Business Case Study: Target SQL

Context

Target is a globally renowned brand and a prominent retailer in the United States. Target makes itself a preferred shopping destination by offering outstanding value, inspiration, innovation and an exceptional guest experience that no other retailer can deliver.

This particular business case focuses on the operations of Target in Brazil and provides insightful information about 100,000 orders placed between 2016 and 2018. The dataset offers a comprehensive view of various dimensions including the order status, price, payment and freight performance, customer location, product attributes, and customer reviews.

By analyzing this extensive dataset, it becomes possible to gain valuable insights into Target's operations in Brazil. The information can shed light on various aspects of the business, such as order processing, pricing strategies, payment and shipping efficiency, customer demographics, product characteristics, and customer satisfaction levels.

Index

Sl.No	Title	Page No.
1	Initial exploration like checking the structure & characteristics of the data	01
2	In-depth Exploration	04
3	Evolution of E-commerce orders in the Brazil region	07
4	Impact on Economy	10
5	Analysis on sales, freight and delivery time	13
6	Analysis based on the payments	18

1. Importing the Data set and doing Initial exploration like checking the structure & characteristics of the data :

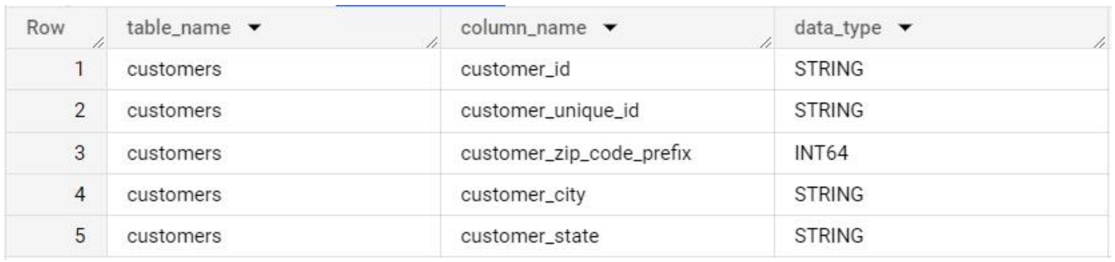
1.1. Data type of all columns in the "customers" table.

Details of the “customers” table with respective column name and data types is fetched using following query.

Query

```
select table_name, column_name, data_type
from`target.INFORMATION_SCHEMA.COLUMNS`
where table_name = "customers"
```

Output



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

Fig: Table 1.1

The output given above is only up to 5 rows of customers table. The ‘data_type’ column is showing the datatype of each column from ‘column_name’.

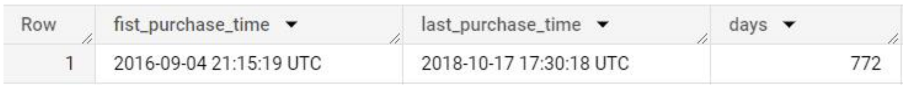
1.2. Get the time range between which the orders were placed.

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. And calculate the difference which is shown in following query.

Query

```
select min(o.order_purchase_timestamp) fist_purchase_time,
max(o.order_purchase_timestamp) last_purchase_time,
date_diff(max(o.order_purchase_timestamp),min(o.order_purchase_timestamp), day) as
days from `target.orders` o
```

Output



Row	fist_purchase_time	last_purchase_time	days
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC	772

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 and days refers to the overall days between these two dates.

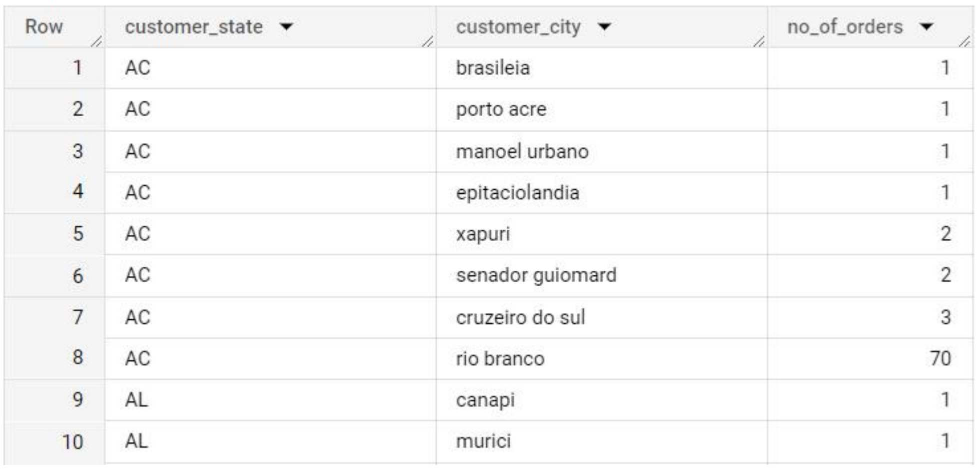
1.3. Count the Cities & States of customers who ordered during the given period.

To find the required record, we need to join the ‘customers’ and ‘orders’ table and extract the ‘customer_state’ and ‘customer_city’ column. And we have created ‘no_of_orders’ column which is showing the number of orders in each city of each state. Shown in the following query.

Query

```
select distinct c.customer_state, c.customer_city, count(o.order_id)as no_of_orders
from `target.customers` c left join `target.orders` o
on c.customer_id = o.customer_id
group by 1,2
order by 1,3
```

Output



Row	customer_state	customer_city	no_of_orders
1	AC	brasileia	1
2	AC	porto acre	1
3	AC	manoel urbano	1
4	AC	epitaciolandia	1
5	AC	xapuri	2
6	AC	senador guiomard	2
7	AC	cruzeiro do sul	3
8	AC	rio branco	70
9	AL	canapi	1
10	AL	murici	1

Fig: Table 1.3

The output given above is only up to 10 rows of whole output obtained. The 'customer_state' ---
And 'customer_city' refers to different state and city present in the country during the whole purchase period. And 'no_of_orders' show the number of order in each city.

Insights

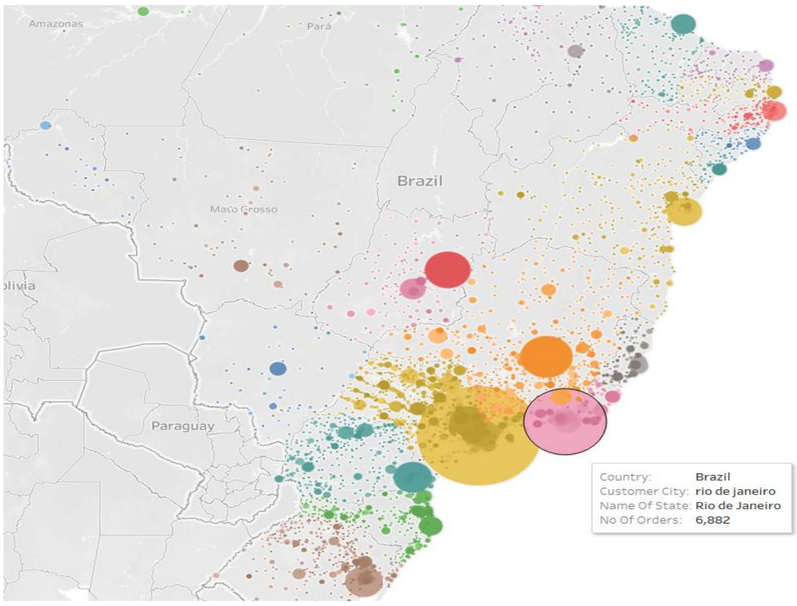


Fig: Map Chart 1.4

Considering the Map chart 1.4, we can see that this is the map of Brazil with different colourful circles of different sizes. Here different colour represents different states , the circles represents different city locations and size of the circle is showing the no. of orders given from different city's. As per example we can see the city 'rio de janeiro' has 6882 orders.

2. In-depth Exploration:

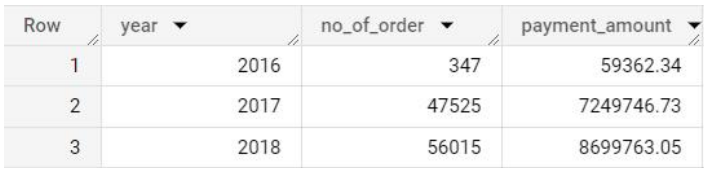
2.1. Is there a growing trend in the no. Of orders placed over the past years.

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. Then we have Extracted the year from the datetime column 'order_purchase_timestamp' as 'year' and we have grouped the no. of orders and payment with 'year' column.

Query

```
select extract(year from o.order_purchase_timestamp) as year, count(o.order_id) as
no_of_order, round(sum(p.payment_value),2) as payment_amount
from `target.orders` o left join `target.payments` p
on o.order_id = p.order_id
group by 1
order by 1
```

Output



Row	year	no_of_order	payment_amount
1	2016	347	59362.34
2	2017	47525	7249746.73
3	2018	56015	8699763.05

Fig: Table 2.1

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

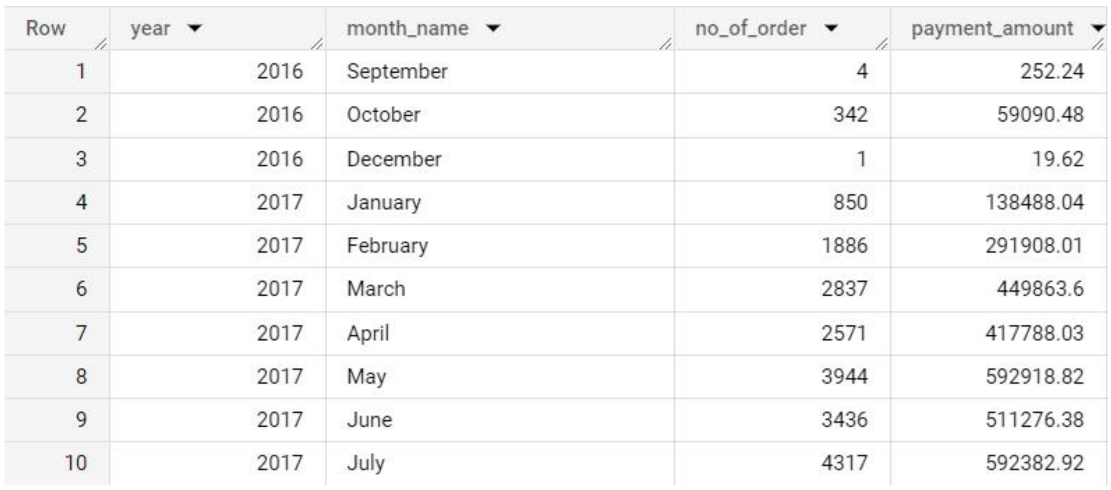
2.2. Can we see some kind of monthly seasonality in terms of the no. Of orders being placed ?

Now in order to get monthly seasonality of the data, we need to get the month wise data for each year to analyze the trend. So we have grouped the 'no_of_order' and 'payment_amount' columns using 'count' and 'sum' aggregate functions and ordered it by the year and month. Shown in the following query.

Query

```
with msi as
(select
extract(year from o.order_purchase_timestamp) as year,
extract(month from o.order_purchase_timestamp) as month,
format_date("%B", o.order_purchase_timestamp) as month_name ,
count(o.order_id) as no_of_order, round(sum(p.payment_value),2) as payment_amount
from `target.orders` o left join `target.payments` p
on o.order_id = p.order_id
group by 1,2,3
order by 1,2)
select year, month_name, no_of_order, payment_amount from msi
```

Output



Row	year	month_name	no_of_order	payment_amount
1	2016	September	4	252.24
2	2016	October	342	59090.48
3	2016	December	1	19.62
4	2017	January	850	138488.04
5	2017	February	1886	291908.01
6	2017	March	2837	449863.6
7	2017	April	2571	417788.03
8	2017	May	3944	592918.82
9	2017	June	3436	511276.38
10	2017	July	4317	592382.92

Fig: Table 2.2

The output given above is only up to 10 rows of complete output. It is showing the required month on month details of no of order placed and payment value obtained for each year.

Insights

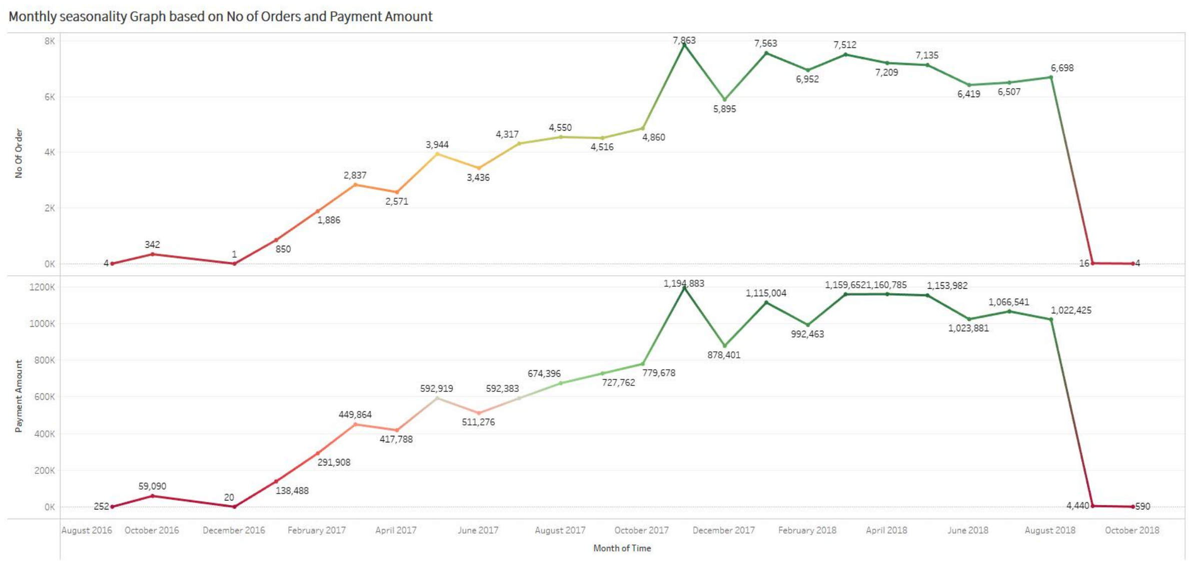


Fig: Graph Chart 2.3

Considering the graph 2.3, we can see that there is a overall gradual increase in 'No of Orders' and 'Payment Amount' from Dec 2016 to Nov 2017. And it is observed that purchasing trend peaked during Nov 2017 to August 2018, but after that both decreased sharply.

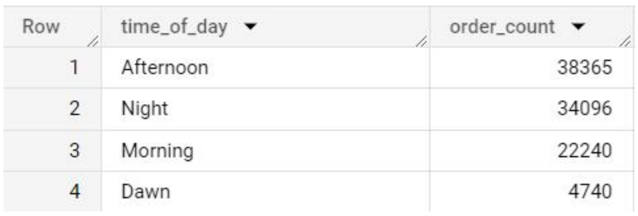
2.3. During what time of the day , do the Brazilian customers mostly place their orders ?

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 o.order_purchase_timestamp) between "00:00:00" and "06:00:00"
then "Dawn"
 when
  extract(time from o.order_purchase_timestamp) between "06:00:01" and "12:00:00"
then "Morning"
 when
  extract(time from o.order_purchase_timestamp) between "12:00:01" and "18:00:00"
then "Afternoon"
 when
  extract(time from o.order_purchase_timestamp) between "18:00:01" and "23:59:59"
then "Night"
 else "NA"
end as time_of_day, count(o.order_purchase_timestamp) as order_count from
`target.orders` o
group by 1
order by 2 desc
```

Output



Row	time_of_day	order_count
1	Afternoon	38365
2	Night	34096
3	Morning	22240
4	Dawn	4740

Fig: Table 2.4

We can see at the above fig.2.4 that Brazilian customers tend to buy mostly in afternoon, between '12:00:01' and '18:00:00' and the least active time is Dawn, between '00.00.00' and '06.00.00'

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 Brazil region:

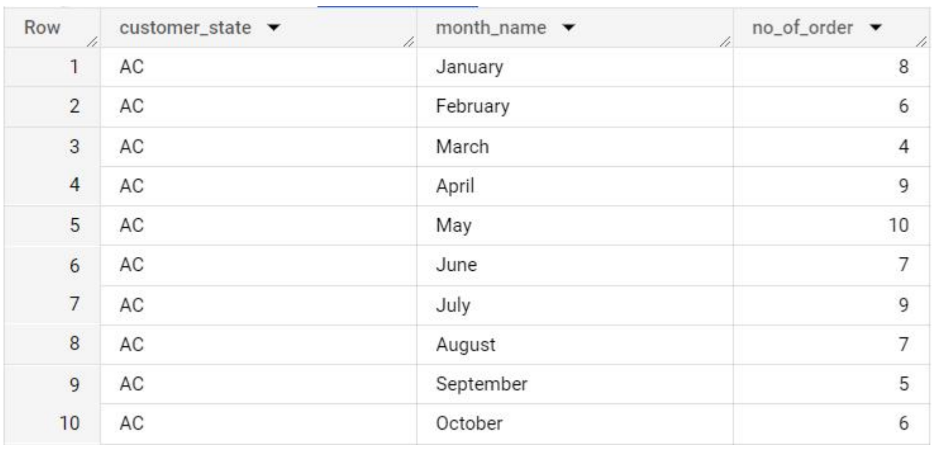
3.1. Get the month on month no. of orders placed in each state.

In order to get the required month on month data by each state, we need to join the ‘customers’ and ‘orders’ table and fetch the month on month number of orders placed in each state. Consider the following code and the output obtained.

Query

```
with mom as
(select c.customer_state, extract(month from o.order_purchase_timestamp) as
month ,format_date("%B", o.order_purchase_timestamp) as month_name,
count(o.order_id) as no_of_order from `target.customers` c join `target.orders` o
on c.customer_id = o.customer_id
group by 1,2,3
order by 1,2)
select customer_state, month_name, no_of_order from mom
```

Output

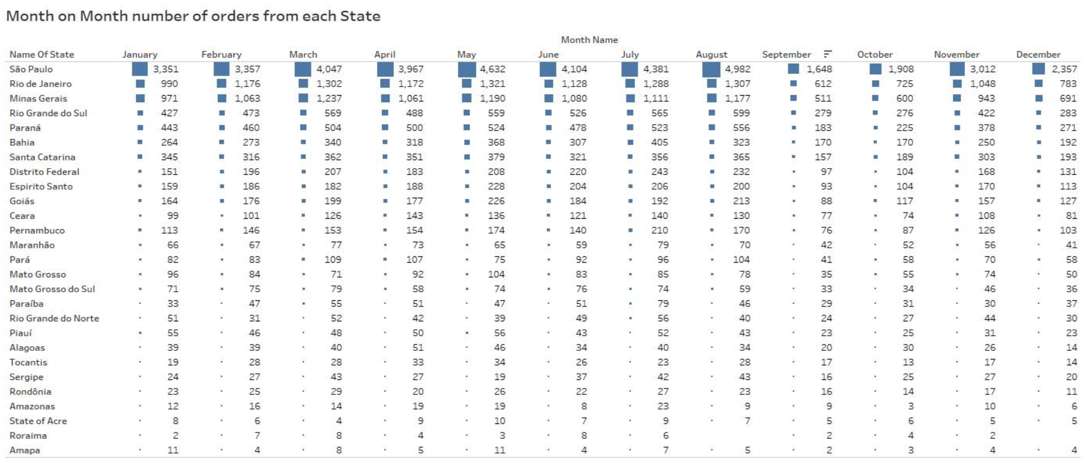


Row	customer_state	month_name	no_of_order
1	AC	January	8
2	AC	February	6
3	AC	March	4
4	AC	April	9
5	AC	May	10
6	AC	June	7
7	AC	July	9
8	AC	August	7
9	AC	September	5
10	AC	October	6

Fig: Table 3.1

The output given above is only up to 10 rows of complete output obtained. We can clearly see the month on month number of orders placed in each state.

Insights



Month on Month number of orders from each State

	Month Name											
Name Of State	January	February	March	April	May	June	July	August	September	October	November	December
São Paulo	3,351	3,357	4,047	3,967	4,632	4,104	4,381	4,982	1,648	1,908	3,012	2,357
Rio de Janeiro	990	1,176	1,302	1,172	1,321	1,128	1,288	1,307	612	725	1,048	783
Minas Gerais	971	1,063	1,237	1,061	1,190	1,080	1,111	1,177	511	600	943	691
Rio Grande do Sul	427	473	569	488	559	526	565	599	279	276	422	283
Paraná	443	460	504	500	524	478	523	556	183	225	378	271
Bahia	264	273	340	318	368	307	405	323	170	170	250	192
Santa Catarina	345	316	362	351	379	321	356	365	157	189	303	193
Distrito Federal	151	196	207	183	208	220	243	232	97	104	168	131
Espirito Santo	159	186	182	188	228	204	206	200	93	104	170	113
Goiás	164	176	199	177	226	184	192	213	88	117	157	127
Ceara	99	101	126	143	136	121	140	130	77	74	108	81
Pernambuco	113	146	153	154	174	140	210	170	76	87	126	103
Maranhão	66	67	77	73	65	59	79	70	42	52	56	41
Pará	82	83	109	107	75	92	96	104	41	58	70	58
Mato Grosso	96	84	71	92	104	83	85	78	35	55	74	50
Mato Grosso do Sul	71	75	79	58	74	76	74	59	33	34	46	36
Paraíba	33	47	55	51	47	51	79	46	29	31	30	37
Rio Grande do Norte	51	31	52	42	39	49	56	40	24	27	44	30
Piauí	55	46	48	50	56	43	52	43	23	25	31	23
Alagoas	39	39	40	51	46	34	40	34	20	30	26	14
Tocantis	19	28	28	33	34	26	23	28	17	13	17	14
Sergipe	24	27	43	27	19	37	42	43	16	25	27	20
Rondônia	23	25	29	20	26	22	27	23	16	14	17	11
Amazonas	12	16	14	19	19	8	23	9	9	3	10	6
State of Acre	8	6	4	9	10	7	9	7	5	6	5	5
Roraima	2	7	8	4	3	8	6		2	4	2	
Amapa	11	4	8	5	11	4	7	5	2	3	4	4

Fig: Heat Map 3.2

Considering the graph 3.2, we can see that ‘Sao Paulo’ or ‘SP’ has the highest number of orders placed in each month and August is the month of highest orders and September is the lowest.

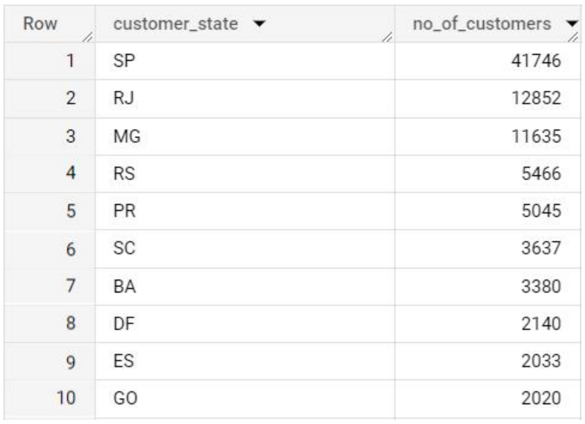
3.2. How are the customers distributed across all the states ?

In order to get the required data we need to use the ‘customers’ table, then we need to count ‘customer_id’ for each state. Consider the following code and the output obtained.

Query

```
select c.customer_state , count(c.customer_id) as no_of_customers
from `target.customers` c
group by 1
order by 2 desc
```

Output



Row	customer_state	no_of_customers
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

Fig: Table 3.3

The output given above is only up to 10 rows of complete output obtained. We can clearly see the state wise distribution of customers.

Insights

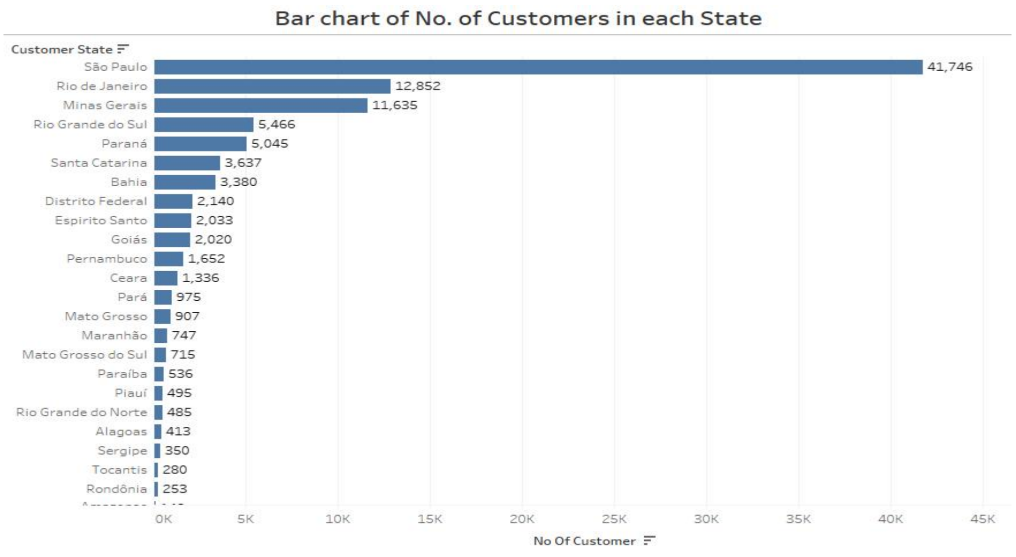


Fig: Bar Chart 3.4

Considering the Bar chart 3.4, the chart shows the number of customer present in each state. It can be observed that state ‘SP’ or ‘Sao Paulo’ has highest no of customer (41746).

Recommendations

1. Since March to August are among 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. Company 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(Analyze the money movement by E-commerce by looking at order prices, freight and others):

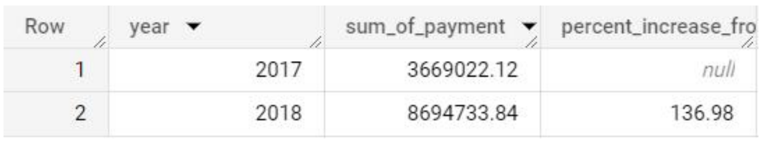
4.1. Get the % increase in the cost of orders from year 2017 to 2018 (including months between Jan to Aug only). You can use the 'payments_value' column in the 'payments' table to get the cost of orders.

To solve this problem, we need to join 'orders' and 'payments' table, then 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 calculate the percentage increase. For that consider the following query.

Query

```
with mom as
(select extract(year from o.order_purchase_timestamp) as year,
round(sum(p.payment_value),2) as sum_of_payment from `target.orders` o left join
`target.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 1
order by 1)
select *, round(((sum_of_payment - lag(sum_of_payment,1)over(order by year))/
lag(sum_of_payment,1)over(order by year)*100),2) as
percent_increase_from_previous_year from mom
order by year
```

Output



Row	year	sum_of_payment	percent_increase_fro
1	2017	3669022.12	null
2	2018	8694733.84	136.98

Fig: Table 4.1

From the output obtained it can be concluded that the revenue of company from selling the products have increase significantly by around 136.98 % 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.

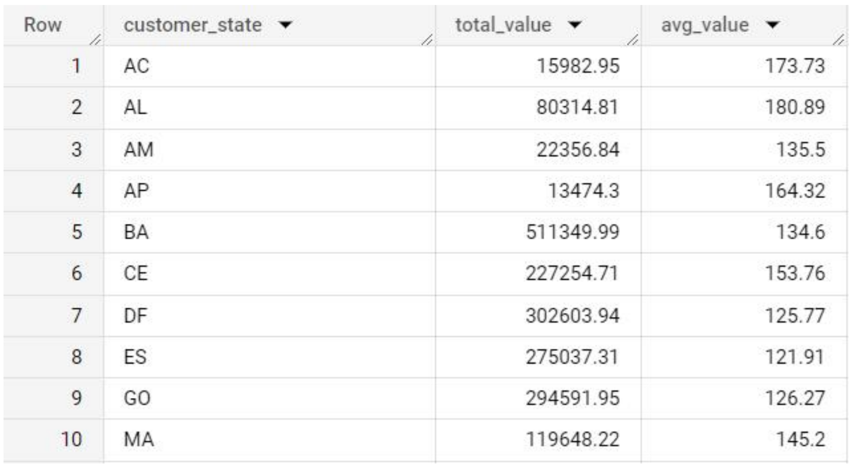
4.2. Calculate the Total & Average value of order price for each state ?

In order to get the required data we need to join 'customers' , 'orders' and 'order_items' table, then we will use aggregate functions to fetch the total and average amount. Consider the following query.

Query

```
select c.customer_state, round(sum(oi.price),2) as total_value ,
round(avg(oi.price),2) as avg_value
from `target.customers` c left join `target.orders` o
on c.customer_id = o.customer_id left join `target.order_items` oi
on o.order_id = oi.order_id
group by 1
order by 1
```

Output



Row	customer_state	total_value	avg_value
1	AC	15982.95	173.73
2	AL	80314.81	180.89
3	AM	22356.84	135.5
4	AP	13474.3	164.32
5	BA	511349.99	134.6
6	CE	227254.71	153.76
7	DF	302603.94	125.77
8	ES	275037.31	121.91
9	GO	294591.95	126.27
10	MA	119648.22	145.2

Fig: Table 4.2

The output given above is only up to 10 rows of whole output obtained. In the output we can see the Total and Average values in each state is shown.

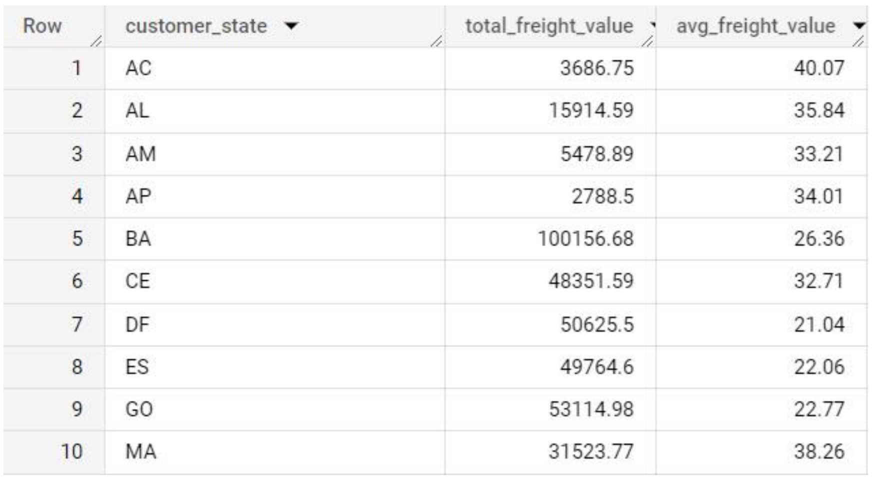
4.3. Calculate the Total & Average value of order freight for each state ?

In order to get the required data we need to join ‘customers’ , ‘orders’ and ‘order_items’ table, then we will use aggregate functions to fetch the total and average amount. Consider the following query

Query

```
select c.customer_state, round(sum(oi.freight_value),2) as total_freight_value ,
round(avg(oi.freight_value),2) as avg_freight_value
from `target.customers` c left join `target.orders` o
on c.customer_id = o.customer_id left join `target.order_items` oi
on o.order_id = oi.order_id
group by 1
order by 1
```

Output



Row	customer_state	total_freight_value	avg_freight_value
1	AC	3686.75	40.07
2	AL	15914.59	35.84
3	AM	5478.89	33.21
4	AP	2788.5	34.01
5	BA	100156.68	26.36
6	CE	48351.59	32.71
7	DF	50625.5	21.04
8	ES	49764.6	22.06
9	GO	53114.98	22.77
10	MA	31523.77	38.26

Fig: Table 4.3

The output given above is only up to 10 rows of whole output obtained. In the output we can see the Total and Average freight values in each state is shown.

Insights

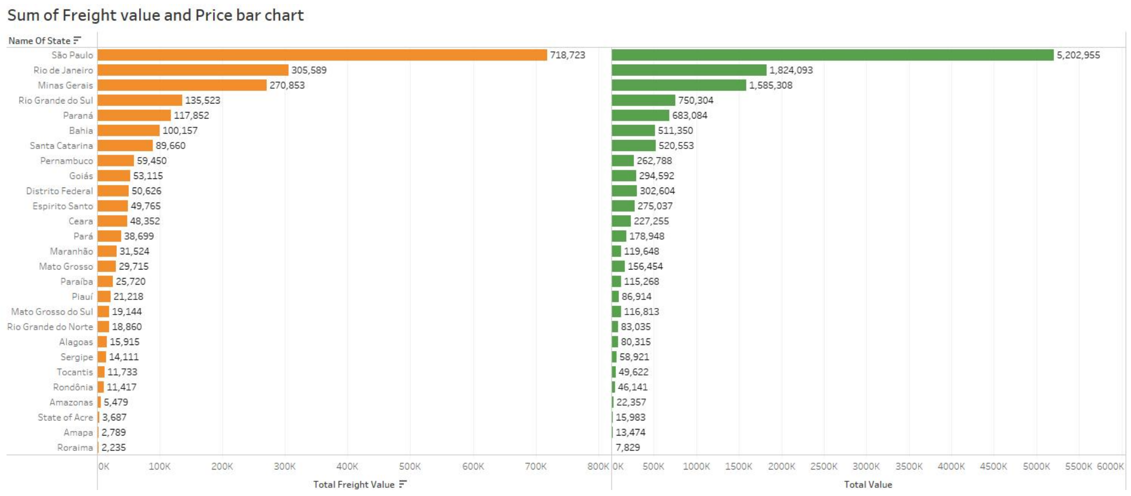


Fig: Bar Chart 4.4

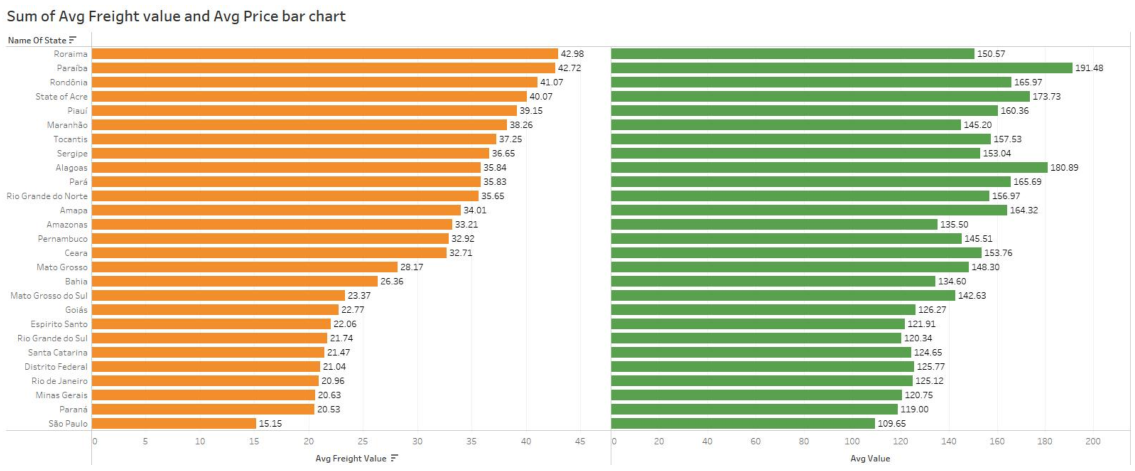


Fig: Bar Chart 4.5

The above Fig: 4.4 and 4.5 shows the output from Q.4.2 and Q.4.3 in a bar chart. We can see in Fig:4.4 both Total Freight value and Total value is the highest in 'Sao Paulo', which is far more than any other state. And in Fig:4.5 'Avg Freight value' and 'Avg value' is mostly similar.

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 Bar chart 4.5 shows there are some states like 'Sao Paulo' , where sum of freight value and total price is very high, So 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 based on sales , freight and delivery time:

5.1. Find the no. of days taken to deliver each order from the order's purchase date as delivery time. Also calculate the difference (in days) between the estimated & actual delivery date of an order.

In order to calculate the delivery time and the difference between the estimated & actual delivery date we can use the given formula:

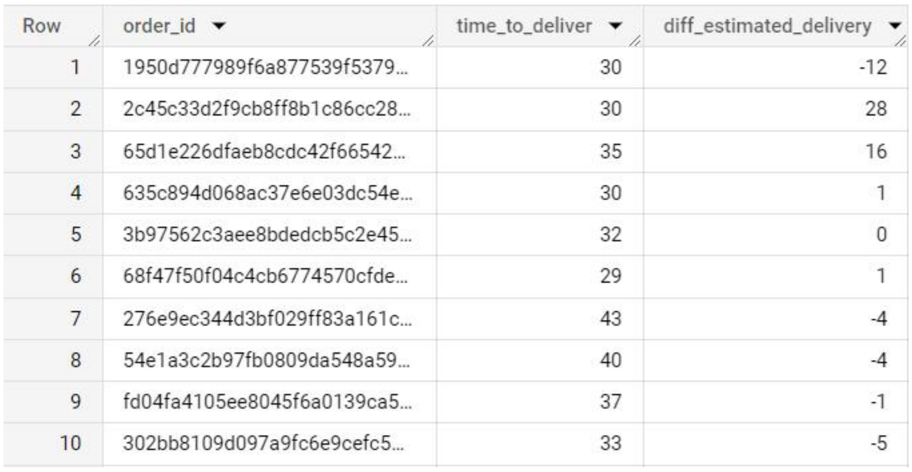
- ➢ Time_to_deliver = (order_delivered_customer_date - order_purchase_timestamp)
- ➢ Diff_estimated_delivery = (order_estimated_delivery_date - order_delivered_customer_date)

Consider the following query to fetch the required output

Query

```
select o.order_id , date_diff(o.order_delivered_customer_date,
o.order_purchase_timestamp, day) as time_to_deliver,
date_diff(o.order_estimated_delivery_date, o.order_delivered_customer_date, day) as
diff_estimated_delivery  from `target.orders` o
```

Output



Row	order_id	time_to_deliver	diff_estimated_delivery
1	1950d777989f6a877539f5379...	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	30	28
3	65d1e226dfaeb8cdc42f66542...	35	16
4	635c894d068ac37e6e03dc54e...	30	1
5	3b97562c3aee8bdedcb5c2e45...	32	0
6	68f47f50f04c4cb6774570cfde...	29	1
7	276e9ec344d3bf029ff83a161c...	43	-4
8	54e1a3c2b97fb0809da548a59...	40	-4
9	fd04fa4105ee8045f6a0139ca5...	37	-1
10	302bb8109d097a9fc6e9cefc5...	33	-5

Fig: Table 5.1

The output given above is only up to 10 rows of whole output obtained. The output show's the time of delivery as 'time_to_deliver' and difference of estimated and delivery date as 'diff_estimated_delivery' of each order.

5.2. Find out the top 5 states with the highest & lowest average freight value.

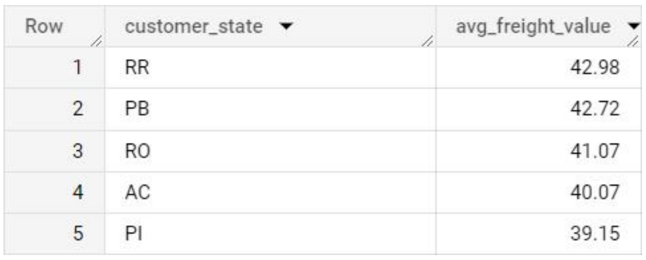
In order to get the required data we need to join the 'customers', 'orders' and 'order_items' table in both the cases of highest and lowest

❖ **Highest :**

Query

```
select c.customer_state, round(avg(oi.freight_value),2) as avg_freight_value from
`target.customers` c join `target.orders` o
on c.customer_id = o.customer_id join `target.order_items` oi
on o.order_id = oi.order_id
group by 1
order by 2 desc
limit 5
```

Output



Row	customer_state	avg_freight_value
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15

Fig: Table 5.2

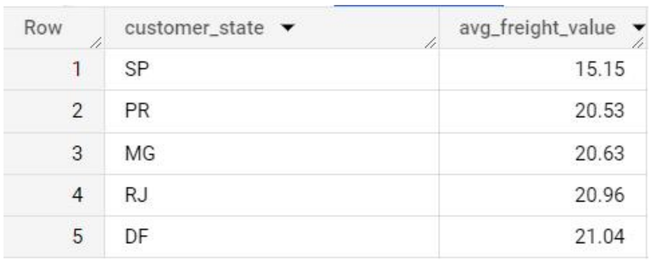
These are the top 5 states with highest average freight value

❖ **Lowest :**

Query

```
select c.customer_state, round(avg(oi.freight_value),2) as avg_freight_value from
`target.customers` c join `target.orders` o
on c.customer_id = o.customer_id join `target.order_items` oi
on o.order_id = oi.order_id
group by 1
order by 2
limit 5
```

Output



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

Fig: Table 5.3

These are the top 5 states with lowest average freight value

5.3. Find out the top 5 states with the highest & lowest average delivery time.

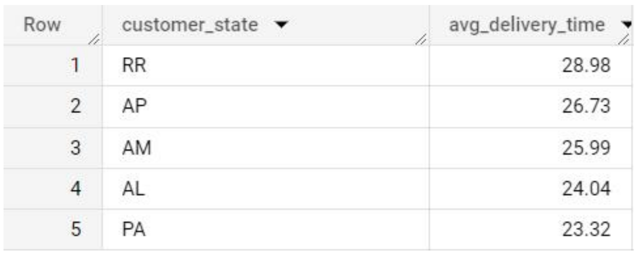
In order to get the required data we need to join the 'customers' and 'orders' table in both the cases of highest and lowest

❖ **Highest :**

Query

```
select c.customer_state,
round(avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,day))
,2) as avg_delivery_time
from `target.customers` c right join `target.orders` o
on c.customer_id = o.customer_id
group by 1
order by 2 desc
limit 5
```

Output



Row	customer_state	avg_delivery_time
1	RR	28.98
2	AP	26.73
3	AM	25.99
4	AL	24.04
5	PA	23.32

Fig: Table 5.4

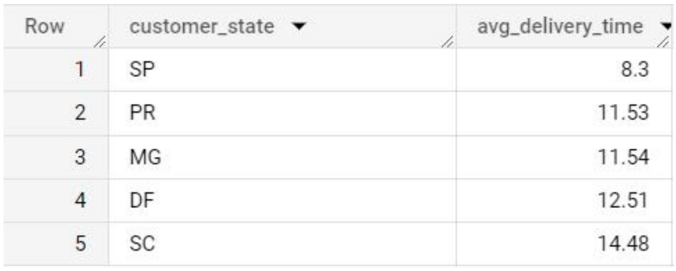
These are the top 5 states with highest average delivery value

❖ **Lowest :**

Query

```
select c.customer_state,
round(avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,day))
,2) as avg_delivery_time
from `target.customers` c right join `target.orders` o
on c.customer_id = o.customer_id
group by 1
order by 2
limit 5
```

Output



Row	customer_state	avg_delivery_time
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48

Fig: Table 5.5

These are the top 5 states with lowest average delivery value

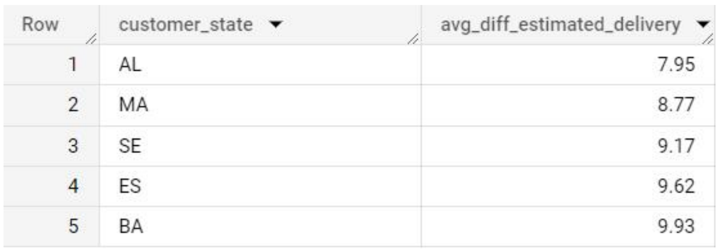
5.4. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

In order to find the required result we can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state, and then we will only fetch top 5 states with fastest delivery. Consider the following query for the required output.

Query

```
select
c.customer_state,round(avg(date_diff(o.order_estimated_delivery_date,o.order_delive
red_customer_date,day)),2) as avg_diff_estimated_delivery
from `target.customers` c right join `target.orders` o
on c.customer_id = o.customer_id
group by 1
order by 2
limit 5
```

Output



Row	customer_state	avg_diff_estimated_delivery
1	AL	7.95
2	MA	8.77
3	SE	9.17
4	ES	9.62
5	BA	9.93

Fig: Table 5.6

These are the top 5 states where the order delivery is really fast compared to estimated delivery, the averages of the difference between actual delivery & estimated delivery is shown in 'avg_diff_estimated_delivery' in days, lesser days the delivery took, greater the accuracy is, in terms of the average difference between the estimated and delivery time.

Insights

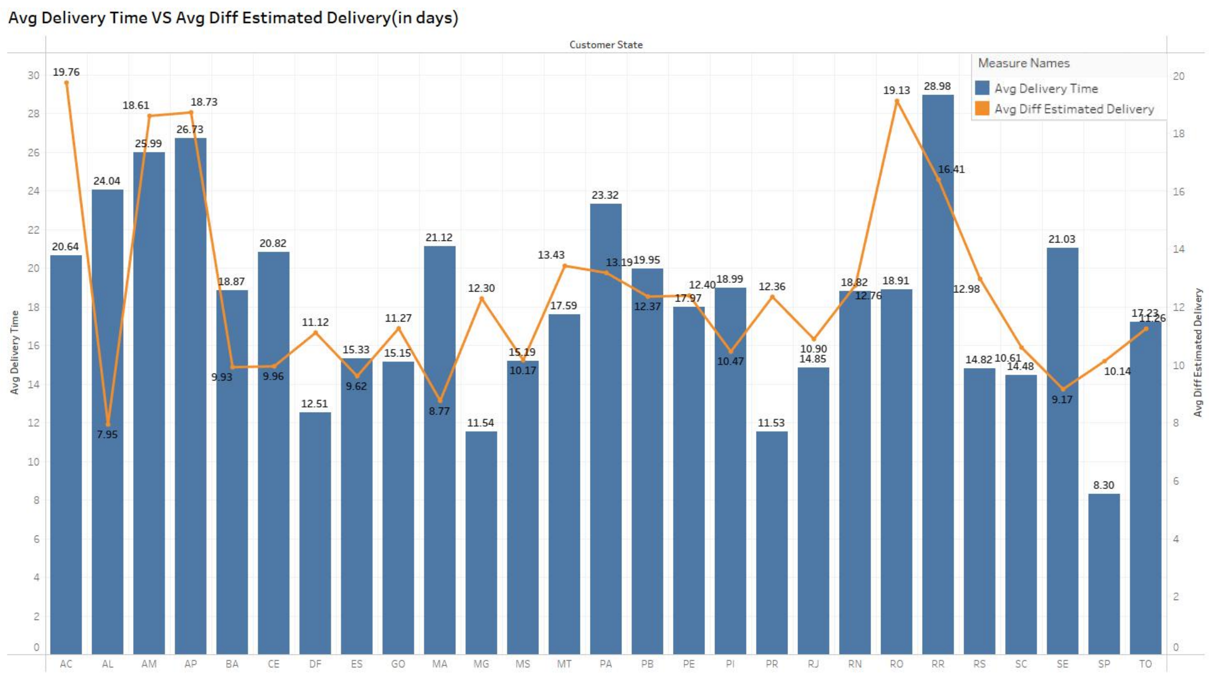


Fig: Bar Chart 5.7

The above Fig: 5.7 represents the 'avg_delivery_time ' and 'avg_diff_estimated_delivery ' in a Bar Chart and Line Chart format, blue Bar chart represents the average delivery time and the yellow Line Chart represents the average difference between delivery and estimated delivery date. So, by looking at this chart we can see that :-

1. If the avg delivery time of each state is more then eventually purchasing experience of the customer will not be good also, which will effect to generate the revenue for the company.

2. States where delivery is really fast and 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.

3. Avg difference between estimated delivery and actual delivery has a high fluctuation in every state which is not good in terms of customer satisfaction in each state.

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.

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.

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 5.5)

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. Analysis based on the Payments:

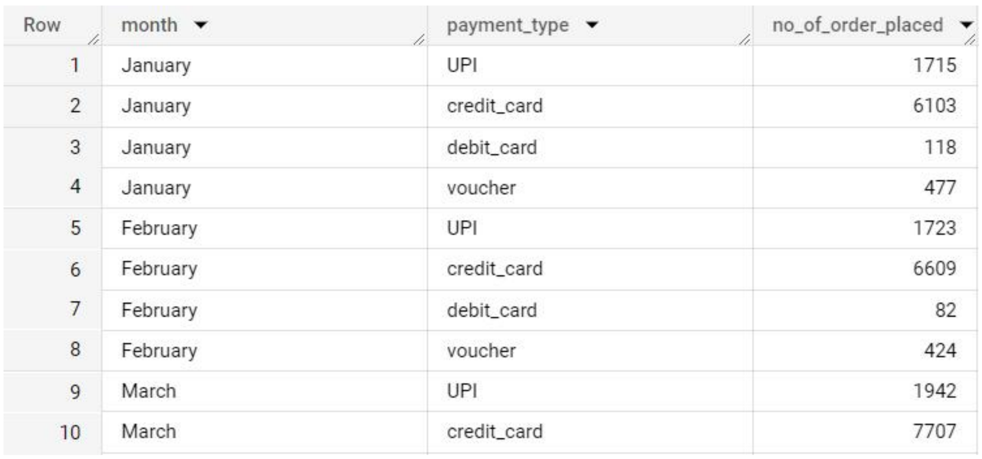
6.1. Find the month on month no. of orders placed using different payment types.

To get the required data we need to join ‘payments’ and ‘orders’ table. Then we need to count the orders as ‘order_id’ and group it by ‘payment_type’ and ‘month’. The following query shows the number of orders placed through each payment type in each month.

Query

```
with mom as
(select format_date("%B", o.order_purchase_timestamp) as month, extract(month from
o.order_purchase_timestamp) as int_month, p.payment_type , count(o.order_id) as
no_of_order_placed
from `target.payments` p left join `target.orders` o
on p.order_id = o.order_id
group by 1,2,3
order by 2,3)
select month, payment_type, no_of_order_placed from mom
```

Output



Row	month	payment_type	no_of_order_placed
1	January	UPI	1715
2	January	credit_card	6103
3	January	debit_card	118
4	January	voucher	477
5	February	UPI	1723
6	February	credit_card	6609
7	February	debit_card	82
8	February	voucher	424
9	March	UPI	1942
10	March	credit_card	7707

Fig: Table 6.1

The above output is showing only up to 10 rows of whole output obtained. By looking at this part of the output we can clearly understand that every month has it’s payment types and each payment type has it’s number of orders shown in the output.

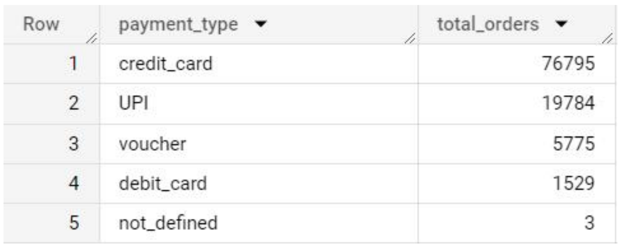
And to see the number of orders in each payment type we need to write a second query.

Query

```
with mom as
(select format_date("%B", o.order_purchase_timestamp) as month, extract(month from
o.order_purchase_timestamp) as int_month, p.payment_type , count(o.order_id) as
no_of_order_placed
from `target.payments` p left join `target.orders` o
on p.order_id = o.order_id
group by 1,2,3
order by 2,3)
select payment_type,sum(no_of_order_placed) total_orders from mom
```

```
group by 1
order by 2 desc
```

Output



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

The above output is showing the number of orders been paid in each payment type, we can clearly see that the highest number of orders were paid through 'credit_card' but 3 numbers of transactions is 'not_defined'.

Insights

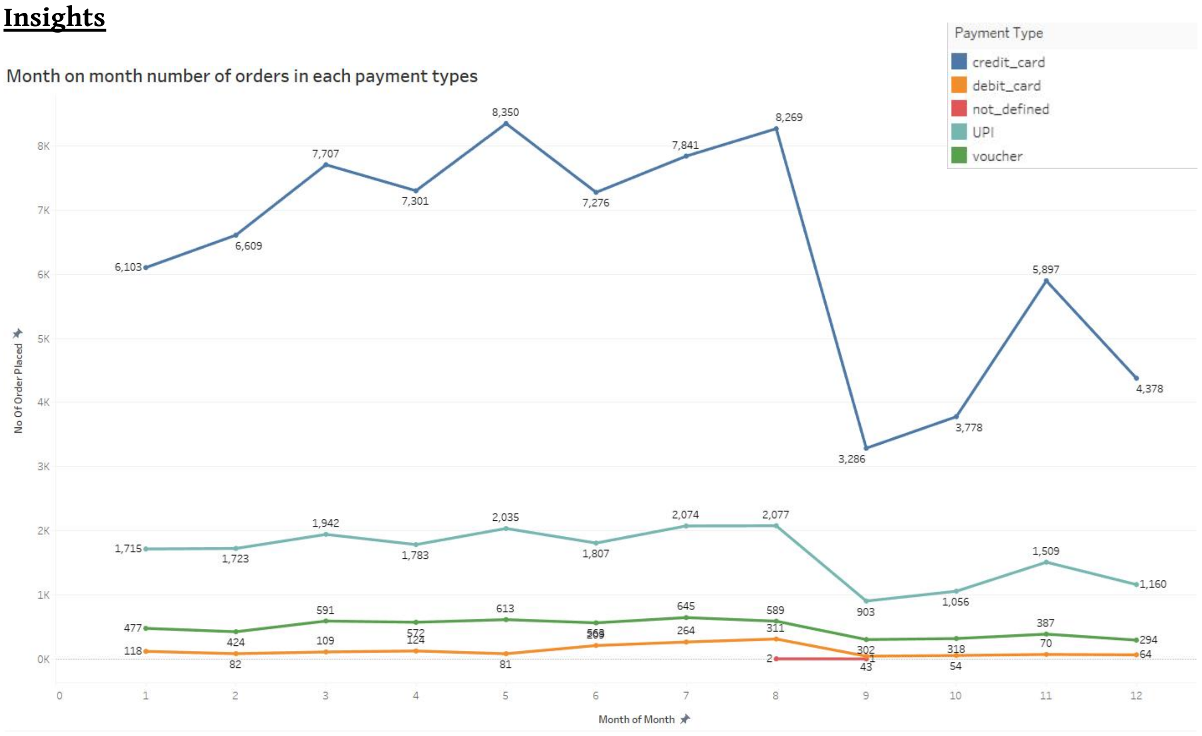


Fig: Graph Chart 6.3

The above Fig: 6.3 represents a Graph Chart, which is showing the month on month number of orders placed using different payment types. Here each graph represents different types of payment methods with different colors, which is shown in the top right corner of the graph as 'Payment Type'. So, by looking at this graph chart we can consider some conclusions :-

1. The 'credit card' remains the preferred payment method throughout the year, so majority of customers opt for credit card payments methods.

2. We can clearly see that from August to September there has been a decrease in the number of orders across all payment types, so we can say at that time customers were buying less products at that time.

3. We can also see that there is small number (3) of transaction happened at the time between August to September which is ‘not_defined’.

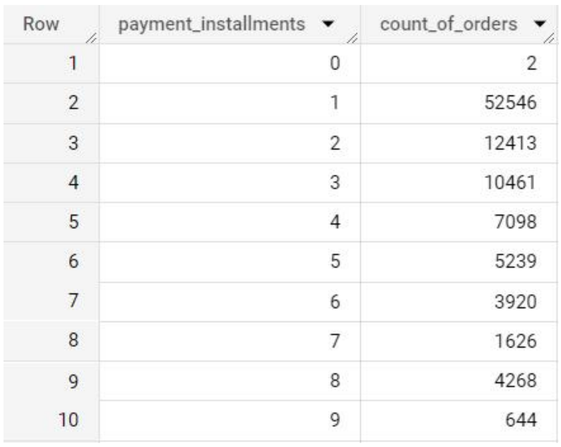
6.2. Find the no. of orders placed on the basis of the payment installments that have been paid.

To get the required data we need to consider the ‘payments’ table, and count all the orders as ‘order_id’ and group it on ‘payment_installment’ column and get the required data of number of orders placed on the basis of payment installments. Consider the following query and the output obtained.

Query

```
select p.payment_installments,count(p.order_id) as count_of_orders
from `target.payments` p
group by 1
order by 1
```

Output



Row	payment_installments	count_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

Fig: Table 6.4

The output given above is only up to 10 rows. The above output obtained shows the number of orders places as ‘count_of_orders’ for each number of payment installments.

Insights

We can clearly see that 1 payment instalment has the maximum number of orders. So, it shows the payment behaviour of customers, that they tend to pay the total amount at once. They don’t want to pay in installments.

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 6.4)

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