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.

customers

SCHEMA | DETAILS | PREVIEW | LINEAGE | DATA PROFILE | DATA QUALITY

Field name	Type	Mode	Key	Collation	Default value	Policy ta
customer_id	STRING	NULLABLE	-	-	-	-
customer_unique_id	STRING	NULLABLE	-	-	-	-
customer_zip_code_prefix	INTEGER	NULLABLE	-	-	-	-
customer_city	STRING	NULLABLE	-	-	-	-
customer_state	STRING	NULLABLE	-	-	-	-

Observations: All the coloumn except customer_zip_code_prifix are string or varchar type and customer_zip_code_prefix is an integer

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

Assumptions: All orders which are purchased or placed are included here and didn't consider whether it is delivered or not. Since we just need to find the time range.

Code:

```
with cte as
(select min(order_purchase_timestamp) as  first_order_timestamp,
       max(order_purchase_timestamp) as  last_order_timestamp
    from `Target.orders` )

select *,
      -- DATE_DIFF(date(cte.last_order_timestamp),date(cte.first_order_timestamp),year) as years,
      -- mod(DATE_DIFF(date(cte.last_order_timestamp),date(cte.first_order_timestamp),month),12) as months
      concat(DATE_DIFF(date(cte.last_order_timestamp),date(cte.first_order_timestamp),year),' years
and',mod(DATE_DIFF(date(cte.last_order_timestamp),date(cte.first_order_timestamp),month),12),'months') as
Duration_of_orders_dataset
      from cte
```

Query results

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

Row	first_order_timestamp	last_order_timestamp	Duration_of_orders_dataset
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC	2 years and 1months

Observations: The first order in the dataset is placed on 04-09-2016 and the last order in the dataset is placed on 17-10-2018 and the time range between them is 2 years and 1 month.

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

```
Code:
select count(distinct customer_city) as City_count,
       count(distinct customer_state) as State_count
  from `Target.customers`
```

Query results

SAVE RESULTS

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

Row	City_count	State_count
1	4119	27

Observations: There are 27 States and 4119 different cities from all states combined from which customers ordered the product during the given time period

II. In-depth Exploration:

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

Row	order_status
1	created
2	shipped
3	approved
4	canceled
5	invoiced
6	delivered
7	processing
8	unavailable

Assumptions: We will find the trend patterns only for the "orders_status" which are marked as "**delivered**"

```
Code:
with cte as
(select order_id,
       format_datetime('%Y/%m',order_purchase_timestamp) as purchased_year_month,
from `Target.orders`
where order_status = 'delivered')

select cte.purchased_year_month,
       count(order_id) as count_orders
       from cte
    group by cte.purchased_year_month
  order by cte.purchased_year_month
```

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTIO

Row	purchased_year_month	count_orders
1	2016/09	1
2	2016/10	265
3	2016/12	1
4	2017/01	750
5	2017/02	1653
6	2017/03	2546
7	2017/04	2303
8	2017/05	3546
9	2017/06	3135
10	2017/07	3872
11	2017/08	4193
12	2017/09	4150

Observations: Yes, we can see the there is clearly a growing trend in number of orders over the year.

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

Assumptions: We will find the monthly seasonality only for the "orders_status" which are marked as "**delivered".**

Code:

```
with cte as
(select order_id,
       format_datetime('%Y/%m',order_purchase_timestamp) as purchased_year_month,
from `Target.orders`
where order_status = 'delivered')

select cte.purchased_year_month,
       count(order_id) as count_orders
       from cte
    group by cte.purchased_year_month
  order by cte.purchased_year_month
```

Observations:

1. With the help of the **above section(A) Query and output** we can see there is a spike in no of orders delivered in **Nov2017** compared to any other previous month

12	2017/09	4150
13	2017/10	4478
14	2017/11	7289
15	2017/12	5513

This sudden spike in number of orders may be due to Nov-15th Brazil's Republic Day and Nov-29th Black Friday. There is a high possibility for giving discount sales during those special days and due to which there is an increase in order count.

2. There is also increase in number of orders delivered on **Jan2018** compared to previous month. This may be due to New Year festival and other religious festivals during the month of Jan.

14	2017/11	7289
15	2017/12	5513
16	2018/01	7069
17	2018/02	6555

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

Assumptions: We will find pattern only for the "orders_status" which are marked as "**delivered"**

```
Code:
with cte as
(select order_id,
        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 day_split
from `Target.orders`
where order_status = 'delivered')

select day_split,
       count(order_id) as order_count
      from cte
    group by day_split
order by 2 desc
```

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON

Row	day_split	order_count
1	Afternoon	36965
2	Night	27522
3	Morning	26919
4	Dawn	5072

Observations: From the above output we can see Brazilian customers mostly place the orders during the afternoon period that is between 13:00 to 18:00 hrs of the day.

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

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

Assumptions: We will find the month-on-month no of orders only for the "orders_status" which are marked as "**delivered**

Code:

```
with cte as
(select order_id,
       format_datetime('%Y/%m',order_purchase_timestamp) as purchased_year_month,
       customer_id
from `Target.orders`
where order_status = 'delivered')

select c.customer_state,
       o.purchased_year_month,
       count(order_id) as orders_count
     from `Target.customers` as c
     inner join
     cte as o
     on c.customer_id = o.customer_id
  group by 1,2
  order by 1,2
```

Query results

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

Row	customer_state	purchased_year_month	orders_count
1	AC	2017/01	2
2	AC	2017/02	3
3	AC	2017/03	2
4	AC	2017/04	5
5	AC	2017/05	8
6	AC	2017/06	4
7	AC	2017/07	5
8	AC	2017/08	4
9	AC	2017/09	5
10	AC	2017/10	5
11	AC	2017/11	5
12	AC	2017/12	5
13	AC	2018/01	6
14	AC	2018/02	3
15	AC	2018/03	2
16	AC	2018/04	4
17	AC	2018/05	2
18	AC	2018/06	3
19	AC	2018/07	4
20	AC	2018/08	3
21	AL	2016/10	1
22	AL	2017/01	2

Analysis: From the above output we can get the state which has the highest number of orders in certain months and prepare logistics and warehouse management accordingly for certain peak months.

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

```
Code:
 select customer_state,

  count(customer_id) as customer_count
 from `Target.customers`
 group by 1
 order by 2 desc
```

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW

Row	customer_state	customer_count
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

Observations: State Code 'SP' has the largest number of customers among all the states of Brazil.

IV. Impact on Economy: Analyse 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).*

Assumptions: We will find the percentage of increase in cost of orders only for the "orders_status" which are marked as "**delivered.**

Code:

```
 -- We will create a cte where only the orders placed in the month between Jan and Aug

 with order_sort as

(select order_id,
       format_datetime('%Y',order_purchase_timestamp) as purchased_year
from `Target.orders`
where order_status = 'delivered' and format_datetime('%m',order_purchase_timestamp) between '01'
and '08'
),
-- We will create a cte to sort by year and total sales and then pivot the table
total_sales_amount_per_year as
(select purchased_year,
       round(sum(payment_value),2) as Total_value
from order_sort as c
inner join
Target.payments as p
on c.order_id = p.order_id
group by 1
order by 1),
pivot_table as
(
  select *
  from( select purchased_year,
               Total_value
            from total_sales_amount_per_year) as tbl1
  pivot
  (
    sum(total_value)
    for purchased_year in ('2017','2018')
  )as p
)

select p.2017 as Total_sales_2017,
       p.2018 as Total_sales_2018,
       round(((p.2018-p.2017)/p.2017) * 100,2) as Percentage_increase_in_sales
      from pivot_table as p
```

Query results

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

Row	Total_sales_2017	Total_sales_2018	Percentage_increase_in_sales
1	3473862.76	8452975.2	143.33

Observations: We can see there is a increase of 143.33% in the total value of orders delivered in the year 2018 compared to 2017.

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

Assumptions: Only orders that where delivered are considered for analysis

Code:

```
select c.customer_state,
       round(avg(p.payment_value),2) as Avg_price,
       round(sum(p.payment_value),2) as Total_price
from Target.orders as o
inner join
Target.payments as p
on o.order_id = p.order_id
inner join
`Target.customers` as c
on o.customer_id = c.customer_id
where o.order_status = 'delivered'
group by 1
order by 2 desc
```

Query results

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

Row	customer_state	Avg_price	Total_price
1	PB	248.33	141545.72
2	AC	234.29	19680.62
3	RO	233.2	60866.2
4	AP	232.33	16262.8
5	AL	227.08	96962.06
6	RR	218.8	10064.62
7	PA	215.92	218295.85
8	SE	208.44	75246.25
9	PI	207.11	108523.97
10	TO	204.27	61485.33
11	CE	199.9	279464.03

Observations:

- The state code PB has the highest Avg_Price and state code SP as lowest Avg_Price
- The state code SP has the highest Total_Price and state code RR as lowest Total_Price

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

Assumptions: Only the order that were delivered are considered for analysis

```
Code:
 select c.customer_state,
       round(avg(oi.freight_value),2) as Avg_freight_price,
       round(sum(oi.freight_value),2) as Total_freight_price
from Target.orders as o
inner join
Target.order_items as oi
on o.order_id = oi.order_id
inner join
`Target.customers` as c
on o.customer_id = c.customer_id
where o.order_status = 'delivered'
group by 1
order by 2 desc
```

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXEC

Row	customer_state	Avg_freight_price	Total_freight_price
1	PB	43.09	25251.73
2	RR	43.09	1982.05
3	RO	41.33	11283.24
4	AC	40.05	3644.36
5	PI	39.12	20457.19
6	MA	38.49	30794.17
7	TO	37.44	11604.86
8	SE	36.57	13714.94
9	AL	35.87	15316.77
10	RN	35.72	18609.12
11	PA	35.63	37552.98
12	AP	34.16	2767.0

Observations:

- The state code PB has the highest Avg_freight_Price and state code SP as lowest Avg_Freight_Price
- The state code SP has the highest Total_Freight_Price and state code RR as lowest Total_Freight_Price

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.
Also, calculate the difference (in days) between the estimated & actual delivery date of an order. Do this in a single query.

Assumptions: Only the orders that were delivered are considered for analysis

Code:

```
select order_id,
       order_purchase_timestamp,
       order_delivered_customer_date,
       order_estimated_delivery_date,
       date_diff(date(order_delivered_customer_date),date(order_purchase_timestamp),day) as
Delivery_Time,
       date_diff(date(order_estimated_delivery_date),date(order_delivered_customer_date),day) as
Estimated_DeliveryDate_Difference
    from `Target.orders`
 where order_status = 'delivered' and order_delivered_customer_date is not null
order by 5
```

Query results

SAVE RESULTS EXPLORE DATA

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

Row	order_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	Delivery_Time	Estimated_DeliveryDate_D
1	1d893dd7ca5f77ebf5f59f0d20...	2017-06-19 08:19:45 UTC	2017-06-19 21:07:52 UTC	2017-06-30 00:00:00 UTC	0	10
2	0922ee1619de7b995648e5a84...	2017-07-11 10:45:44 UTC	2017-07-12 20:49:42 UTC	2017-08-14 00:00:00 UTC	1	32
3	785011ed7352847ae26ea2336...	2017-10-30 10:58:54 UTC	2017-10-31 22:23:41 UTC	2017-11-10 00:00:00 UTC	1	9
4	df4c0cdd0a9a9b0d0e9f75553...	2017-10-30 13:10:42 UTC	2017-10-31 19:21:45 UTC	2017-11-10 00:00:00 UTC	1	9
5	9202a540dd3900ebeb2ff790e...	2018-01-22 08:05:13 UTC	2018-01-23 19:51:47 UTC	2018-02-06 00:00:00 UTC	1	13
6	354ca44327bb555017ef42fd9...	2017-01-30 16:33:23 UTC	2017-01-31 19:13:10 UTC	2017-03-15 00:00:00 UTC	1	42
7	a132f4bdf1eb9803546cef73ca...	2018-07-19 11:01:33 UTC	2018-07-20 20:41:52 UTC	2018-08-01 00:00:00 UTC	1	11
8	fada5293c9a23323a95ffe4129...	2018-07-12 12:25:22 UTC	2018-07-13 20:38:28 UTC	2018-08-01 00:00:00 UTC	1	18
9	bc1b85147b5edbb7cbefcf5c1...	2018-07-24 11:50:06 UTC	2018-07-25 20:56:35 UTC	2018-08-01 00:00:00 UTC	1	6
10	d28b9fc7e25679fbae70211d3...	2018-07-24 01:08:25 UTC	2018-07-25 12:16:55 UTC	2018-08-01 00:00:00 UTC	1	6
11	2f7d729d01da93cdb12badf11...	2017-12-18 09:44:19 UTC	2017-12-19 19:04:24 UTC	2018-01-05 00:00:00 UTC	1	16

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

Assumptions: Only the orders that were delivered are considered for analysis

Code:

```
with cte as
(select c.customer_state,
       round(avg(oi.freight_value),2) as Avg_freight_price,
       dense_rank() over(order by round(avg(oi.freight_value),2) desc) as
High_to_Low_Avg_freight_rank,
       dense_rank() over(order by round(avg(oi.freight_value),2)) as Low_to_High_Avg_freight_rank
from Target.orders as o
inner join
Target.order_items as oi
on o.order_id = oi.order_id
inner join
`Target.customers` as c
on o.customer_id = c.customer_id
group by 1)
```

```
 (continuation.........)

select tbl1.num as rank,
       Highest_5_states_FreightAvg,
       Highest_5_Avg_freight_price,
       Lowest_5_states_FreightAvg,
       Lowest_5_Avg_freight_price
from (select distinct customer_state as Highest_5_states_FreightAvg,
        Avg_freight_price as Highest_5_Avg_freight_price,
        High_to_Low_Avg_freight_rank as num
      from cte
    where High_to_Low_Avg_freight_rank in (1,2,3,4,5)) as tbl1
inner join
     (select distinct customer_state as Lowest_5_states_FreightAvg,
        Avg_freight_price as Lowest_5_Avg_freight_price,
        Low_to_High_Avg_freight_rank as num
      from cte
    where Low_to_High_Avg_freight_rank in (1,2,3,4,5)) as tbl2
  on tbl1.num = tbl2.num
  order by tbl1.num
```

ry results

NFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

rank	Highest_5_states_FreightAvg	Highest_5_Avg_freight_price	Lowest_5_states_FreightAvg	Lowest_5_Avg_freight_price
1	RR	43.09	SP	15.12
1	PB	43.09	SP	15.12
2	RO	41.33	PR	20.47
3	AC	40.05	MG	20.63
4	PI	39.12	RJ	20.91
5	MA	38.49	DF	21.07

Observations: Since there are two same Highest_Avg_frieght_price between RR and PB rank 1 is given to both due to which the the Lowest_Avg_frieght_price of SP (which is the 1st rank in lowest is repeated twice) to match the Highest rank count join.

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

Assumptions: Only the orders that were delivered are considered for analysis

Code:

```
with cte as
(select c.customer_state,
       avg(date_diff(order_delivered_customer_date, order_purchase_timestamp,day)) as
Avg_delivery_time,
       dense_rank() over(order by avg(date_diff(order_delivered_customer_date,
order_purchase_timestamp,day)) desc) as High_to_Low_Avg_delivery_time_rnk,
       dense_rank() over(order by avg(date_diff(order_delivered_customer_date,
order_purchase_timestamp,day))) as Low_to_High_Avg_delivery_time_rnk
from Target.orders as o
inner join
`Target.customers` as c
on o.customer_id = c.customer_id
where o.order_status = 'delivered'
group by 1
)

select Highest_5_states_DeliveryTime_Avg,
       Highest_5_DeliveryTime_Days_Avg,
       Lowest_5_states_DeliveryTime_Avg,
       Lowest_5_DeliveryTime_Days_Avg,
from (select customer_state as Highest_5_states_DeliveryTime_Avg,
        round(Avg_delivery_time,2) as Highest_5_DeliveryTime_Days_Avg,
        High_to_Low_Avg_delivery_time_rnk as num
      from cte
    where High_to_Low_Avg_delivery_time_rnk in (1,2,3,4,5)
    order by 3) as tbl1
inner join
     (select customer_state as Lowest_5_states_DeliveryTime_Avg,
        round(Avg_delivery_time,2) as Lowest_5_DeliveryTime_Days_Avg,
        Low_to_High_Avg_delivery_time_rnk as num
      from cte
    where Low_to_High_Avg_delivery_time_rnk in (1,2,3,4,5)
    order by 3) as tbl2
  on tbl1.num = tbl2.num
  order by tbl1.num
```

Query results

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

Row	Highest_5_states_DeliveryTime_Av	Highest_5_DeliveryTi	Lowest_5_states_DeliveryTime_Av	Lowest_5_DeliveryTi
1	RR	28.98	SP	8.3
2	AP	26.73	PR	11.53
3	AM	25.99	MG	11.54
4	AL	24.04	DF	12.51
5	PA	23.32	SC	14.48

Observations:

State code RR has the highest Delivery Time and State code SP as the lowest Delivery Time

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.

Assumptions: Only the orders that were delivered are considered for analysis

Code:

```
with cte as
(select c.customer_state,
       round(avg(date_diff(order_delivered_customer_date,order_estimated_delivery_date,day)),3)
as Avg_delivery_time,
       dense_rank() over(order by
avg(date_diff(order_delivered_customer_date,order_estimated_delivery_date,day))) as rnk
from Target.orders as o
inner join
`Target.customers` as c
on o.customer_id = c.customer_id
where o.order_status = 'delivered'
group by 1
)

select customer_state,
       Avg_delivery_time
  from cte
where rnk<=5
order by rnk
```

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW

Row	customer_state	Avg_delivery_time
1	AC	-19.763
2	RO	-19.132
3	AP	-18.731
4	AM	-18.607
5	RR	-16.415

Observations: State code AC has the fastest Avg Delivery time which means on an average the orders are delivered 20 days before the estimated delivery date

VI. Analysis based on the payments:

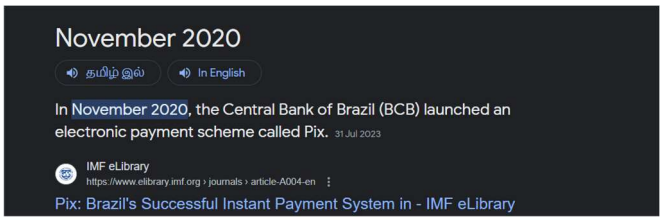
A. Find the month-on-month no. of orders placed using different payment types

JOB INFORMATION | RESULTS | CHAR

Row	payment_type
1	credit_card
2	voucher
3	not_defined
4	debit_card
5	UPI

Assumptions: Only orders that were marked as delivered are considered for analysis

Doubts: There is no UPI system in Brazil as of now but there is an almost similar payment system called PRIX but that was launched at Nov 2020 based on info in



But the problem is our dataset has only orders placed between **September 2016 to October 2018**. Theoretically UPI shouldn't be there. I guess other Payment service similar to PayPal or something related to it is renamed in the dataset as "UPI" for convenience of the learners. So, with that note I will proceed further with the question as UPI was present at that time.

Code:

```
with cte as
(select order_id,
       format_datetime('%Y/%m',order_purchase_timestamp) as purchased_year_month,
       customer_id
from `Target.orders`
where order_status = 'delivered'),
payment_sort as
(select c.purchased_year_month,
        if(p.payment_type = 'credit_card',1,0) as Credit_card,
        if(p.payment_type = 'voucher',1,0) as Voucher,
        if(p.payment_type = 'debit_card',1,0) as Debit_card,
        if(p.payment_type = 'UPI',1,0) as UPI,
        if(p.payment_type = 'not_defined',1,0) as Not_defined
from cte as c
inner join
Target.payments as p
using (order_id)
)

select purchased_year_month,
       sum(Credit_card) as Credit_card,
       sum(Voucher) as Voucher,
       sum(Debit_card) as Debit_card,
       sum(UPI) as UPI,
       sum(Not_defined) as Not_defined
from payment_sort
group by 1
order by 1
```

Query results

SAVE RESUL

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

Row	purchased_year_month	Credit_card	Voucher	Debit_card	UPI	Not_defined
1	2016/10	209	20	2	51	0
2	2016/12	1	0	0	0	0
3	2017/01	542	60	9	188	0
4	2017/02	1257	108	13	371	0
5	2017/03	1908	197	30	565	0
6	2017/04	1772	165	25	474	0
7	2017/05	2733	285	29	740	0
8	2017/06	2373	235	26	689	0
9	2017/07	2974	342	20	811	0
10	2017/08	3186	272	33	902	0
11	2017/09	3183	277	43	868	0
12	2017/10	3416	276	51	955	0

Observations: Most of the customers prefer Credit card over any other mode of payment

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

Code:

```
select payment_installments,
       count(p.order_id) as total_orders
    from Target.payments as p
    inner join
    Target.orders as o
    using(order_id)
  where order_status = 'delivered'
  group by 1
order by 1
```

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW

Row	payment_installment	total_orders
1	0	2
2	1	50929
3	2	12075
4	3	10164
5	4	6891
6	5	5095
7	6	3804
8	7	1563
9	8	4136
10	9	618
11	10	5150
12	11	22

Observations: From the above result we can during the time period we can see highest customer count instalment paid count on 1st instalment