I. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset.

A. Data type of all columns in the “customers” table.

QUERY:

```
SELECT COLUMN_NAME,DATA_TYPE
FROM ecommerce-406417.target.INFORMATION_SCHEMA.COLUMNS
WHERE table_name = 'customers'
```

Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW

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

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

QUERY:

```
select min(order_purchase_timestamp) as `first_order`,max(order_purchase_timestamp) as `last_order`
from `target.orders`;
```

Query results

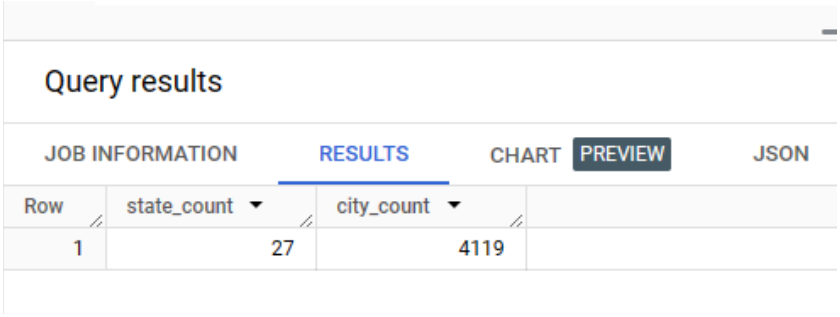
JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON

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

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

QUERY:

```
select count(distinct(customer_state)) as `state_count`,count(distinct(customer_city)) as `city_count`
from `target.customers`
where customer_id in (
                 select customer_id
                from `target.orders`
                where order_purchase_timestamp between (select
                min(order_purchase_timestamp) from `target.orders`)
                AND (select max(order_purchase_timestamp) from `target.orders`)) ;
```



II. In-depth Exploration:

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

QUERY:

```
with `YOY_order_count` as (
select month,
sum(case when year=2016 then order_count else 0 end ) as `Y16_orders`,
sum(case when year=2017 then order_count else 0 end ) as `Y17_orders`,
sum(case when year=2018 then order_count else 0 end ) as `Y18_orders`
from
(
select extract(year from order_purchase_timestamp) as `year`,extract(month from
order_purchase_timestamp) as `month`,count(order_id)
`order_count`
from `target.orders`
group by year,month
)
group by month
)
select month,Y16_orders,Y17_orders,Y18_orders,
round(if(Y16_orders=0,Y17_orders,(Y17_orders/Y16_orders)-1),2) as `YOY_2017`,
round(if(Y17_orders=0,Y18_orders,(Y18_orders/Y17_orders)-1),2) as `YOY_2018`
from YOY_order_count
order by month;
```

Query results

SAVE RESULTS | EXPLO

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

Row	month	Y16_orders	Y17_orders	Y18_orders	YOY_2017	YOY_2018
1	1	0	800	7269	800.0	8.09
2	2	0	1780	6728	1780.0	2.78
3	3	0	2682	7211	2682.0	1.69
4	4	0	2404	6939	2404.0	1.89
5	5	0	3700	6873	3700.0	0.86
6	6	0	3245	6167	3245.0	0.9
7	7	0	4026	6292	4026.0	0.56
8	8	0	4331	6512	4331.0	0.5
9	9	4	4285	16	1070.25	-1.0
10	10	324	4631	4	13.29	-1.0
11	11	0	7544	0	7544.0	-1.0
12	12	1	5673	0	5672.0	-1.0

INSIGHTS-

1. By looking at YOY_2017 we can spot positive increments in the number of orders placed during the year 2017 vs year 2016.
2. Fourth quarter of year 2017 record highest sales in terms of number of orders placed.
3. Year 2018, first quarter recorder highest sales and fourth quarter recorded negative growth rate.

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

```
select month,
sum(case when year=2016 then order_count else 0 end ) as `Y16_orders`,
sum(case when year=2017 then order_count else 0 end ) as `Y17_orders`,
sum(case when year=2018 then order_count else 0 end ) as `Y18_orders`
from
(
select extract(year from order_purchase_timestamp) as `year`,extract(month
from order_purchase_timestamp) as `month`,count(order_id) as `order_count`
from `target.orders`
group by year,month
order by year,month
)
group by month
order by month
;
```

Query results

SA

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

Row	month	Y16_orders	Y17_orders	Y18_orders
1	1	0	800	7269
2	2	0	1780	6728
3	3	0	2682	7211
4	4	0	2404	6939
5	5	0	3700	6873
6	6	0	3245	6167
7	7	0	4026	6292
8	8	0	4331	6512
9	9	4	4285	16
10	10	324	4631	4
11	11	0	7544	0
12	12	1	5673	0

INSIGHTS :

1. We can not specify a common monthly seasonality for all three years.
2. If we observe all three years individually we can spot the maximum sale month.
    a. October 2016- number of order 324.
    b. November 2017-number of order 7544.
    c. January 2018-number of orders 7269.
3. Similarly can also identify best selling quarters..

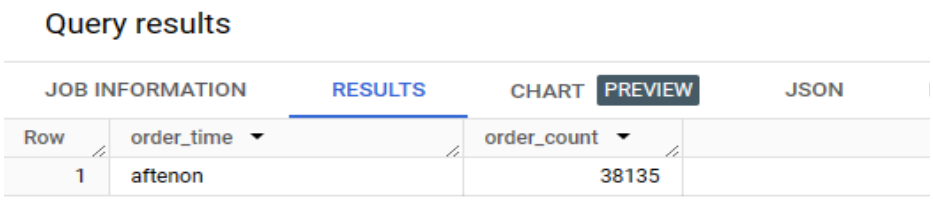
C. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

QUERY:

```
with `cte` as
(select order_time,
count(order_id) as `order_count`
from
(select
order_id,extract(year from order_purchase_timestamp) as `year`,extract(month from
order_purchase_timestamp) as `month`,
case when extract(time from order_purchase_timestamp) between ("00:00:00") and ("06:59:59") then
"Dawn"
when extract(time from order_purchase_timestamp) between ("07:00:00") and ("12:59:59") then
"Morning"
when extract(time from order_purchase_timestamp) between ("13:00:00") and ("18:59:59") then "aftenon"
when extract(time from order_purchase_timestamp) between ("19:00:00") and ("23:59:59") then "night"
end as `order_time`
from `target.orders`
)
group by order_time
```

```
order by order_time)

select order_time,order_count
from `cte`
where order_count=
(select max(order_count) from `cte`);
```



INSIGHTS :

1. Output suggested afternoon is the high time when Brazilian citizens shop.
2. Target can draw customer's attention by giving additional discounts to shops during the evening or other time of the day.
3. Also can increase the sale advertisement during the afternoon.
4. For better customer experience can increase the number of staff during rush hour.

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

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

QUERY:

```
select customer_state,month,
sum(case when year=2016 then order_count else 0 end ) as `y_16`,
sum(case when year=2017 then order_count else 0 end ) as `y_17`,
sum(case when year=2018 then order_count else 0 end ) as `y_18`
from
(select c.customer_state,
extract(year from o.order_purchase_timestamp) as `year`,
extract(month from o.order_purchase_timestamp) as `month`,
count(o.customer_id) as `order_count`
from `target.orders` o inner join `target.customers` c
on o.customer_id=c.customer_id
group by c.customer_state,year,month
order by year,month)
group by customer_state,month
order by month,customer_state;
```

Query results

SAVE RESULTS EXPLORE

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

Row	customer_state	month	y_16	y_17	y_18
1	AC	1	0	2	6
2	AL	1	0	2	37
3	AM	1	0	0	12
4	AP	1	0	0	11
5	BA	1	0	25	239
6	CE	1	0	9	90
7	DF	1	0	13	138
8	ES	1	0	12	147
9	GO	1	0	18	146
10	MA	1	0	9	57

Results per page: 50 1 – 50 of 322

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

```
select customer_state,count(distinct(customer_unique_id)) as `unique_customer_count`
from `target.customers`
group by customer_state
order by customer_state;
```

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON

Row	customer_state	unique_customer_co
1	AC	77
2	AL	401
3	AM	143
4	AP	67
5	BA	3277
6	CE	1313
7	DF	2075
8	ES	1964
9	GO	1952
10	MA	726

INSIGHTS :

1.By looking at the output we can identify states which have maximum and minimum number of unique customers.

2. Accordingly we can plan marketing strategy,advertising ,discount schemes and logistics for each state .

3. For instance we can improve customer satisfaction by increasing logistic staff ,fast delivery and less customer service response time.
4. Similarly can improve sales in those states where sales are low by giving discounts and additional free services.

IV. 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).

QUERY:

```
with `y_17` as (
select extract(year from order_purchase_timestamp) as `year`,extract(month from
order_purchase_timestamp) `month`,sum(payment_value)`order_value_17`
from `target.orders` o inner join `target.payments` p
on o.order_id=p.order_id
where extract(date from order_purchase_timestamp)  between "2017-01-01" and "2017-08-31"
group by year ,month
order by year,month
),
`y_18` as
 (
select extract(year from order_purchase_timestamp) as `year`,extract(month from
order_purchase_timestamp) `month`,sum(payment_value)`order_value_18`
from `target.orders` o inner join `target.payments` p
on o.order_id=p.order_id
where extract(date from order_purchase_timestamp)  between "2018-01-01" and "2018-08-31"
group by year ,month
order by year,month
)

select y_17.month,round(order_value_17)as `order_value_17` ,round(order_value_18) as
`order_value_18`,round((((order_value_18-order_value_17)/ order_value_17)*100)) as
`price_diff_percentage`
from y_17 full join y_18
on y_17.month=y_18.month
order by y_17.month;
```

Query results

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

Row	month	order_value_17	order_value_18	price_diff_percentag
1	1	138488.0	1115004.0	705.0
2	2	291908.0	992463.0	240.0
3	3	449864.0	1159652.0	158.0
4	4	417788.0	1160785.0	178.0
5	5	592919.0	1153982.0	95.0
6	6	511276.0	1023880.0	100.0
7	7	592383.0	1066541.0	80.0
8	8	674396.0	1022425.0	52.0

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

QUERY:

```
select c.customer_state,round(sum(price),2) as `total_order_price`,
round(avg(price),2) as `avg_order_price`
from `target.orders` o inner join `target.customers` c
on o.customer_id=c.customer_id
inner join `target.order_items` oi
on o.order_id=oi.order_id
group by c.customer_state
order by c.customer_state;
```

Query results

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

Row	customer_state	total_order_price	avg_order_price
1	AC	15982.95	173.73
2	AL	80314.81	180.89
3	AM	22356.84	135.5
4	AP	13474.3	164.32
5	BA	511349.99	134.6
6	CE	227254.71	153.76
7	DF	302603.94	125.77
8	ES	275037.31	121.91
9	GO	294591.95	126.27
10	MA	119648.22	145.2

Results per page:

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

QUERY:

```
select c.customer_state,round(sum(freight_value),2) as `total_order_price`,
round(avg(freight_value),2) as `avg_order_price`
from `target.orders` o inner join `target.customers` c
on o.customer_id=c.customer_id
inner join `target.order_items` oi
on o.order_id=oi.order_id
group by c.customer_state
order by c.customer_state;
```

Query results

SAVE R

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

Row	customer_state	total_order_price	avg_order_price
1	AC	3686.75	40.07
2	AL	15914.59	35.84
3	AM	5478.89	33.21
4	AP	2788.5	34.01
5	BA	100156.68	26.36
6	CE	48351.59	32.71
7	DF	50625.5	21.04
8	ES	49764.6	22.06
9	GO	53114.98	22.77
10	MA	31523.77	38.26

V. 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.

QUERY:

```
select order_id,date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as `time to
deliver`,
date_diff(order_estimated_delivery_date,order_delivered_customer_date,day) as
`diff_estimated_delivery`
from `target.orders`;
```

Query results

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

Row	order_id	time to deliver	diff_estimated_delive
1	1950d777989f6a877539f5379...	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	30	28
3	65d1e226dfaeb8cdc42f66542...	35	16
4	635c894d068ac37e6e03dc54e...	30	1
5	3b97562c3aee8bdedcb5c2e45...	32	0
6	68f47f50f04c4cb6774570cfde...	29	1
7	276e9ec344d3bf029ff83a161c...	43	-4
8	54e1a3c2b97fb0809da548a59...	40	-4
9	fd04fa4105ee8045f6a0139ca5...	37	-1
10	302bb8109d097a9fc6e9cefc5...	33	-5

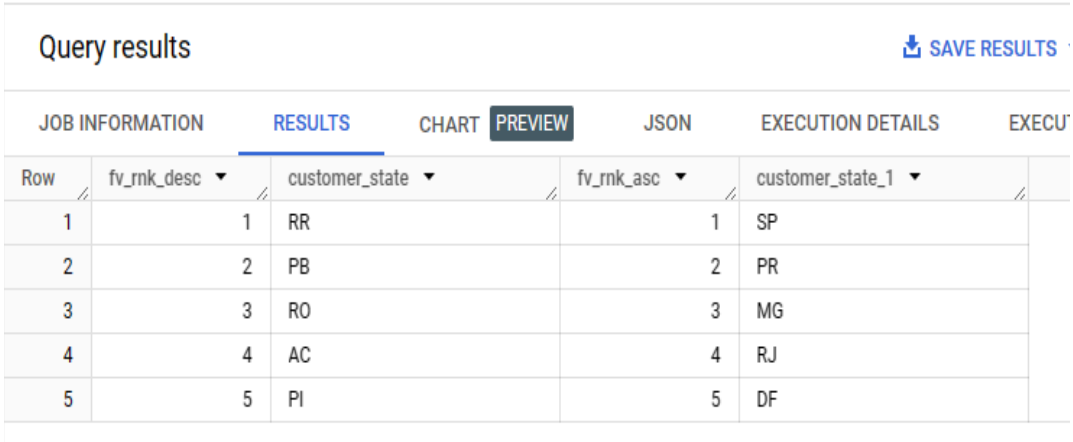
Results per page:

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

QUERY:

```
with `top_5` as (
select c.customer_state,
avg(oi.freight_value) as `avg_freight_value`,
dense_rank()over(order by avg(oi.freight_value)desc) as `fv_rnk_desc`
from `target.customers` c inner join `target.orders` o
on c.customer_id=o.customer_id
inner join `target.order_items` oi
on o.order_id=oi.order_id
group by c.customer_state
order by `fv_rnk_desc`
),
`lowest_5` as
(
select c.customer_state,
avg(oi.freight_value) as `avg_freight_value`,
dense_rank()over(order by avg(oi.freight_value) asc)`fv_rnk_asc`
from `target.customers` c inner join `target.orders` o
on c.customer_id=o.customer_id
inner join `target.order_items` oi
on o.order_id=oi.order_id
group by c.customer_state
order by `fv_rnk_asc`
)
```

```
select t.`fv_rnk_desc`,t.customer_state,l.`fv_rnk_asc`,l.customer_state
from top_5 t inner join lowest_5 l
on t.`fv_rnk_desc`=l.`fv_rnk_asc`
order by t.`fv_rnk_desc`,l.`fv_rnk_asc`
limit 5;
```



Query results

SAVE RESULTS

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

Row	fv_rnk_desc	customer_state	fv_rnk_asc	customer_state_1
1	1	RR	1	SP
2	2	PB	2	PR
3	3	RO	3	MG
4	4	AC	4	RJ
5	5	PI	5	DF

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

QUERY:

```
with `cte` as
(select
customer_state,round(avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)),2)
as `avg_delivery_time`,
dense_rank()over(order by
avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day))desc) as `rnk_desc`,
dense_rank()over(order by
avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day))asc) as `rnk_asc`
from `target.orders` o inner join `target.customers` c
on o.customer_id=c.customer_id
group by customer_state)

select c1.rnk_desc,c1.customer_state,c1.avg_delivery_time as `higest_avg_delivery_time`
,c2.customer_state,c2.avg_delivery_time as `lowest_avg_delivery_time`
from cte c1 inner join cte c2
on c1.rnk_desc=c2.rnk_asc
order by c1.rnk_desc,c2.rnk_asc
limit 5
```

Query results

SAVE RESULTS | EXPLORE DA

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

Row	rnk_desc	customer_state	higest_avg_delivery_	customer_state_1	lowest_avg_delivery_
1	1	RR	28.98	SP	8.3
2	2	AP	26.73	PR	11.53
3	3	AM	25.99	MG	11.54
4	4	AL	24.04	DF	12.51
5	5	PA	23.32	SC	14.48

INSIGHTS :

1. Fast delivery increases sales growth. Customers are more likely to order multiple times if delivery services are fast.
2. To ensure fast delivery we can increase warehouse, can analyze historical order data for better sales prediction and stock up accordingly.
3. Can also introduce employee incentives for fast delivery.

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

QUERY:

```
select customer_state,round(`avg_actual_delivery_time`,2) as
`avg_actual_delivery_time`,round(`avg_estimated_delivery_time`,2) as
`avg_estimated_delivery_time`,round((`avg_estimated_delivery_time`-`avg_actual_delivery_time`),2) as
`time_diff`
from (
select customer_state,
avg(date_diff(order_delivered_customer_date,order_purchase_timestamp, day))as
`avg_actual_delivery_time`,
avg(date_diff(order_estimated_delivery_date,order_purchase_timestamp,day)) as
`avg_estimated_delivery_time`
from `target.orders` o inner join `target.customers` c
on o.customer_id=c.customer_id
where order_status="delivered"
group by customer_state
)
order by time_diff desc
```

Query results

SAVE R

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

Row	customer_state	avg_actual_delivery_t	avg_estimated_deliv	time_diff
1	AC	20.64	40.73	20.09
2	RO	18.91	38.39	19.47
3	AP	26.73	45.87	19.13
4	AM	25.99	44.92	18.94
5	RR	28.98	45.63	16.66
6	MT	17.59	31.37	13.77
7	PA	23.32	36.79	13.48
8	RS	14.82	28.16	13.34
9	RN	18.82	31.87	13.05
10	PR	11.53	24.25	12.73

Results per page: 50

VI. Analysis based on the payments:

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

```
select extract(year from order_purchase_timestamp) as `year`, extract(month from
order_purchase_timestamp) as`month`,
payment_Type,count(distinct(o.order_id)) as`count_of_order`
from  `target.payments` p full join `target.orders` o
on p.order_id=o.order_id
group by year,month,payment_type
order by year,month;
```

Query results

SAVE RESUI

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

Row	year	month	payment_Type	count_of_order
1	2016	9	credit_card	3
2	2016	9	*null*	1
3	2016	10	UPI	63
4	2016	10	credit_card	253
5	2016	10	voucher	11
6	2016	10	debit_card	2
7	2016	12	credit_card	1
8	2017	1	UPI	197
9	2017	1	credit_card	582
10	2017	1	debit_card	9

Results per page: 50

INSIGHTS :

1. Credit card and UPI are the top two used methods for transactions by customers.
2. By ensuring fast and seamless transactions we can improve the checkout experience of customers .

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

QUERY:

```
select payment_sequential,count(order_id) as `order_count`
from `target.payments`
where payment_sequential>=1
group by payment_sequential
order by order_count desc
```

Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW

Row	payment_sequential	order_count
1	1	99360
2	2	3039
3	3	581
4	4	278
5	5	170
6	6	118
7	7	82
8	8	54
9	9	43
10	10	34

INSIGHTS :

1. Output suggests customers who made one time payment are higher .
2. We can provide incentives for one time payments.