
distinct customer_state,
round(avg(freight_value) over(partition by customer_state),0) as Avg_freight_value
from
(select customer_state,freight_value
from
`SQL_CASE_STUDY.orders`o
inner join
 `SQL_CASE_STUDY.order_items` oi on o.order_id=oi.order_id
inner join
`SQL_CASE_STUDY.customers`c on o.customer_id=c.customer_id)X
order by Avg_freight_value desc
Limit 5
```

Query (Top 5 for lowest average freight value)

```
select
distinct customer_state,
round(avg(freight_value) over(partition by customer_state),0) as Avg_freight_value
from
(select customer_state,freight_value
from
`SQL_CASE_STUDY.orders`o
inner join
 `SQL_CASE_STUDY.order_items` oi on o.order_id=oi.order_id
inner join
`SQL_CASE_STUDY.customers`c on o.customer_id=c.customer_id)X
order by Avg_freight_value asc
Limit 5
```

Top 5 for highest average freight value

JOB INFORMATION | RESULTS | CHART | JSON

Row	customer_state	Avg_freight_value
1	PB	43.0
2	RR	43.0
3	RO	41.0
4	AC	40.0
5	PI	39.0

Top 5 for lowest average freight value

JOB INFORMATION | RESULTS | CHART | JSON

Row	customer_state	Avg_freight_value
1	SP	15.0
2	PR	21.0
3	DF	21.0
4	RJ	21.0
5	SC	21.0

Inference : The queries calculate the average freight value for each state and identify the top 5 states with the highest and lowest average freight values, respectively. By using the `LIMIT` clause, the results are restricted to the top 5 states, which are then sorted in descending or ascending order of their average freight values. Moreover lowest average freight value has been a boon for SP in terms of making it number one under total order value.

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

Query (Top 5 for highest average delivery value)

```
select
distinct customer_state,
round(avg(X.time_to_deliver) over(partition by customer_state),2) as avg_delivery_time
from
(select customer_state,
Date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as time_to_deliver
from
`SQL_CASE_STUDY.orders` o
inner join
`SQL_CASE_STUDY.customers` c
on o.customer_id=c.customer_id) X
order by avg_delivery_time desc
LIMIT 5
```

Query (Top 5 for lowest average delivery value)

```
select
distinct customer_state,
round(avg(X.time_to_deliver) over(partition by customer_state),2) as
avg_delivery_time
from
(select customer_state,
Date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as
time_to_deliver
from
`SQL_CASE_STUDY.orders` o
inner join
`SQL_CASE_STUDY.customers` c
on o.customer_id=c.customer_id) X
order by avg_delivery_time asc
LIMIT 5
```

Top 5 for highest average delivery value

JOB INFORMATION | RESULTS | CHART | JSON

Row	customer_state	avg_delivery_time
1	RR	28.98
2	AP	26.73
3	AM	25.99
4	AL	24.04
5	PA	23.32

Top 5 for lowest average delivery value

JOB INFORMATION | RESULTS | CHART | JSON

Row	customer_state	avg_delivery_time
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48

Inference **:** RR has the highest average delivery time, making it a big concern for the company. If we are a careful observer, then, it's easy to notice that the total order sales was lowest for RR. May be slow delivery issues are the major factors for low sales.Moreover, the lowest average delivery time for the state SP has been a perk, pointing out to a fact, that SP has the highest order value in comparison to all the other states.

5.4 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.

Query:

```
select
distinct customer_state,
round(avg(X.diff_estimated_delivery) over(partition by customer_state),2) as
avg_diff_delivery_time
from
(select customer_state,
Date_diff(order_delivered_customer_date,order_estimated_delivery_date,day) as
diff_estimated_delivery
from
`SQL_CASE_STUDY.orders` o
inner join
`SQL_CASE_STUDY.customers` c
on o.customer_id=c.customer_id) X
order by avg_diff_delivery_time
Limit 5
```

JOB INFORMATION | RESULTS | CHART | JSON

Row	customer_state	avg_diff_delivery_tim
1	AC	-19.76
2	RO	-19.13
3	AP	-18.73
4	AM	-18.61
5	RR	-16.41

Inference : These are the 5 states in the above given table where the average of difference between actual delivery and estimated delivery is lowest. Negative value for this average is an indication that the difference between actual delivery and estimated delivery is negative, which means that the product reached way before its estimated delivery time.

Analysis based on the payments

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

Query :

```
select FORMAT_TIMESTAMP('%B',order_purchase_timestamp) as
month,payment_type,Count(payment_type) as count_of_orders
from
`SQL_CASE_STUDY.orders` o inner join `SQL_CASE_STUDY.payments` p
on o.order_id=p.order_id
group by month,payment_type
order by
CASE
   WHEN month = 'January' THEN 1
   WHEN month = 'February' THEN 2
   WHEN month = 'March' THEN 3
   WHEN month = 'April' THEN 4
   WHEN month = 'May' THEN 5
   WHEN month = 'June' THEN 6
   WHEN month = 'July' THEN 7
   WHEN month = 'August' THEN 8
   WHEN month = 'September' THEN 9
   WHEN month = 'October' THEN 10
   WHEN month = 'November' THEN 11
   WHEN month = 'December' THEN 12
END;
```

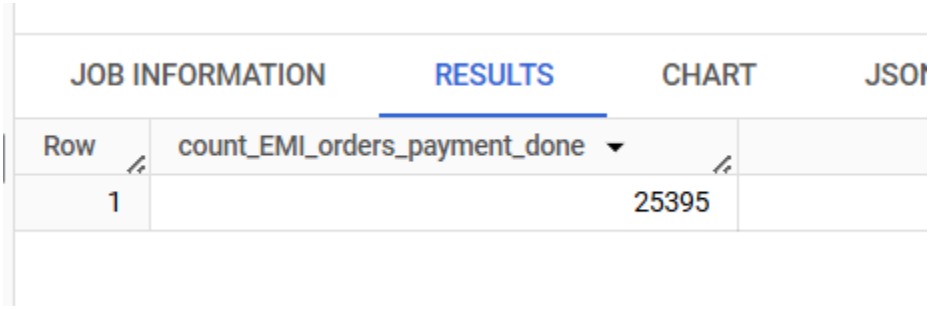
Row	month	payment_type	count_of_orders
1	January	credit_card	6103
2	January	UPI	1715
3	January	voucher	477
4	January	debit_card	118
5	February	UPI	1723
6	February	credit_card	6609
7	February	voucher	424
8	February	debit_card	82
9	March	credit_card	7707
10	March	UPI	1942

Inference : After grouping over month (extracted from timestamp of an order) and payment_types, we are able to calculate the count of orders corresponding to the month and payment type combinations.

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

Query:

```
select Count(distinct order_id) as count_EMI_orders_payment_done
from `SQL_CASE_STUDY.payments`
where payment_type="credit_card" and payment_installments=payment_sequential
```



Row	count_EMI_orders_payment_done
1	25395

Inference : The query calculates the number of distinct orders paid via credit card where all installments have been completed. By filtering the payments table for `payment_type` as "credit_card" and matching `payment_installments` with `payment_sequential`, it identifies fully paid installment orders. The result shows there are 25,395 such orders, indicating successful completion of all credit card installments.