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

a. Data type of columns in a table

Customers:

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

Geolocation:

Field name	Type	Mode
geolocation_zip_code_prefix	INTEGER	NULLABLE
geolocation_lat	FLOAT	NULLABLE
geolocation_lng	FLOAT	NULLABLE
geolocation_city	STRING	NULLABLE
geolocation_state	STRING	NULLABLE

Order_items:

Field name	Type	Mode
order_id	STRING	NULLABLE
order_item_id	INTEGER	NULLABLE
product_id	STRING	NULLABLE
seller_id	STRING	NULLABLE
shipping_limit_date	TIMESTAMP	NULLABLE
price	FLOAT	NULLABLE
freight_value	FLOAT	NULLABLE

Order_reviews:

Field name	Type	Mode
review_id	STRING	NULLABLE
order_id	STRING	NULLABLE
review_score	INTEGER	NULLABLE
review_comment_title	STRING	NULLABLE
review_creation_date	TIMESTAMP	NULLABLE
review_answer_timestamp	TIMESTAMP	NULLABLE

Orders:

Field name	Type	Mode
order_id	STRING	NULLABLE
customer_id	STRING	NULLABLE
order_status	STRING	NULLABLE
order_purchase_timestamp	TIMESTAMP	NULLABLE
order_approved_at	TIMESTAMP	NULLABLE
order_delivered_carrier_date	TIMESTAMP	NULLABLE
order_delivered_customer_date	TIMESTAMP	NULLABLE
order_estimated_delivery_date	TIMESTAMP	NULLABLE

Payments:

Field name	Type	Mode
order_id	STRING	NULLABLE
payment_sequential	INTEGER	NULLABLE
payment_type	STRING	NULLABLE
payment_installments	INTEGER	NULLABLE
payment_value	FLOAT	NULLABLE

Products:

Field name	Type	Mode
product_id	STRING	NULLABLE
product_category	STRING	NULLABLE
product_name_length	INTEGER	NULLABLE
product_description_length	INTEGER	NULLABLE
product_photos_qty	INTEGER	NULLABLE
product_weight_g	INTEGER	NULLABLE
product_length_cm	INTEGER	NULLABLE
product_height_cm	INTEGER	NULLABLE
product_width_cm	INTEGER	NULLABLE

Seller:

Field name	Type	Mode
seller_id	STRING	NULLABLE
seller_zip_code_prefix	INTEGER	NULLABLE
seller_city	STRING	NULLABLE
seller_state	STRING	NULLABLE

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

```
SELECT
MIN(order_purchase_timestamp) AS start_date,
MAX(order_purchase_timestamp) AS end_date
FROM `target-410421.Target_dataset.orders`
```

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

3. Count the Cities and states of customers who ordered during the given period.

```
SELECT DISTINCT
customer_city,customer_state
FROM`target-410421.Target_dataset.customers` AS c
JOIN`target-410421.Target_dataset.orders` AS o
```

```
ON c.customer_id=o.customer_id
```

Row	customer_city	customer_state
1	rio de janeiro	RJ
2	sao leopoldo	RS
3	general salgado	SP
4	brasilia	DF
5	paranavai	PR
6	cuiaba	MT
7	sao luis	MA
8	maceio	AL
9	hortolandia	SP
10	varzea grande	MT

Q2) In-depth exploration:

2.1)Isthereagrowingtrendone-commerceinBrazil?

How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?

```
SELECT
EXTRACT(YEAR FROM order_purchase_timestamp) as Year_of_purchase,
EXTRACT(MONTH FROM order_purchase_timestamp) as
Month_of_purchase,
COUNT(order_id) as No_of_orders,
FROM `target-410421.Target_dataset.orders`
Group by 1,2
Order by 1,2
```

Row	Year_of_purchase	Month_of_purchase	No_of_orders
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026

2.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

```
SELECT
CASE
WHEN EXTRACT(hour FROM timestamp(order_purchase_timestamp)) BETWEEN 0 AND 6 THEN
'Dawn'
WHEN EXTRACT(hour FROM timestamp(order_purchase_timestamp)) BETWEEN 7 AND 12 THEN
'Morning'
WHEN EXTRACT(hour FROM timestamp(order_purchase_timestamp)) BETWEEN 13 AND 18 THEN
'Afternoon'
WHEN EXTRACT(hour FROM timestamp(order_purchase_timestamp)) BETWEEN 19 AND 23 THEN
'Night'
END AS Time_of_day,
COUNT(DISTINCT order_id) AS No_of_orders
 FROM `target-410421.Target_dataset.orders`
 GROUP BY 1
 ORDER BY 2 DESC;
```

Row	Time_of_day	No_of_orders
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

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

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

```
SELECT EXTRACT (YEAR from (o.order_purchase_timestamp)) as Year,
EXTRACT (MONTH from ( o.order_purchase_timestamp)) as Month_of_purchase,
c.customer_state,
COUNT(o.order_id) as No_of_orders
FROM `target-410421.Target_dataset.orders` o
JOIN `target-410421.Target_dataset.customers` c
ON o.customer_id = c.customer_id
Group by customer_state, Month_of_purchase, Year
Order by Year, Month_of_purchase, No_of_orders
LIMIT 10;
```

Row	Year	Month_of_purchase	customer_state	No_of_orders
1	2016	9	RS	1
2	2016	9	RR	1
3	2016	9	SP	2
4	2016	10	PB	1
5	2016	10	RR	1
6	2016	10	PI	1
7	2016	10	AL	2
8	2016	10	MT	3
9	2016	10	SE	3
10	2016	10	ES	4

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

```
SELECT
COUNT (customer_unique_id) AS No_of_customers,
customer_state
FROM `Target_dataset.customers`
GROUP BY 2
LIMIT 10;
```

Row	No_of_customers	customer_state
1	485	RN
2	1336	CE
3	5466	RS
4	3637	SC
5	41746	SP
6	11635	MG
7	3380	BA
8	12852	RJ
9	2020	GO
10	747	MA

4. 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.

```
with cte as (
SELECT DISTINCT
EXTRACT (year from (o.order_purchase_timestamp)) as Year,
EXTRACT (month from (o.order_purchase_timestamp)) as Month,
round(SUM(p.payment_value),2) AS monthly_sales,
FROM `Target_dataset.payments` p
join `Target_dataset.orders` o on p.order_id = o.order_id
WHERE EXTRACT (year from (o.order_purchase_timestamp)) between
2017 and 2018
and EXTRACT (month from (o.order_purchase_timestamp)) between 1
and 8
group by Year, Month
order by Year, Month)
SELECT *,
LEAD (monthly_sales,8) OVER (ORDER BY cte.year, cte.Month asc) as
next_year_sales,
round((LEAD (monthly_sales,8) OVER (ORDER BY cte.year, cte.Month
asc) - monthly_sales)/monthly_sales*100, 2) as pct_inc
from cte
order by cte.year, cte.Month;
```

Row	Year	Month	monthly_sales	next_year_sales	pct_inc
1	2017	1	138488.04	1115004.18	705.13
2	2017	2	291908.01	992463.34	239.99
3	2017	3	449863.6	1159652.12	157.78
4	2017	4	417788.03	1160785.48	177.84
5	2017	5	592918.82	1153982.15	94.63
6	2017	6	511276.38	1023880.5	100.26
7	2017	7	592382.92	1066540.75	80.04
8	2017	8	674396.32	1022425.32	51.61

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

```
SELECT
customer_state,
ROUND(SUM(price),2) as sum_of_price,
ROUND(AVG(price),2) as avg_price,
ROUND(SUM(freight_value),2) as sum_of_freight_value,
ROUND(AVG(freight_value),2) as avg_freight_value
FROM `Target_dataset.order_items`oi
JOIN`Target_dataset.orders` o
ON oi.order_id = o.order_id
JOIN `Target_dataset.customers` c
ON o.customer_id = c.customer_id
group by customer_state
LIMIT 10;
```

Row	customer_state	sum_of_price	avg_price	sum_of_freight_value	avg_freight_value
1	SP	5202955.05	109.65	718723.07	15.15
2	RJ	1824092.67	125.12	305589.31	20.96
3	PR	683083.76	119.0	117851.68	20.53
4	SC	520553.34	124.65	89660.26	21.47
5	DF	302603.94	125.77	50625.5	21.04
6	MG	1585308.03	120.75	270853.46	20.63
7	PA	178947.81	165.69	38699.3	35.83
8	BA	511349.99	134.6	100156.68	26.36
9	GO	294591.95	126.27	53114.98	22.77
10	RS	750304.02	120.34	135522.74	21.74

5. 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.

```
SELECT order_id,
customer_id,
```

```
DATE_DIFF(order_estimated_delivery_date,order_purchase_timestamp, Day) AS  Estimated,
DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, Day) AS  Purchasing,
DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, Day) AS  Delivery
FROM`Target_dataset.orders`
```

2. Also, calculate the difference (in days) between the estimated & actual delivery date of an order.

```
SELECT
date_diff(order_delivered_customer_date,order_purchase_timestamp,day) AStime_to_delivery,
date_diff(order_delivered_customer_date,order_estimated_delivery_date,day) ASdiff_estimated_time
FROM`Target_dataset.orders`
```

Row	time_to_delivery	diff_estimated_t
1	30	12
2	30	-28
3	35	-16
4	30	-1
5	32	0
6	29	-1
7	43	4
8	40	4
9	37	1
10	33	5

3. Group data by state, take mean of freight_value, time_to_delivery, Diff_estimated_delivery

```
SELECT
c.customer_state,
ROUND(AVG(oi.freight_value),2) AS  avg_freight_value,
ROUND(AVG(Timestamp_diff(o.order_delivered_customer_date,o.order_p
urchase_timestamp, day)),2) AS  avg_time_to_delivery,
round(avg(Timestamp_diff(o.order_estimated_delivery_date,
o.order_delivered_customer_date, day)),2) AS
avg_diff_estimated_delivery
FROM`Target_dataset.orders` o
JOIN `Target_dataset.customers` c
ON  o.customer_id = c.customer_id
```

```
JOIN`Target_dataset.order_items` oi
ON o.order_id=oi.order_id
WHERE order_purchase_timestamp is not null
AND order_delivered_customer_date is not null
AND order_estimated_delivery_date is not null
GROUP BY customer_state
LIMIT10;
```

Row	customer_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_delivery
1	RJ	20.91	14.69	11.14
2	MG	20.63	11.52	12.4
3	SC	21.51	14.52	10.67
4	SP	15.11	8.26	10.27
5	GO	22.56	14.95	11.37
6	RS	21.61	14.71	13.2
7	BA	26.49	18.77	10.12
8	MT	28.0	17.51	13.64
9	SE	36.57	20.98	9.17
10	PE	32.69	17.79	12.55

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

HIGHEST:

```
SELECT c.customer_state,
ROUND(AVG(oi.freight_value),2) AS Avg_freight_value
FROM `Target_dataset.order_items` oi
JOIN `Target_dataset.orders` o ON oi.order_id = o.order_id
JOIN `Target_dataset.customers` c ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY AVG(freight_value) DESC
LIMIT 5;
```

Row	customer_state	Avg_freight_value
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15

LOWEST:

```
SELECT c.customer_state,
ROUND(AVG(oi.freight_value),2) AS Avg_freight_value
FROM `Target_dataset.order_items` oi
JOIN `Target_dataset.orders` o ON oi.order_id = o.order_id
JOIN `Target_dataset.customers` c ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY AVG(freight_value) ASC
LIMIT 5;
```

Row	customer_state	Avg_freight_value
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04

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

   HIGHEST:

```
SELECT
c.customer_state,
ROUND(AVG(Timestamp_diff(o.order_purchase_timestamp,o.order_delivered_custom
er_date, day)),2) AS avg_time_to_delivery
FROM `Target_dataset.orders` o
JOIN `Target_dataset.customers` c
ON o.customer_id = c.customer_id
GROUP  BY customer_state
ORDER BY avg_time_to_delivery DESC
LIMIT 5;
```

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

LOWEST:

```
SELECT
c.customer_state,
ROUND(AVG(Timestamp_diff(o.order_purchase_timestamp,o.order_delivered_custom
er_date, day)),2) AS avg_time_to_delivery
FROM `Target_dataset.orders` o
JOIN `Target_dataset.customers` c
ON o.customer_id = c.customer_id
GROUP  BY customer_state
ORDER BY avg_time_to_delivery ASC
```

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

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

```
SELECT c.customer_state,
ROUND(avg(Timestamp_diff(o.order_estimated_delivery_date,
o.order_delivered_customer_date, day)),2) as
avg_diff_estimated_delivery
FROM `Target_dataset.orders` o
JOIN`Target_dataset.customers` c
ON o.customer_id = c.customer_id
GROUP BY customer_state
ORDER BY avg_diff_estimated_delivery asc
LIMIT 5;
```

Row	customer_state	avg_diff_estimated_delivery
1	AL	7.95
2	MA	8.77
3	SE	9.17
4	ES	9.62
5	BA	9.93

6. Analysis based on the payments:

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

```
SELECT
p.payment_type,
EXTRACT (year FROM (o.order_purchase_timestamp)) as Year,
EXTRACT (month FROM (o.order_purchase_timestamp)) as Month_of_purchase,
COUNT(o.order_id) as No_of_orders
FROM `Target_dataset.payments` p
JOIN `Target_dataset.orders` o
ON p.order_id = o.order_id
group by Month_of_purchase, payment_type, Year
order by Year, Month_of_purchase
LIMIT 10;
```

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

2. Find the no. of orders placed based on the payment installments that have been paid.

```
SELECT
payment_installments,
COUNT(order_id) as No_of_orders
FROM `Target_dataset.payments`
GROUP BY payment_installments;
```

Row	payment_installments	No_of_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