Project- Target SQL

Submitted by Mr Marshal Harsh Mathew

Data set- Target , Brazil

Synopsis

Data analysis based on the data set from Target to derive valuable insights on the sale, customer behavior etc. Based on the data we are trying to find the trends in different aspects by using Bigquery.

This project is divided into different categories and in each category different queries are used to get insights

**Category 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

Query

1. select
2. \*
3. from `Bigquery\_project\_marsh050223`.INFORMATION\_SCHEMA.COLUMNS;

Result

|  |  |  |  |
| --- | --- | --- | --- |
| table\_schema | table\_name | column\_name | ordinal\_position |
| Bigquery\_project\_marsh050223 | customers | customer\_id | 1 |
| Bigquery\_project\_marsh050223 | customers | customer\_unique\_id | 2 |
| Bigquery\_project\_marsh050223 | customers | customer\_zip\_code\_prefix | 3 |
| Bigquery\_project\_marsh050223 | customers | customer\_city | 4 |
| Bigquery\_project\_marsh050223 | customers | customer\_state | 5 |
| Bigquery\_project\_marsh050223 | geolocations | geolocation\_zip\_code\_prefix | 1 |
| Bigquery\_project\_marsh050223 | geolocations | geolocation\_lat | 2 |
| Bigquery\_project\_marsh050223 | geolocations | geolocation\_lng | 3 |
| Bigquery\_project\_marsh050223 | geolocations | geolocation\_city | 4 |
| Bigquery\_project\_marsh050223 | geolocations | geolocation\_state | 5 |
| Bigquery\_project\_marsh050223 | order\_items | order\_id | 1 |
| Bigquery\_project\_marsh050223 | order\_items | order\_item\_id | 2 |
| Bigquery\_project\_marsh050223 | order\_items | product\_id | 3 |
| Bigquery\_project\_marsh050223 | order\_items | seller\_id | 4 |
| Bigquery\_project\_marsh050223 | order\_items | shipping\_limit\_date | 5 |
| Bigquery\_project\_marsh050223 | order\_items | price | 6 |
| Bigquery\_project\_marsh050223 | order\_items | freight\_value | 7 |
| Bigquery\_project\_marsh050223 | order\_reviews | review\_id | 1 |
| Bigquery\_project\_marsh050223 | order\_reviews | order\_id | 2 |
| Bigquery\_project\_marsh050223 | order\_reviews | review\_score | 3 |
| Bigquery\_project\_marsh050223 | order\_reviews | review\_comment\_title | 4 |
| Bigquery\_project\_marsh050223 | order\_reviews | review\_creation\_date | 5 |
| Bigquery\_project\_marsh050223 | order\_reviews | review\_answer\_timestamp | 6 |
| Bigquery\_project\_marsh050223 | Orders | order\_id | 1 |
| Bigquery\_project\_marsh050223 | Orders | customer\_id | 2 |
| Bigquery\_project\_marsh050223 | Orders | order\_status | 3 |
| Bigquery\_project\_marsh050223 | Orders | order\_purchase\_timestamp | 4 |
| Bigquery\_project\_marsh050223 | Orders | order\_approved\_at | 5 |
| Bigquery\_project\_marsh050223 | Orders | order\_delivered\_carrier\_date | 6 |
| Bigquery\_project\_marsh050223 | Orders | order\_delivered\_customer\_date | 7 |
| Bigquery\_project\_marsh050223 | Orders | order\_estimated\_delivery\_date | 8 |
| Bigquery\_project\_marsh050223 | payments | order\_id | 1 |
| Bigquery\_project\_marsh050223 | payments | payment\_sequential | 2 |
| Bigquery\_project\_marsh050223 | payments | payment\_type | 3 |
| Bigquery\_project\_marsh050223 | payments | payment\_installments | 4 |
| Bigquery\_project\_marsh050223 | payments | payment\_value | 5 |
| Bigquery\_project\_marsh050223 | products | product\_id | 1 |
| Bigquery\_project\_marsh050223 | products | product\_category | 2 |
| Bigquery\_project\_marsh050223 | products | product\_name\_length | 3 |
| Bigquery\_project\_marsh050223 | products | product\_description\_length | 4 |
| Bigquery\_project\_marsh050223 | products | product\_photos\_qty | 5 |
| Bigquery\_project\_marsh050223 | products | product\_weight\_g | 6 |
| Bigquery\_project\_marsh050223 | products | product\_length\_cm | 7 |
| Bigquery\_project\_marsh050223 | products | product\_height\_cm | 8 |
| Bigquery\_project\_marsh050223 | products | product\_width\_cm | 9 |
| Bigquery\_project\_marsh050223 | Sellers | seller\_id | 1 |
| Bigquery\_project\_marsh050223 | Sellers | seller\_zip\_code\_prefix | 2 |
| Bigquery\_project\_marsh050223 | Sellers | seller\_city | 3 |
| Bigquery\_project\_marsh050223 | Sellers | seller\_state | 4 |

Insight

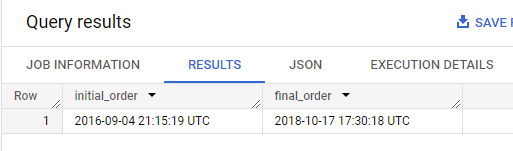
The data is stored in different columns and the primary keys like order\_id, customer\_id, product\_id, seller\_id, state\_id will help in joining tables to get the insigts

2.Time period of orders

**Query**

1. select
2. min(order\_purchase\_timestamp) as initial\_order,
3. max(order\_purchase\_timestamp) as final\_order
4. from`Bigquery\_project\_marsh050223.orders`;

Result

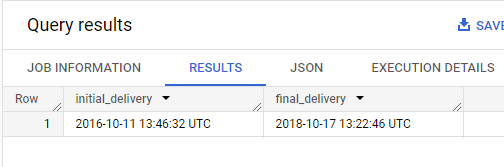


3.Time period of order delivery to customer

Query

1. select
2. min(order\_delivered\_customer\_date) as initial\_delivery,
3. max(order\_delivered\_customer\_date) as final\_delivery
4. from`Bigquery\_project\_marsh050223.orders`;

Result



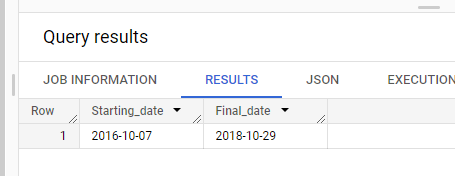
Time period of reviews

The data table was wrong in capturing the date .The year, date and month columns got interchanged which is corrected in the below query

**Query**

1. select
2. min(date\_act) as Starting\_date,
3. max(date\_act) as Final\_date
4. from(
5. select
6. date((t.day\_new),t.month\_new,t.year\_new) as date\_act
7. from
8. (
9. select
10. extract(day from review\_answer\_timestamp)+2000 as day\_new,
11. extract(month from review\_answer\_timestamp) as month\_new,
12. extract(year from review\_answer\_timestamp) as year\_new
13. from `Bigquery\_project\_marsh050223.order\_reviews`
14. order by year\_new desc)t
15. )

Result



Cities and States of customers ordered during the given period

Query

select

customer\_city as City,

customer\_state as State

from `Bigquery\_project\_marsh050223.customers`

group by customer\_city,customer\_state

order by customer\_city;

Result



Insights

Time periods are slights different in tables but the data is mainly from 2016 September to 2018 October.

**Category 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?

A.The growth of e commerce in the given period of time ( year on year)

Query

select

distinct t.Year\_of\_Sale,

round(sum(t.price) over(partition by t.Year\_of\_Sale)) as Total\_revenue\_in\_USD,

round(sum(t.Price\_with\_shipping) over(partition by t.Year\_of\_Sale)) as Total\_revenue\_with\_shipping\_cost\_in\_USD

from

(select

\*,

extract(year from od.order\_purchase\_timestamp) as Year\_of\_Sale,

extract(month from od.order\_purchase\_timestamp) as Month\_of\_Sale,

ore.price+ore.freight\_value as Price\_with\_shipping

from `Bigquery\_project\_marsh050223.orders` od left join `Bigquery\_project\_marsh050223.order\_items` ore

                                                          on od.order\_id=ore.order\_id

                                              left join`Bigquery\_project\_marsh050223.customers` cr  on od.

                                                          customer\_id=cr.customer\_id

)t

order by t.Year\_of\_Sale;

Result



Insight

The sale is increasing year by year as per the given data set. The e commerce is booming as per the data, so more customers will be opting for online sale. We need to forecast the demand and more sellers need to be introduced to the e commerce business.

B. Cumulative Sale- MOM

Query

select

distinct t.Year\_of\_Sale,

t.Month\_of\_Sale,

round(sum(t.price) over(partition by t.Year\_of\_Sale order by t.Month\_of\_Sale range between unbounded preceding and current row)) as Cumulative\_Sale

from

(select

\*,

extract(year from od.order\_purchase\_timestamp) as Year\_of\_Sale,

extract(month from od.order\_purchase\_timestamp) as Month\_of\_Sale,

ore.price+ore.freight\_value as Price\_with\_shipping,

extract(time from od.order\_purchase\_timestamp) as Order\_time

from `Bigquery\_project\_marsh050223.orders` od left join `Bigquery\_project\_marsh050223.order\_items` ore

                                                          on od.order\_id=ore.order\_id

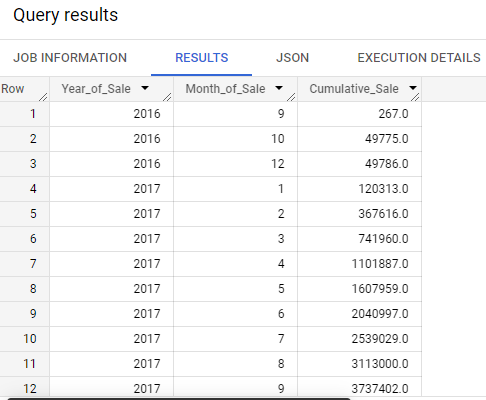
                                              left join`Bigquery\_project\_marsh050223.customers` cr  on od.

                                                          customer\_id=cr.customer\_id

)t

order by t.Year\_of\_Sale,t.Month\_of\_Sale;

Result



C. Peak months of Sale

Query

select

\*

from

(

select

distinct t.Year\_of\_Sale,

t.Month\_of\_Sale,

round(sum(t.price) over(partition by t.Year\_of\_Sale,T.Month\_of\_Sale)) as Sum\_price

from

(select

\*,

extract(year from od.order\_purchase\_timestamp) as Year\_of\_Sale,

extract(month from od.order\_purchase\_timestamp) as Month\_of\_Sale,

ore.price+ore.freight\_value as Price\_with\_shipping,

extract(time from od.order\_purchase\_timestamp) as Order\_time

from `Bigquery\_project\_marsh050223.orders` od left join `Bigquery\_project\_marsh050223.order\_items` ore

                                                          on od.order\_id=ore.order\_id

                                              left join`Bigquery\_project\_marsh050223.customers` cr  on od.

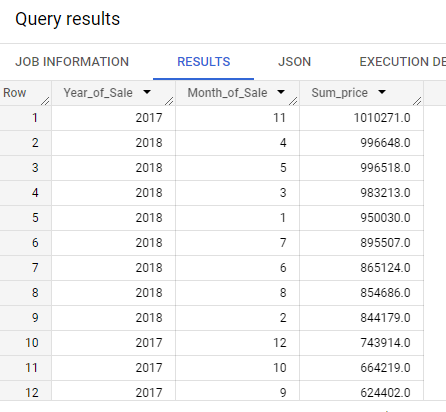
                                                          customer\_id=cr.customer\_id

)t

)

order by Sum\_price desc

Result



The Peak month is 2017 November, thanks giving festival and starting of summer in Brazil might be the reason of increase in Sale.

**Category 3**

Evolution of E-commerce orders in the Brazil region:

1.Get month on month orders by states

Query

select

distinct t.customer\_state as State,

t.Year\_of\_Sale as Year,

t.Month\_of\_Sale as Month,

count(distinct t.order\_id) over(partition by t.Year\_of\_Sale,t.Month\_of\_Sale,t.customer\_state) as MOM\_orders

from

(

select

od.order\_id,

cr.customer\_unique\_id,

cr.customer\_state,

extract(year from od.order\_purchase\_timestamp) as Year\_of\_Sale,

extract(month from od.order\_purchase\_timestamp) as Month\_of\_Sale,

ore.price+ore.freight\_value as Price\_with\_shipping,

extract(time from od.order\_purchase\_timestamp) as Order\_time,

od.order\_purchase\_timestamp,

min(od.order\_purchase\_timestamp)over (partition by cr.customer\_unique\_id order by od.order\_purchase\_timestamp rows between unbounded preceding and unbounded following) as First\_order\_date

from `Bigquery\_project\_marsh050223.orders` od left join `Bigquery\_project\_marsh050223.order\_items` ore

                                                          on od.order\_id=ore.order\_id

                                              left join`Bigquery\_project\_marsh050223.customers` cr  on od.

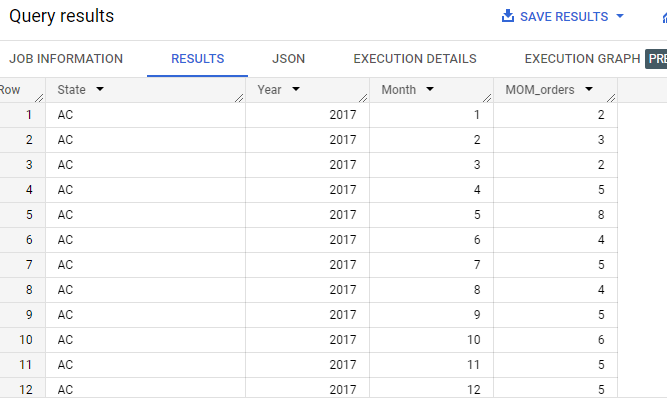
                                                          customer\_id=cr.customer\_id

where lower(od.order\_status)!='canceled'

)t

order by 1,2,3;

Result



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

Query

select

t.Purchase\_time\_Brazil\_std\_time,

count(distinct t.order\_id) as Orders\_placed

from(

select

order\_id,

order\_purchase\_timestamp,

case when time\_sub(TIME (extract(time from order\_purchase\_timestamp)), interval 3 HOUR) between '03:00:00' and '07:00:00' then'Dawn '

when time\_sub(TIME (extract(time from order\_purchase\_timestamp)), interval 3 HOUR) between '07:00:01' and '11:59:59' then 'Morning'

when time\_sub(TIME (extract(time from order\_purchase\_timestamp)), interval 3 HOUR) between '12:00:00' and '18:00:00' then 'After\_noon'

else 'Night'

end as Purchase\_time\_Brazil\_std\_time

from `Bigquery\_project\_marsh050223.orders`

where lower(order\_status)!="canceled"

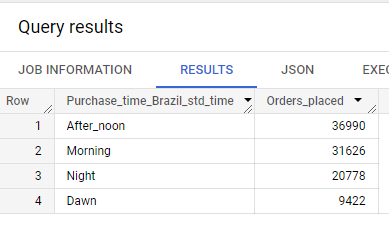
order by order\_purchase\_timestamp desc

)t

group by t.Purchase\_time\_Brazil\_std\_time

order by count(distinct order\_id) desc;

Result



Insight

We have calculated the orders based on the Brazilia timing which is 3 hours behind UTC.The customers are placing orders mainly during afternoon and morning. So during these timing more adds need to played on the app/ or other available screens which may push the sale up.

**Category 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

Query

SELECT

round(((Next\_year-Cost\_of\_Order) \*100/cost\_of\_order)) as Percentage\_of\_increase\_in\_sale

from

(

select

Year,

Cost\_of\_Order,

lead(Cost\_of\_order)over(order by Year) as Next\_Year

from

(

select

distinct t.Year,

round(sum(t.payment\_value)) as Cost\_of\_order

from

(

select

p.order\_id,

o.order\_status,

extract(year from o.order\_purchase\_timestamp) as Year,

extract(month from o.order\_purchase\_timestamp) as Month,

p.payment\_value

from `Bigquery\_project\_marsh050223.payments` p  left join `Bigquery\_project\_marsh050223.orders` o

                                                      on p.order\_id=o.order\_id

)t

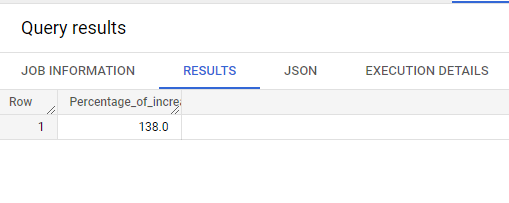
where t.order\_status!='canceled' and t.Year in (2017,2018) and t.Month between 1 and 8

group by t.year

))

limit 1;

Result



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

Query

select

distinct t.customer\_state as State,

round(avg(t.price)over(partition by t.customer\_state)) as Mean\_price,

round(avg(t.freight\_value)over(partition by t.customer\_state)) as Mean\_freight,

round(sum(t.price)over(partition by t.customer\_state)) as Sum\_price,

round(sum(t.freight\_value)over(partition by t.customer\_state)) as Sum\_freight,

from

(

select

distinct od.order\_id,

cr.customer\_state,

sum(ore.freight\_value) as freight\_value,

sum(ore.price) as price,

from `Bigquery\_project\_marsh050223.orders` od left join `Bigquery\_project\_marsh050223.order\_items` ore

                                                          on od.order\_id=ore.order\_id

                                              left join`Bigquery\_project\_marsh050223.customers` cr  on od.

                                                          customer\_id=cr.customer\_id

where lower(od.order\_status)!='canceled'

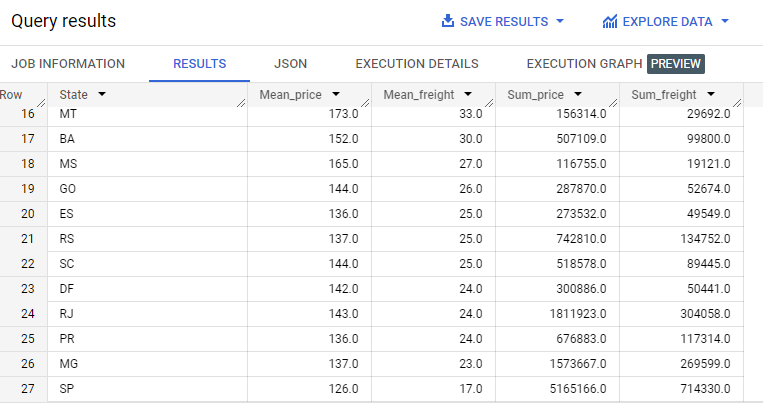
group by od.order\_id, cr.customer\_state

)t

order by Mean\_freight desc;

Result





Insight

The freight cost is more in RR which is the state Roraima and state PB , which is the state Paraiba which are respectively the northernmost and easternmost states of Brazil.The RR state is full of Amzon rain forest and high terrain so transportation cost is more and the state PB is also very far from the main The least cost of freight is at Sao polo, the financial capital of brazil.

When we look at the seller details based on the orders,

Query

select

distinct seller\_id,

count\_products,

seller\_state

from

(

select

count(product\_id) over (partition by seller\_id) as count\_products,

seller\_id,

seller\_state

from

(

select

distinct o.order\_id,

o.product\_id,

o.seller\_id,

s.seller\_state,

from `Bigquery\_project\_marsh050223.order\_items` o inner join`Bigquery\_project\_marsh050223.sellers` s

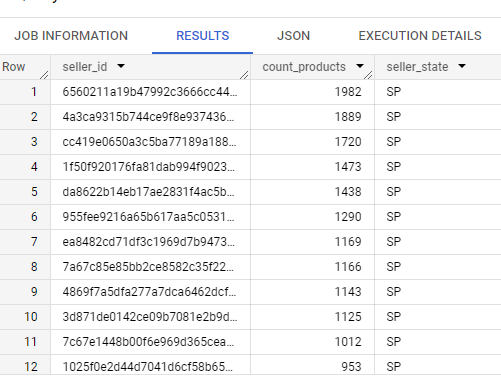
                                          on o.seller\_id=s.seller\_id

)

)

order by count\_products desc

Result



Checking the sellers from state RR and PB

Query

select

distinct seller\_id,

count\_products,

seller\_state

from

(

select

count(product\_id) over (partition by seller\_id) as count\_products,

seller\_id,

seller\_state

from

(

select

distinct o.order\_id,

o.product\_id,

o.seller\_id,

s.seller\_state,

from `Bigquery\_project\_marsh050223.order\_items` o inner join`Bigquery\_project\_marsh050223.sellers` s

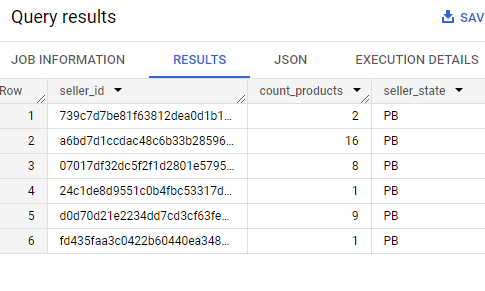
                                          on o.seller\_id=s.seller\_id

)

)

where seller\_state in ('RR','PB')

Result



The top selling sellers are based out of SP , so the average freight cost is lesser. Only six sellers are from PB and there are no sellers from RR which is the reason for the high average freight costs which indicates customer from these states are mainly buying from sellers from other states. So Target should try to find some sellers is RR, PB or nearby states which might bring down the freight cost to these states.

Map of Brazil



**Category 5**

Analysis on sales, freight and delivery time

1.Calculate days between purchasing, delivering and estimated delivery

Query

select

order\_id,

date\_diff(order\_estimated\_delivery\_date,order\_purchase\_timestamp,day) as Diff\_est\_order,

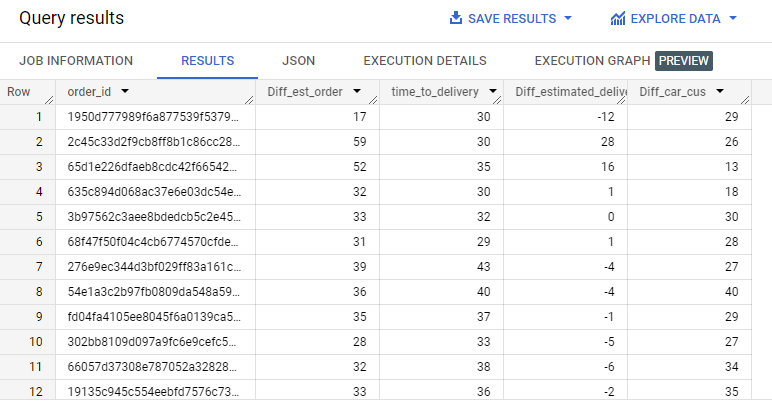
date\_diff(order\_delivered\_customer\_date,order\_purchase\_timestamp,day) as time\_to\_delivery,

date\_diff(order\_estimated\_delivery\_date,order\_delivered\_customer\_date,day) as Diff\_estimated\_delivery,

date\_diff(order\_delivered\_customer\_date,order\_delivered\_carrier\_date,day) as Diff\_car\_cus

from `Bigquery\_project\_marsh050223.orders`

Result



2.Find time\_to\_delivery & diff\_estimated\_delivery. Formula for the same given below:

time\_to\_delivery = order\_delivered\_customer\_date-order\_purchase\_timestamp

diff\_estimated\_delivery = order\_estimated\_delivery\_date-order\_delivered\_customer\_date

Query

select

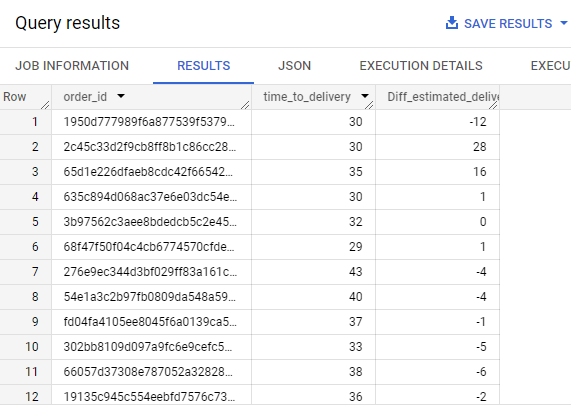
order\_id,

date\_diff(order\_delivered\_customer\_date,order\_purchase\_timestamp,day) as time\_to\_delivery,

date\_diff(order\_estimated\_delivery\_date,order\_delivered\_customer\_date,day) as Diff\_estimated\_delivery,

from `Bigquery\_project\_marsh050223.orders`

Result



Query

select

round(countif(Diff\_estimated\_delivery<0)\*100/count(order\_id),2) as percentage\_delayed\_orders

from

(

select

order\_id,

date\_diff(order\_delivered\_customer\_date,order\_purchase\_timestamp,day) as time\_to\_delivery,

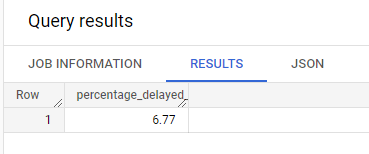
date\_diff(order\_estimated\_delivery\_date,order\_delivered\_customer\_date,day) as Diff\_estimated\_delivery,

from `Bigquery\_project\_marsh050223.orders`

where date\_diff(order\_estimated\_delivery\_date,order\_delivered\_customer\_date,day) is not null

)

Result



Insights

6.77% of the orders got delivered after the estimated time which needs to come down.

3.Group data by state, take mean of freight\_value, time\_to\_delivery, diff\_estimated\_delivery

Query

select

customer\_state,

round(avg(fre\_val),2) as avg\_fr,

round(avg(time\_to\_delivery),2) as avg\_timedel,

round(avg(Diff\_estimated\_delivery),2) as avg\_diff\_estdel,

from

(

select

distinct order\_id,

sum(freight\_value) over (partition by order\_id) as fre\_val,

customer\_state,

time\_to\_delivery,

Diff\_estimated\_delivery

from

(

select

o.order\_id,

t.freight\_value,

c.customer\_state,

date\_diff(o.order\_delivered\_customer\_date,o.order\_purchase\_timestamp,day) as time\_to\_delivery,

date\_diff(o.order\_estimated\_delivery\_date,o.order\_delivered\_customer\_date,day) as Diff\_estimated\_delivery,

from `Bigquery\_project\_marsh050223.orders` o left join `Bigquery\_project\_marsh050223.order\_items` t on

                                                  o.order\_id=t.order\_id

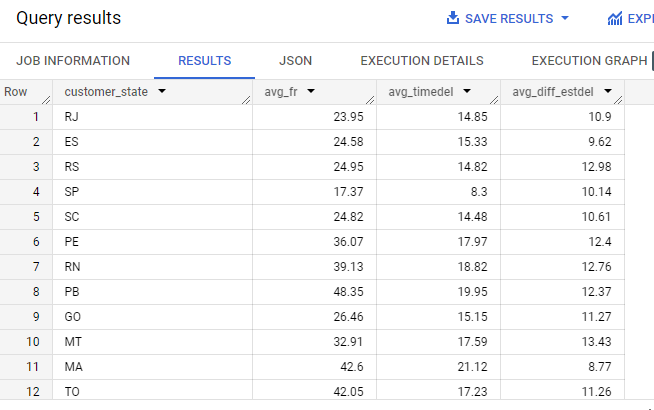
                                              left join `Bigquery\_project\_marsh050223.customers`c on

                                                  o.customer\_id=c.customer\_id

))

group by customer\_state

Result



4. Sort the data to get the following:

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

Query- lowest 5 states

select

customer\_state,

round(avg(fre\_val),2) as avg\_fr,

#round(avg(time\_to\_delivery),2) as avg\_timedel,

#round(avg(Diff\_estimated\_delivery),2) as avg\_diff\_estdel,

from

(

select

distinct order\_id,

sum(freight\_value) over (partition by order\_id) as fre\_val,

customer\_state,

time\_to\_delivery,

Diff\_estimated\_delivery

from

(

select

o.order\_id,

t.freight\_value,

c.customer\_state,

date\_diff(o.order\_delivered\_customer\_date,o.order\_purchase\_timestamp,day) as time\_to\_delivery,

date\_diff(o.order\_estimated\_delivery\_date,o.order\_delivered\_customer\_date,day) as Diff\_estimated\_delivery,

from `Bigquery\_project\_marsh050223.orders` o left join `Bigquery\_project\_marsh050223.order\_items` t on

                                                  o.order\_id=t.order\_id

                                              left join `Bigquery\_project\_marsh050223.customers`c on

                                                  o.customer\_id=c.customer\_id

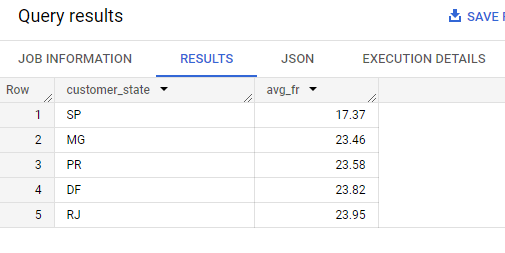
))

group by customer\_state

order by avg(fre\_val)

limit 5;

Result



Query – Top 5 States

select

customer\_state,

round(avg(fre\_val),2) as avg\_fr,

#round(avg(time\_to\_delivery),2) as avg\_timedel,

#round(avg(Diff\_estimated\_delivery),2) as avg\_diff\_estdel,

from

(

select

distinct order\_id,

sum(freight\_value) over (partition by order\_id) as fre\_val,

customer\_state,

time\_to\_delivery,

Diff\_estimated\_delivery

from

(

select

o.order\_id,

t.freight\_value,

c.customer\_state,

date\_diff(o.order\_delivered\_customer\_date,o.order\_purchase\_timestamp,day) as time\_to\_delivery,

date\_diff(o.order\_estimated\_delivery\_date,o.order\_delivered\_customer\_date,day) as Diff\_estimated\_delivery,

from `Bigquery\_project\_marsh050223.orders` o left join `Bigquery\_project\_marsh050223.order\_items` t on

                                                  o.order\_id=t.order\_id

                                              left join `Bigquery\_project\_marsh050223.customers`c on

                                                  o.customer\_id=c.customer\_id

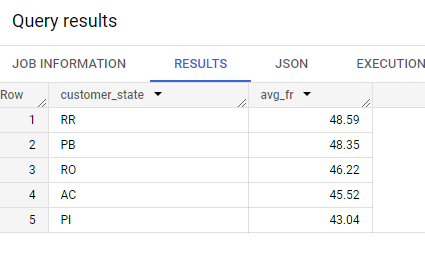
))

group by customer\_state

order by avg(fre\_val) desc

limit 5;

Result



Top 5 states with highest/lowest average time to delivery

Query – lowest five states

select

customer\_state,

#round(avg(fre\_val),2) as avg\_fr,

round(avg(time\_to\_delivery),2) as avg\_timedel,

#round(avg(Diff\_estimated\_delivery),2) as avg\_diff\_estdel,

from

(

select

distinct order\_id,

sum(freight\_value) over (partition by order\_id) as fre\_val,

customer\_state,

time\_to\_delivery,

Diff\_estimated\_delivery

from

(

select

o.order\_id,

t.freight\_value,

c.customer\_state,

date\_diff(o.order\_delivered\_customer\_date,o.order\_purchase\_timestamp,day) as time\_to\_delivery,

date\_diff(o.order\_estimated\_delivery\_date,o.order\_delivered\_customer\_date,day) as Diff\_estimated\_delivery,

from `Bigquery\_project\_marsh050223.orders` o left join `Bigquery\_project\_marsh050223.order\_items` t on

                                                  o.order\_id=t.order\_id

                                              left join `Bigquery\_project\_marsh050223.customers`c on

                                                  o.customer\_id=c.customer\_id

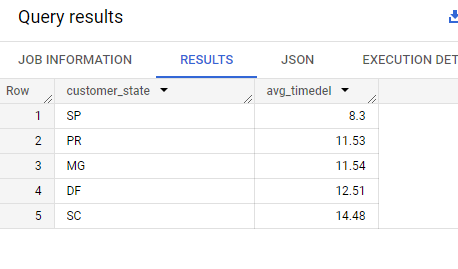
))

group by customer\_state

order by avg(time\_to\_delivery)

limit 5;

Result



Query top five states

select

customer\_state,

#round(avg(fre\_val),2) as avg\_fr,

round(avg(time\_to\_delivery),2) as avg\_timedel,

#round(avg(Diff\_estimated\_delivery),2) as avg\_diff\_estdel,

from

(

select

distinct order\_id,

sum(freight\_value) over (partition by order\_id) as fre\_val,

customer\_state,

time\_to\_delivery,

Diff\_estimated\_delivery

from

(

select

o.order\_id,

t.freight\_value,

c.customer\_state,

date\_diff(o.order\_delivered\_customer\_date,o.order\_purchase\_timestamp,day) as time\_to\_delivery,

date\_diff(o.order\_estimated\_delivery\_date,o.order\_delivered\_customer\_date,day) as Diff\_estimated\_delivery,

from `Bigquery\_project\_marsh050223.orders` o left join `Bigquery\_project\_marsh050223.order\_items` t on

                                                  o.order\_id=t.order\_id

                                              left join `Bigquery\_project\_marsh050223.customers`c on

                                                  o.customer\_id=c.customer\_id

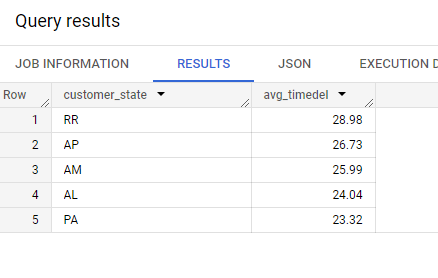
))

group by customer\_state

order by avg(time\_to\_delivery) desc

limit 5;

Result



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

Query top 5 states with fastest delivery compared to estimated delivery

select

customer\_state,

#round(avg(fre\_val),2) as avg\_fr,

#round(avg(time\_to\_delivery),2) as avg\_timedel,

round(avg(Diff\_estimated\_delivery),2) as avg\_diff\_estdel,

from

(

select

distinct order\_id,

sum(freight\_value) over (partition by order\_id) as fre\_val,

customer\_state,

time\_to\_delivery,

Diff\_estimated\_delivery

from

(

select

o.order\_id,

t.freight\_value,

c.customer\_state,

date\_diff(o.order\_delivered\_customer\_date,o.order\_purchase\_timestamp,day) as time\_to\_delivery,

date\_diff(o.order\_estimated\_delivery\_date,o.order\_delivered\_customer\_date,day) as Diff\_estimated\_delivery,

from `Bigquery\_project\_marsh050223.orders` o left join `Bigquery\_project\_marsh050223.order\_items` t on

                                                  o.order\_id=t.order\_id

                                              left join `Bigquery\_project\_marsh050223.customers`c on

                                                  o.customer\_id=c.customer\_id

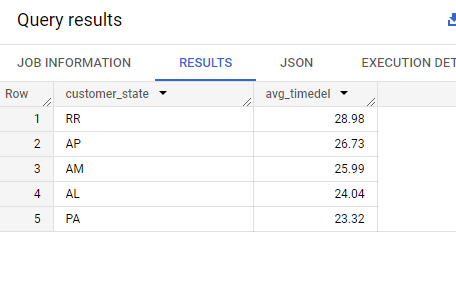
))

group by customer\_state

order by avg(Diff\_estimated\_delivery) desc

limit 5;

Result



Query -five states with lowest delivery time compared to estimated delivery

select

customer\_state,

#round(avg(fre\_val),2) as avg\_fr,

#round(avg(time\_to\_delivery),2) as avg\_timedel,

round(avg(Diff\_estimated\_delivery),2) as avg\_diff\_estdel,

from

(

select

distinct order\_id,

sum(freight\_value) over (partition by order\_id) as fre\_val,

customer\_state,

time\_to\_delivery,

Diff\_estimated\_delivery

from

(

select

o.order\_id,

t.freight\_value,

c.customer\_state,

date\_diff(o.order\_delivered\_customer\_date,o.order\_purchase\_timestamp,day) as time\_to\_delivery,

date\_diff(o.order\_estimated\_delivery\_date,o.order\_delivered\_customer\_date,day) as Diff\_estimated\_delivery,

from `Bigquery\_project\_marsh050223.orders` o left join `Bigquery\_project\_marsh050223.order\_items` t on

                                                  o.order\_id=t.order\_id

                                              left join `Bigquery\_project\_marsh050223.customers`c on

                                                  o.customer\_id=c.customer\_id

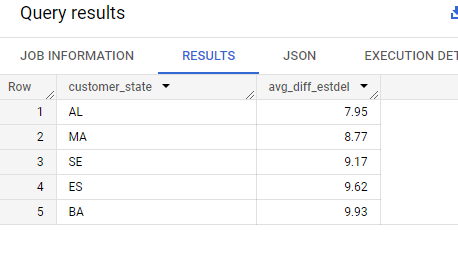
))

group by customer\_state

order by avg(Diff\_estimated\_delivery)

limit 5;

Result



**Category 6 Payment Type Analysis**

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

Query:

select

Year,

Month,

Payment\_type,

count(order\_id) as Count\_orders,

from

(

select

extract(year from order\_purchase\_timestamp) as Year,

extract(month from order\_purchase\_timestamp) as Month,

order\_id,

case when UPI>0 and Voucher>0 and credit\_card>0 and debit\_card>0 then "all"

     when UPI>0 and Voucher>0 and credit\_card>0 and debit\_card=0 then 'UPI+VOUCHER+CREDIT\_CARD'

     when UPI>0 and Voucher>0 and credit\_card=0 and debit\_card>0 then 'UPI+VOUCHER+CREDIT\_CARD'

     when UPI>0 and Voucher=0 and credit\_card>0 and debit\_card>0 then 'UPI+CREDIT\_CARD+DEBIT\_CARD'

     when UPI=0 and Voucher>0 and credit\_card>0 and debit\_card>0 then 'VOUCHER+CREDIT\_CARD+DEBIT\_CARD'

     when UPI>0 and Voucher>0 and credit\_card=0 and debit\_card=0 then 'UPI+VOUCHER'

     when UPI>0 and Voucher=0 and credit\_card>0 and debit\_card=0 then 'UPI+CREDIT\_CARD'

     when UPI>0 and Voucher=0 and credit\_card=0 and debit\_card>0 then 'UPI+DEBIT\_CARD'

     when UPI=0 and Voucher>0 and credit\_card>0 and debit\_card=0 then 'VOUCHER+CREDIT\_CARD'

     when UPI=0 and Voucher>0 and credit\_card=0 and debit\_card>0 then 'VOUCHER+DEBIT\_CARD'

     when UPI=0 and Voucher=0 and credit\_card>0 and debit\_card>0 then 'DEBIT\_CARD+CREDIT\_CARD'

     when UPI>0 and Voucher=0 and credit\_card=0 and debit\_card=0 then 'UPI'

     when UPI=0 and Voucher>0 and credit\_card=0 and debit\_card=0 then 'VOUCHER'

     when UPI=0 and Voucher=0 and credit\_card>0 and debit\_card=0 then 'CREDIT\_CARD'

     when UPI=0 and Voucher=0 and credit\_card=0 and debit\_card>0 then 'DEBIT\_CARD'

     ELSE 'NA'

     END AS PAYMENT\_TYPE

     from

     (

select

order\_id,

order\_purchase\_timestamp,

sum(UPI) as UPI,

sum(Voucher) as Voucher,

sum(credit\_card) as credit\_card,

sum(debit\_card) as debit\_card,

sum(not\_defined) as not\_define\_new

from(

select

p.order\_id,

o.order\_purchase\_timestamp,

case when lower(p.payment\_type)="upi" then 1 else 0 end as UPI,

case when lower(p.payment\_type)="voucher" then 1 else 0 end as Voucher,

case when lower(p.payment\_type)="credit\_card" then 1 else 0 end as credit\_card,

case when lower(p.payment\_type)="debit\_card" then 1 else 0 end as debit\_card,

case when lower(p.payment\_type)="not\_defined" then 1 else 0 end as not\_defined,

from `Bigquery\_project\_marsh050223.payments` p left join `Bigquery\_project\_marsh050223.orders` o on p.order\_id=o.order\_id

)

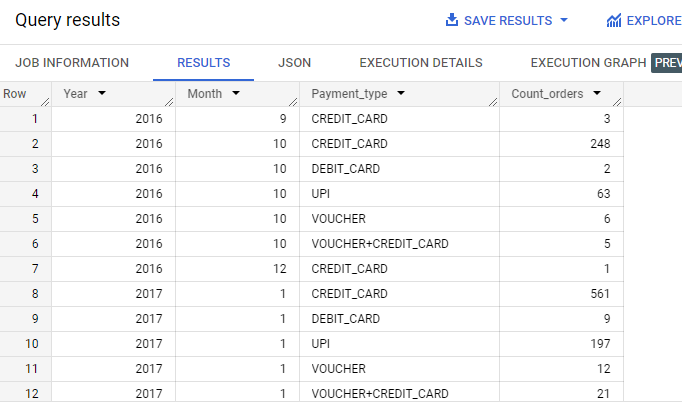
group by order\_id,order\_purchase\_timestamp

     )

)   group by Year,Month,payment\_type

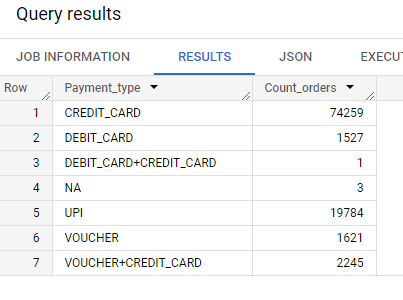
order by Year,Month,payment\_type;

Result



Insights:

Credit card is the most popular payment option in the given period of time followed by the UPI. Three cases are there where payment type in not defined and a few customers used multiple payment options together to complete the transaction.



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

Query

select

distinct payment\_installments,

count(order\_id) as count\_orders

from

(

select

order\_id,

payment\_installments,

nos,

max\_install

from(

select

order\_id,

payment\_installments,

nos,

max(nos) over(partition by order\_id) as max\_install

from(

select

order\_id,

payment\_installments,

row\_number()over(partition by order\_id) as nos

from `Bigquery\_project\_marsh050223.payments`

)

)

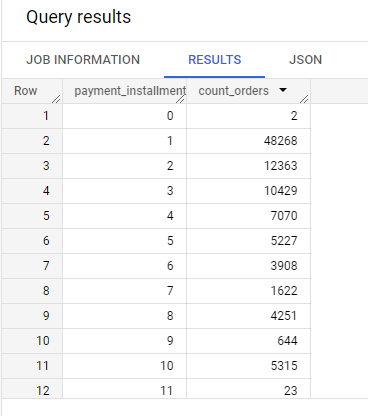
where nos=max\_install

)

group by payment\_installments

order by payment\_installments

Result



Fast moving products in the given time period

Query

select

\*

from

(

select

distinct product\_id,

product\_category,

count(order\_item\_id) over(partition by product\_id) as count\_of\_products

from

(

SELECT

o.order\_id,

o.order\_item\_id,

o.product\_id,

p.product\_category,

p.product\_name\_length

from `Bigquery\_project\_marsh050223.order\_items` o left join `Bigquery\_project\_marsh050223.products` p

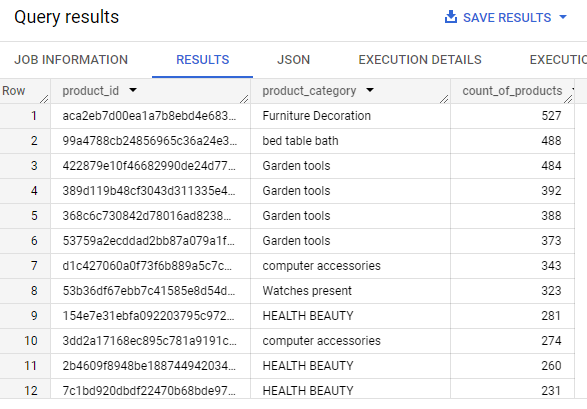
                                                on o.product\_id=p.product\_id

)

)

order by count\_of\_products desc;

Result



Seller with avg review less than 2

Query

select

\*

from

(

select

distinct seller\_id,

round(avg(review\_score)over(partition by seller\_id),2) as avg\_review

from

(

select

o.order\_id,

ot.product\_id,

ov.review\_score,

ot.seller\_id

from `Bigquery\_project\_marsh050223.orders` o left join `Bigquery\_project\_marsh050223.order\_reviews`  ov on o.order\_id=ov.order\_id

                                             left join `Bigquery\_project\_marsh050223.order\_items` ot on

                                             o.order\_id=ot.order\_id

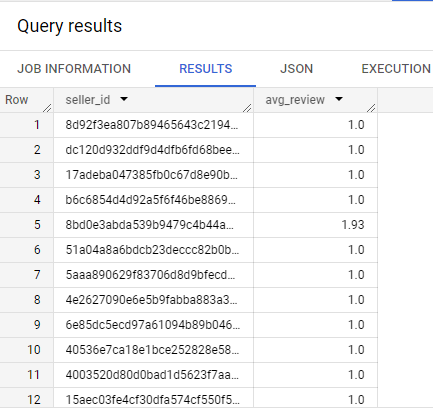
where review\_score is not null

)

)

where avg\_review <2

Result



Suggestions

* E commerce is booming in Brazil from 2017 , so more products need to be introduced to the business.
* More sellers need to introduce in farthest places or nearby places to reduce the freight cost.
* Need to revisit the list of sellers with lower rating and check on the reason , then need to work on improving it.
* Need to improve the delivery of the products with in the estimated time , for which the current percentage is only 93%.
* Customers are mainly using credit card as payment option and also using the EMI option. So need to associate with all major banks to give credit card offers to customers so that more customers will go for sale.
* Need to plan well before the peak months like November by storing more products.