Business Case Study: Target SQL

Index

1.Initial exploration of dataset like checking the characteristics of data

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

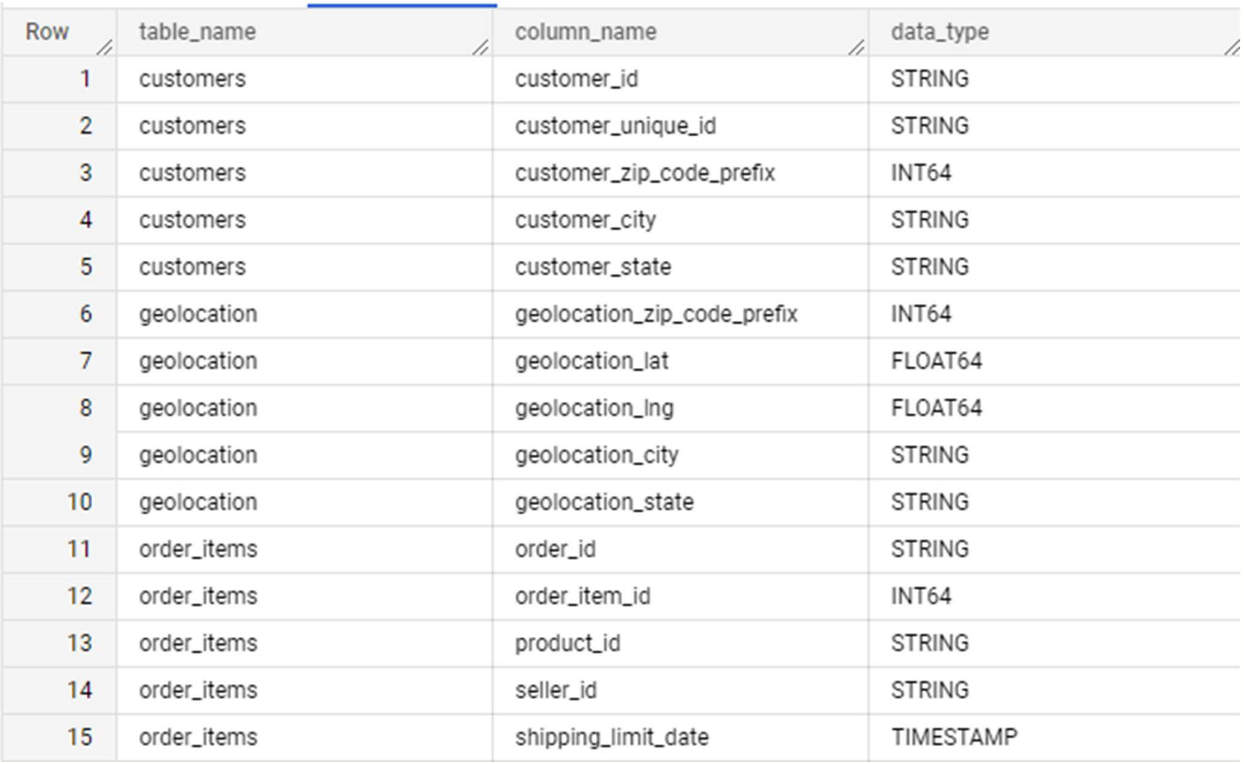
1.Data type of columns in a table

Details of 8 tables with their respective column name and data types can be fetched using following query

Query

```
select table_name,column_name,data_type from
`target-scaler-sql-
project.target_company_dataset.INFORMATION_SCHEMA.COLUMNS` where table_name IN
('customers','geolocation','order_items','orders','payments','products','sellers')
order by 1;
```

Output of Query



Row	table_name	column_name	data_type
1	customers	customer_id	STRING
2	customers	customer_unique_id	STRING
3	customers	customer_zip_code_prefix	INT64
4	customers	customer_city	STRING
5	customers	customer_state	STRING
6	geolocation	geolocation_zip_code_prefix	INT64
7	geolocation	geolocation_lat	FLOAT64
8	geolocation	geolocation_lng	FLOAT64
9	geolocation	geolocation_city	STRING
10	geolocation	geolocation_state	STRING
11	order_items	order_id	STRING
12	order_items	order_item_id	INT64
13	order_items	product_id	STRING
14	order_items	seller_id	STRING
15	order_items	shipping_limit_date	TIMESTAMP

Fig: Table 1.1

The output given above is only up to 15 rows of whole output obtained. The 'data_type' column showing the datatype of each corresponding to each 'column_name' of a table.

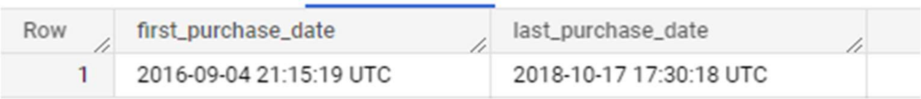
1.2. Time period for which the data is given

To find the time period for which data is given ,we have to extract the minimum and maximum date out of 'order_purchase_timestamp' column of 'orders.csv' table. Referring to the following query

Query

```
select min(order_purchase_timestamp) first_purchase_date,
max(order_purchase_timestamp) last_purchase_date
from `target_company_dataset.orders`;
```

Output of Query



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

Fig: Table 1.2

The ‘first_purchase_date’ refers to the date of first purchase and ‘last_purchase_date’ refers to the date of last purchase available in record.

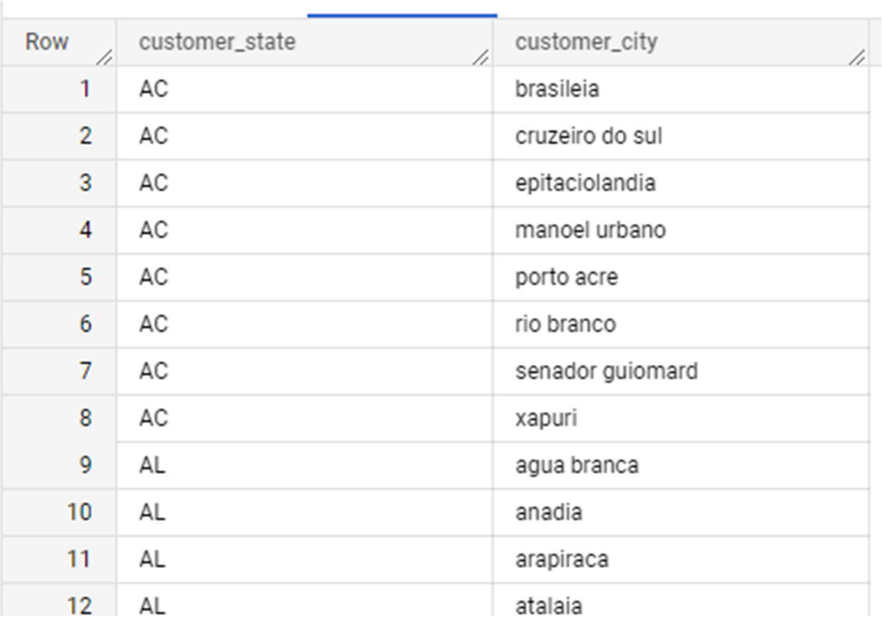
1.3. Cities and States of customers ordered during the given period

To find the required record, we need to join the ‘customers’ and ‘orders’ table through common ‘customer_id’ column and extract the ‘customer_state’ and ‘customer_city’ column within given time span of ‘order_purchase_timestamp’ column. Referring to the following query

Query

```
select distinct c.customer_state, c.customer_city from target_company_dataset.customers c join t
arget_company_dataset.orders o
on c.customer_id=o.customer_id
where
o.order_purchase_timestamp between
( select min(order_purchase_timestamp) from `target_company_dataset.orders`)
and
( select max(order_purchase_timestamp) from `target_company_dataset.orders`)
order by 1,2;
```

Output of Query



Row	customer_state	customer_city
1	AC	brasileia
2	AC	cruzeiro do sul
3	AC	epitaciolandia
4	AC	manoel urbano
5	AC	porto acre
6	AC	rio branco
7	AC	senador guiomard
8	AC	xapuri
9	AL	agua branca
10	AL	anadia
11	AL	arapiraca
12	AL	atalaia

Fig: Table 1.3

The output given above is only up to 12 rows of whole output obtained. The ‘customer_state’ and ‘customer_city’ refers to different state and city present in the city during the whole purchase period.

2.In-depth Exploration

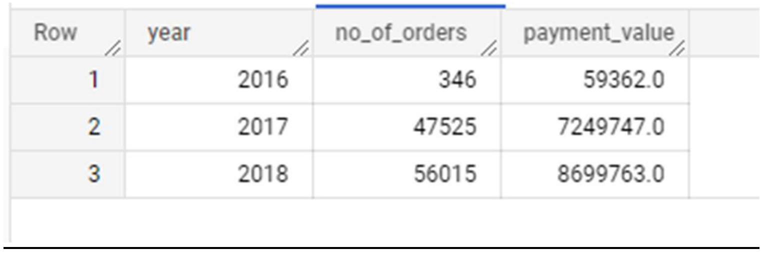
2.1. Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?

For required data we need to show how no of orders and purchasing value is changing year on year basis . For that we need to join the 'orders' and 'payments' table through common 'order_id' column. Group the data year wise and apply the count and sum aggression to 'order_id' and 'payment_value' column .Referring to following query

Query

```
select extract(year from order_purchase_timestamp) as year ,
count(o.order_id) as no_of_orders,
round(sum(p.payment_value)) payment_value
from `target_company_dataset.orders` o join `target_company_dataset.payments` p
on o.order_id=p.order_id
group by year
order by 1;
```

Output of Query



Row	year	no_of_orders	payment_value
1	2016	346	59362.0
2	2017	47525	7249747.0
3	2018	56015	8699763.0

Fig: Table 2.1

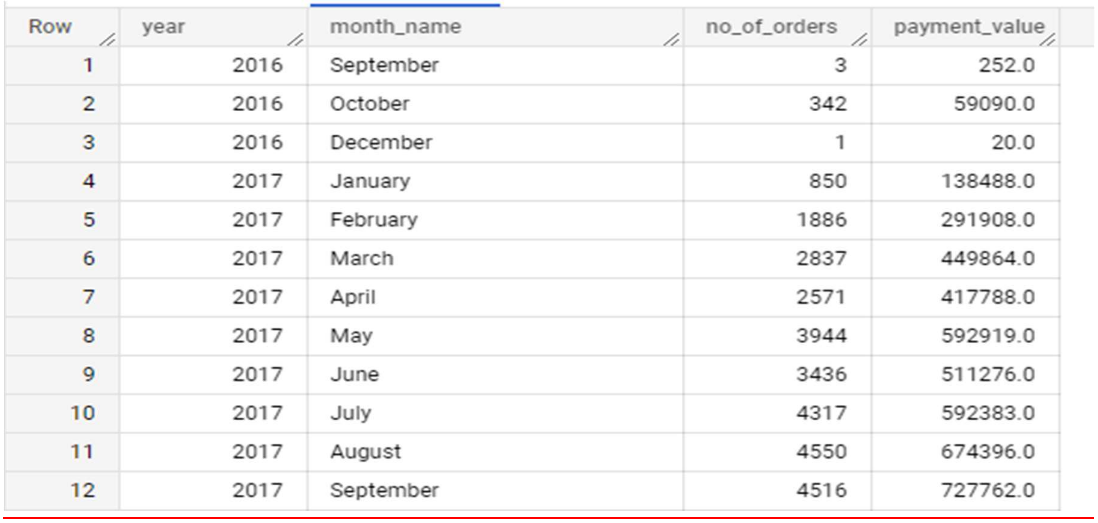
From output given above, it is obvious that no of orders as well as payment value has increased significantly year on year basis. So, from given data we can conclude *that there is a growing trend of e-commerce in Brazil.*

Now to get "seasonality" of the data, we need to get the month wise data for each year to analyze the trend. Referring to the following query

Query

```
with cte as
(select extract(year from order_purchase_timestamp) as year ,
format_date('%B', order_purchase_timestamp) as month_name,
extract(month from order_purchase_timestamp) as month,
count(o.order_id) as no_of_orders,round(sum(p.payment_value)) payment_value
from `target_company_dataset.orders` o join `target_company_dataset.payments` p
on o.order_id=p.order_id
group by 1,2,3)
select year,month_name,no_of_orders ,payment_value from cte
order by year,month;
```

Output of Query



Row	year	month_name	no_of_orders	payment_value
1	2016	September	3	252.0
2	2016	October	342	59090.0
3	2016	December	1	20.0
4	2017	January	850	138488.0
5	2017	February	1886	291908.0
6	2017	March	2837	449864.0
7	2017	April	2571	417788.0
8	2017	May	3944	592919.0
9	2017	June	3436	511276.0
10	2017	July	4317	592383.0
11	2017	August	4550	674396.0
12	2017	September	4516	727762.0

Fig: Table 2.2

The output given above is only up to 12 rows of complete output obtained. It is showing the required month on month details of no of order placed and payment value obtained for each year.

Insights

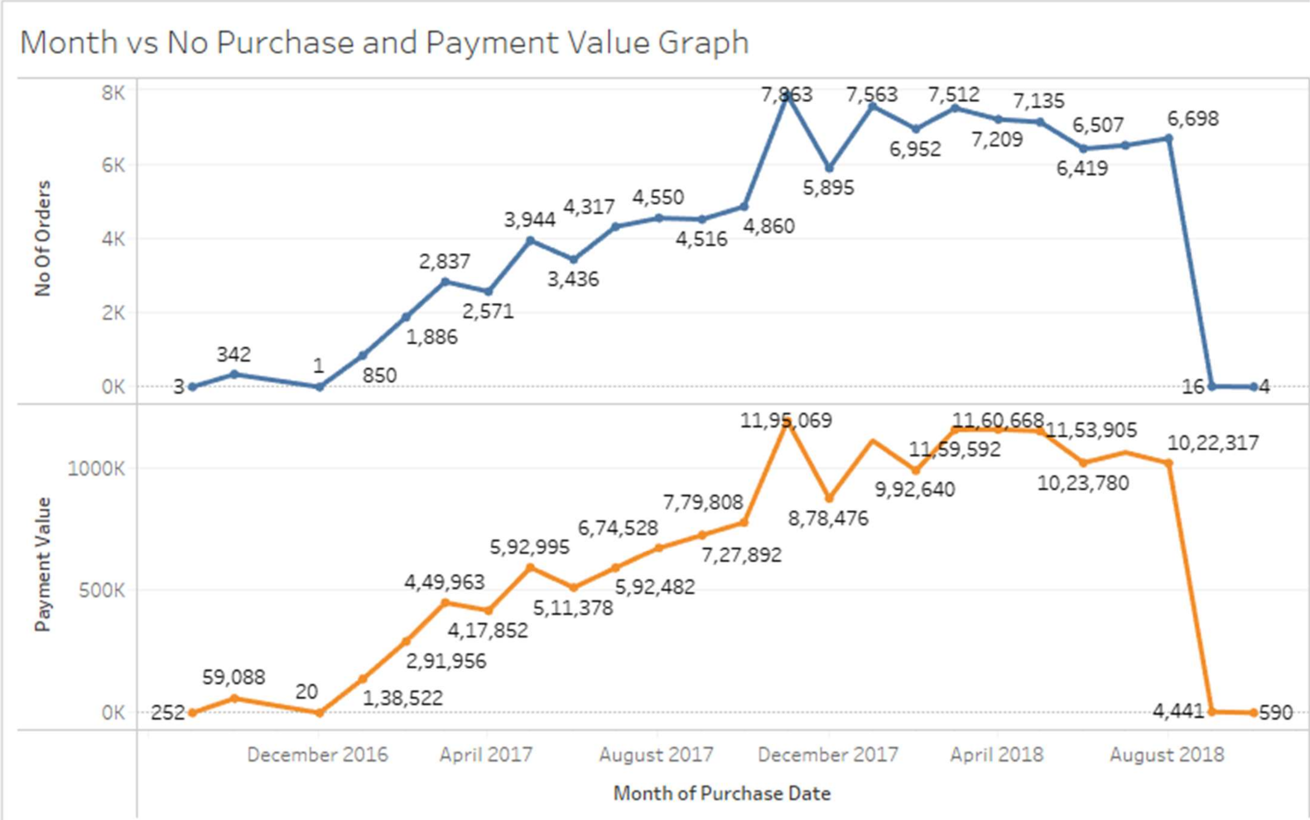


Fig: Graph 2.3

Consider the graph 2.3, we can see that there is a overall gradual increase in no of orders as well as purchasing value monthly. However, it is to be observed that purchasing trend peaked during November 2017 to August 2018, but after that numbers decreased sharply.

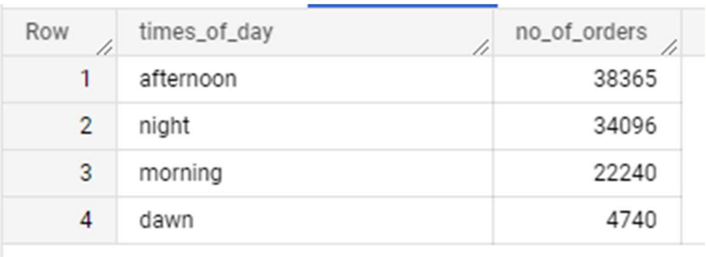
2.2. What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?

To acquire the desired data, we need to analyse the time at which the purchasing is done , for that we need to broadly divide the day into morning, evening, night and dawn and after that we have look for the number of orders falls in different times of day. For that we need to refer to the query given below

Query

```
select
case
when extract(time from order_purchase_timestamp ) between '00:00:00' and '06:00:00'
then 'dawn'
when extract(time from order_purchase_timestamp ) between '06:00:01' and '12:00:00'
then 'morning'
when extract(time from order_purchase_timestamp ) between '12:00:01' and '18:00:00'
then 'afternoon'
when extract(time from order_purchase_timestamp ) between '18:00:01' and '23:59:59'
then 'night'
else 'NA'
end as times_of_day,
count(order_purchase_timestamp) as no_of_orders
from `target_company_dataset.orders`
group by times_of_day
order by no_of_orders desc;
```

Output of Query



Row	times_of_day	no_of_orders
1	afternoon	38365
2	night	34096
3	morning	22240
4	dawn	4740

Fig: Table 2.4

We can observe from the output obtained that Brazilian customers tend to buy mostly in afternoon i.e. between '12:00:01' and '18:00:00' followed by night. The least active time is Dawn.

Recommendations

1.Since the customers tend to buy in afternoon mostly, the company can increase the number of staffs in the stores in afternoon to assist the customers, moreover no of payment counters can be increased in afternoon for better convenience of customers visiting the store. (Referring to table 2.4)

2.For online orders, the server of the website should be working smooth during the peak hours. The transaction gateway should be fast so as to complete the transaction and customers can get best experience while purchasing.

3.As it can be observed that there is sharp decrease in no of orders and payment value after august 2018. So immediate steps need to be taken in order to increase the inflow of orders. Company can think of launching pay later policy where customer can buy now and pay later, this can attract customers (Referring graph 2.3)

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

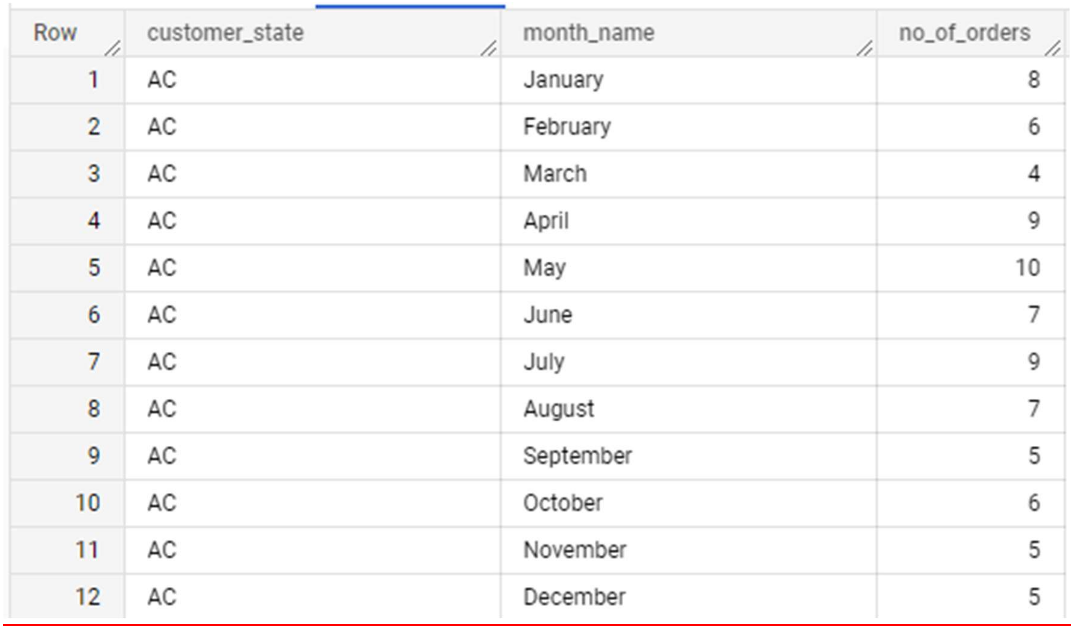
3.1. Get month on month orders by states

To get the required data, we need to join the ‘customers’ and ‘orders’ table and fetch the month wise no of orders placed for each customer state. Consider the following code and the output obtained

Query

```
with temp as (
select c.customer_state ,format_datetime('%B',order_purchase_timestamp) as month_name,
extract(month from order_purchase_timestamp) as month_no,count(o.order_id) no_of_orders
from `target_company_dataset.customers` c join `target_company_dataset.orders` o
on c.customer_id=o.customer_id
group by 1,2,3)
select customer_state,month_name,no_of_orders from temp
order by 1,month_no asc;
```

Output of Query



Row	customer_state	month_name	no_of_orders
1	AC	January	8
2	AC	February	6
3	AC	March	4
4	AC	April	9
5	AC	May	10
6	AC	June	7
7	AC	July	9
8	AC	August	7
9	AC	September	5
10	AC	October	6
11	AC	November	5
12	AC	December	5

Fig: Table 3.1

The output given above is only up to 12 rows of complete output obtained. We can clearly see the month wise no of orders placed for each customer state.

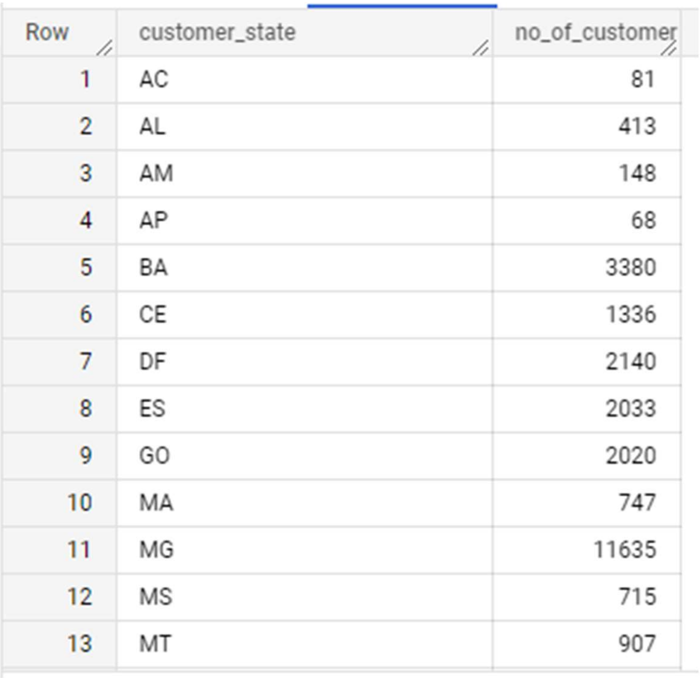
3.2. Distribution of customers across the states in Brazil

To get the required data firstly we need to join the ‘customers’ and ‘orders’ table ,then we can get the ddistribution of customers across different states.

Query

```
select c.customer_state , count(c.customer_id) no_of_customer
from `target_company_dataset.customers` c join `target_company_dataset.orders` o
on c.customer_id=o.customer_id
group by 1
order by 1;
```

Output of Query



Row	customer_state	no_of_customer
1	AC	81
2	AL	413
3	AM	148
4	AP	68
5	BA	3380
6	CE	1336
7	DF	2140
8	ES	2033
9	GO	2020
10	MA	747
11	MG	11635
12	MS	715
13	MT	907

Fig: Table 3.2

The output given above is only up to 13 rows of complete output obtained. We can clearly see the state wise distribution of customers.

Insights

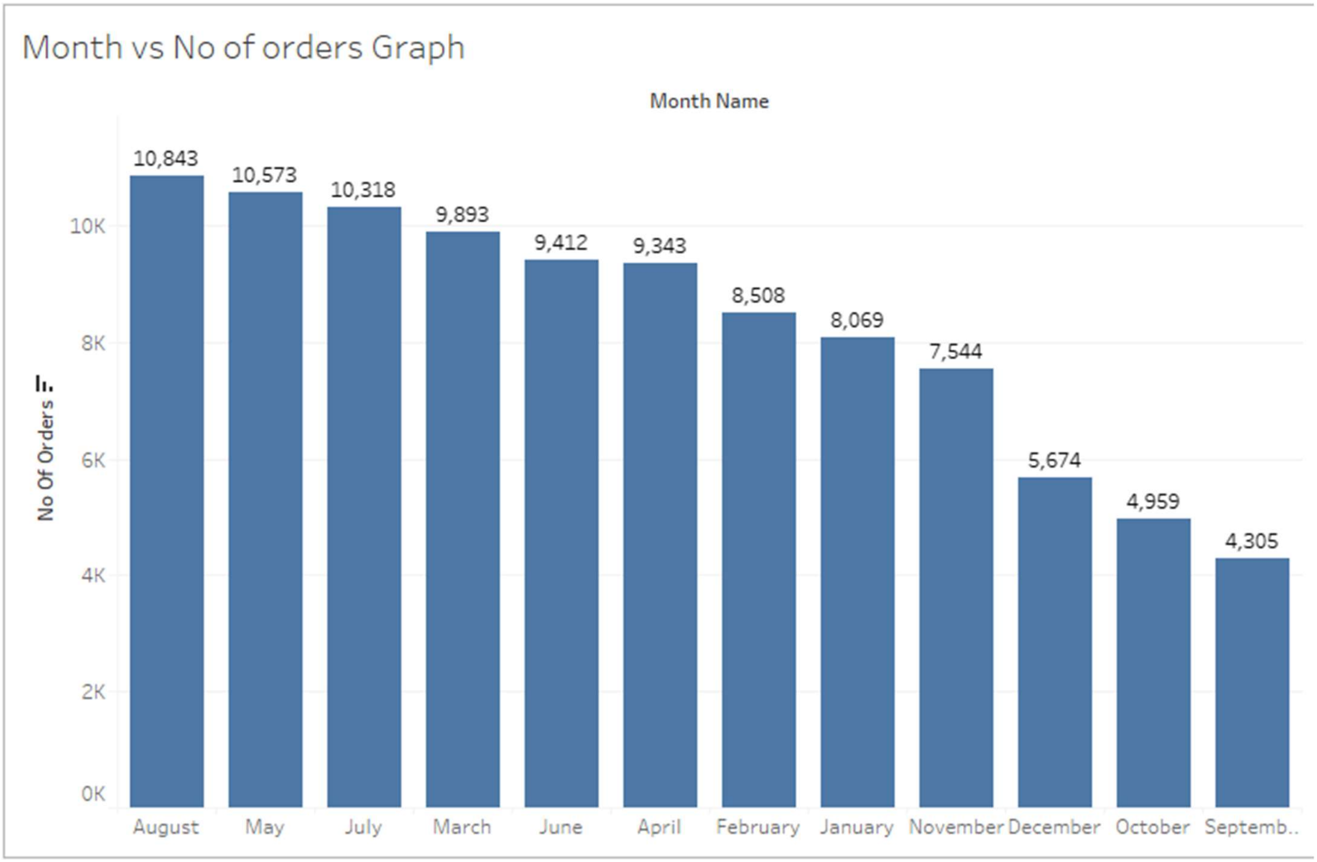


Fig: Graph 3.3

Consider the graph 3.3, it can be observed that no of orders placed in month of August is highest and September being the lowest.

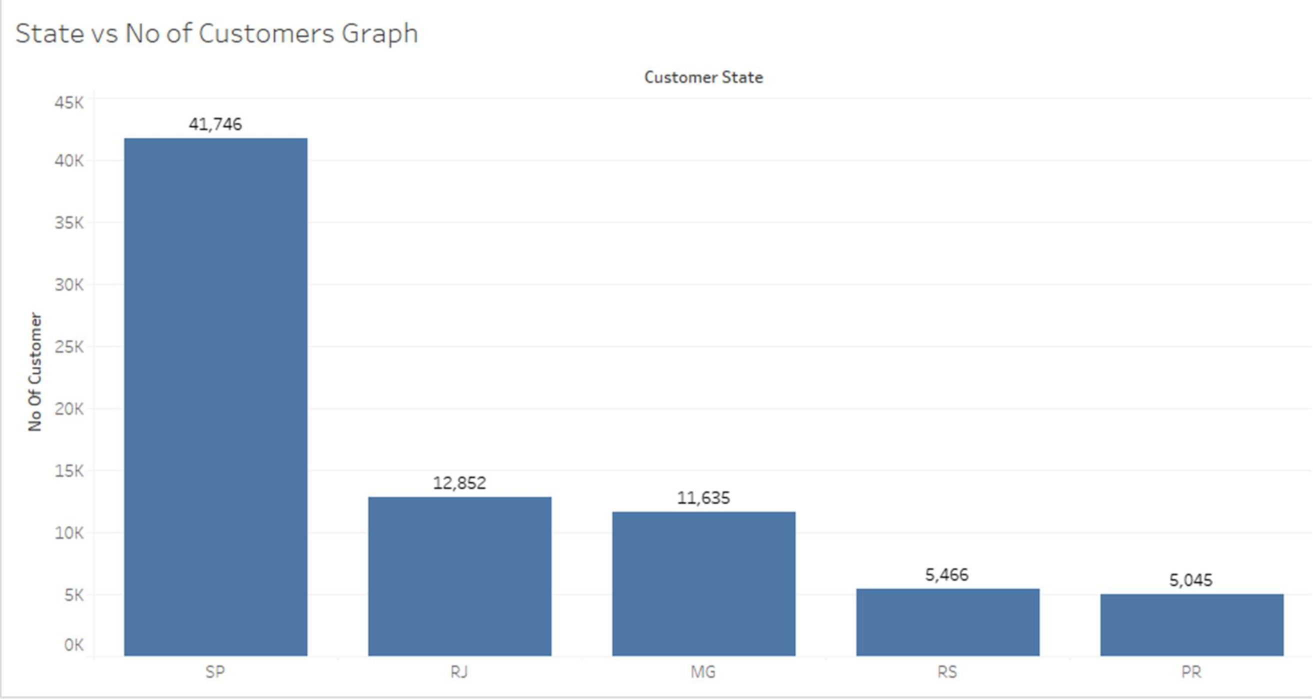


Fig: Graph 3.4

Consider the graph 3.4, the graph is plotted for the states having highest no of customers and data is sorted for top 5 states. It can be observed that state SP has highest no of customer.

Recommendations

1.Since March to August are amongst highest months in terms of no of order placed, Company can have more storage of selling products in these months so that Products couldn't get "out of stock". (Refer to graph 3.3)

2.Comapny can think of opening new stores in top 5 states having highest no of customers. This can increase the revenue of company.

3. For states where no of customers is least, Company can invest in marketing and advertisement in order to attract more customers.

4.Impact on Economy

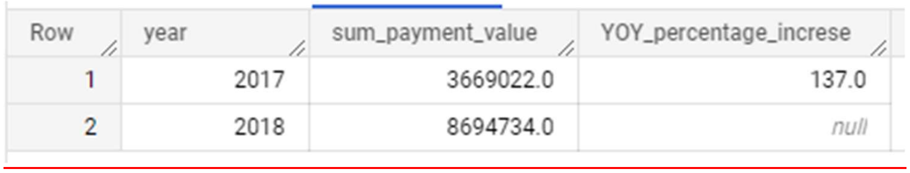
4.1. Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use “payment_value” column in payments table

To solve this problem, we need to add up ‘payment_value’ data separately for 2017 and 2018 including months between January to August only. And then we need to compare the change between the data of 2017 and 2018. For that consider the following query

Query

```
with cte as
(select extract(year from order_purchase_timestamp) as year,round(sum(payment_value))
as sum_payment_value
from `target_company_dataset.payments` p join `target_company_dataset.orders` o
on p.order_id=o.order_id
where extract(month from order_purchase_timestamp) between 1 and 8
group by 1
order by 1)
select *  ,
round(((lead(sum_payment_value) over(order by year asc)-
sum_payment_value)/sum_payment_value)*100,1) as YOY_percentage_increse
from cte
order by cte.year;
```

Output of Query



Row	year	sum_payment_value	YOY_percentage_increse
1	2017	3669022.0	137.0
2	2018	8694734.0	null

Fig: Table 4.1

Insights

From the output obtained it can be concluded that the revenue of company from selling the products have increase significantly by around 137 % from 2017 to 2018 in the months extending from January to August. That can be considered as very good performance by the company year on year basis.

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

To get the required data we need to join the ‘customers’ and ‘orders’ table through common ‘customer_id’ column, then need to select required column and applying aggregate function will fetch the desired result

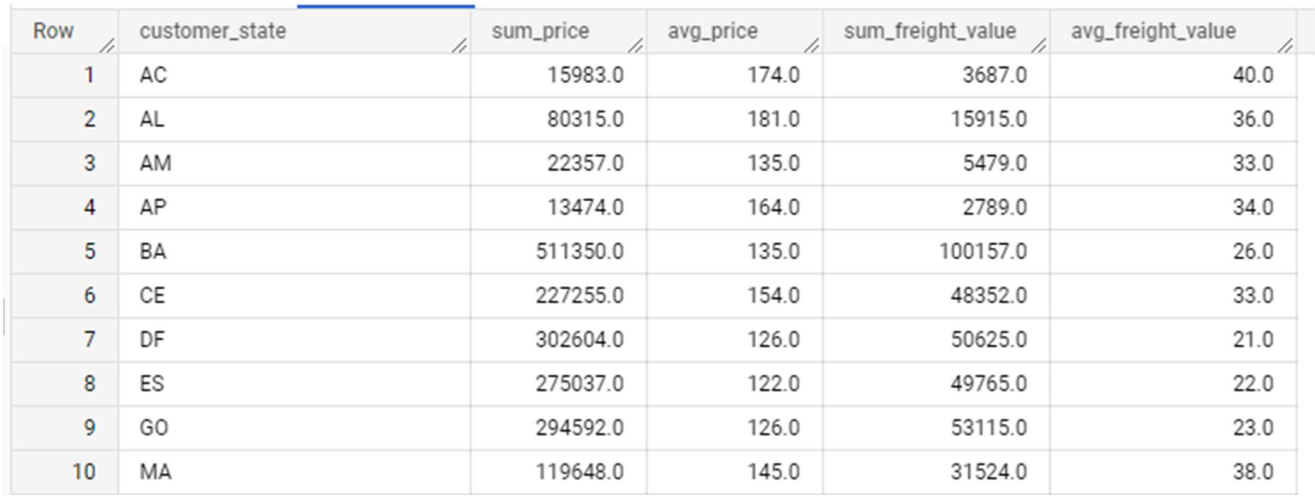
Query

```
select c.customer_state ,round(sum(op.price))sum_price,round(avg(op.price))avg_price,

round(sum(op.freight_value))sum_freight_value,round(avg(op.freight_value))avg_freight_value

from `target_company_dataset.customers` c
join `target_company_dataset.orders` o
on c.customer_id=o.customer_id
join `target_company_dataset.order_items` op
on o.order_id=op.order_id
group by 1
order by 1;
```

Output of Query



Row	customer_state	sum_price	avg_price	sum_freight_value	avg_freight_value
1	AC	15983.0	174.0	3687.0	40.0
2	AL	80315.0	181.0	15915.0	36.0
3	AM	22357.0	135.0	5479.0	33.0
4	AP	13474.0	164.0	2789.0	34.0
5	BA	511350.0	135.0	100157.0	26.0
6	CE	227255.0	154.0	48352.0	33.0
7	DF	302604.0	126.0	50625.0	21.0
8	ES	275037.0	122.0	49765.0	22.0
9	GO	294592.0	126.0	53115.0	23.0
10	MA	119648.0	145.0	31524.0	38.0

Fig: Table 4.2

The output given above is only up to 10 rows of whole output obtained. In the complete output we can clearly obtain Mean & Sum of price and freight value by customer state.

Insights

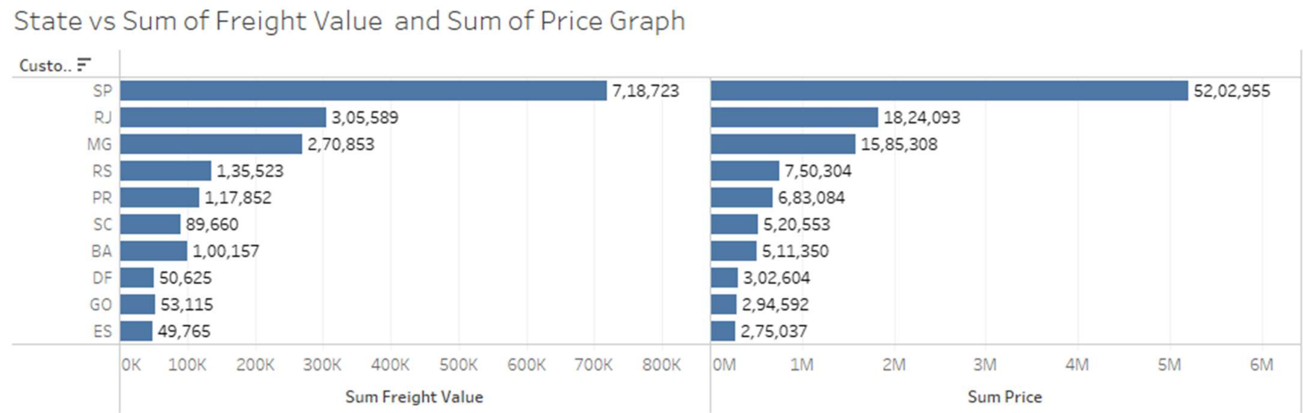


Fig: Graph 4.3

The above graph shows the top 10 states where total of freight value i.e., Price rate at which a product is delivered from one point to another, and sum of price i.e., Actual price of the products ordered is maximum.

Recommendations

1.The company has shown very healthy growth of revenue from year 2017 to 2018, So it recommended that company should invest more and open new stores across all locations of country. (Referring to table 4.1)

2.As graph 4.3 shows there are some states where sum of freight value and total price is very high, Company can consider opening of various new stores in these states and reduce the freight value, from that customers of these states will get the product in cheaper cost.

5.Analysis on sales, freight and delivery time

5.1. Calculate days between purchasing, delivering and estimated delivery

To get the required data, following formula can be considered:

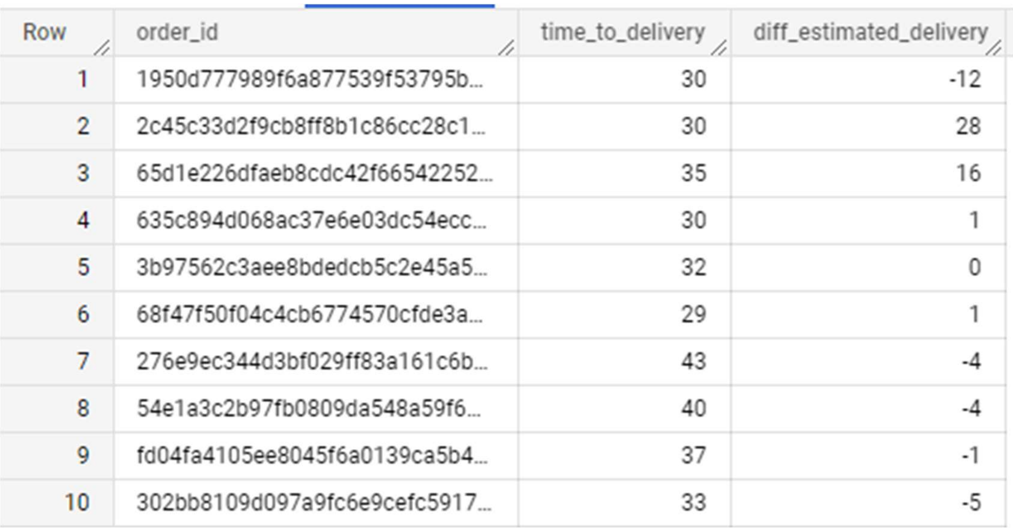
- time_to_delivery = order_purchase_timestamp-order_delivered_customer_date
- diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

Query required to fetch the data is as follows

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 `target_company_dataset.orders`;
```

Output of Query



Row	order_id	time_to_delivery	diff_estimated_delivery
1	1950d777989f6a877539f53795b...	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28c1...	30	28
3	65d1e226dfaeb8cdc42f66542252...	35	16
4	635c894d068ac37e6e03dc54ecc...	30	1
5	3b97562c3aee8bdedcb5c2e45a5...	32	0
6	68f47f50f04c4cb6774570cfde3a...	29	1
7	276e9ec344d3bf029ff83a161c6b...	43	-4
8	54e1a3c2b97fb0809da548a59f6...	40	-4
9	fd04fa4105ee8045f6a0139ca5b4...	37	-1
10	302bb8109d097a9fc6e9cefc5917...	33	-5

Fig: Table 5.1

The output given above is only up to 10 rows of whole output obtained. The output shown gives the required data for each available 'order_id'.

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

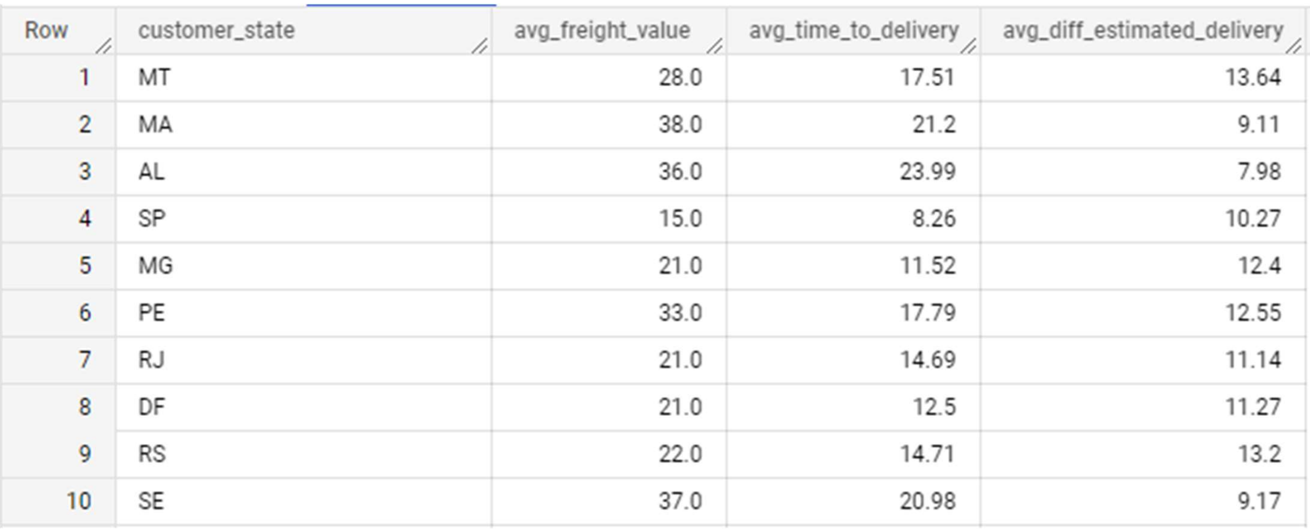
For required data we need to first join the 'orders' and 'customers' table then extracting the columns will lead to result. Consider the following query

Query

```
select c.customer_state,round(avg(oi.freight_value)) as avg_freight_value ,
round(avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)),2)
avg_time_to_delivery,
round(avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)),2)
as avg_diff_estimated_delivery
from `target_company_dataset.orders` o join `target_company_dataset.customers` c
on o.customer_id=c.customer_id
join `target_company_dataset.order_items` oi
```

```
on oi.order_id=o.order_id
group by 1;
```

Output of Query



Row	customer_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_delivery
1	MT	28.0	17.51	13.64
2	MA	38.0	21.2	9.11
3	AL	36.0	23.99	7.98
4	SP	15.0	8.26	10.27
5	MG	21.0	11.52	12.4
6	PE	33.0	17.79	12.55
7	RJ	21.0	14.69	11.14
8	DF	21.0	12.5	11.27
9	RS	22.0	14.71	13.2
10	SE	37.0	20.98	9.17

Fig: Table 5.2

The output given above is only up to 10 rows of whole output obtained. The output shown gives the asked data for each available 'customer_state'.

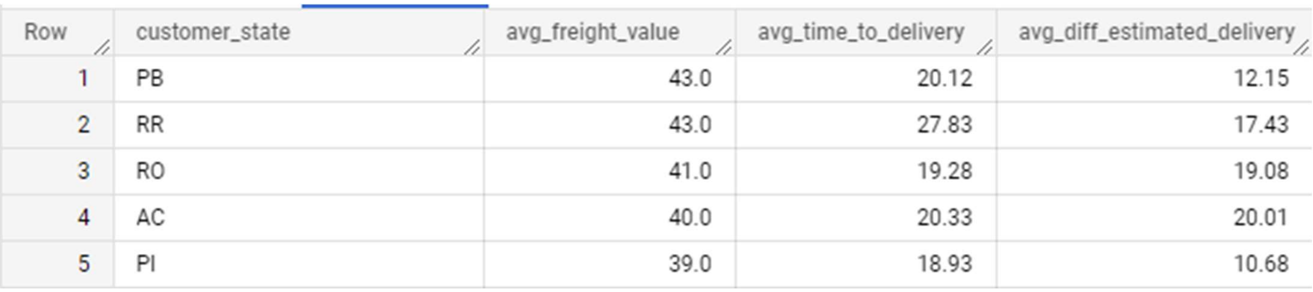
5.3. Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5

Consider the following query to get the Top 5 states with highest average freight value - sort in descending order of their value

Query

```
select c.customer_state,round(avg(oi.freight_value)) as avg_freight_value ,
round(avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)),2) avg_time_to_
delivery,
round(avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)),2) as avg_
diff_estimated_delivery
from `target_company_dataset.orders` o join `target_company_dataset.customers` c
on o.customer_id=c.customer_id
join `target_company_dataset.order_items` oi
on oi.order_id=o.order_id
group by 1
order by 2 desc
limit 5;
```

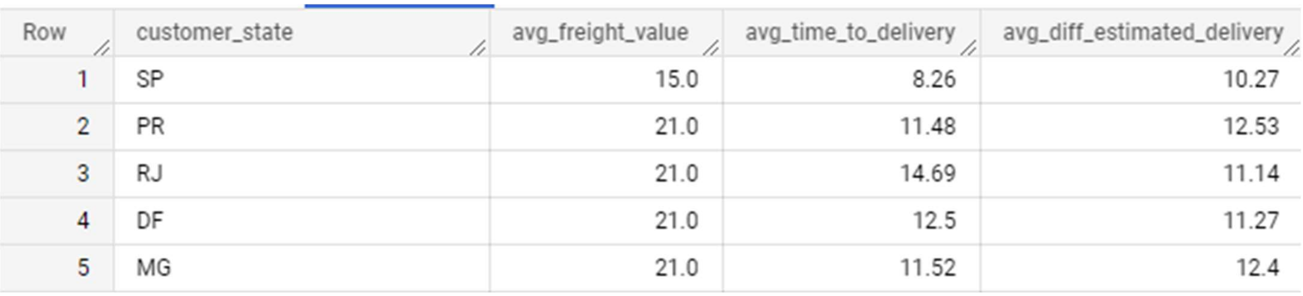
Output of Query



Row	customer_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_delivery
1	PB	43.0	20.12	12.15
2	RR	43.0	27.83	17.43
3	RO	41.0	19.28	19.08
4	AC	40.0	20.33	20.01
5	PI	39.0	18.93	10.68

Fig: Table 5.3

Similarly, if we change the order by clause to ‘asc’ (ascending order), the result of query will lead to bottom 5 states with lowest average freight value - sort in ascending order of their value as shown in table 5.4.



Row	customer_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_delivery
1	SP	15.0	8.26	10.27
2	PR	21.0	11.48	12.53
3	RJ	21.0	14.69	11.14
4	DF	21.0	12.5	11.27
5	MG	21.0	11.52	12.4

Fig: Table 5.4

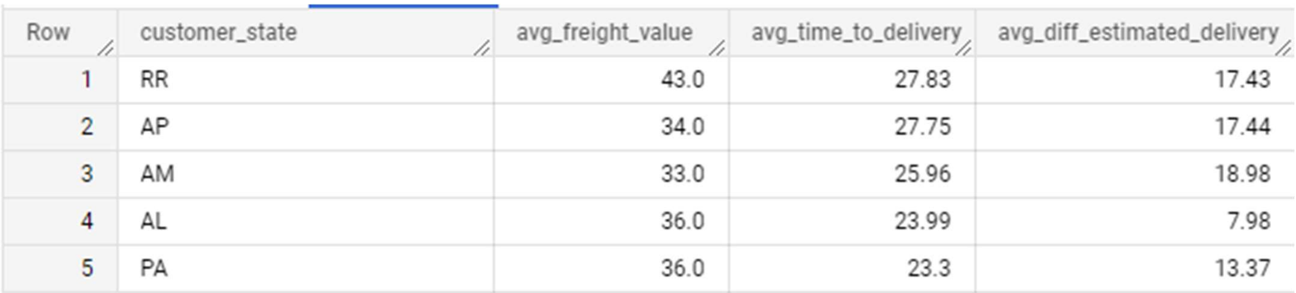
5.4. Top 5 states with highest/lowest average time to delivery

Consider the following query to get the Top 5 states with highest average time to delivery - sort in descending order of their value

Query

```
select c.customer_state,round(avg(oi.freight_value)) as avg_freight_value ,
round(avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)),2) avg_time_to_
delivery,
round(avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)),2) as avg_
diff_estimated_delivery
from `target_company_dataset.orders` o join `target_company_dataset.customers` c
on o.customer_id=c.customer_id
join `target_company_dataset.order_items` oi
on oi.order_id=o.order_id
group by 1
order by 3 desc
limit 5;
```

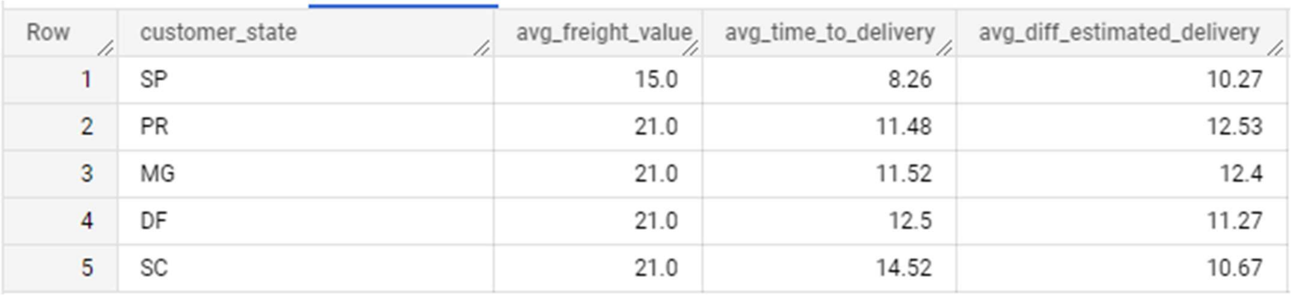
Output of Query



Row	customer_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_delivery
1	RR	43.0	27.83	17.43
2	AP	34.0	27.75	17.44
3	AM	33.0	25.96	18.98
4	AL	36.0	23.99	7.98
5	PA	36.0	23.3	13.37

Fig Table 5.5

Similarly, if we change the order by clause to ‘asc’ (ascending order), the result of query will lead to bottom 5 states with lowest average time to delivery - sorted in ascending order of their value as shown in table 5.6.



Row	customer_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_delivery
1	SP	15.0	8.26	10.27
2	PR	21.0	11.48	12.53
3	MG	21.0	11.52	12.4
4	DF	21.0	12.5	11.27
5	SC	21.0	14.52	10.67

Fig: Table 5.6

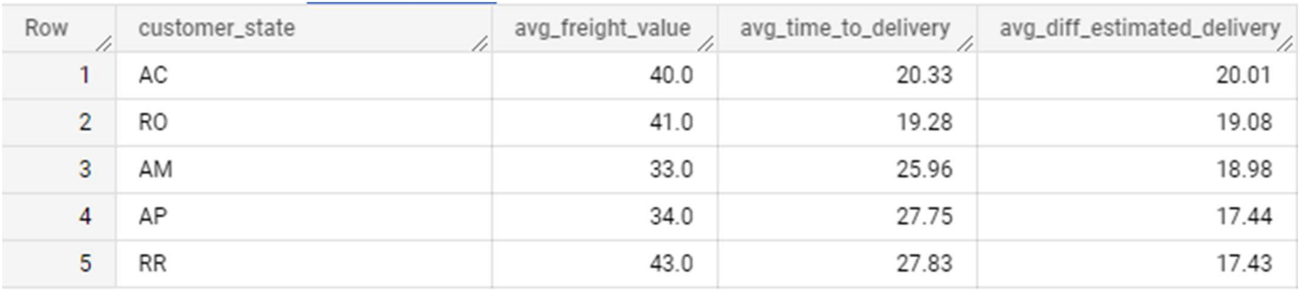
5.5.Top 5 states where delivery is really fast/ not so fast compared to estimated date

Consider the following query to get the Top 5 where delivery is really fast compared to estimated date

Query

```
select c.customer_state,round(avg(oi.freight_value)) as avg_freight_value ,
round(avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)),2) avg_time_to_
delivery,
round(avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)),2) as avg_
diff_estimated_delivery
from `target_company_dataset.orders` o join `target_company_dataset.customers` c
on o.customer_id=c.customer_id
join `target_company_dataset.order_items` oi
on oi.order_id=o.order_id
group by 1
order by 4 desc
limit 5;
```

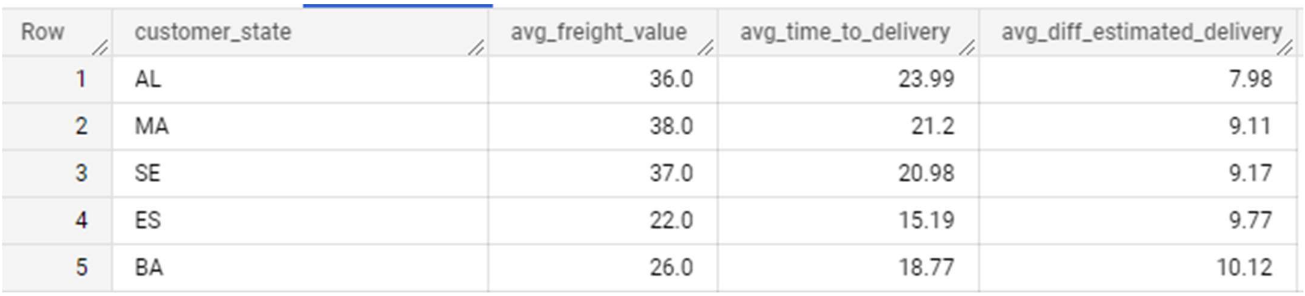
Output of Query



Row	customer_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_delivery
1	AC	40.0	20.33	20.01
2	RO	41.0	19.28	19.08
3	AM	33.0	25.96	18.98
4	AP	34.0	27.75	17.44
5	RR	43.0	27.83	17.43

Fig: Table 5.7

Similarly, we can get bottom 5 states where delivery is not so fast compared to estimated date. Result is as shown in table 5.8



Row	customer_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_delivery
1	AL	36.0	23.99	7.98
2	MA	38.0	21.2	9.11
3	SE	37.0	20.98	9.17
4	ES	22.0	15.19	9.77
5	BA	26.0	18.77	10.12

Fig: Table 5.8

Insights

1. The states having highest average freight value is not good sign for business of company. Since if delivering cost is more the customer will not order items.

2.If the time taken to deliver the item is more than eventually purchasing experience of customers will be not good and revenue from those states will decrease.

3.States where delivery is really fast i.e., order is being delivered well before the estimated time is good sign for company business. This will increase the trust of customers in company and they will tend to buy more.

Recommendations

1.Company should decrease fright value for the states where its average value is high. Decrement in fright value leads to cheaper the value of items and eventually increase the no. of customers. (Refer to Table 5.3)

2.To decrease the average time of delivery, company can think of opening new stores in the states where its average value is high. Decreasing the time of delivery eventually sends good reviews for company. (Refer to Table 5.3)

3. Company can focus the bottom 5 states where delivery is not so fast compared to estimated date and try to increase the delivery speed by increasing the no. of delivering staffs. By doing so, review and eventually revenue of company will increase. (Refer to Table 5.8)

4. company can think of awarding the best performing stores whose delivery are fast and user review are good, by giving higher bonus to staffs. This will increase the participation of employees within company.

6. Payment type analysis

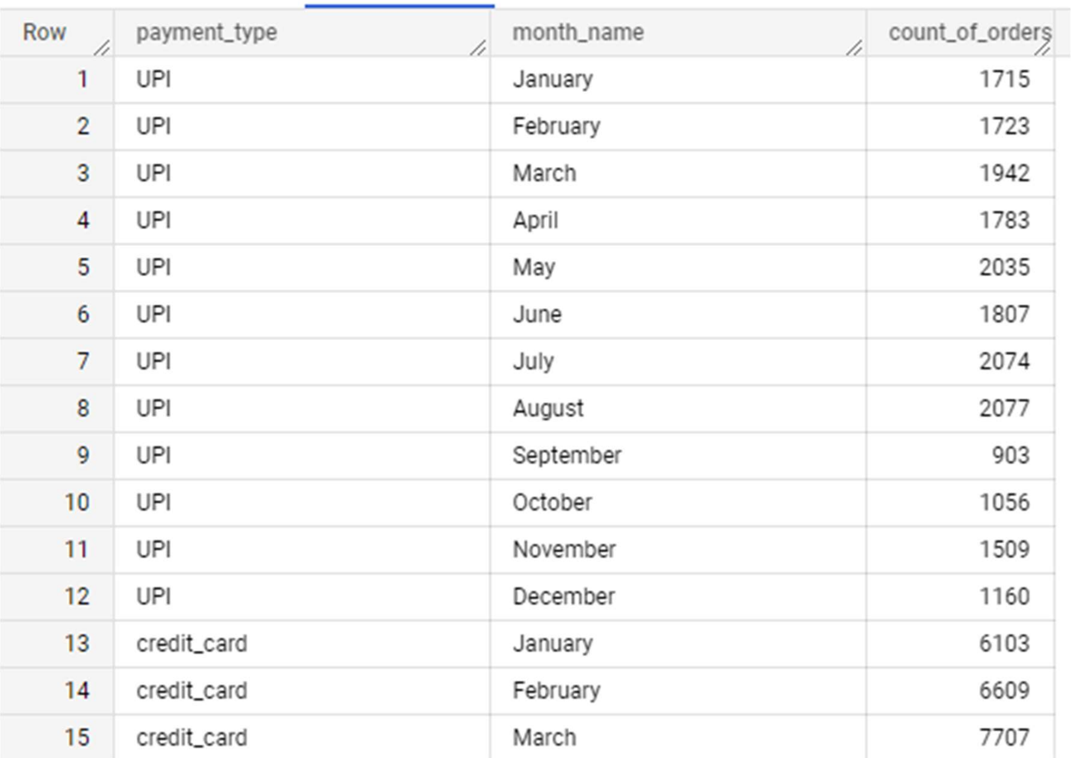
6.1. Month over Month count of orders for different payment types

To get the required data, we need to join ‘payments’ and ‘orders’ table through common ‘order_id’ column . After that we have to select ‘payment_type’, ’month_name’ and ‘count_of_orders’ column as result. The complete query is given below

Query

```
with temp as (
select p.payment_type ,format_datetime('%B',order_purchase_timestamp) as month_name,
extract(month from order_purchase_timestamp) as month_no,count(o.order_id) count_of_orders
from `target_company_dataset.payments` p join `target_company_dataset.orders` o
on p.order_id=o.order_id
group by 1,2,3)
select payment_type,month_name,count_of_orders from temp
order by 1,month_no asc;
```

Output of Query



Row	payment_type	month_name	count_of_orders
1	UPI	January	1715
2	UPI	February	1723
3	UPI	March	1942
4	UPI	April	1783
5	UPI	May	2035
6	UPI	June	1807
7	UPI	July	2074
8	UPI	August	2077
9	UPI	September	903
10	UPI	October	1056
11	UPI	November	1509
12	UPI	December	1160
13	credit_card	January	6103
14	credit_card	February	6609
15	credit_card	March	7707

Fig: Table 6.1

The output given above is only up to 15 rows of whole output obtained. In the complete output we can clearly obtain Month over Month count of orders for different payment types.

Insights

To get the more insights into the data we need to find the which payment type customers of country are preferring over the other. For that we need to get the total no of orders made via different payment type.

Have a look at following query and the output obtained:

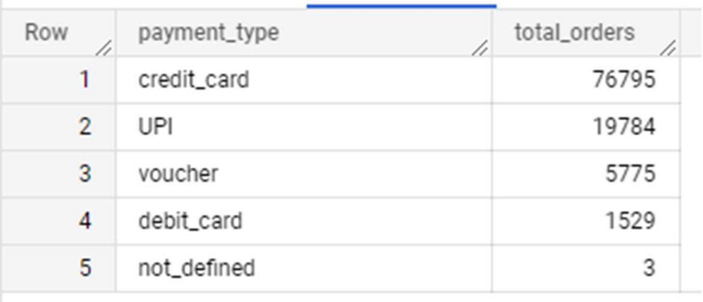
Query

```
with temp as (

select p.payment_type ,format_datetime('%B',order_purchase_timestamp) as month_name,extract(mont
h from order_purchase_timestamp) as month_no,count(o.order_id) count_of_orders

from `target_company_dataset.payments` p join `target_company_dataset.orders` o
on p.order_id=o.order_id
group by 1,2,3)
select payment_type,sum(count_of_orders) total_orders from temp
group by 1
order by 2 desc
```

Output of Query



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

Fig: Table 6.2

From output we can clearly see that, customers are using mostly credit card while purchasing the items, followed by UPI payment. Customers are hardly using debit card for purchase.

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

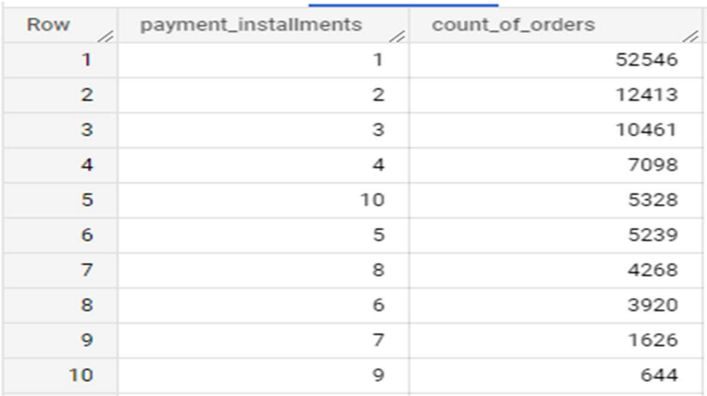
To get the required data we need to consider the 'payments' table,group the data on 'payment_installment' column and get the required Count of orders based on the no. of payment instalments. Consider the following query and the result obtained from it

Query

```
select payment_installments,count(order_id)count_of_orders

from `target_company_dataset.payments`
group by 1
order by 2 desc;
```

Output of Query



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

Fig: Table 6.3

The output given above is only up to 10 rows of whole output obtained. The above output obtained having count of orders arranged in descending order. The whole output can be obtained by running the query.

Insights

Since the output obtained is arranged in decreasing order of ‘count_of_orders’. We can clearly see that payment instalment with value 1 is having maximum number of orders. This shows the payment behaviour of customers, they tend to pay the total outstanding amount in one go. They don’t want to pay in instalments.

Recommendations

1.Since credit card is most popular method of payment, Company can think of giving lucrative discount on purchase through credit card in order to attract more customers. By doing this more customers will buy and also purchasing value per customers will also increase. (Refer to table 6.2)

2. Company can think of giving no cost EMI options on credit and debit card. This will push the customers to spend more as they need not to pay instantly. Also, no cost EMI options on debit card will attract customers doesn’t own a credit card. Accordingly, payment instalment will increase and revenue of company will also increase. (Refer to table 6.2 and 6.3)

3. Company can think of launching its own card and give the extra discount while purchasing through this card.