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

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

Query:

```
SELECT
column_name,data_type
FROM business_case.INFORMATION_SCHEMA.COLUMNS
where table_name='target_customers'
```

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/Inference:

From the above, we can see that **customer_id, customer_unique_id, customer_city, customer_state** fall under **STRING** datatype and **customer_zip_cose_prefix** fall under **INTEGER** datatype.

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

Query:

```
SELECT DISTINCT
FIRST_VALUE(order_purchase_timestamp) over (order by order_purchase_timestamp asc)
as first_order,
FIRST_VALUE(order_purchase_timestamp) over (order by order_purchase_timestamp desc)
as last_order
FROM `target-8001.business_case.target_orders`
```

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

Insights/Inference:

Therefore from the analysis, the first order is placed at 2016-09-04 and the last order at 2018-10-17

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

```
SELECT Count(Distinct geolocation_city) as cities, count(distinct
geolocation_state)as states
FROM `target-8001.business_case.target_geolocation`
```

Row	cities	states
1	8011	27

Insights/Inference:

Therefore there are 8011 cities and 27 different states in the given dataset.

2.In-depth Exploration:

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

```
With a as
(Select year,count(order_id)as current_year_orders
from
(SELECT *, Extract(year from order_purchase_timestamp)as year FROM `target-
8001.business_case.target_orders` )
group by year
order by year asc)

, b as
(Select year, current_year_orders,ifnull(LAG(a.current_year_orders)over(order by
year asc ),a.current_year_orders) as previous_year from a
order by year)

Select *, ((b.current_year_orders-previous_year)/previous_year)*100 as
percentage_increase
from b
```

Row	year	current_year_orders	previous_year	percentage_increase
1	2016	329	329	0.0
2	2017	45101	329	13608.51063829...
3	2018	54011	45101	19.75565951974...

Insights/Inference:

By the analysis, there is growth in the orders over past every years.

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

Query:

```
With base as
(Select year, month,count(order_id)as order_permonth
from
(SELECT *, Extract(year from order_purchase_timestamp)as year,Extract(month from
order_purchase_timestamp)as month
FROM `target-8001.business_case.target_orders`)
group by year, month
order by year asc,month asc)

Select year,month,order_permonth from base
where base.order_permonth= (Select MAX(order_permonth)from base)
```

Row	year	peakmonth	order_permonth
1	2017	11	7544

Insights/Inference:

As per the analysis, 11th month of 2017 year the orders were peak.

2.C.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:

```
With base as(SELECT Extract(Hour from order_purchase_timestamp)as
hour,count(order_id)as orders
FROM `target-8001.business_case.target_orders`
group by hour)

,base1 as
(Select *,CASE When hour between 0 and 6 then 'Dawn'
         When hour between 7 and 12 then 'Morning'
         When hour between 13 and 18 then 'Afternon'
         When hour between 19 and 23 then 'Night'
         END AS Day_Timing
      from base)

,base2 as
(Select Day_Timing, Sum(orders)as Total_orders from base1
```

```
Group by Day_Timing)

Select * from base2
where Total_orders =
(Select MAX(base2.Total_orders) from base2)
```

Row	Day_Timing	Total_orders
1	Afternon	38135

Insights/Inference:

In Afternoon, the orders placed by the customers are more.

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

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

Query:

```
      With base as(SELECT *,Extract(month from order_purchase_timestamp)as month
FROM `target-8001.business_case.target_orders` )

Select b.customer_state,a.month,count(a.order_id) as orders_placed from base as a
left join `target-8001.business_case.target_customers`as b
on a.customer_id=b.customer_id
Group by a.month,b.customer_state
order by b.customer_state asc,a.month asc
```

Row	customer_state	month	orders_placed
1	AC	1	8
2	AC	2	6
3	AC	3	4
4	AC	4	9
5	AC	5	10
6	AC	6	7
7	AC	7	9
8	AC	8	7
9	AC	9	5
10	AC	10	6

3.B. How are the customers distributed across all the states?
Query:

```
SELECT customer_state,Count(customer_id)as customer_count FROM `target-
8001.business_case.target_customers`
GROUP BY customer_state
order by customer_state asc
```

Row	customer_state	customer_count
1	AC	81
2	AL	413
3	AM	148
4	AP	68
5	BA	3380
6	CE	1336
7	DF	2140
8	ES	2033
9	GO	2020
10	MA	747

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

4.A. 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:

```
With base as(SELECT *, Extract(year from order_purchase_timestamp)as
year,Extract(month from order_purchase_timestamp)as month
FROM `target-8001.business_case.target_orders` )
,
base1 as
(Select a.year,SUM(b.payment_value)as cost from base as a
left join `target-8001.business_case.target_payments` as b
on a.order_id=b.order_id
where a.year in(2017,2018) and a.month Between 1 and 8
Group by a.year
order by a.year asc)

,base2 as
(Select *,ifnull(LAG(cost) over (order by year asc),cost) as previous_year_cost,
from base1
order by year asc)

Select *,((cost-previous_year_cost)/previous_year_cost)*100 as Percentage
from base2
```

Row	year	cost	previous_year_cost	Percentage_Increase
1	2017	3669022.119999...	3669022.119999...	0.0
2	2018	8694733.839999...	3669022.119999...	136.9768716466...

Insights/Inference:

In 2018 there were approximately 137% increase in the cost of orders when compared to previous year.

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

Query:

```
Select c.customer_state,SUM(a.payment_value) as
total_order_price,AVG(a.payment_value)as average_order_price
from `target-8001.business_case.target_payments` as a
inner join `target-8001.business_case.target_orders`as b
on a.order_id = b.order_id
inner join `target-8001.business_case.target_customers` as c
on b.customer_id = c.customer_id
Group by c.customer_state
order by c.customer_state
```

Row	customer_state	total_order_price	average_order_price
1	AC	19680.61999999...	234.2930952380...
2	AL	96962.05999999...	227.0774238875...
3	AM	27966.93	181.6034415584...
4	AP	16262.79999999...	232.3257142857...
5	BA	616645.8200000...	170.8160166204...
6	CE	279464.0299999...	199.9027396280...
7	DF	355141.0800000...	161.1347912885...
8	ES	325967.55	154.7069530137...
9	GO	350092.3100000...	165.7634043560...
10	MA	152523.0200000...	198.8566101694...
11	MG	1872257.260000...	154.7064336473...

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

Query:

```
Select c.customer_state,SUM(a.freight_value) as
total_freight_value,AVG(a.freight_value)as average_freight_value
```

```
from `target-8001.business_case.target_order_items` as a
inner join `target-8001.business_case.target_orders`as b
on a.order_id = b.order_id
inner join `target-8001.business_case.target_customers` as c
on b.customer_id = c.customer_id
Group by c.customer_state
order by c.customer_state
```

Row	customer_state	total_freight_value	average_freight_valu
1	AC	3686.750000000...	40.07336956521...
2	AL	15914.58999999...	35.84367117117...
3	AM	5478.890000000...	33.20539393939...
4	AP	2788.500000000...	34.00609756097...
5	BA	100156.6799999...	26.36395893656...
6	CE	48351.58999999...	32.71420162381...
7	DF	50625.49999999...	21.04135494596...
8	ES	49764.59999999...	22.05877659574...
9	GO	53114.97999999...	22.76681525932...
10	MA	31523.77000000...	38.25700242718...

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

5.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.
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_estimated_delivery_date - order_delivered_customer_date

Query:

```
SELECT order_id,
TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp , DAY) as
timetaken_to_deliver,
TIMESTAMP_DIFF(order_estimated_delivery_date ,order_delivered_customer_date ,DAY)as
diff_estimated_delivery
FROM `target-8001.business_case.target_orders`
```

Row	order_id	timetaken_to_deliver	diff_estimated_delivery
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
11	66057d37308e787052a32828...	38	-6

Insights/Inference:

The timetaken_to_deliver column, shows the time taken for the order to reach the customer.

In the diff_estimated_delivery column,

- the positive value indicates the no of days before the estimated delivery, the order reached.
- the negative value indicates the no of days after the estimated delivery, the order reached.

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

Query:

```
With base as
(Select c.customer_state,AVG(a.freight_value)as average_freight_value
from `target-8001.business_case.target_order_items` as a
inner join `target-8001.business_case.target_orders`as b
on a.order_id = b.order_id
inner join `target-8001.business_case.target_customers` as c
on b.customer_id = c.customer_id
Group by c.customer_state
order by average_freight_value
Limit 5),

base1 as
(Select c.customer_state,AVG(a.freight_value)as average_freight_value
from `target-8001.business_case.target_order_items` as a
inner join `target-8001.business_case.target_orders`as b
on a.order_id = b.order_id
inner join `target-8001.business_case.target_customers` as c
on b.customer_id = c.customer_id
Group by c.customer_state
order by average_freight_value desc
Limit 5)
```

```
Select * from base
union all
Select * from base1
```

Row	customer_state	average_freight_valu
1	SP	15.14727539041...
2	PR	20.53165156794...
3	MG	20.63016680630...
4	RJ	20.96092393168...
5	DF	21.04135494596...
6	RR	42.98442307692...
7	PB	42.72380398671...
8	RO	41.06971223021...
9	AC	40.07336956521...
10	PI	39.14797047970...

Insights/Inference:

From the above output, the first 5(1-5) shows the top 5 states with lowest average freight value and the following 5(6-10) shows the top 5 states with highest average freight value.

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

Query:

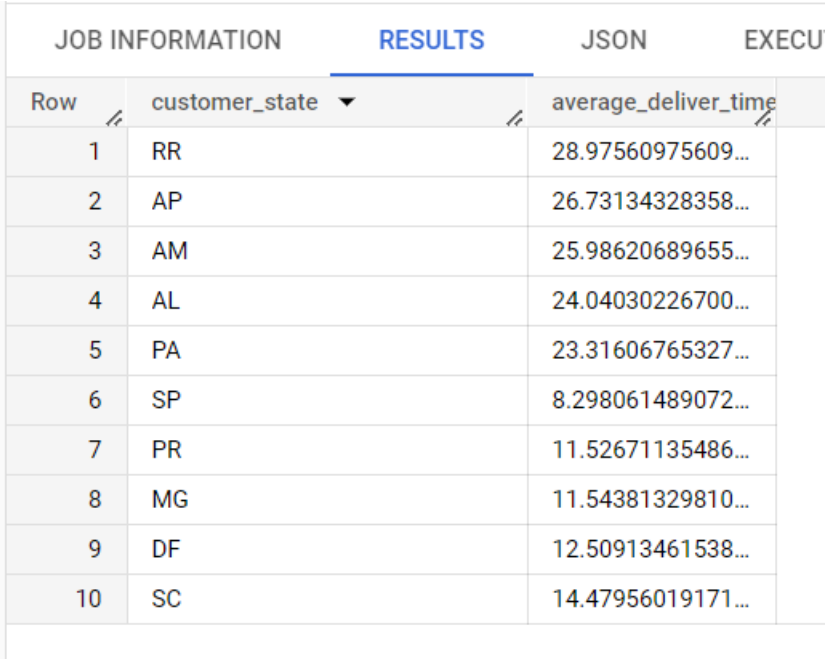
```
With base as
(SELECT order_id,
TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp , DAY) as
timetaken_to_deliver,
FROM `target-8001.business_case.target_orders`),
base1 as
(Select c.customer_state,Avg(a.timetaken_to_deliver)as average_deliver_time
From base as a
inner join `target-8001.business_case.target_orders` as b
on a.order_id=b.order_id
inner join `target-8001.business_case.target_customers` as c
on b.customer_id=c.customer_id
group by c.customer_state
order by c.customer_state),

base2 as
(Select * from base1
order by base1.average_deliver_time asc limit 5),

base3 as
(Select * from base1
order by base1.average_deliver_time desc limit 5)

Select * from base2
union all
```

```
Select * from base3
```



JOB INFORMATION | RESULTS | JSON | EXECU

Row	customer_state	average_deliver_time
1	RR	28.97560975609...
2	AP	26.73134328358...
3	AM	25.98620689655...
4	AL	24.04030226700...
5	PA	23.31606765327...
6	SP	8.298061489072...
7	PR	11.52671135486...
8	MG	11.54381329810...
9	DF	12.50913461538...
10	SC	14.47956019171...

Insights/Inference:

From the above output, the first 5(1-5) shows the top 5 states with highest average delivery time and the following 5(6-10) shows the top 5 states with lowest average delivery time.

5.D. 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:

```
With base as
(Select
a.order_id,a.order_delivered_customer_date,a.order_estimated_delivery_date,b.custom
er_state
from `target-8001.business_case.target_orders` as a
inner join
`target-8001.business_case.target_customers` as b
on a.customer_id=b.customer_id
where a.order_delivered_customer_date is not null),

base1 as
(Select
customer_state,TIMESTAMP_DIFF(order_delivered_customer_date,order_estimated_deliver
y_date, DAY) as average_fast_delivery
from base)

Select customer_state, AVG(base1.average_fast_delivery)as fast_delivery from base1
group by customer_state
order by fast_delivery desc
limit 5
```

Row	customer_state	fast_delivery
1	AL	-7.94710327455...
2	MA	-8.76847977684...
3	SE	-9.17313432835...
4	ES	-9.61854636591...
5	BA	-9.93488943488...

Insights/Inference:

The above is the top 5 fastest states that deliver the order earlier when compared to other

6.Analysis based on the payments:

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

```
with base as
(SELECT order_id, Extract(year from order_purchase_timestamp)as year,Extract(month
from order_purchase_timestamp)as month FROM `target-
8001.business_case.target_orders` )
Select a.year,a.month,b.payment_type,count(b.order_id)orders from base as a
inner join `target-8001.business_case.target_payments`as b
on a.order_id=b.order_id
group by a.year,a.month,b.payment_type
order by a.year,a.month asc
```

Row	year	month	payment_type	orders
1	2016	9	credit_card	3
2	2016	10	credit_card	254
3	2016	10	UPI	63
4	2016	10	voucher	23
5	2016	10	debit_card	2
6	2016	12	credit_card	1
7	2017	1	credit_card	583
8	2017	1	UPI	197
9	2017	1	voucher	61
10	2017	1	debit_card	9
11	2017	2	credit_card	1356

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

Query:

```
Select payment_installments,count(order_id)orders
FROM `target-8001.business_case.target_payments`
where payment_sequential>=1
group by payment_installments
```

Row	payment_installment	orders
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626
9	8	4268
10	9	644
11	10	5328

Recommendations:

Row	year	month	orders
13	2017	10	4631
14	2017	11	7544
15	2017	12	5673
16	2018	1	7269
17	2018	2	6728
18	2018	3	7211
19	2018	4	6939
20	2018	5	6873
21	2018	6	6167
22	2018	7	6292
23	2018	8	6512
24	2018	9	16
25	2018	10	4

From the analysis, it is evident that the last 2 months, there was huge drop in orders. So it is recommended to take action at most possible to regain the orders.