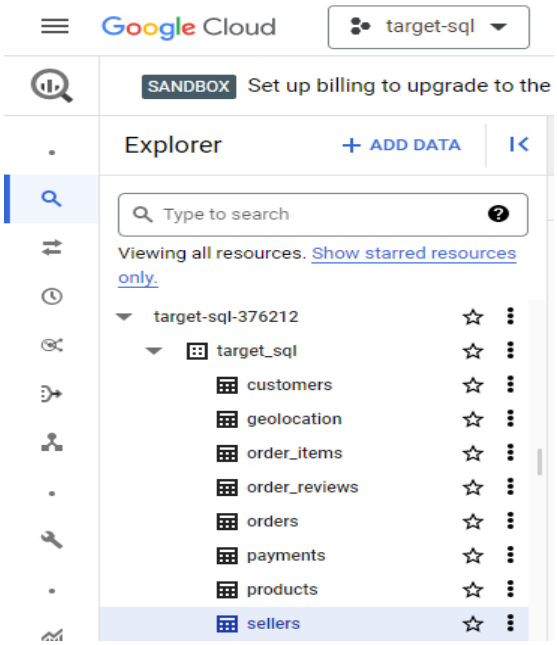
TARGET SQL PROJECT

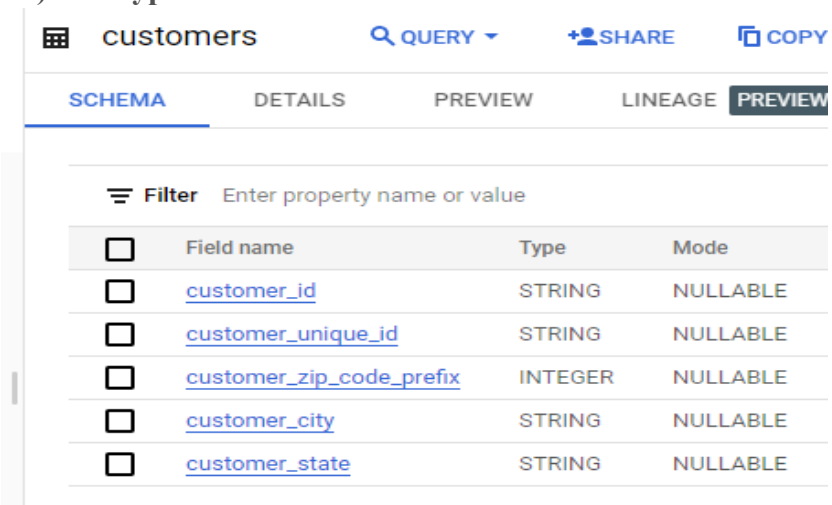
Name : Anusha Jadhav
Batch : DSML AUG 2022 BEGINNER MORNING

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



Dataset imported successfully from the drive.

1.1)Data type of customers columns in a table



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

1.2)Time period for which the data is given

*Unsaved query 2 | *Unsaved query 3

RUN | SAVE | SHARE | SCHEDULE | MORE

```
/*1.2] Time period for which the data is given  */
select first_value(order_purchase_timestamp) over(order by order_purchase_timestamp)as first_date,
        last_value(order_purchase_timestamp) over(order by order_purchase_timestamp desc) as last_date
from   `target_sql.orders`
limit 1
```

Query results | SAVE RESULTS

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

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

1.3).Cities and States of customers ordered during the given period

RUN | SAVE | SHARE | SCHEDULE | MORE

```
/* Cities and States of customers ordered during the given period */
select  customer_city,customer_state
from    `target_sql.customers`
group by customer_city,customer_state
```

Query results

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

ow	customer_city	customer_state
1	acu	RN
2	ico	CE
3	ipe	RS
4	ipu	CE
5	ita	SC
6	itu	SP
7	jau	SP
8	luz	MG
9	poa	SP
10	uba	MG

PERSONAL HISTORY | PROJECT HISTORY

2.In-depth Exploration:

2.1)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?

RUN SAVE SHARE SCHEDULE MORE

```sql
/* Is there a growing trend on e-commerce in Brazil? */
select  extract(month from order_purchase_timestamp)as month,
        extract(year from order_purchase_timestamp)  as year,
        count(*) as no_of_orders
from `target_sql.orders`
group by month,year
order by month,year
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION

Row	month	year	no_of_orders
1	1	2017	800
2	1	2018	7269
3	2	2017	1780
4	2	2018	6728
5	3	2017	2682
6	3	2018	7211
7	4	2017	2404
8	4	2018	6939
9	5	2017	3700
10	5	2018	6873

PERSONAL HISTORY PROJECT HISTORY

Yes , there is a growing trend observed. When we look for seasonality in months irrespective of years, the highest number of orders were placed in the month of august (winter season with dry weather) but there is sharp fall of orders in september(onset of rainy season).

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

RUN SAVE SHARE SCHEDULE MORE

```sql
/*What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)? */
with orders_per_hour as (
    select count(*) as no_of_orders,
           extract(hour from order_purchase_timestamp) as hour
    from    `target_sql.orders`
    group by hour)
select  sum(no_of_orders) as Total_orders,
        case
          when hour between 1 and 6 then 'dawn'
          when hour between 5 and 13 then 'morning'
          when hour between 12 and 19 then 'afternoon'
          else 'night'
          end as Timings
from    orders_per_hour
group by Timings
order by Total_orders
```

Query results

JOB INFORMATION | RESULTS | JSON

Row	Total_orders	Timings
1	2848	dawn
2	24743	night
3	34251	morning
4	37599	afternoon

PERSONAL HISTORY | PROJECT HISTORY

According to the given data, customers tend to purchase more during afternoon(i.e from 12pm to 18pm) and the least purchase is done during the dawn (i.e from 1am to 5am)

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

3.1)Get month on month orders by states

RUN | SAVE | SHARE | SCHEDULE | MORE

```
select  c.customer_state as State,
        extract(month from o.order_purchase_timestamp)as Month,
        count(o.order_id) as Number_of_orders
from    `target_sql.orders`as o
left join `target_sql.customers`as c
on o.customer_id = c.customer_id
group by State,Month
order by State,Month
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DE

Row	State	Month	Number_of_ord
1	AC	1	8
2	AC	2	6
3	AC	3	4
4	AC	4	9
5	AC	5	10
6	AC	6	7
7	AC	7	9
8	AC	8	7
9	AC	9	5
10	AC	10	6
11	AC	11	5
12	AC	12	5
13	AL	1	39

From the above result state: AC has highest no of orders in the month of May.

3.2)Distribution of customers across the states in Brazil

RUN SAVE SHARE SCHEDULE

```
/*Distribution of customers across the states in Brazil*/
select customer_state as Customer_State,
       customer_city as Customer_City,
       count(customer_id) as number_of_customers
from `target_sql.customers`
group by Customer_State, Customer_City
order by number_of_customers desc
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTI

Row	Customer_State	Customer_City	number_of_cust
1	SP	sao paulo	15540
2	RJ	rio de janeiro	6882
3	MG	belo horizonte	2773
4	DF	brasilia	2131
5	PR	curitiba	1521
6	SP	campinas	1444
7	RS	porto alegre	1379
8	BA	salvador	1245
9	SP	guarulhos	1189
10	SP	sao bernardo do campo	938
11	RJ	niteroi	849
12	SP	santo andre	796
13	SP	osasco	746

Total records are 4310 which cannot be plotted on a graph.Instead distribution of customers by each state is done below

RUN SAVE SHARE SCHEDULE

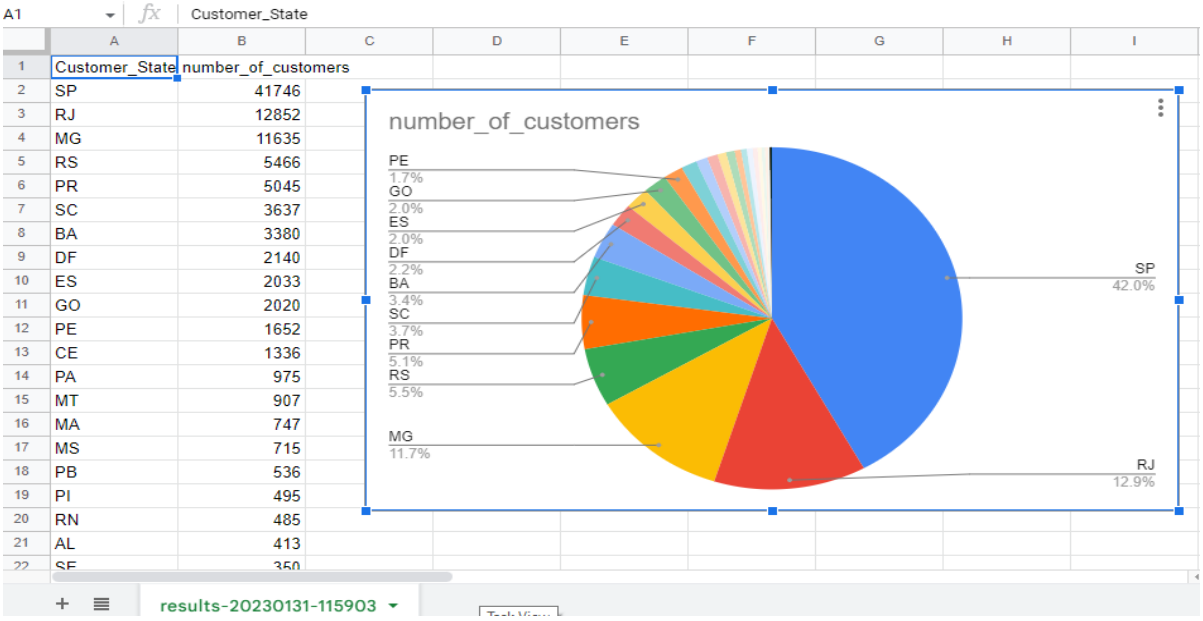
```
select customer_state as Customer_State,
       count(customer_id) as number_of_customers
from `target_sql.customers`
group by Customer_State
order by number_of_customers desc
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECU

Row	Customer_State	number_of_cust
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

Now here we have only 27 records which is plotted in pie chart. The States in order: SP>RJ>MG> …..>RR has the distribution of no of customers across Brazil



Customer_State	number_of_customers
SP	41746
RJ	12852
MG	11635
RS	5466
PR	5045
SC	3637
BA	3380
DF	2140
ES	2033
GO	2020
PE	1652
CE	1336
PA	975
MT	907
MA	747
MS	715
PB	536
PI	495
RN	485
AL	413
SE	350

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

4.1) 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

RUN SAVE SHARE SCHEDULE MORE

```
with p1_2017 as
( select    payment_value,
            extract (year from order_purchase_timestamp) as Year,
            extract (month from order_purchase_timestamp) as Month
  from    `target_sql.orders`as o
  join     `target_sql.payments` as p
  on o.order_id = p.order_id
  where     extract (year from order_purchase_timestamp) = 2017 and
            extract (month from order_purchase_timestamp) between 1 and 8
),
p2_2018 as
( select    payment_value,
            extract (year from order_purchase_timestamp) as Year,
            extract (month from order_purchase_timestamp) as Month
  from    `target_sql.orders`as o
  join     `target_sql.payments` as p
  on o.order_id = p.order_id
  where     extract (year from order_purchase_timestamp) = 2018 and
            extract (month from order_purchase_timestamp) between 1 and 8
)
select  round(((sum(p2.payment_value) - sum(p1.payment_value))/ sum(p1.payment_value))*100,2) as
        percentage_of_cost_increase_from_2017_to_2018
from    p1_2017 as p1
join    p2_2018 as p2
on p1.Month = p2.Month
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION D

Row	percentage_of_cost_increase_from_2017_to_2018
1	4.21

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

RUN | SAVE | SHARE | SCHEDULE | MORE

```
/*Mean & Sum of price and freight value by customer state*/
select  c.customer_state,
        avg(oi.price) as Mean_Price,
        sum(oi.price) as Total_Price,
        avg(oi.freight_value) as Mean_freight_value,
        sum(oi.freight_value) as Total_freight_value,
        avg(oi.price + oi.freight_value) as Mean_Cost_of_order,
        sum(oi.price + oi.freight_value) as Total_Cost_of_order
from    `target_sql.order_items`as oi
join    `target_sql.orders`as o
on oi.order_id = o.order_id
join     `target_sql.customers`as c
on o.customer_id = c.customer_id
group by c.customer_state
```

Query results

SAVE RESULTS | EXPL

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

Row	customer_state	Mean_Price	Total_Price	Mean_freight_va	Total_freight_val	Mean_Cost_of_c	Total_Cost_of_
1	SP	109.653629...	5202955.05...	15.1472753...	718723.069...	124.800904...	5921678.11...
2	RJ	125.117818...	1824092.66...	20.9609239...	305589.310...	146.078742...	2129681.97...
3	PR	119.004139...	683083.760...	20.5316515...	117851.680...	139.535790...	800935.440...
4	SC	124.653577...	520553.340...	21.4703687...	89660.2600...	146.123946...	610213.600...
5	DF	125.770548...	302603.939...	21.0413549...	50625.4999...	146.811903...	353229.440...
6	MG	120.748574...	1585308.02...	20.6301668...	270853.460...	141.378740...	1856161.48...
7	PA	165.692416...	178947.809...	35.8326851...	38699.3000...	201.525101...	217647.109...
8	BA	134.601208...	511349.990...	26.3639589...	100156.679...	160.965167...	611506.670...
9	GO	126.271731...	294591.949...	22.7668152...	53114.9799...	149.038546...	347706.930...
10	RS	120.337453...	750304.020...	21.7358043...	135522.740...	142.073257...	885826.760...

State :PB has the highest Mean-Price, Mean-Freight-value, Mean-Cost-of-order and State:SP Being the lowest of all

5.Analysis on sales, freight and delivery time

5.1)Calculate days between purchasing, delivering and estimated delivery

RUN SAVE SHARE SCHEDULE MORE

```
/*Calculate days between purchasing, delivering and estimated delivery*/
select  order_id,
        date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as btw_purchase_to_delivery_date
        date_diff(order_delivered_customer_date,order_estimated_delivery_date,day) as delay_in_delivery,
        date_diff(order_estimated_delivery_date,order_purchase_timestamp,day) as btw_purchase_to_estimated
from    `target_sql.orders`
order by btw_purchase_to_delivery_date desc,delay_in_delivery desc,btw_purchase_to_estimated
```

Query results SAVE RE

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW

Row	order_id	btw_purchase_to_delivery_date	delay_in_delivery	btw_purchase_to_estimated
1	ca07593549f1816d26a572e06...	209	181	28
2	1b3190b2dfa9d789e1f14c05b...	208	188	19
3	440d0d17af552815d15a9e41a...	195	165	30
4	285ab9426d6982034523a855f...	194	166	28
5	0f4519c5f1c541ddec9f21b3bd...	194	161	32
6	2fb597c2f772eca01b1f5c561b...	194	155	39
7	47b40429ed8cce3aee9199792...	191	175	15
8	2fe324febf907e3ea3f2aa9650...	189	167	22
9	2d7561026d542c8dbd8f0daea...	188	159	28
10	c27815f7e3dd0b926b5855262...	187	162	25

For above there are 99441 orders done in the dataset .

5.2)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

RUN SAVE SHARE SCHEDULE MORE This query will p

```
select  order_id,
        date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as time_to_delivery,
        date_diff(order_delivered_customer_date,order_estimated_delivery_date,day) as diff_estimated_delivery
from    `target_sql.orders`
order by time_to_delivery desc,diff_estimated_delivery
```

Press Al

Query results SAVE RESULTS EXI

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

Row	order_id	time_to_delivery	diff_estimated_delivery
1	ca07593549f1816d26a572e06...	209	181
2	1b3190b2dfa9d789e1f14c05b...	208	188
3	440d0d17af552815d15a9e41a...	195	165
4	2fb597c2f772eca01b1f5c561b...	194	155
5	0f4519c5f1c541ddec9f21b3bd...	194	161
6	285ab9426d6982034523a855f...	194	166
7	47b40429ed8cce3aee9199792...	191	175
8	2fe324febf907e3ea3f2aa9650...	189	167
9	2d7561026d542c8dbd8f0daea...	188	159
10	437222e3fd1b07396f1d9ba8c...	187	144

Clearly from the first row : estimated_delivery says 181 days,but the item got delivered 28 days later on 209th day of the purchase.

5.3)Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

*Unsaved query14 target_sql

RUN SAVE SHARE SCHEDULE MORE Query compl

```
/* Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery  */
select  customer_state,
        round(avg(freight_value),2) as Mean_freight_value,
        round(avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day))) as Mean_time_to_delivery,
        round(avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day))) as Mean_diff_estimated_delivery
from    `target_sql.order_items` as oi
join `target_sql.orders`  as o
on    oi.order_id = o.order_id
join `target_sql.customers` as c
```

Query results SAVE RES

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

Row	customer_state	Mean_freight_value	Mean_time_to_delivery	Mean_diff_estimated_delivery
1	SP	15.15	8.0	10.0
2	PR	20.53	11.0	13.0
3	MG	20.63	12.0	12.0
4	RJ	20.96	15.0	11.0
5	DF	21.04	13.0	11.0
6	SC	21.47	15.0	11.0
7	RS	21.74	15.0	13.0
8	ES	22.06	15.0	10.0
9	GO	22.77	15.0	11.0
10	MS	23.37	15.0	10.0

5.4 Sort the data to get the following:

a)Top 5 states with lowest average freight value - sort in asc limit 5

```
/*Top 5 states with lowest average freight value - sort in desc/asc limit 5
select  customer_state,
        round(avg(freight_value),2) as Mean_freight_value,
from    `target_sql.order_items` as oi
join `target_sql.orders`  as o
on    oi.order_id = o.order_id
join `target_sql.customers` as c
on    o.customer_id = c.customer_id
group by c.customer_state
order by Mean_freight_value
limit 5
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTI

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

b)Top 5 states with highest average time to delivery

RUN | SAVE | SHARE | SCHEDULE | MORE

```
select  customer_state,
        round(avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day))) as Mean_time_to_delivery
from    `target_sql.order_items` as oi
join `target_sql.orders`  as o
on    oi.order_id = o.order_id
join `target_sql.customers` as c
on    o.customer_id = c.customer_id
group by c.customer_state
order by Mean_time_to_delivery desc
limit 5
```

Press Alt+F1 f

Query results | SAVE RESULTS | EXPLOR

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

Row	customer_state	Mean_time_to_delivery
1	AP	28.0
2	RR	28.0
3	AM	26.0
4	AL	24.0
5	PA	23.0

Mean-Time-Delivery = AVG(order_delivered_customer_date - order_purchase_timestamp)

c)Top 5 states where delivery is really fast/ not so fast compared to estimated date

```
/*Top 5 states where delivery is really fast/ not so fast compared to estimated date*/
select  customer_state,
round(avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day))) as Mean_diff_estimated_delivery
from    `target_sql.order_items` as oi
join `target_sql.orders`  as o
on    oi.order_id = o.order_id
join `target_sql.customers` as c
on    o.customer_id = c.customer_id
group by c.customer_state
order by Mean_diff_estimated_delivery
limit 5
```

Press Alt+F1 for ac

Query results

SAVE RESULTS EXPLORE DAT

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

ow	customer_state	Mean_diff_estim
1	AL	8.0
2	SE	9.0
3	MA	9.0
4	SP	10.0
5	BA	10.0

6.Payment type analysis:

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

RUN SAVE SHARE SCHEDULE MORE

```
/* Month over Month count of orders for different payment types */
select  extract(month from order_purchase_timestamp) as Month,
        payment_type,
        count(o.order_id) as no_of_orders
from    `target_sql.orders` as o
join   `target_sql.payments` as p
on o.order_id = p.order_id
group by Month,payment_type
order by Month,no_of_orders
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DET

Row	Month	payment_type	no_of_orders
1	1	debit_card	118
2	1	voucher	477
3	1	UPI	1715
4	1	credit_card	6103
5	2	debit_card	82
6	2	voucher	424
7	2	UPI	1723
8	2	credit_card	6609
9	3	debit_card	109
10	3	voucher	591

Payments done by credit cards is more as compared to other payment_type

6.2)Count of orders based on the no. of payment instalments

RUN | SAVE | SHARE | SCHEDULE | MOR

```
/* Count of orders based on the no. of payment installments */
select payment_installments, count(order_id) as no_of_orders
from   `target_sql.payments`
group by payment_installments
order by payment_installments, no_of_orders
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

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

7. Actionable Insights:

- The given time period for the Dataset is in between 09-sept-2016 and 17-oct-2018
- There is consistent increase in number of orders from 2016 to 2018 as shown in the graph below:

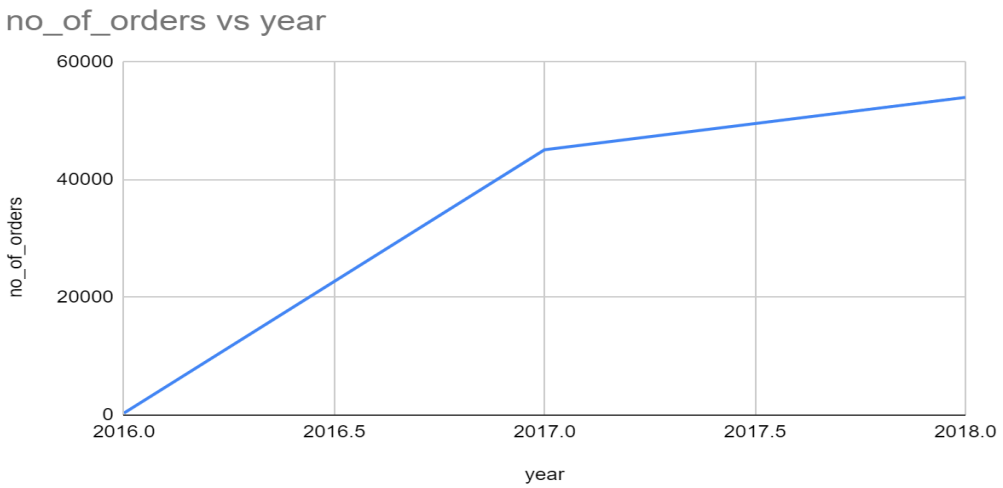
RUN SAVE SHARE SCHEDULE MO

```
select
        extract(year from order_purchase_timestamp)  as year,
        count(*) as no_of_orders
from `target_sql.orders`
group by year
order by year
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

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



-
- The issue of delay in delivery of the orders has to be looked into inorder to maximise the purchase and ensure brand trust for the customers. It can done by:
    - Introducing more of number of warehouses in order to reduce scarcity of products by looking into geolocations table for every state
    - How to speed up deliveries during extreme weather conditions
    - Having a sufficient delivery network system becomes very important as customers tend to purchase more during holidays.
- Number of orders purchased via credit cards is more in the given dataset and then comes UPI transaction

RUN SAVE SHARE SCHEDUL

```
select  payment_type,
        count(o.order_id) as no_of_orders
from    `target_sql.orders` as o
join    `target_sql.payments` as p
on o.order_id = p.order_id
group by payment_type
order by no_of_orders desc
```

Query results

JOB INFORMATION | RESULTS | JSON | EXEC

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

- Top three product category with highest cost price ordered by customers

RUN SAVE SHARE SCHEDULE MORE

```
select distinct p.product_id,o.order_id,product_category ,(price + freight_value)as total_price
from `target_sql.orders`as o
join `target_sql.order_items` as oi
on o.order_id = oi.order_id
join `target_sql.products` as p
on p.product_id = oi.product_id
group by o.order_id,p.product_id,product_category, total_price
order by total_price desc,p.product_id
limit 3
```

Query results SAVE RESULTS

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

Row	product_id	order_id	product_category	total_price
1	489ae2aa008f021502940f251...	0812eb902a67711a1cb742b3c...	housewares	6929.31
2	69c590f7ffc7bf8db97190b6cb...	fefacc66af859508bf1a7934ea...	PCs	6922.21
3	1bdf5e6731585cf01aa8169c7...	f5136e38d1a14a4dbd87dff67d...	Art	6726.66

Highest purchased category is housewares and health-beauty being the cheapest category purchased.

8.Recommendations:

- As we know that payments done by credit cards are way more than debit cards ,hence in order to engage more customers who only own debit cards, a high value of discounts have to be given.
- Tracking each customer's purchases will help the target to recommend product_category items they are interested in.
- Customer table should include age and gender for better understanding of interests of customer
- Review_score from reviews table can be used to eliminate items with review_score=1 given by more than 1000 customers