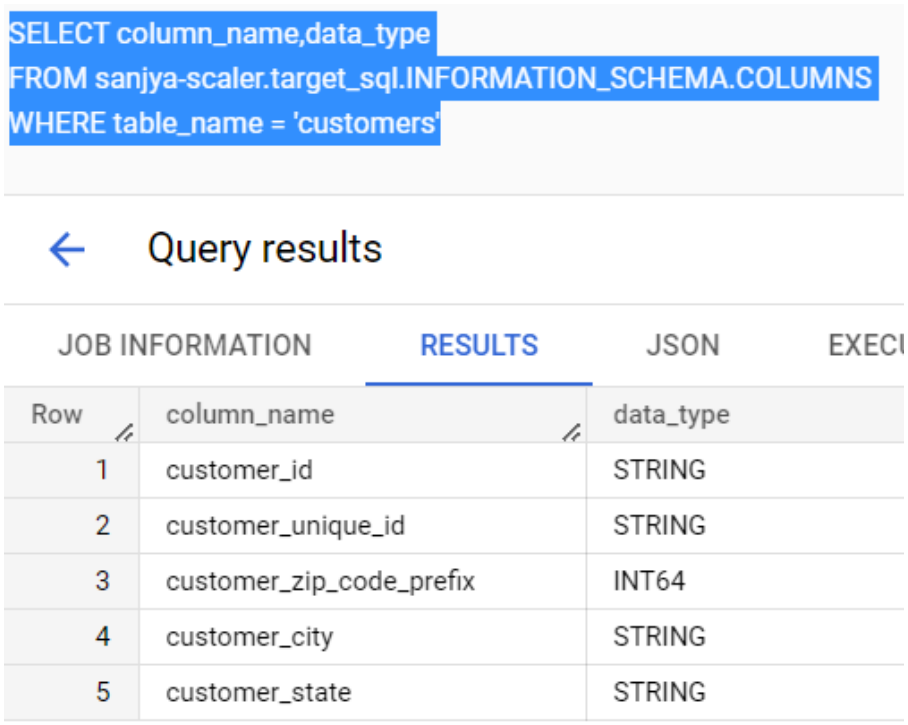
1.Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset

1. **Data type of columns in a table**

   SELECT column_name,data_type FROM sanjya-scaler.target_sql.INFORMATION_SCHEMA.COLUMNS
   WHERE table_name = 'customers'



```
SELECT column_name,data_type
FROM sanjya-scaler.target_sql.INFORMATION_SCHEMA.COLUMNS
WHERE table_name = 'customers'
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECU

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

order_reviews

QUERY SHARE COPY

SCHEMA DETAILS PREVIEW LINEAGE

Filter Enter property name or value

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

geolocation

QUERY SHARE COPY

SCHEMA DETAILS PREVIEW LINEAGE

Filter Enter property name or value

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

order_items QUERY SHARE C

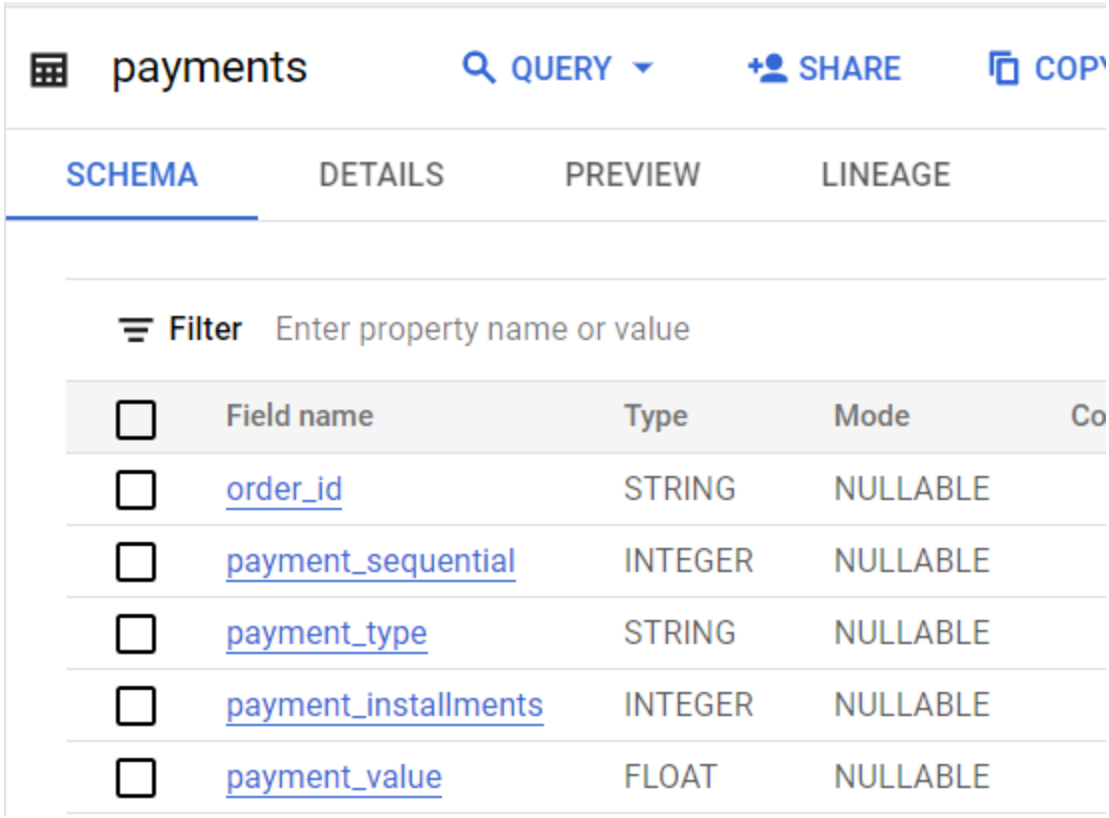
SCHEMA DETAILS PREVIEW LINEAGE

Field name	Type	Mode	C
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	

orders QUERY SHARE COPY SNAP

SCHEMA DETAILS PREVIEW LINEAGE

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
QUERY
SHARE
COPY
SCHEMA
DETAILS
PREVIEW
LINEAGE
Filter
Enter property name or value
Field name
Type
Mode
Co
order_id
STRING
NULLABLE
payment_sequential
INTEGER
NULLABLE
payment_type
STRING
NULLABLE
payment_installments
INTEGER
NULLABLE
payment_value
FLOAT
NULLABLE

products

QUERY SHARE COPY

SCHEMA DETAILS PREVIEW LINEAGE

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

sellers

QUERY SHARE COPY

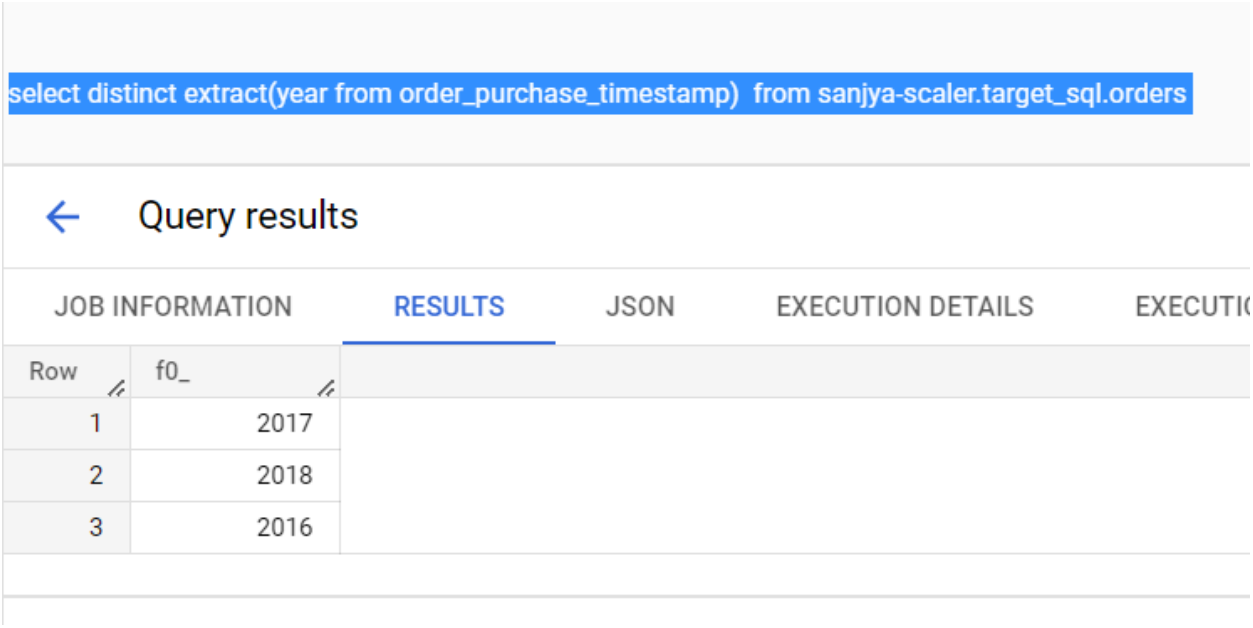
SCHEMA DETAILS PREVIEW LINEAGE

Filter Enter property name or value

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

2.Time period for which the data is given

select distinct extract(year from order_purchase_timestamp) from sanjya-scaler.target_sql.orders



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

select distinct c.customer_id, c.customer_state,c.customer_city from sanjya-scaler.target_sql.customers c left join sanjya-scaler.target_sql.orders o on c.customer_id=o.customer_id where extract(year from o.order_purchase_timestamp) in (2016,2017,2018) order by c.customer_state,c.customer_city

```
select distinct c.customer_id, c.customer_state,c.customer_city from sanjya-scaler.target_sql.customers c left join sanjya-scaler.target_sql.orders o
on c.customer_id=o.customer_id where extract(year from o.order_purchase_timestamp) in (2016,2017,2018) order by c.customer_state,c.customer_city
```

Query results

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

Row	customer_id	customer_state	customer_city
1	b1161707c5b3711b7cf6213c1...	AC	brasileia
2	757bbd8c61a5fd67d5b8c18ef...	AC	cruzeiro do sul
3	f23c4b530f6d7d421de1e38d3...	AC	cruzeiro do sul
4	ee0ab5a9747a11f916fff4b4fc3...	AC	cruzeiro do sul
5	8a45f7d2b87f16a273fd86e9d5...	AC	epitaciolandia
6	013bdb994a9c8f09fde3f5f543...	AC	manoel urbano

2.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?**

   Definitely there was a growing trend from 2016-2018.

   From 2016-2017 there was a huge increase in sales,from 2017-2018 slight increasing in sales

   select * from

   (

   select count(order_id) as order_count,extract(year from order_purchase_timestamp) as year from sanjya-scaler.target_sql.orders group by extract(year from order_purchase_timestamp)

) tbl

   order by year

```
select * from
(
select count(order_id) as order_count,extract(year from order_purchase_timestamp)  as year from sanjya-scaler.target_sql.orders
group by extract(year from order_purchase_timestamp)
) tbl
order by year
```

Query results

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

Row	order_count	year
1	329	2016
2	45101	2017
3	54011	2018

```
with tbl1 as
(
select *,
dense_rank() over(partition by year order by order_count desc) as row_rank
  from
(
select count(order_id) as order_count,extract(year from order_purchase_timestamp) as year,extract(month from order_purchase_timestamp)  as month,
 from sanjya-scaler.target_sql.orders
group by extract(year from order_purchase_timestamp),extract(month from order_purchase_timestamp)
) tbl
)
```

select order_count,year,month from tbl1 where row_rank<=3 order by year asc,order_count ;

JOB INFORMATION | RESULTS | JSON | E

Row	order_count	year	month
1	1	2016	12
2	4	2016	9
3	324	2016	10
4	4631	2017	10
5	5673	2017	12
6	7544	2017	11
7	6939	2018	4
8	7211	2018	3
9	7269	2018	1

From this result we can say that there is a growth in e-commerce starting from October to january.

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

```
select count(order_id) as no_of_orders,
        Daytime
        from
        (select
        order_id,
        extract(hour from order_purchase_timestamp) as hour,
```

```
case when extract(hour from order_purchase_timestamp) between 3 and 10 then 'Morning'
when  extract(hour from order_purchase_timestamp) between 11 and 15 then 'Afternoon'
 when extract(hour from order_purchase_timestamp) between 16 and 19 then 'Dawn'
else 'Night'
end as Daytime
from sanjya-scaler.target_sql.orders
)
group by Daytime
order by no_of_orders desc
```

Query results

JOB INFORMATION | RESULTS | JSON | E

Row	no_of_orders	Daytime
1	32114	Afternoon
2	26423	Night
3	24576	Dawn
4	16328	Morning

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

1. **Get month on month orders by states**

```
select count(order_id) as no_of_orders,customer_state,month from
(
```

```
select o.order_id,c.customer_state,extract(month from order_purchase_timestamp) as month from sanjya-scaler.target_sql.orders o  inner join sanjya-scaler.target_sql.customers c on o.customer_id=c.customer_id

)

group by customer_state, month

order by customer_state,month
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DET

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

The above query gives the number of orders in each month for every state .

```
with tbl as

(

select count(order_id) as no_of_orders,customer_state,month,

dense_rank() over(partition by customer_state order by count(order_id) desc) as high_orders
```

```
 from

(

select o.order_id,c.customer_state,extract(month from order_purchase_timestamp) as month from sanjya-scaler.target_sql.orders o  inner join sanjya-scaler.target_sql.customers c on o.customer_id=c.customer_id

)

group by customer_state, month

order by customer_state,month

)

select no_of_orders,customer_state,month from tbl where high_orders=1 order by customer_state
```

Query results

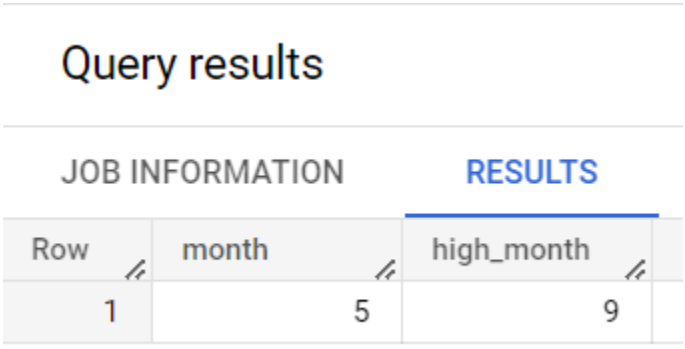
JOB INFORMATION | RESULTS | JSON | EXECUTION DET

Row	no_of_orders	customer_state	month
1	10	AC	5
2	51	AL	4
3	23	AM	7
4	11	AP	1
5	11	AP	5

The above query gives the month with highest number of orders in each state of brazil

```
select month,high_month from

(
```

```
with tbl as

(

select count(order_id) as no_of_orders,customer_state,month,

dense_rank() over(partition by customer_state order by count(order_id) desc) as high_orders

 from

(

select o.order_id,c.customer_state,extract(month from order_purchase_timestamp) as month from sanjya-scaler.target_sql.orders o  inner join sanjya-scaler.target_sql.customers c on o.customer_id=c.customer_id

)

group by customer_state, month

order by customer_state,month

)

select month,count(month) as high_month from tbl where high_orders=1 group by month order by count(month) desc

)  limit 1
```



Query results

JOB INFORMATION | RESULTS

Row	month	high_month
1	5	9

Above one represents the month in which we got the highest number of orders most of the times.

2.Distribution of customers across the states in Brazil

select count(customer_id) no_of_customers,customer_state from sanjya-scaler.target_sql.customers group by customer_state order by count(customer_id) desc

Query results

JOB INFORMATION | RESULTS | JSON | EXECU

Row	no_of_customers	customer_state
1	41746	SP
2	12852	RJ
3	11635	MG
4	5466	RS
5	5045	PR
6	3637	SC
7	3380	BA

4.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**

with tbl as

(

select sum(payment_value) as payment_value,year from

(

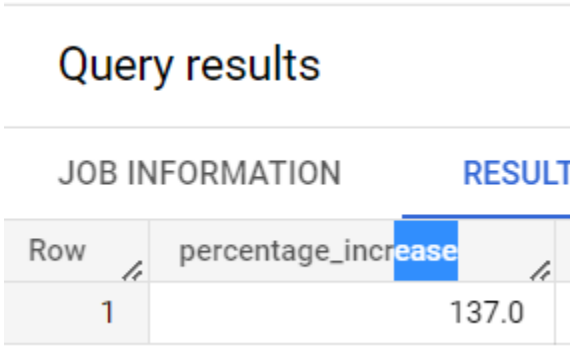
select p.payment_value,extract(year from o.order_purchase_timestamp) as year,extract(month from o.order_purchase_timestamp) as month from sanjya-scaler.target_sql.orders o inner join sanjya-scaler.target_sql.payments p on o.order_id=p.order_id where extract(year from

o.order_purchase_timestamp) in (2017,2018) and extract(month from o.order_purchase_timestamp) between 1 and 8

) group by year

)

select round((((select payment_value from tbl where year=2018)-(select payment_value from tbl where year=2017))/(select payment_value from tbl where year=2017))*100) as percentage_increase



The above result represents the percentage increase from 2017 to 2018

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

```
select c.customer_state,sum(oi.price) as price_sum,sum(oi.freight_value) as
            freight_value_sum,sum(oi.price)/count(oi.price) as price_mean,

sum(oi.freight_value)/count(oi.freight_value) as freight_mean

from sanjya-scaler.target_sql.order_items oi left join sanjya-scaler.target_sql.orders o on
            oi.order_id=o.order_id inner join sanjya-scaler.target_sql.customers c on
            o.customer_id=c.customer_id
```

```
group by c.customer_state order by c.customer_state
```

Row	customer_state	price_sum	freight_value_sum	price_mean	freight_mean
1	AC	15982.9499...	3686.7500000000...	173.727717...	40.0733695...
2	AL	80314.8099...	15914.589999999...	180.889211...	35.8436711...
3	AM	22356.8400...	5478.8900000000...	135.496000...	33.2053939...
4	AP	13474.2999...	2788.5000000000...	164.320731...	34.0060975...
5	BA	511349.990...	100156.67999999...	134.601208...	26.3639589...
6	CE	227254.709...	48351.589999999...	153.758261...	32.7142016...
7	DF	302603.939...	50625.499999999...	125.770548...	21.0413549...

5. Analysis on sales, freight and delivery time

1. Calculate days between purchasing, delivering and estimated delivery

```
select order_id,date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as
delivery_period,date_diff(order_delivered_customer_date,order_estimated_delivery_date,day) as
no_of_days_late from sanjya-scaler.target_sql.orders order by order_id asc
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_id	delivery_period	no_of_days_late
1	00010242fe8c5a6d1ba2dd792...	7	-8
2	00018f77f2f0320c557190d7a1...	16	-2
3	000229ec398224ef6ca0657da...	7	-13
4	00024acbcdf0a6daa1e931b03...	6	-5
5	00042b26cf59d7ce69dfabb4e...	25	-15
6	00048cc3ae777c65dbb7d2a06...	6	-14
7	00054e8431b9d7675808bcb8...	8	-16
8	000576fe39319847cbb9d288c...	5	-15

2.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 order_id,date_diff(order_purchase_timestamp,order_delivered_customer_date,day) as time_to_delivery,

date_diff(order_estimated_delivery_date,order_delivered_customer_date,day) as diff_estimated_delivery

from sanjya-scaler.target_sql.orders order by order_id
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_id	time_to_delivery	diff_estimated_delivery
1	00010242fe8c5a6d1ba2dd792...	-7	8
2	00018f77f2f0320c557190d7a1...	-16	2
3	000229ec398224ef6ca0657da...	-7	13
4	00024acbcdf0a6daa1e931b03...	-6	5
5	00042b26cf59d7ce69dfabb4e...	-25	15
6	00048cc3ae777c65dbb7d2a06...	-6	14

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

```
select c.customer_state,

round(sum(oi.freight_value)/count(oi.freight_value)) as mean_freight_value,

round(sum(date_diff(order_purchase_timestamp,order_delivered_customer_date,day))/count(date_diff(order_pur
chase_timestamp,order_delivered_customer_date,day))) as mean_time_to_delivery,

round(sum(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day))/count(date_diff(orde
r_estimated_delivery_date,order_delivered_customer_date,day))) as mean_diff_estimated_delivery

from sanjya-scaler.target_sql.order_items oi  inner join sanjya-scaler.target_sql.orders o   on
oi.order_id=o.order_id

inner join sanjya-scaler.target_sql.customers c on c.customer_id=o.customer_id

group by c.customer_state

order by c.customer_state asc
```

Query results

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

Row	customer_state	mean_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
1	AC	40.0	-20.0	20.0
2	AL	36.0	-24.0	8.0
3	AM	33.0	-26.0	19.0
4	AP	34.0	-28.0	17.0
5	BA	26.0	-19.0	10.0
6	CE	33.0	-21.0	10.0
7	DF	21.0	-13.0	11.0
8	ES	22.0	-15.0	10.0
9	GO	23.0	-15.0	11.0
10	MA	38.0	-21.0	9.0

Results per pag

Sort the data to get the following:

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

```sql
with tbl as

(

select c.customer_state,

round(sum(oi.freight_value)/count(oi.freight_value)) as mean_freight_value,

round(sum(date_diff(order_purchase_timestamp,order_delivered_customer_date,day))/count(date_diff(order_purcha
se_timestamp,order_delivered_customer_date,day))) as mean_time_to_delivery,

round(sum(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day))/count(date_diff(order_e
stimated_delivery_date,order_delivered_customer_date,day))) as mean_diff_estimated_delivery

from sanjya-scaler.target_sql.order_items oi  inner join sanjya-scaler.target_sql.orders o   on
oi.order_id=o.order_id

inner join sanjya-scaler.target_sql.customers c on c.customer_id=o.customer_id

group by c.customer_state

order by c.customer_state asc

)



select customer_state,mean_freight_value as avg_freight_value from tbl order by avg_freight_value desc limit
5
```

JOB INFORMATION | RESULTS | JSON

Row	customer_state	avg_freight_valu
1	RR	43.0
2	PB	43.0
3	RO	41.0
4	AC	40.0
5	PI	39.0

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

```
with tbl as

(

select c.customer_state,

round(sum(oi.freight_value)/count(oi.freight_value)) as mean_freight_value,

round(sum(date_diff(order_purchase_timestamp,order_delivered_customer_date,day))/count(date_diff(order_purcha
se_timestamp,order_delivered_customer_date,day))) as mean_time_to_delivery,

round(sum(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day))/count(date_diff(order_e
stimated_delivery_date,order_delivered_customer_date,day))) as mean_diff_estimated_delivery

from sanjya-scaler.target_sql.order_items oi  inner join sanjya-scaler.target_sql.orders o   on
oi.order_id=o.order_id

inner join sanjya-scaler.target_sql.customers c on c.customer_id=o.customer_id

group by c.customer_state

order by c.customer_state asc
```

```
)

select customer_state,mean_time_to_delivery as avg_time_to_delivery from tbl order by avg_time_to_delivery
desc limit 5
```

JOB INFORMATION | RESULTS | JSON | EXECUTI

Row	customer_state	avg_time_to_delivery
1	SP	-8.0
2	PR	-11.0
3	MG	-12.0
4	DF	-13.0
5	RS	-15.0

7. states where delivery is really fast/ not so fast compared to estimated date

```
with tbl as

(

select c.customer_state,

round(sum(oi.freight_value)/count(oi.freight_value)) as mean_freight_value,

round(sum(date_diff(order_purchase_timestamp,order_delivered_customer_date,day))/count(date_diff(order_purchase_t
imestamp,order_delivered_customer_date,day))) as mean_time_to_delivery,

round(sum(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day))/count(date_diff(order_estim
ated_delivery_date,order_delivered_customer_date,day))) as mean_diff_estimated_delivery
```

```
from sanjya-scaler.target_sql.order_items oi  inner join sanjya-scaler.target_sql.orders o   on
oi.order_id=o.order_id

inner join sanjya-scaler.target_sql.customers c on c.customer_id=o.customer_id

group by c.customer_state

order by c.customer_state asc

)



select customer_state,mean_diff_estimated_delivery as avg_diff_estimated_delivery from tbl order by
avg_diff_estimated_delivery desc limit 5
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAIL

Row	customer_state	avg_diff_estimated_delivery
1	AC	20.0
2	AM	19.0
3	RO	19.0
4	RR	17.0
5	AP	17.0

6. Payment type analysis:

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

```
with tbl as

(select o.order_id,extract(month from o.order_purchase_timestamp) as month,p.payment_type from
sanjya-scaler.target_sql.orders o inner join sanjya-scaler.target_sql.payments p on o.order_id=p.order_id)

select count(order_id) as no_of_orders,month,payment_type from tbl

group by month,payment_type
```

```
order by month,payment_type
```

JOB INFORMATION | RESULTS | JSON | E

Row	no_of_orders	month	payment_type
1	1715	1	UPI
2	6103	1	credit_card
3	118	1	debit_card
4	477	1	voucher
5	1723	2	UPI
6	6609	2	credit_card
7	82	2	debit_card
8	424	2	voucher

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

```
select count(order_id) as no_of_orders,payment_installments from sanjya-scaler.target_sql.payments group by payment_installments order by count(order_id) desc
```

Row	no_of_orders	payment_installments
1	52546	1
2	12413	2
3	10461	3
4	7098	4
5	5328	10
6	5239	5
7	4268	8
8	3920	6
9	1626	7
10	644	9

7.Actionable Insights

- If we observe there are increase in number of orders every year in the months of october-January
- There would be less customers in the morning and more customers in the afternoon and evening.
- From the 3rd answer,we can say that in the month of “MAY”,we get the highest number of orders most of the times.Because there might be some seasonal vacations.
- In the states SP,RJ and MG we got more customers.
- From question number 4 ,from 2017-2018 between January and August there was 137% increase in cost of orders.This means year by year people are showing interest in online shopping.
- In the states RR,PB,RO,AC and PI,delivery charges are more .
- In the states RR,AP,AC, AM,RO,there are less customers.So where customers are less delivery charges are more and vice versa.
- RR,AP,AC,AM and RO states are having less number of orders.
- SP,PR,MG,DF,RS are states having the highest average time to delivery because these states have more number of customers and therefore more number of orders.
- AC,AM,RO,RR,AP, In these states the products are getting delivered very fast and before the estimated time because these states have less customers and few orders.
- Most of the online payments are made using credit cards.
- Most of the customers are paying the bills in single payments or else in 2,3,4 and 20 installments.

8.Recommendations

- Most of the members are using the credit cards.So if we make a deal with the credit cards on giving an offer on payments with the credit card.we might get more orders.It will be useful to bot the credit card companies and shopping marts.
- Most of the customers visit the shopping malls or order the products in the afternoon and evening.So we should make sure that all the products must have stock at the time and more staff should be there in the marts and malls.
- Some states of Brazil have very few number of orders and high delivery charges.We should try to find out the reason why there are less customers and we should try decreasing the delivery charges or no delivery charge for some period of time to find the pattern and reason of having few customers.
- Some states are having more customers and less delivery charges.Any way the people getting the products very late.We should try to decrease this time because there are chances that the customers may choose other options of buying the products because of late delivery.
- In seasonal vacations, we usually get more orders,in this time we should attract the customers by giving appropriate offers to the customers based on the season,gender and age.
- In the month of october-january will get more customers,so every year by this time we should get ready by new offers,new marketing and innovative ideas.