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

1. Data type of columns in a table

A. Customers Table:

customer_id	STRING
customer_unique_id	STRING
customer_zip_code_prefix	INTEGER
customer_city	STRING
customer_state	STRING

B. Geolocation:

geolocation_zip_code_prefix	INTEGER
geolocation_lat	FLOAT
geolocation_lng	FLOAT
geolocation_city	STRING
geolocation_state	STRING

C. order_items:

order_id	STRING
order_item_id	INTEGER
product_id	STRING
seller_id	STRING
shipping_limit_date	TIMESTAMP
price	FLOAT
freight_value	FLOAT

D. Order_review:

review_id	STRING
order_id	STRING
review_score	INTEGER
review_comment_title	STRING
review_creation_date	TIMESTAMP
review_answer_timestamp	TIMESTAMP

G. Orders:

order_id	STRING
customer_id	STRING
order_status	STRING
order_purchase_timestamp	TIMESTAMP
order_approved_at	TIMESTAMP
order_delivered_carrier_date	TIMESTAMP
order_delivered_customer_date	TIMESTAMP
order_estimated_delivery_date	TIMESTAMP

H. Payments:

order_id	STRING
payment_sequential	INTEGER
payment_type	STRING
payment_installments	INTEGER
payment_value	FLOAT

I. Products:

product_id	STRING
product_category	STRING
product_name_length	INTEGER
product_description_length	INTEGER
product_photos_qty	INTEGER
product_weight_g	INTEGER
product_length_cm	INTEGER
product_height_cm	INTEGER
product_width_cm	INTEGER

J. Seller:

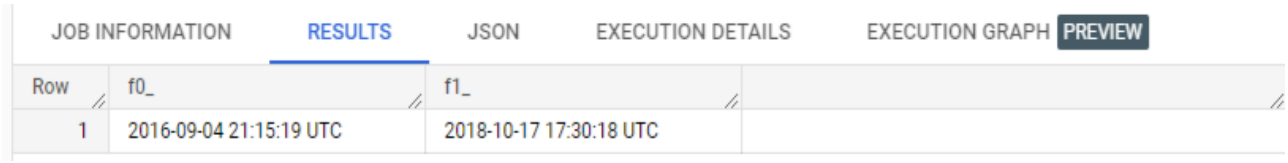
seller_id	STRING
seller_zip_code_prefix	INTEGER
seller_city	STRING
seller_state	STRING

2. Time period for which the data is given

A. QUERY:

```
SELECT MIN(order_purchase_timestamp), MAX(order_purchase_timestamp)
FROM target.orders
```

Table:



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

Row	f0_	f1_
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

3. Cities and States of customers ordered during the given period

A. QUERY:

```
SELECT c.customer_city, c.customer_state, COUNT(*) Orders
FROM `target.orders` o JOIN `target.customers` c USING (customer_id)
GROUP BY c.customer_city, c.customer_state
ORDER BY COUNT(*) DESC
LIMIT 5
```

Table:

Row	customer_city	customer_state	Orders
1	sao paulo	SP	15540
2	rio de janeiro	RJ	6882
3	belo horizonte	MG	2773
4	brasilia	DF	2131
5	curitiba	PR	1521

Insight:

1. These are the places from where we had a potential customer and have the highest orders placed

Recommendations:

1. Increase the stocks in this area and showcase products which are relevant to buyers

Table:

Row	customer_city	customer_state	Orders
1	caem	BA	1
2	avai	SP	1
3	bodo	RN	1
4	cipo	BA	1
5	bora	SP	1

Insight:

1. These are the areas where we had a least no of customers and produces less revenue.

Recommendations:

1. Take a marketing team increase the awareness of the products that could benefit them.
2. See the competition and try to do something innovative from competitor.
3. Pick up the pain point and find the solution that could relate our domain and create a relief for the customers.

In-depth Exploration:

1. Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?

A. There is Increase in e-commerce market but there are specific months where sales are high

QUERY:

```
WITH cte1 AS (
  SELECT EXTRACT(YEAR FROM order_purchase_timestamp) year,
    EXTRACT(MONTH FROM order_purchase_timestamp) month
  FROM target.orders
)
SELECT month, COUNT(*) sales
FROM cte1
GROUP BY month
ORDER BY COUNT(*) DESC
```

TABLE:

Row	month	sales
1	8	10843
2	5	10573
3	7	10318
4	3	9893
5	6	9412
6	4	9343
7	2	8508
8	1	8069
9	11	7544
10	12	5674
11	10	4959
12	9	4305

Insight:

1. 8, 5, 7 are the months with highest sales
2. 9, 10, 12 are the months with lowest sales

Recommendations:

1. In the high growing months try to keep the stocks full for the products that are sold at high pace and try to showcase products that are less sold because the probability of selling the items in these months increases
2. In the low selling months try to maintain only those stocks which has a high demand

2. What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?

QUERY:

```
CREATE VIEW target.v1 AS
SELECT EXTRACT(HOUR FROM order_purchase_timestamp) AS hr1
FROM target.orders
WHERE EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 6 AND 7 ;

CREATE VIEW target.v2 AS
SELECT EXTRACT(HOUR FROM order_purchase_timestamp) AS hr2
FROM target.orders
WHERE EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 ;

CREATE VIEW target.v3 AS
SELECT EXTRACT(HOUR FROM order_purchase_timestamp) AS hr2
FROM target.orders
WHERE EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 ;

CREATE VIEW target.v4 AS
SELECT EXTRACT(HOUR FROM order_purchase_timestamp) AS hr2
FROM target.orders
WHERE EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 19 AND 24 ;

SELECT CASE
WHEN COUNT(v1) > 0 THEN "DAWN" ELSE ""
END, COUNT(v1)
FROM target.v1

UNION ALL

SELECT CASE
WHEN COUNT(v2) > 0 THEN "MORNING" ELSE ""
END, COUNT(v2)
FROM target.v2

UNION ALL

SELECT CASE
WHEN COUNT(v3) > 0 THEN "AFTERNOON" ELSE ""
END, COUNT(v3)
FROM target.v3

UNION ALL

SELECT CASE
WHEN COUNT(v4) > 0 THEN "NIGHT" ELSE ""  END,  COUNT(v4)FROM target.v4
```

TABLE:

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXE

Row	f0_	f1_
1	AFTERNOON	38135
2	DAWN	1733
3	MORNING	27733
4	NIGHT	28331

Insights:

1. Afternoon, Morning, Night has a good number of active buyers.

Recommendations:

1. Try to maintain servers and take care of traffic, as traffic increases the database efficiency decreases that could create a loss of customers.

Evolution of E-commerce orders in the Brazil region:

1. Get month on month orders by states

QUERY:

```
WITH cte1 AS (
  SELECT EXTRACT(YEAR FROM order_purchase_timestamp) year, EXTRACT(MONTH FROM order
_purchase_timestamp) month, order_id, customer_state
  FROM target.orders JOIN target.customers USING (customer_id)
)

SELECT customer_state state, COUNT(*) sales
FROM cte1
GROUP BY state
ORDER BY sales DESC
```

TABLE:

Row	state	sales
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

Insights:

1. SP, RJ, MG has the highest sales
2. GO, ES, DF has the low sales

Recommendations:

1. Take care of the servers in high potential states
2. Increase the marketing cost in the low performing state to spread awareness, try to give vouchers, gift carts, discounts that attracts the customers.

2. Distribution of customers across the states in Brazil

QUERY:

```
SELECT customer_city, COUNT(*) AS No_Of_Customers
FROM target.customers
GROUP BY customer_city
ORDER BY COUNT(*) DESC
LIMIT 10
```

TABLE:

Row	customer_city	No_Of_Custome
1	sao paulo	15540
2	rio de janeiro	6882
3	belo horizonte	2773
4	brasilia	2131
5	curitiba	1521
6	campinas	1444
7	porto alegre	1379
8	salvador	1245
9	guarulhos	1189
10	sao bernardo do campo	938

Insights:

1. Only top 2 rows have the large customer base.

Recommendations:

1. Market the products in such a way that demand should increase and supply chain automatically increases.

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

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

   QUERY:

```
WITH cte1 AS (
  SELECT EXTRACT(YEAR FROM order_purchase_timestamp) year , payment_value, order_id
  FROM target.orders JOIN target.payments USING (order_id)
  WHERE EXTRACT(YEAR FROM order_purchase_timestamp) BETWEEN 2017 AND 2018
  AND EXTRACT(MONTH FROM order_purchase_timestamp) BETWEEN 1 AND 8
),

cte2 AS (
  SELECT product_category, SUM(payment_value) as val_2017
  FROM cte1 JOIN target.order_items USING (order_id)
    JOIN target.products USING (product_id)
  WHERE year = 2017
  GROUP BY product_category
),

cte3 AS (
  SELECT product_category, SUM(payment_value) as val_2018
  FROM cte1 JOIN target.order_items USING (order_id)
    JOIN target.products USING (product_id)
  WHERE year = 2018
  GROUP BY product_category
)


SELECT product_category, val_2017, val_2018,
  ROUND(((val_2018 - val_2017)/val_2018) * 100) percentage_inc
FROM cte2 JOIN cte3 USING (product_category)
ORDER BY percentage_inc DESC
```

TABLE:

Row	product_category	val_2017	val_2018	percentage_inc
1	HOUSE PASTALS OVEN AND C...	222.51	50333.3400...	100.0
2	Construction Tools Illumination	1072.43	66220.7600...	98.0
3	CONSTRUCTION SECURITY TO...	847.310000...	51110.2600...	98.0
4	party articles	83.41	3628.37000...	98.0
5	Construction Tools Construction	6047.97999...	207117.979...	97.0

Insights:

1. These are the product_category whose sales has been increased

Recommendations:

1. Increase the production of these products and try to upgrade the quality and increase the variations of these products

TABLE:

Row	product_category	val_2017	val_2018	percentage_inc
1	cds music dvds	720.980000...	117.58	-513.0
2	IMAGE IMPORT TABLETS	6753.88	1970.25999...	-243.0
3	Fashion Sport	1640.30999...	505.37	-225.0
4	Blu Ray DVDs	5125.38000...	2123.04999...	-141.0
5	Fashion Men's Clothing	7616.76000...	3178.97	-140.0

Insights:

1. These are the product_category whose sales has been decreased

Recommendations:

1. These products are eating the revenue try to remove the products and invest those revenue in generating innovative products
2. Another option is try to upgrade the product in a way that people love it.

2. Mean & Sum of price and freight value by customer state

QUERY:

```
WITH cte1 AS (
  SELECT customer_state,
    ROUND(SUM(price)) AS sum_of_price,
    ROUND(SUM(price)/COUNT(price)) AS Mean_of_price,
    ROUND(SUM(freight_value)) AS sum_of_freight_value,
    ROUND(SUM(freight_value)/COUNT(freight_value)) AS Mean_freight_value,
  FROM target.orders JOIN target.order_items USING (order_id)
    JOIN target.customers USING (customer_id)
  GROUP BY customer_state
)

SELECT *,
  ROUND(100-(((Mean_of_price - Mean_freight_value)/Mean_of_price)*100)) AS
  percentage_btw_pf
FROM cte1
ORDER BY percentage_btw_pf DESC
```

TABLE:

Row	customer_state	sum_of_price	Mean_of_price	sum_of_freight_	Mean_freight_	percentage_btw_pf
1	RR	7829.0	151.0	2235.0	43.0	28.0
2	MA	119648.0	145.0	31524.0	38.0	26.0
3	RO	46141.0	166.0	11417.0	41.0	25.0
4	SE	58921.0	153.0	14111.0	37.0	24.0
5	PI	86914.0	160.0	21218.0	39.0	24.0
6	AM	22357.0	135.0	5479.0	33.0	24.0
7	PE	262788.0	146.0	59450.0	33.0	23.0
8	PB	115268.0	191.0	25720.0	43.0	23.0
9	RN	83035.0	157.0	18860.0	36.0	23.0
10	TO	49622.0	158.0	11733.0	37.0	23.0

Insight:

1. Percentage_btw_pf = it is the % of mean_freight_value in mean_price

Recommendations:

1. The percentage is too high as compared to the product, If the service charges increase without a reason, then there is a chance of decrease in demand of that category. So, try to reduce the transport charge as low as possible.

Analysis on sales, freight and delivery time

1. Calculate days between purchasing, delivering and estimated delivery

   QUERY:

```
WITH cte1 AS (
  SELECT product_category,
    CASE
      WHEN order_delivered_customer_date IS NULL THEN NULL
      ELSE (DATE_DIFF(DATE(order_delivered_customer_date),
       DATE(order_purchase_timestamp), DAY)) END AS Time_delivery,
    CASE
      WHEN order_estimated_delivery_date IS NULL THEN NULL
      ELSE (DATE_DIFF(DATE(order_estimated_delivery_date),
       DATE(order_purchase_timestamp), DAY)) END AS Time_estimated
  FROM target.orders JOIN target.order_items USING (order_id)
    JOIN target.products USING (product_id)
)

SELECT *
FROM cte1
WHERE Time_delivery IS NOT NULL
ORDER BY Time_delivery DESC, Time_estimated DESC
```

   TABLE:

Row	product_category	Time_delivery	Time_estimated
1	automotive	210	29
2	Cool Stuff	208	20
3	Games consoles	196	31
4	Furniture office	195	40
5	musical instruments	195	29
6	Watches present	194	33
7	Casa Construcao	191	16
8	Furniture Decoration	190	23
9	automotive	188	29
10	ELECTRICES 2	188	26

   Insight:

   1. The delivery time is much more as compared to estimated time

   Recommendations:

   1. Try to partnership with some companies like FedEx, CSX which has the good transport facility and help the customer by delivering the product as soon as possible

2. Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:
    1. time_to_delivery = order_purchase_timestamp - order_delivered_customer_date
    2. diff_estimated_delivery = order_estimated_delivery_date- order_delivered_customer_date

QUERY:

```
SELECT product_category, order_id,
  CASE
    WHEN order_delivered_customer_date IS NULL THEN NULL
    ELSE (DATE_DIFF(DATE(order_delivered_customer_date),
       DATE(order_purchase_timestamp), DAY))
  END AS time_to_delivery ,
  CASE
    WHEN order_estimated_delivery_date IS NULL THEN NULL
    ELSE (DATE_DIFF(DATE(order_estimated_delivery_date),
       DATE(order_delivered_customer_date), DAY))
  END AS diff_estimated_delivery

FROM target.orders JOIN target.order_items USING (order_id)
  JOIN target.products USING (product_id)
WHERE product_category IS NOT NULL AND order_purchase_timestamp IS NOT NULL
ORDER BY time_to_delivery DESC, diff_estimated_delivery
LIMIT 10
```

TABLE:

Row	product_category	order_id	time_to_delivery	diff_estimated_c
1	automotive	ca07593549f1816d26a572e06...	210	-181
2	Cool Stuff	1b3190b2dfa9d789e1f14c05b...	208	-188
3	Games consoles	440d0d17af552815d15a9e41a...	196	-165
4	musical instruments	285ab9426d6982034523a855f...	195	-166
5	Furniture office	2fb597c2f772eca01b1f5c561b...	195	-155
6	Watches present	0f4519c5f1c541ddec9f21b3bd...	194	-161
7	Casa Construcao	47b40429ed8cce3aee9199792...	191	-175
8	Furniture Decoration	2fe324febf907e3ea3f2aa9650...	190	-167
9	ELECTRICES 2	c27815f7e3dd0b926b5855262...	188	-162
10	automotive	2d7561026d542c8dbd8f0daea...	188	-159

Insights:

1. Time of delivery is crossing the estimated time
2. This is not happening only for a specific category of product, Its happening to all categories

Recommendations:

1. Try to increase the distribution chain.
2. Partner with transport companies or try to build one for own.
3. Optimize the transport like use bikes for small products, vans for intermediate size of products and trucks for extremely large products.

3. Sort the data to get the following:

    1. Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5

```
SELECT customer_state, AVG(freight_value) AS Average_value
FROM target.order_items JOIN target.orders USING (order_id)
  JOIN target.customers USING (customer_id)
GROUP BY customer_state
ORDER BY AVG(freight_value) DESC
LIMIT 5;
```

TOP 5:

Row	customer_state	Average_value
1	RR	42.9844230...
2	PB	42.7238039...
3	RO	41.0697122...
4	AC	40.0733695...
5	PI	39.1479704...

BOTTOM 5:

Row	customer_state	Average_Value
1	SP	15.1472753...
2	PR	20.5316515...
3	MG	20.6301668...
4	RJ	20.9609239...
5	DF	21.0413549...

Insights:

1. Freight_value plays a major role in product selection by the customer, If the freight_value even touches the 50% of the amount customers would be unhappy to buy the products

Recommendations:

1. As the top states have the high value try to decrease the value by cost cutting in unnecessary budgets like
    a. Use electric vehicles which moves in economy speed and take less charge
    b. Less use of Man Power, 1 Person needs to use the 2-wheeler, only 2 people needs to use the truck accordingly...

2. Top 5 states with highest/lowest average time to delivery

```
SELECT customer_city,
  AVG(DATE_DIFF(DATE(order_delivered_customer_date),
       DATE(order_purchase_timestamp), DAY)) AS avg_delivery_time

FROM target.orders JOIN target.order_items USING (order_id)
  JOIN target.customers USING (customer_id)
WHERE order_delivered_customer_date IS NOT NULL
GROUP BY customer_city
ORDER BY avg_delivery_time DESC
LIMIT 5
```

HEIGHEST:

Row	customer_city	avg_delivery_tim
1	novo brasil	148.0
2	capinzal do norte	109.0
3	adhemar de barros	98.0
4	santa cruz de goias	86.6666666...
5	arace	86.5

LOWEST:

Row	customer_city	avg_delivery_tim
1	iomere	3.0
2	siriji	3.0
3	pedra bela	4.0
4	divino das laranjeiras	4.0
5	contenda	4.0

Insights:

1. Top 5 cities take more time to deliver.
2. Bottom 5 cities take less time to deliver.

Recommendations:

1. Increase the distribution system.
2. Use light motor vehicles so that it could get into most of the areas.

3. Top 5 states where delivery is really fast/ not so fast compared to estimated date

```
SELECT customer_city,
  AVG(CASE
    WHEN order_estimated_delivery_date IS NULL THEN 0
    ELSE (DATE_DIFF(DATE(order_estimated_delivery_date),
       DATE(order_purchase_timestamp), DAY))
  END) AS avg_delivery_time

FROM target.orders JOIN target.order_items USING (order_id)
  JOIN target.customers USING (customer_id)
GROUP BY customer_city
ORDER BY avg_delivery_time DESC
LIMIT 5
```

FAST:

Row	customer_city	avg_delivery_tim
1	juruti	88.0
2	portalegre	85.0
3	alvorada d'oeste	82.0
4	japoata	67.0
5	apuarema	65.0

NOT SO FAST:

Row	customer_city	avg_delivery_tim
1	claro dos pocoes	9.0
2	pereiras	10.0
3	cipo-guacu	11.0
4	meridiano	12.0
5	ibema	13.0

Insights:

1. Top 5 cities estimations are good
2. Bottom 5 cities estimations are very low

Recommendations:

1. Increase the distribution system at low estimated areas
2. Partner with transport companies or build one for own.

Payment type analysis:

1. Month over Month count of orders for different payment types

QUERY:

```
WITH cte1 AS (
  SELECT EXTRACT(YEAR FROM order_purchase_timestamp) year,
    EXTRACT(MONTH FROM order_purchase_timestamp) month, payment_type
  FROM target.orders JOIN target.payments USING (order_id)
)

SELECT year, month, payment_type, COUNT(*) AS No_Of_orders
FROM cte1
GROUP BY year, month, payment_type
ORDER BY year, month
LIMIT 10
```

TABLE:

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

Insights:

1. credit card, UPI payments are more as compared to other transactions

Recommendations:

1. Try to accept all bank credit cards and all type of cards like VISA, MasterCard etc.
2. Enable the UPI transactions all over the places.

2. Count of orders based on the no. of payment installments

```
SELECT payment_type, COUNT(*) No_of_payments
FROM target.payments
GROUP BY payment_type
```

TABLE:

Row	payment_type	No_of_payments
1	credit_card	76795
2	voucher	5775
3	not_defined	3
4	debit_card	1529
5	UPI	19784

INSTALLMENT PAYMENTS: Credit_card, Debit_card, not_defined

```
SELECT payment_type, COUNT(*) No_of_payments
FROM target.payments
WHERE payment_type IN ('credit_card', 'debit_card', 'not_defined')
GROUP BY payment_type
```

TABLE:

Row	payment_type	No_of_payments
1	credit_card	76795
2	not_defined	3
3	debit_card	1529

Insights:

1. credit card, UPI payments are more as compared to other transactions

Recommendations:

1. Try to accept all bank credit cards and all type of cards like VISA, MasterCard etc.
2. Enable the UPI transactions all over the places.