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:

```
select * from `scaler-sql-374010.Target_project.INFORMATION_SCHEMA.COLUMNS`
```

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

Row	table_catalog	table_schema	table_name	column_name	ordinal_position	is_nullable
1	scaler-sql-374010	Target_project	order_items	order_id	1	YES
2	scaler-sql-374010	Target_project	order_items	order_item_id	2	YES
3	scaler-sql-374010	Target_project	order_items	product_id	3	YES
4	scaler-sql-374010	Target_project	order_items	seller_id	4	YES
5	scaler-sql-374010	Target_project	order_items	shipping_limit_date	5	YES
6	scaler-sql-374010	Target_project	order_items	price	6	YES
7	scaler-sql-374010	Target_project	order_items	freight_value	7	YES
8	scaler-sql-374010	Target_project	sellers	seller_id	1	YES
9	scaler-sql-374010	Target_project	sellers	seller_zip_code_prefix	2	YES
10	scaler-sql-374010	Target_project	sellers	seller_city	3	YES
11	scaler-sql-374010	Target_project	sellers	seller_state	4	YES
12	scaler-sql-374010	Target_project	geolocation	geolocation_zip_code_prefix	1	YES
13	scaler-sql-374010	Target_project	geolocation	geolocation_lat	2	YES
14	scaler-sql-374010	Target_project	geolocation	geolocation_lng	3	YES

a) Time period for which the data is given.

- From 2016-09-04 to 2018-10-17

```
SELECT  min(order_purchase_timestamp) as min_date, max(order_purchase_timestamp) as max_date FROM `scaler-sql-374010.Target_project.orders`
```

Query results

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

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

b) Cities and States of customers ordered during the given period

```
SELECT count(distinct geolocation_city) as city_count
FROM `scaler-sql-374010.Target_project.geolocation`
```

Press Alt+F1 for Accessibility Optior

Query results SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXE >

w	city_count
1	8011

```
SELECT count(distinct g.geolocation_state) as states_cnt,count(distinct g.geolocation_city) as city_cnt FROM
 `scaler-sql-374010.Target_project.orders` o
join
`scaler-sql-374010.Target_project.customers` c on o.customer_id=c.customer_id
join
`scaler-sql-374010.Target_project.geolocation` g on c.customer_zip_code_prefix=g.geolocation_zip_code_prefix
```

Query results SAVE RESULTS

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

Row	states_cnt	city_cnt
1	27	5812

2. In-depth Exploration:

a) 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?

```
SELECT
Concat(substring(cast(order_purchase_timestamp as string),1,4),substring(cast(order_purchase_timestamp as string),6,2)) as year_month,
count(order_id) as order_count

 FROM
 `scaler-sql-374010.Target_project.orders`
  group by year_month
 order by year_month
```

Query results

JOB INFORMATION | RESULTS | JSON | EXE

Row	year_month	order_count
5	201702	1780
6	201703	2682
7	201704	2404
8	201705	3700
9	201706	3245
10	201707	4026
11	201708	4331
12	201709	4285
13	201710	4631
14	201711	7544
15	201712	5673
16	201801	7269

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

```
SELECT

case
```

```
when EXTRACT(hour FROM order_purchase_timestamp AT TIME ZONE "UTC-
3") >4 and EXTRACT(hour FROM order_purchase_timestamp AT TIME ZONE "UTC-3") <8   then "Dawn"
when EXTRACT(hour FROM order_purchase_timestamp AT TIME ZONE "UTC-
3") >= 8 and EXTRACT(hour FROM order_purchase_timestamp AT TIME ZONE "UTC-3") <12 then "Morning"
when EXTRACT(hour FROM order_purchase_timestamp AT TIME ZONE "UTC-
3") >=12 and EXTRACT(hour FROM order_purchase_timestamp AT TIME ZONE "UTC-3") <20  then "Afternoon"
else "night"
end as grp,
count(order_id) as order_count
 FROM
 `scaler-sql-374010.Target_project.orders`
 group by grp
```

JOB INFORMATION | RESULTS | JSON | EXECUT

Row	grp	order_count
1	Morning	25660
2	night	10596
3	Afternoon	49256
4	Dawn	13929

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

a) Get month on month orders by states.

```
SELECT
Concat(substring(cast(o.order_purchase_timestamp as string),1,4),substring(cast(o.order_purchase_timestamp as string),6,2)) as
year_month,
c.customer_state,
count(order_id) as order_count

 FROM
```

```
 `scaler-sql-374010.Target_project.orders` o join
 `scaler-sql-374010.Target_project.customers` c on o.customer_id=c.customer_id
group by year_month,
c.customer_state

 order by year_month
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUT

Row	year_month	customer_state	order_count
1	201609	RR	1
2	201609	RS	1
3	201609	SP	2
4	201610	SP	113
5	201610	RS	24
6	201610	RJ	56
7	201610	MT	3
8	201610	GO	9
9	201610	MG	40

b) Distribution of customers across the states in Brazil

```
SELECT
g.geolocation_state,
count(distinct c.customer_id) as customer_cnt,
round((count(distinct c.customer_id)/sum(count(distinct c.customer_id)) over())*100,2) as  cust_perc_by_state
FROM
 `scaler-sql-374010.Target_project.customers` c
 full join
`scaler-sql-374010.Target_project.geolocation` g on c.customer_zip_code_prefix=g.geolocation_zip_code_prefix
group by
g.geolocation_state
order by customer_cnt desc
```

INFORMATION | RESULTS | JSON | EXECUTION DETAILS

geolocation_state	customer_cnt	cust_perc_by_st
SP	41731	41.93
RJ	12839	12.9
MG	11624	11.68
RS	5473	5.5
PR	5034	5.06
SC	3651	3.67
BA	3371	3.39
ES	2027	2.04
GO	2011	2.02
DF	1974	1.98
PE	1648	1.66
CE	1332	1.34
PA	972	0.98

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

   a) 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.

```
select temp.o_year,temp.pay_value,
lag(temp.pay_value)over (order by o_year ) as prev_val,

(temp.pay_value-lag(temp.pay_value)over (order by o_year ))  as inc,
round(((temp.pay_value-
lag(temp.pay_value)over (order by o_year ))/ lag(temp.pay_value)over (order by o_year ))*100,2)  as percent_inc
from (
```

```
SELECT
EXTRACT(year FROM order_purchase_timestamp AT TIME ZONE "UTC-3") as o_year,
sum(p.payment_value) as pay_value,
FROM
 `scaler-sql-374010.Target_project.orders` o
join
`scaler-sql-374010.Target_project.payments` p on o.order_id=p.order_id
group by
o_year
) as temp

order by o_year
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRA

Row	o_year	pay_value	prev_val	inc	percent_inc
1	2016	59362.3400...	*null*	*null*	*null*
2	2017	7249931.08...	59362.3400...	7190568.74...	12113.01
3	2018	8699578.68...	7249931.08...	1449647.60...	20.0

b) Mean & Sum of price and freight value by customer state

```
select

c.customer_state,
round(sum(oi.price),2) as total_price,
round(avg(oi.price),2) as avg_price
from `scaler-sql-374010.Target_project.order_items` oi
join
 `scaler-sql-374010.Target_project.orders`o on oi.order_id=o.order_id
 join
  `scaler-sql-374010.Target_project.customers` c on o.customer_id=c.customer_id
 group by
 c.customer_state
 order by total_price desc
```

Row	customer_state	total_price	avg_price
1	SP	5202955.05	109.65
2	RJ	1824092.67	125.12
3	MG	1585308.03	120.75
4	RS	750304.02	120.34
5	PR	683083.76	119.0
6	SC	520553.34	124.65
7	BA	511349.99	134.6
8	DF	302603.94	125.77
9	GO	294591.95	126.27
10	ES	275037.31	121.91
11	PE	262788.03	145.51

```
select

c.customer_state,
round(sum(freight_value),2) as sum_val,
round(avg(freight_value),2) as avg_val
from `scaler-sql-374010.Target_project.order_items` oi
join
 `scaler-sql-374010.Target_project.orders`o on oi.order_id=o.order_id
 join
  `scaler-sql-374010.Target_project.customers` c on o.customer_id=c.customer_id
 group by
 c.customer_state
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	sum_val	avg_val
1	SP	718723.07	15.15
2	RJ	305589.31	20.96
3	PR	117851.68	20.53
4	SC	89660.26	21.47
5	DF	50625.5	21.04
6	MG	270853.46	20.63
7	PA	38699.3	35.83
8	BA	100156.68	26.36
9	GO	53114.98	22.77

5. Analysis on sales, freight and delivery time

a) Calculate days between purchasing, delivering and estimated delivery.

```
SELECT

sum(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, day)) as days_deliver_purchase,
sum(DATE_DIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date, day)) as days_deliver_estimed,
sum(DATE_DIFF(o.order_estimated_delivery_date,o.order_purchase_timestamp, day)) as days_estimed_purchase

from
 `scaler-sql-374010.Target_project.orders` o
```

days_deliver_purchase	days_deliver_estimed	days_estimed_purchase
1166789	-1057185	232313

```
SELECT
c.customer_state,
sum(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, day)) as days_deliver_purchase,
sum(DATE_DIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date, day)) as days_deliver_estimed,
sum(DATE_DIFF(o.order_estimated_delivery_date,o.order_purchase_timestamp, day)) as days_estimed_purchase

from
 `scaler-sql-374010.Target_project.orders` o  join
 `scaler-sql-374010.Target_project.customers` c on o.customer_id=c.customer_id
 group by
 c.customer_state
 order by
 c.customer_state
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTIO

Row	customer_state	days_deliver_pu	days_deliver_est	days_estimed_p
1	AC	1651	-1581	3302
2	AL	9544	-3155	13309
3	AM	3768	-2698	6624
4	AP	1791	-1255	3108
5	BA	61429	-32348	98144
6	CE	26626	-12736	41332
7	DF	26019	-23127	51493
8	ES	30587	-19189	51381
9	GO	29650	-22050	54025

b) Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:

- time_to_delivery = order_purchase_timestamp-order_delivered_customer_date
- diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

```
SELECT

round(avg(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, day)),2) as avg_time_to_delivery,
round(avg(DATE_DIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date, day)),2) as avg_diff_estimated_delivery
from
 `scaler-sql-374010.Target_project.orders` o
```

Row	avg_time_to_delivery	avg_diff_estimated_delivery
1	12.09	-10.96

c) 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(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, day)),2) as avg_time_to_delivery,
round(avg(DATE_DIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date, day)),2) as avg_diff_estimated_delivery
from
 `scaler-sql-374010.Target_project.orders` o
 join
`scaler-sql-374010.Target_project.customers`c on o.customer_id=c.customer_id
join
`scaler-sql-374010.Target_project.order_items` oi on o.order_id=oi.order_id
group by
c.customer_state
order by
avg_freight_value desc,
avg_time_to_delivery desc , avg_diff_estimated_delivery desc
```

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

Row	customer_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimat
1	RR	42.98	27.83	-17.43
2	PB	42.72	20.12	-12.15
3	RO	41.07	19.28	-19.08
4	AC	40.07	20.33	-20.01
5	PI	39.15	18.93	-10.68
6	MA	38.26	21.2	-9.11
7	TO	37.25	17.0	-11.46
8	SE	36.65	20.98	-9.17
9	AL	35.84	23.99	-7.98
10	PA	35.83	23.3	-13.37

d) Sort the data to get the following:
e) Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5
f) Top 5 states with highest/lowest average time to delivery
g) Top 5 states where delivery is really fast/ not so fast compared to estimated date

```
Top 5 states with highest freight value
SELECT
c.customer_state,
round(avg(oi.freight_value),2) as avg_freight_value
from
 `scaler-sql-374010.Target_project.orders` o
 join
`scaler-sql-374010.Target_project.customers`c on o.customer_id=c.customer_id
join
`scaler-sql-374010.Target_project.order_items` oi on o.order_id=oi.order_id
group by
c.customer_state
order by
avg_freight_value desc
```

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

Top 5 states with lowest freight value

```
SELECT
c.customer_state,
round(avg(oi.freight_value),2) as avg_freight_value
from
 `scaler-sql-374010.Target_project.orders` o
 join
`scaler-sql-374010.Target_project.customers`c on o.customer_id=c.customer_id
join
`scaler-sql-374010.Target_project.order_items` oi on o.order_id=oi.order_id
group by
c.customer_state
order by
avg_freight_value
limit 5
```

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

Top 5 states with highest average time to delivery

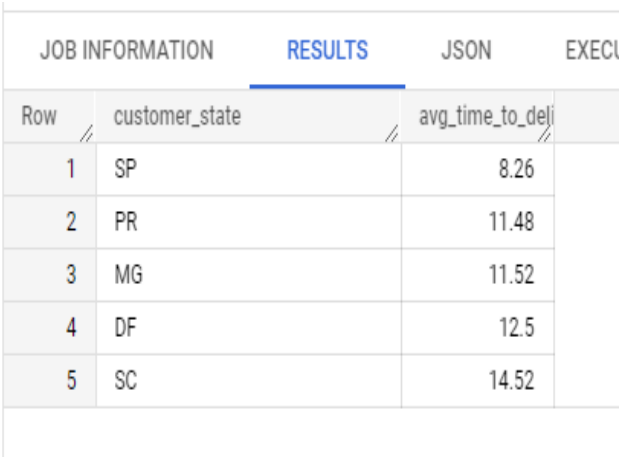
```
SELECT
c.customer_state,
round(avg(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timest
amp, day)),2) as avg_time_to_delivery
from
 `scaler-sql-374010.Target_project.orders` o
 join
`scaler-sql-374010.Target_project.customers`c on o.customer_id=c.customer_id
join
`scaler-sql-374010.Target_project.order_items` oi on o.order_id=oi.order_id
group by
c.customer_state
order by
avg_time_to_delivery desc
limit 5
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_time_to_delivery
1	RR	27.83
2	AP	27.75
3	AM	25.96
4	AL	23.99
5	PA	23.3

Top 5 states with lowest average time to delivery

```
SELECT
c.customer_state,
round(avg(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_tim
estamp, day)),2) as avg_time_to_delivery
from
 `scaler-sql-374010.Target_project.orders` o
 join
`scaler-sql-
374010.Target_project.customers`c on o.customer_id=c.customer_id
join
`scaler-sql-374010.Target_project.order_items` oi on o.order_id=oi.order_id
group by
c.customer_state
order by
avg_time_to_delivery
limit 5
```



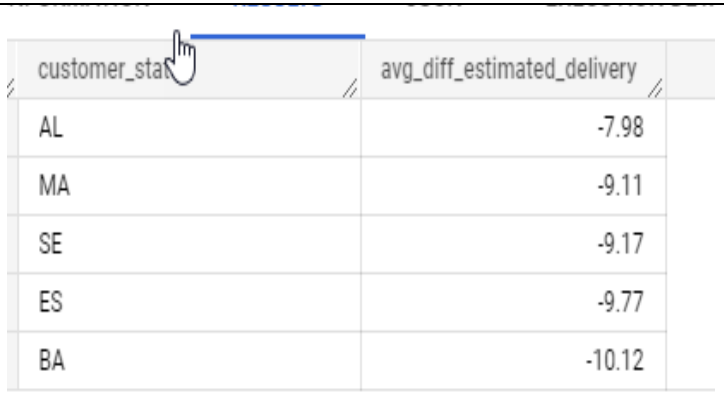
JOB INFORMATION | RESULTS | JSON | EXECU

Row	customer_state	avg_time_to_deli
1	SP	8.26
2	PR	11.48
3	MG	11.52
4	DF	12.5
5	SC	14.52

Top 5 states where delivery is really fast compared to estimated date

```
SELECT
c.customer_state,
round(avg(DATE_DIFF(o.order_delivered_customer_date, o.order_estimated_del
ivery_date, day)),2) as avg_diff_estimated_delivery
from
 `scaler-sql-374010.Target_project.orders` o
 join
`scaler-sql-
374010.Target_project.customers`c on o.customer_id=c.customer_id
join
`scaler-sql-374010.Target_project.order_items` oi on o.order_id=oi.order_id
```

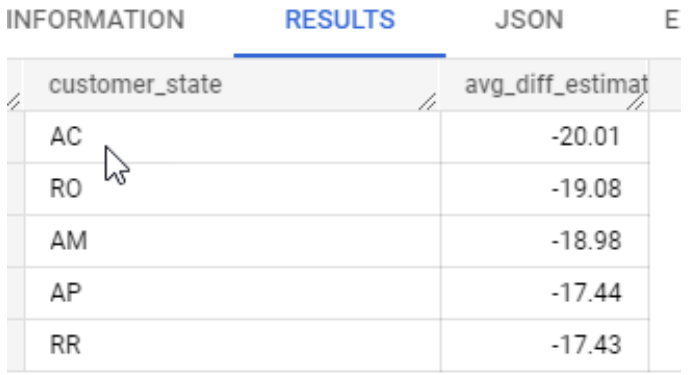
```
group by
c.customer_state
order by
avg_diff_estimated_delivery desc
limit 5
```



customer_sta	avg_diff_estimated_delivery
AL	-7.98
MA	-9.11
SE	-9.17
ES	-9.77
BA	-10.12

Top 5 states where delivery is not so fast compared to estimated date

```
SELECT
c.customer_state,
round(avg(DATE_DIFF(o.order_delivered_customer_date, o.order_estimated_del
ivery_date, day)),2) as avg_diff_estimated_delivery
from
 `scaler-sql-374010.Target_project.orders` o
 join
`scaler-sql-
374010.Target_project.customers`c on o.customer_id=c.customer_id
join
`scaler-sql-374010.Target_project.order_items` oi on o.order_id=oi.order_id
group by
c.customer_state
order by
avg_diff_estimated_delivery
limit 5
```



INFORMATION RESULTS JSON E

customer_state	avg_diff_estimat
AC	-20.01
RO	-19.08
AM	-18.98
AP	-17.44
RR	-17.43

6. Payment type analysis:

a) Month over Month count of orders for different payment types

```
select
payment_type,
count(order_id) as order_cnt

from
`scaler-sql-374010.Target_project.payments`

group by
payment_type
order by order_cnt desc
```

payment_type	order_cnt
credit_card	76795
UPI	19784
voucher	5775
debit_card	1529
not_defined	3

```
SELECT year_month,credit_card,UPI,voucher,debit_card
FROM

(select
```

```
Concat(substring(cast(o.order_purchase_timestamp as string),1,4),substring(cast(o.order_purchase_timestamp as string),
6,2)) as year_month,
p.payment_type,
count(o.order_id) as order_cnt

from
`scaler-sql-374010.Target_project.orders` o
join
`scaler-sql-374010.Target_project.payments`p on o.order_id=p.order_id


group by
year_month,
payment_type
order by year_month  ) as temp
PIVOT
(
SUM(order_cnt) FOR payment_type IN ("credit_card","UPI","voucher","debit_card" )
) AS PivotTable   order by year_month
```

NFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH | PR

year_month	credit_card	UPI	voucher	debit_card
201701	583	197	61	9
201702	1356	398	119	13
201703	2016	590	200	31
201704	1846	496	202	27
201705	2853	772	289	30
201706	2463	707	239	27
201707	3086	845	364	22
201708	3284	938	294	34
201709	3283	903	287	43
201710	3524	993	291	52
201711	5897	1509	387	70
201712	4377	1160	294	64
201801	5520	1518	416	109
201802	5253	1325	305	69

a) Count of orders based on the no. of payment installments

```
select
payment_installments,
count(order_id) as order_cnt,
round((count( order_id)/sum(count(order_id))over())*100,3) as percent
from
`scaler-sql-374010.Target_project.payments`

group by
payment_installments
order by payment_installments
```

JOB INFORMATION | RESULTS | JSON | EX

Row	payment_installr	order_cnt	percent
1	0	2	0.002
2	1	52546	50.58
3	2	12413	11.949
4	3	10461	10.07
5	4	7098	6.832
6	5	5239	5.043
7	6	3920	3.773
8	7	1626	1.565
9	8	4268	4.108

7. Insights

- There are 27 states and 8011 cities in the geo location table and I noticed the orders from 5812 cities only and there are no orders from 2199 cities.
- There is an increase in orders month over month and there are highest sales happened in 201711. Since data is there is no significant data to conclude the seasonality behavior in data.
- It's clearly showing most of the orders in afternoon and Morning.
- Around 42 %of customers from SP state and 15 states have less than 1% of customers. There is an opportunity to increase the customer base in the states by introducing discounts or some other marketing strategies.
- There is 20% increase in the payments from 2017 to 2018
- SP state has minimum avg price and PB state has maximum avg price.
- RR state orders have maximum avg freight value and SP state has low avg freight value

8. Recommendations

- Most of the orders are placed in the afternoon and morning, so we should make sure that we have IT infra system to manage the transactions load.
- Around 42 %of customers come from one SP state only and 15 states have less than 1% of customers. There is an opportunity to increase the customer base in the states by introducing discounts or some other marketing strategies.
- Need to explore the options to decrease the delivery time for the top 10 states with highest average time to delivery.