TARGET CUSTOMER ANALYSIS :
A REPORT ON THE ANALYSIS OF CUSTOMER BEHAVIOUR AND COMPANY GROWTH

- **DATATYPES :**

Lets check the data types of the columns .
Tbale-wise information for the database.

SQL Server Query :

```
USE "TargetCustomerAnalysis"
SELECT
TABLE_CATALOG,
TABLE_SCHEMA,
TABLE_NAME,
COLUMN_NAME,
DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS;
```

Query Result :

Results | Messages

	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	DATA_TYPE
1	TargetCustomerAnalysis	dbo	order_items	"order_id"	nvarchar
2	TargetCustomerAnalysis	dbo	order_items	"order_item_id"	nvarchar
3	TargetCustomerAnalysis	dbo	order_items	"product_id"	nvarchar
4	TargetCustomerAnalysis	dbo	order_items	"seller_id"	nvarchar
5	TargetCustomerAnalysis	dbo	order_items	"shipping_limit_date"	nvarchar
6	TargetCustomerAnalysis	dbo	order_items	"price"	nvarchar
7	TargetCustomerAnalysis	dbo	order_items	"freight_value"	nvarchar
8	TargetCustomerAnalysis	dbo	customers	"customer_id"	nvarchar
9	TargetCustomerAnalysis	dbo	customers	"customer_unique_id"	nvarchar
10	TargetCustomerAnalysis	dbo	customers	"customer_zip_code_prefix"	nvarchar
11	TargetCustomerAnalysis	dbo	customers	"customer_city"	nvarchar
12	TargetCustomerAnalysis	dbo	customers	"customer_state"	nvarchar
13	TargetCustomerAnalysis	dbo	geolocation	"geolocation_zip_code_prefix"	nvarchar
14	TargetCustomerAnalysis	dbo	geolocation	"geolocation_lat"	nvarchar
15	TargetCustomerAnalysis	dbo	geolocation	"geolocation_lng"	nvarchar

- **TIME RANGE**

The time range for which the data is available.

SQL Server Query :

```
select min(["order_purchase_timestamp"]) as first_order_date, max(["order_purchase_timestamp"]) as last_order_date
from TargetCustomerAnalysis.dbo.orders;
```

Query Result :

Results | Messages

	first_order_date	last_order_date
1	2016-09-04 21:15:19	2018-10-17 17:30:18

So, we can see that first order purchased on 4th September , 2016 and the last order purchased on 17th of October in 2018. Amost two years of data records are registered in the database.

- **CUSTOMER DISTRIBUTION**

Let's look at state and city wise customer counts first.

SQL Server Query :

```
select temp2.*, round(((city_count*1.0 / state_count)*100),2) as city_count_percent
from
(select temp.*, sum(city_count) over(partition by ["customer_state"]) as state_count
from(
select ["customer_state"],["customer_city"], count(distinct ["customer_unique_id"]) city_count
from TargetCustomerAnalysis.dbo.customers
group by ["customer_state"],["customer_city"]
)temp)temp2
order by state_count desc, city_count desc;
```

Query Result :

	"customer_state"	"customer_city"	city_count	state_count	city_count_percent
1	SP	sao paulo	14984	40345	37.140000000000
2	SP	campinas	1398	40345	3.470000000000
3	SP	guarulhos	1153	40345	2.860000000000
4	SP	sao bernardo do campo	908	40345	2.250000000000
5	SP	santo andre	768	40345	1.900000000000
6	SP	osasco	717	40345	1.780000000000
7	SP	santos	692	40345	1.720000000000
8	SP	sao jose dos campos	666	40345	1.650000000000
9	SP	sorocaba	610	40345	1.510000000000
10	SP	jundiai	547	40345	1.360000000000
11	SP	ribeirao preto	489	40345	1.210000000000
12	SP	barueri	419	40345	1.040000000000
13	SP	mogi das cruzes	371	40345	0.920000000000
14	SP	piracicaba	360	40345	0.890000000000
15	SP	sao jose do rio preto	330	40345	0.820000000000

	"customer_state"	"customer_city"	city_count	state_count	city_count_percent
625	SP	bora	1	40345	0.000000000000
626	SP	boraceia	1	40345	0.000000000000
627	SP	alfredo marcondes	1	40345	0.000000000000
628	SP	ajapi	1	40345	0.000000000000
629	SP	agisse	1	40345	0.000000000000
630	RJ	rio de janeiro	6620	12396	53.400000000000
631	RJ	niteroi	811	12396	6.540000000000
632	RJ	nova iguacu	432	12396	3.480000000000
633	RJ	sao goncalo	399	12396	3.220000000000
634	RJ	duque de caxias	262	12396	2.110000000000
635	RJ	campos dos goytacaz	235	12396	1.900000000000

So, Sao Paolo, Rio de Janeiro and Minas Gerais have the most customers. And Sao Paolo, Rio de Janeiro and Belo Horizonte are the top 3 cities in Sao Paolo, Rio de Janeiro and Minas Gerais respectively with respect to total customer counts.

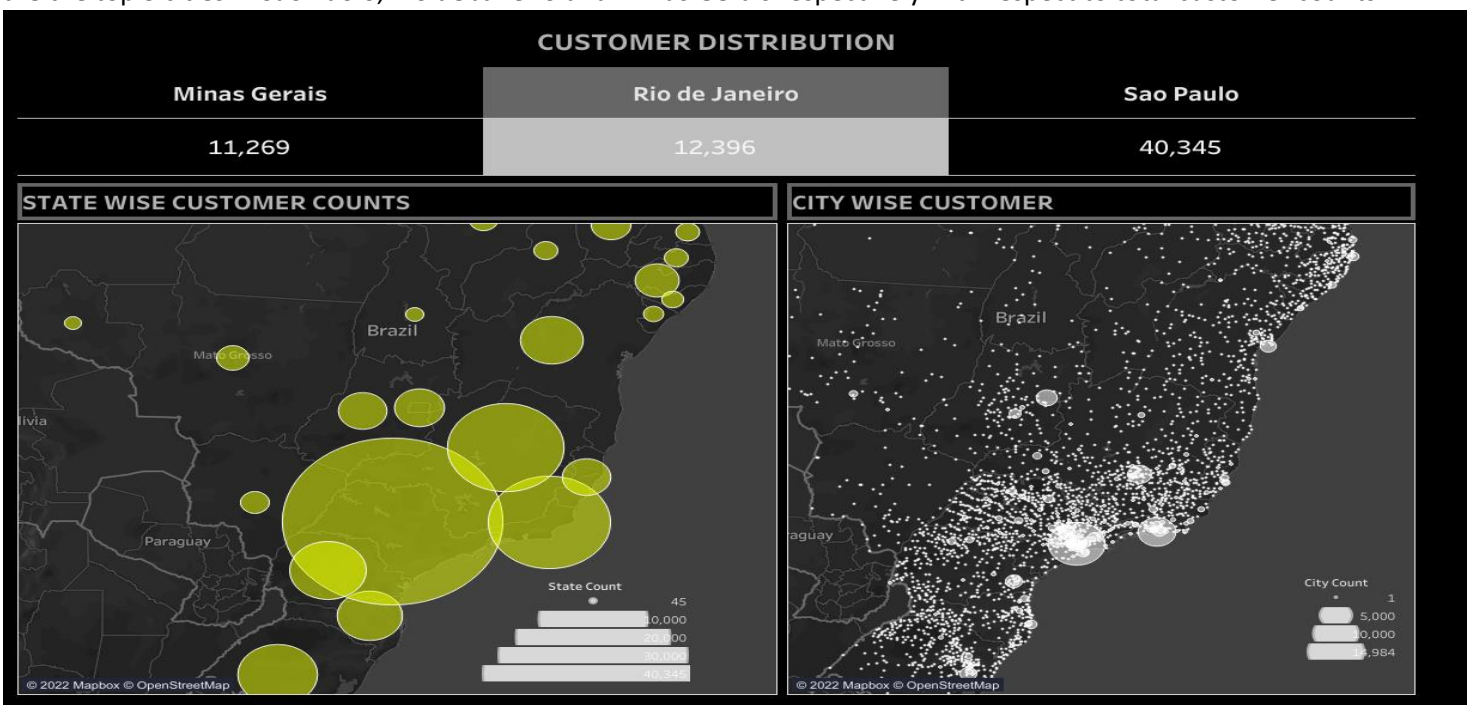


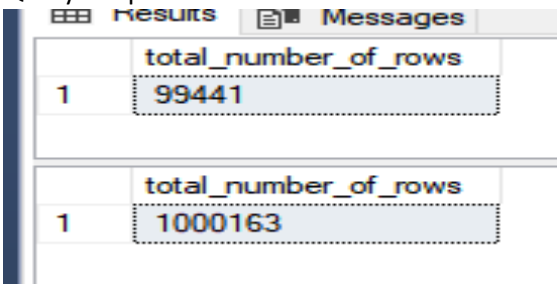
Tableau Dashboard : https://public.tableau.com/app/profile/writabrata.dey/viz/CUSTOMERDISTRIBUTION/Dashboard1

Here to resolve the unknowns I tried to use the geolocation table. But it seems geolocation table has 10,00,163 records while the customers table contains 99,441, so basically geolocations table contains all possible zip codes latitude and longitude. So the map is being plotted using customers table information only.

SQL Server Query :

```
-- Total number of records in geolocation and customer tables.
-- TOTAL ROWS in customers table is 99,441
select max(row_num) as total_number_of_rows
from
(select
ROW_NUMBER() over(order by ["customer_state"]) as row_num
from TargetCustomerAnalysis.dbo.customers)x

-- TOTAL ROWS in geolocation table is 10,00,163
select max(row_num) as total_number_of_rows
from
(select
ROW_NUMBER() over(order by ["geolocation_state"]) as row_num
from TargetCustomerAnalysis.dbo.geolocation)x
```

Query Output :



- **TREND CURVE**

Lets try to understand the trends about the company over the two years.

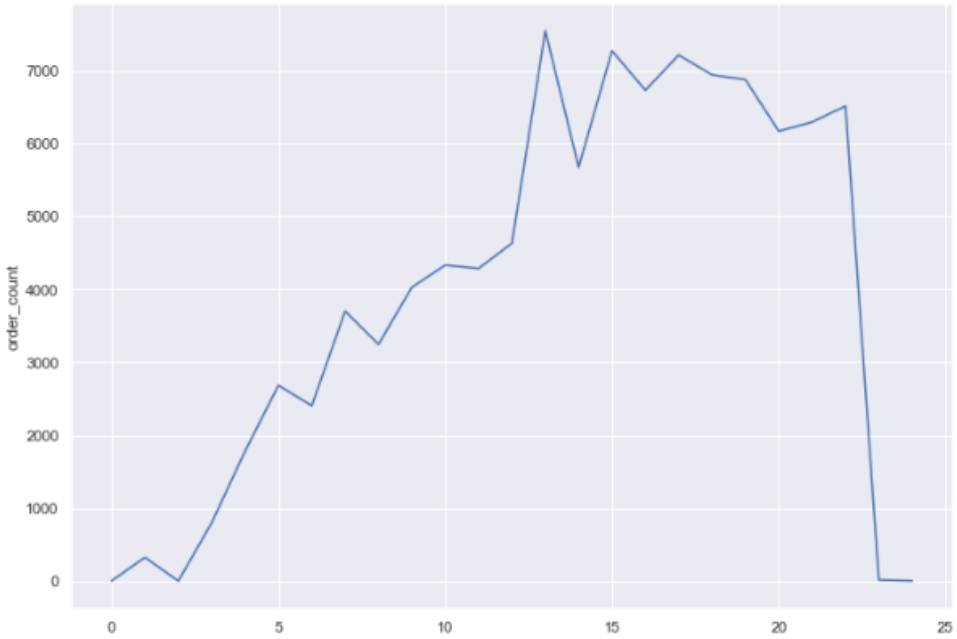
SQL Server Query :

```
select temp2.*,
(((temp2.order_count - temp2.previous_month_order_count) * 1.0) / nullif(temp2.previous_month_order_count,0) )*100 as
growth_percent
from
(select temp.*,
lag(order_count,1,0) over(order by purchase_year,month_number) as previous_month_order_count
from
(select DATENAME(YEAR,["order_purchase_timestamp"]) as purchase_year,
DATENAME(month, ["order_purchase_timestamp"]) as purchase_month, DATEPART(MM,["order_purchase_timestamp"]) as
month_number,
count(distinct ["order_id"]) as order_count
from TargetCustomerAnalysis.dbo.orders
group by  DATENAME(YEAR,["order_purchase_timestamp"]), DATENAME(month,
["order_purchase_timestamp"]),DATEPART(MM,["order_purchase_timestamp"])
)temp)temp2;
```

Query Output :

	purchase_year	purchase_month	month_number	order_count	previous_month_order_count	growth_percent
1	2016	September	9	4	0	NULL
2	2016	October	10	324	4	8000.000000000000
3	2016	December	12	1	324	-99.691358024600
4	2017	January	1	800	1	79900.000000000000
5	2017	February	2	1780	800	122.500000000000
6	2017	March	3	2682	1780	50.674157303300
7	2017	April	4	2404	2682	-10.365398956000
8	2017	May	5	3700	2404	53.910149750400
9	2017	June	6	3245	3700	-12.297297297200
10	2017	July	7	4026	3245	24.067796610100
11	2017	August	8	4331	4026	7.575757575700
12	2017	September	9	4285	4331	-1.062110367100

Thus it can be seen that the company had the growth in 2017 mainly. Since then on its keeping shinning.



Now, of course we have to exclude the last data points at both ends , as those are not complete and outliers also. Apart from those the growth is pretty much high.

- **SEASONALITY CHECK**

SQL Server Query :

```
with base as
(select orders.["order_id"],["order_purchase_timestamp"], datename(month, ["order_purchase_timestamp"]) as month
from TargetCustomerAnalysis.dbo.orders)
select month, count(distinct ["order_id"]) as frequency
from base
group by month
order by frequency desc
```

Query Output :

	month	frequency
1	August	10843
2	May	10573
3	July	10318
4	March	9893
5	June	9412
6	April	9343
7	February	8508
8	January	8069
9	November	7544
10	December	5674
11	October	4959
12	September	4305

So, from March to August demand is in peak, then it falls gradually.

- **SHIFT-WISE FREQUENCY**

Now let find out the timings when people usually purchase most

SQL Server Query :

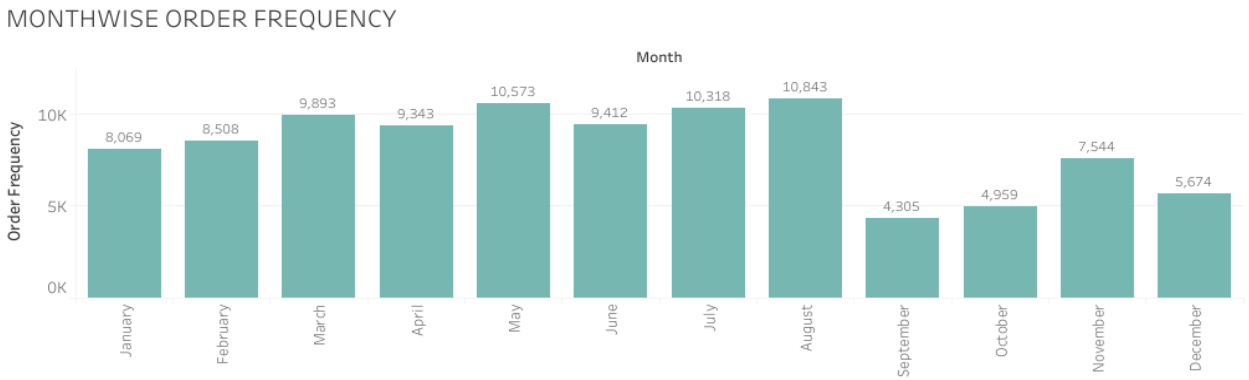
```
with base as
(select temp.["order_purchase_timestamp"],
case when hour <= 3 or hour > 21 then 'night'
when hour > 3 and hour < 7 then 'dawn'
when hour >= 7 and hour < 12 then 'morning'
when hour >= 12 and hour < 17 then 'afternoon'
else 'evening'
end as shifts
from
(select ["order_purchase_timestamp"], datename(HOUR, ["order_purchase_timestamp"]) as hour
from TargetCustomerAnalysis.dbo.orders)temp)

select  base.shifts, count(shifts) as frequency
from base
group by shifts;
```

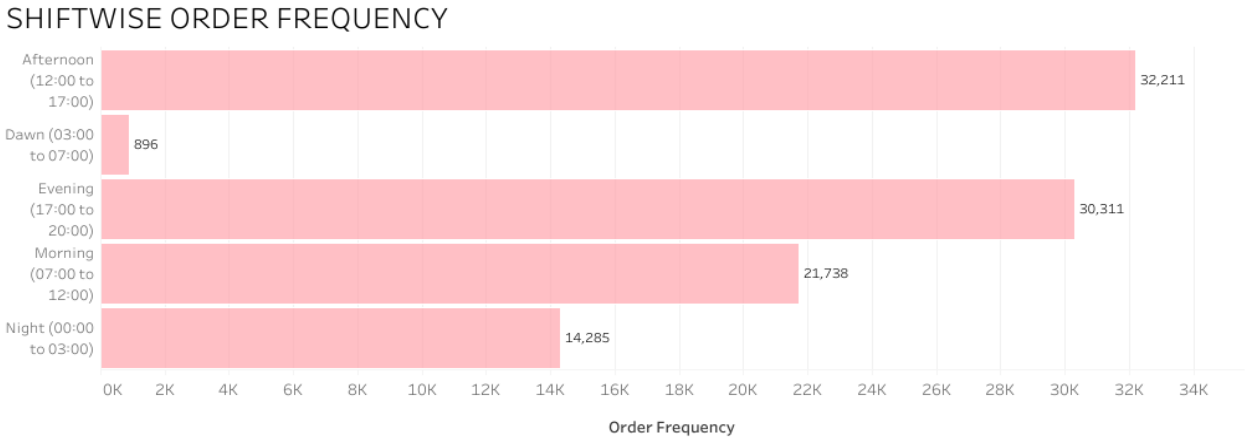
Query Output :

Results Messages

	shifts	frequency
1	dawn	896
2	night	14285
3	morning	21738
4	evening	30311
5	afternoon	32211

So, most of the orders are being placed in the afternoon shifts that is post 12 at noon upto 5PM in the afternoon. And as expected least amount of orders are placed usually at dawn that is from 3AM in the morning to 7AM.





- **PERCENTAGE INCREASE IN ORDER COSTS**

If we want to look at the cost of order increased from 2017 to 2018, that's can be calculated and onlu January to August data is included here.

SQL Server Query :

```
select percent_inc as percent_inc_order_cost_2017_18
from
(select (next_year_order_cost - order_cost_avg) / order_cost_avg * 100 as percent_inc
from
(select base3.*,
LEAD(order_cost_avg,1,0) over(order by year) as next_year_order_cost
from
(select year, AVG(payment_value) as order_cost_avg
from
(select datename(year,base.["order_purchase_timestamp"]) as year, payment_value
from
(select orders.["order_id"], orders.["order_purchase_timestamp"], payments.payment_value
from payments
join orders
on orders.["order_id"] = payments.order_id
where (orders.["order_purchase_timestamp"] between '2017-01-01 00:00:00' and '2017-08-01 00:00:00')
or (orders.["order_purchase_timestamp"] between '2018-01-01 00:00:00' and '2018-08-01 00:00:00')
)base
)base2
group by year
)base3
)base4
```

)base5

where percent_inc > 0

Query Output :

	percent_inc_order_cost_2017_18
1	3.52528313530285

So, the cost growth basically 3.53%.

- **AVERAGE AND TOTAL PRICE VALUE AND FREIGHT VALUE (STATE-WISE)**

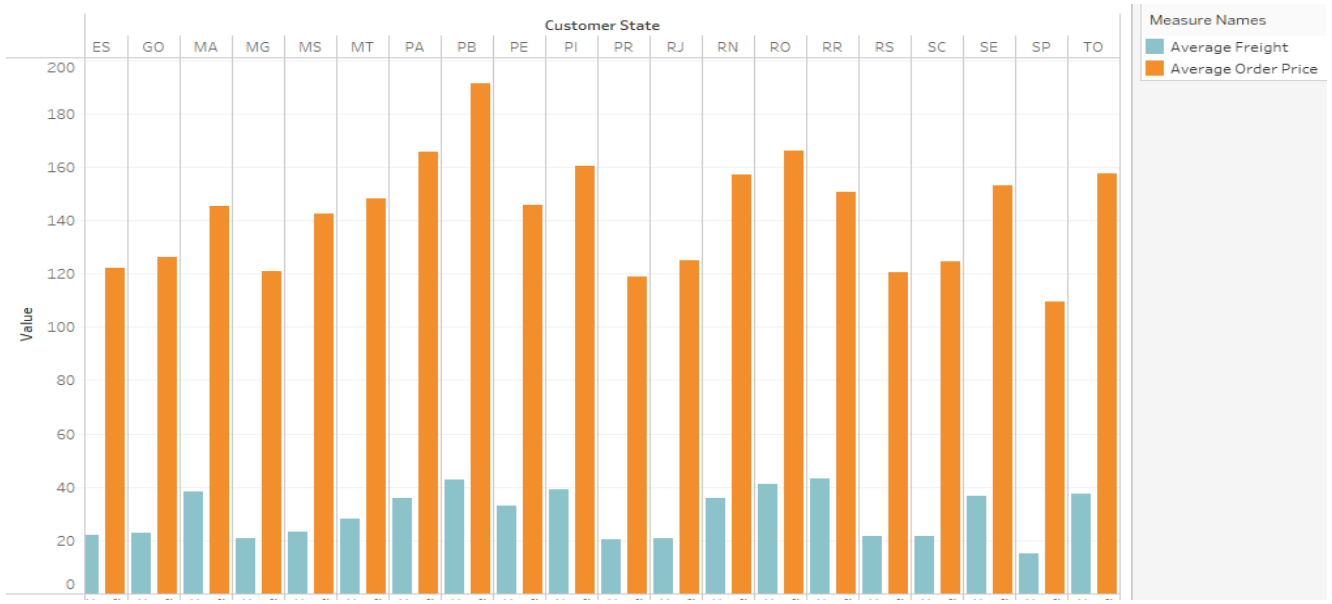
SQL Server Query:

```
select base2.["customer_state"], round(sum(base2.price),2) as total_order_price, round(AVG(price),2) as avg_order_price,
round(SUM(base2.freight_value),2) as total_freight, round(avg(base2.freight_value),2) as avg_freight
from
(select customers.["customer_state"],base.price, base.freight_value
from
(select orders.["customer_id"], cast(["price"] as float) as price, cast(["freight_value"] as float) as freight_value
from TargetCustomerAnalysis.dbo.order_items
join orders
on orders.["order_id"] = order_items.["order_id"])base

join customers
on customers.["customer_id"] = base.["customer_id"])base2
group by ["customer_state"]
```

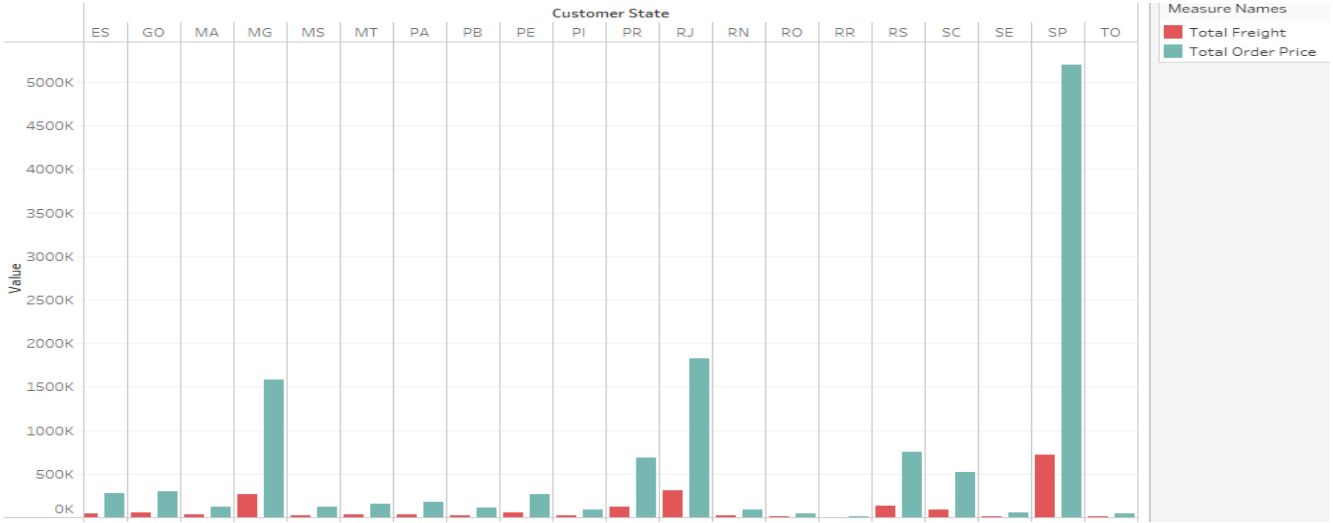
Query Output :

	"customer_state"	total_order_price	avg_order_price	total_freight	avg_freight
1	PE	262788.03	145.51	59449.66	32.92
2	PB	115268.08	191.48	25719.73	42.72
3	PA	178947.81	165.69	38699.3	35.83
4	RS	750304.02	120.34	135522.74	21.74
5	AC	15982.95	173.73	3686.75	40.07
6	BA	511349.99	134.6	100156.68	26.36
7	SP	5202955.05	109.65	718723.07	15.15
8	SC	520553.34	124.65	89660.26	21.47
9	SE	58920.85	153.04	14111.47	36.65
10	MA	119648.22	145.2	31523.77	38.26
11	TO	49621.74	157.53	11732.68	37.25
12	RO	46140.64	165.97	11417.38	41.07
13	DF	302603.94	125.77	50625.5	21.04
14	MT	156453.53	148.3	29715.43	28.17

Average :



Total :



- **PRICE AND FREIGHT VALUE EVALUATION OVER TIME**

Google Big Query :

```
select months, round(avg(freight_value),2) as freight_value_avg, round(avg(total_price),2) as total_price_avg

from

  (select orders.order_id, orders.order_purchase_timestamp,date_trunc(orders.order_purchase_timestamp,month) as months, cast(order_items.price as float64) as price,cast(order_items.freight_value as float64) as freight_value, (cast(order_items.price as float64)+cast(order_items.freight_value as float64)) as total_price

  from `big-query-tutorials-358515.target_customer_analysis.orders` as orders

  join `big-query-tutorials-358515.target_customer_analysis.order_items` as order_items

  on order_items.order_id = orders.order_id

  )

group by months

order by months
```

[There was an issue with SQL Server while trying to truncate the dates, so went for Google Big Query Sandbox.]

Query Output :

Thus total price and freight may vary a lot over the regions and states but over the time the average pretty much remained constant.

- **DELIVERY TIME ANALYSIS**

Lets calculate first the percentage of delivered and unavailable products.

SQL Query :

```
select ["order_status"], no_of_orders,
(no_of_orders*1.0/total_order_count)*100 as order_percent
from
(select base.*,
sum(no_of_orders) over() as total_order_count
from
(select ["order_status"], count(distinct ["order_id"]) as no_of_orders
from TargetCustomerAnalysis.dbo.orders
group by ["order_status"])base)base2;
```

Query Output :

	"order_status"	no_of_orders	order_percent
1	approved	2	0.002011242800
2	canceled	625	0.628513389800
3	created	5	0.005028107100
4	delivered	96478	97.020343721400
5	invoiced	314	0.315765127000
6	processing	301	0.302692048500
7	shipped	1107	1.113222916100
8	unavailable	609	0.612423447000

So pretty much a large amount of products in the dataset are getting delivered.
Now lets focus on the delivery times statewise.

SQL Server Query :

```
select main.["customer_state"], avg(time_to_delivery) as time_to_delivery_avg, avg(diff_estimated_delivery) as
diff_estimated_delivery_avg, avg(estimated_days_for_delivery) as estimated_days_for_delivery_avg,
round(avg(freight_value),2) as freight_value_avg
from
(select base2.*, cast(items.["freight_value"] as float) as freight_value
from
(select
base.*,customers.["customer_city"],customers.["customer_state"]
from
```

```
(select ["customer_id"],["order_id"],
["order_purchase_timestamp"],["order_delivered_customer_date"],["order_estimated_delivery_date"],
DATEDIFF(DD, ["order_purchase_timestamp"],["order_delivered_customer_date"]) as time_to_delivery,


DATEDIFF(DD, ["order_delivered_customer_date"],["order_estimated_delivery_date"]) as diff_estimated_delivery,
DATEDIFF(DD, ["order_purchase_timestamp"],["order_estimated_delivery_date"]) as estimated_days_for_delivery
from TargetCustomerAnalysis.dbo.orders
where (["order_status"] = 'delivered'))
base
join TargetCustomerAnalysis.dbo.customers as customers
on base.["customer_id"] = customers.["customer_id"]
where time_to_delivery >= 0)
base2
join TargetCustomerAnalysis.dbo.order_items as items
on items.["order_id"] = base2.["order_id"])
main
GROUP BY main.["customer_state"];
```

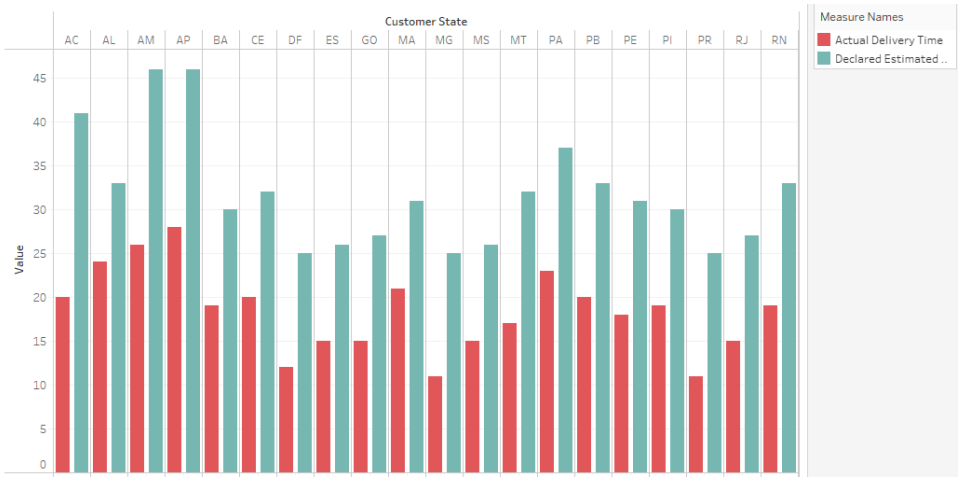
Query Output :

Results Messages

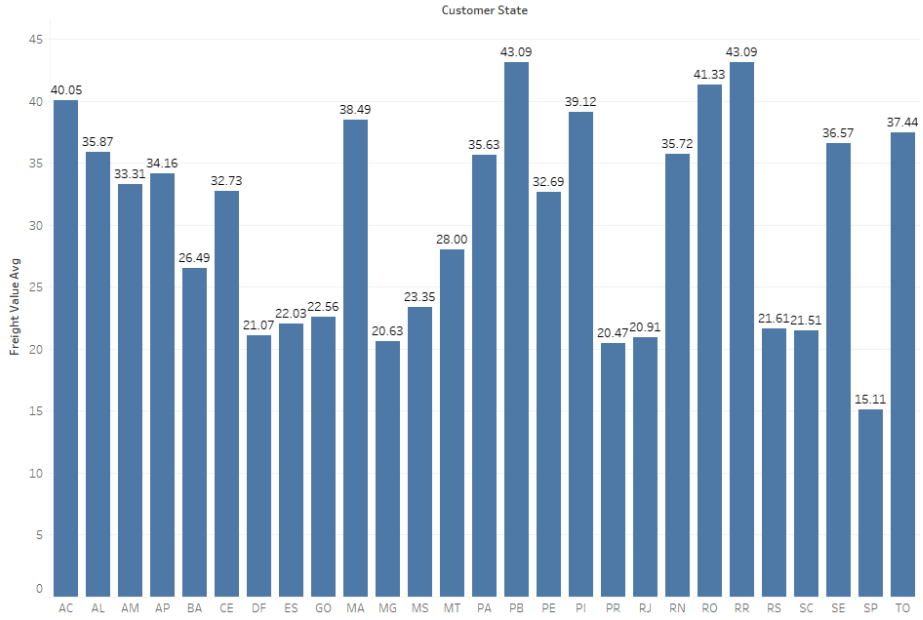
	"customer_state"	time_to_delivery_avg	diff_estimated_delivery_avg	estimated_days_for_delivery_avg	freight_value_avg
1	PE	18	13	31	32.69
2	PB	20	13	33	43.09
3	PA	23	14	37	35.63
4	RS	15	14	29	21.61
5	AC	20	20	41	40.05
6	BA	19	10	30	26.49
7	SP	8	11	19	15.11
8	SC	14	11	26	21.51
9	SE	21	10	31	36.57

Here,
‘time_to_delivery_avg’ : time it takes to get delivered from purchase date to the date customer receives the parcel.
‘diff_estimated_delivery_avg’ : This represents the difference between the estimated delivery date and the date customer recieves the parcel.
‘estimated_days_for_delivery_avg’: declared estimated days it takes to reach a customer.
‘freight_value_avg’ : average freight charge statewise.

Here is the plot containing declared estimated time and actual delivery time. Pretty much every where, it didn’t take more time than estimated time for the product to get delivered .



Here's a visualization representing the freight value average over all the states



- **Looking at Top 5's :**

SQL Server Query :

- TOP 5 states with lowest freight charge as a whole

```
select top(5) x.*
from
(select main.["customer_state"], avg(time_to_delivery) as time_to_delivery_avg, avg(diff_estimated_delivery) as
diff_estimated_delivery_avg, avg(estimated_days_for_delivery) as estimated_days_for_delivery_avg,
round(avg(freight_value),2) as freight_value_avg
from
(select base2.*, cast(items.["freight_value"] as float) as freight_value
from
(select
base.*,customers.["customer_city"],customers.["customer_state"]
from
```

```
(select ["customer_id"],["order_id"],
["order_purchase_timestamp"],["order_delivered_customer_date"],["order_estimated_delivery_date"],
DATEDIFF(DD, ["order_purchase_timestamp"],["order_delivered_customer_date"]) as time_to_delivery,
DATEDIFF(DD, ["order_delivered_customer_date"],["order_estimated_delivery_date"]) as diff_estimated_delivery,
DATEDIFF(DD, ["order_purchase_timestamp"],["order_estimated_delivery_date"]) as estimated_days_for_delivery
from TargetCustomerAnalysis.dbo.orders
where (["order_status"] = 'delivered'))
base
join TargetCustomerAnalysis.dbo.customers as customers
on base.["customer_id"] = customers.["customer_id"]
where time_to_delivery >= 0)
base2
join TargetCustomerAnalysis.dbo.order_items as items
on items.["order_id"] = base2.["order_id"])
main
GROUP BY main.["customer_state"])x
order by freight_value_avg
```

Query Output :

Results | Messages

	"customer_state"	time_to_delivery_avg	diff_estimated_delivery_avg	estimated_days_for_delivery_avg	freight_value_avg
1	SP	8	11	19	15.11
2	PR	11	13	25	20.47
3	MG	11	13	25	20.63
4	RJ	15	12	27	20.91
5	DF	12	12	25	21.07

- **Top 5 states with highest freight charge as a whole**

```
select top(5) x.*
from
(select main.["customer_state"], avg(time_to_delivery) as time_to_delivery_avg, avg(diff_estimated_delivery) as
diff_estimated_delivery_avg, avg(estimated_days_for_delivery) as estimated_days_for_delivery_avg,
round(avg(freight_value),2) as freight_value_avg
from
```

```
(select base2.*, cast(items.["freight_value"] as float) as freight_value
from
(select
base.*,customers.["customer_city"],customers.["customer_state"]
from
(select ["customer_id"],["order_id"],
["order_purchase_timestamp"],["order_delivered_customer_date"],["order_estimated_delivery_date"],
DATEDIFF(DD, ["order_purchase_timestamp"],["order_delivered_customer_date"]) as time_to_delivery,
DATEDIFF(DD, ["order_delivered_customer_date"],["order_estimated_delivery_date"]) as diff_estimated_delivery,
```

```
DATEDIFF(DD, ["order_purchase_timestamp"],["order_estimated_delivery_date"]) as estimated_days_for_delivery
from TargetCustomerAnalysis.dbo.orders
where (["order_status"] = 'delivered'))
base
join TargetCustomerAnalysis.dbo.customers as customers
on base.["customer_id"] = customers.["customer_id"]
where time_to_delivery >= 0)
base2
join TargetCustomerAnalysis.dbo.order_items as items
on items.["order_id"] = base2.["order_id"])
main
GROUP BY main.["customer_state"])x
order by freight_value_avg desc
```

Query Output :

	"customer_state"	time_to_delivery_avg	diff_estimated_delivery_avg	estimated_days_for_delivery_avg	freight_value_avg
1	PB	20	13	33	43.09
2	RR	28	18	46	43.09
3	RO	19	20	39	41.33
4	AC	20	20	41	40.05
5	PI	19	11	30	39.12

- **Top 5 states with highest freight charge Year-wise**

```
select top5.*
from
(select row_num.*,
ROW_NUMBER() over(partition by purchase_year order by freight_value_avg desc) as top5_yearwise
from
(select main.["customer_state"],main.purchase_year, avg(time_to_delivery) as time_to_delivery_avg,
avg(diff_estimated_delivery) as diff_estimated_delivery_avg, avg(estimated_days_for_delivery) as
estimated_days_for_delivery_avg, round(avg(freight_value),2) as freight_value_avg
from
(select base2.*, cast(items.["freight_value"] as float) as freight_value
from
```

```
(select
base.*,customers.["customer_city"],customers.["customer_state"]
from
(select ["customer_id"],["order_id"], ["order_purchase_timestamp"],DATENAME(YEAR,["order_purchase_timestamp"]) as
purchase_year,["order_delivered_customer_date"],["order_estimated_delivery_date"],
DATEDIFF(DD, ["order_purchase_timestamp"],["order_delivered_customer_date"]) as time_to_delivery,
DATEDIFF(DD, ["order_delivered_customer_date"],["order_estimated_delivery_date"]) as diff_estimated_delivery,
DATEDIFF(DD, ["order_purchase_timestamp"],["order_estimated_delivery_date"]) as estimated_days_for_delivery
from TargetCustomerAnalysis.dbo.orders
```

```
where (["order_status"] = 'delivered'))
base
join TargetCustomerAnalysis.dbo.customers as customers
on base.["customer_id"] = customers.["customer_id"]
where time_to_delivery >= 0)
base2
join TargetCustomerAnalysis.dbo.order_items as items
on items.["order_id"] = base2.["order_id"])
main
GROUP BY main.["customer_state"], main.purchase_year)row_num)top5
where top5.top5_yearwise < 6;
```

Query Output :

	"customer_state"	purchase_year	time_to_delivery_avg	diff_estimated_delivery_avg	estimated_days_for_delivery_avg	freight_value_avg	top5_yearwise
1	PE	2016	18	47	65	49.06	1
2	ES	2016	24	39	64	44.01	2
3	CE	2016	27	47	75	39.28	3
4	PI	2016	27	40	67	36.09	4
5	MA	2016	24	41	65	32.73	5
6	RO	2017	19	20	39	39.95	1
7	AC	2017	21	20	41	38.58	2
8	PB	2017	22	13	35	37.99	3
9	MA	2017	21	11	32	35.83	4
10	SE	2017	23	9	33	35.62	5
11	RR	2018	26	22	48	51.96	1
12	PB	2018	19	12	32	47.83	2
13	RO	2018	20	19	39	43.14	3
14	PI	2018	19	10	29	42.92	4
15	AC	2018	19	22	42	42.75	5

This is a whole picture of fright values evolution as it shows the top 5s in each year.

- **Top 5 states with highest time_to_delivery**

SQL Server Query :

```
select top(5) x.*
from
(select main.["customer_state"], avg(time_to_delivery) as time_to_delivery_avg, avg(diff_estimated_delivery) as
diff_estimated_delivery_avg, avg(estimated_days_for_delivery) as estimated_days_for_delivery_avg,
round(avg(freight_value),2) as freight_value_avg
from
(select base2.*, cast(items.["freight_value"] as float) as freight_value
from
(select
base.*,customers.["customer_city"],customers.["customer_state"]
from
```

```
(select ["customer_id"],["order_id"],
["order_purchase_timestamp"],["order_delivered_customer_date"],["order_estimated_delivery_date"],
DATEDIFF(DD, ["order_purchase_timestamp"],["order_delivered_customer_date"]) as time_to_delivery,
DATEDIFF(DD, ["order_delivered_customer_date"],["order_estimated_delivery_date"]) as diff_estimated_delivery,
DATEDIFF(DD, ["order_purchase_timestamp"],["order_estimated_delivery_date"]) as estimated_days_for_delivery
from TargetCustomerAnalysis.dbo.orders
where (["order_status"] = 'delivered'))
base
join TargetCustomerAnalysis.dbo.customers as customers
on base.["customer_id"] = customers.["customer_id"]
where time_to_delivery >= 0)
base2
join TargetCustomerAnalysis.dbo.order_items as items
on items.["order_id"] = base2.["order_id"])
main
GROUP BY main.["customer_state"])x
order by time_to_delivery_avg desc
```

SQL Output :

	"customer_state"	time_to_delivery_avg	diff_estimated_delivery_avg	estimated_days_for_delivery_avg	freight_value_avg
1	RR	28	18	46	43.09
2	AP	28	18	46	34.16
3	AM	26	19	46	33.31
4	AL	24	8	33	35.87
5	PA	23	14	37	35.63

- **Top5 lowest average time_to_delivery**

SQL Server Query :

```
select top(5) x.*
from
(select main.["customer_state"], avg(time_to_delivery) as time_to_delivery_avg, avg(diff_estimated_delivery) as
diff_estimated_delivery_avg, avg(estimated_days_for_delivery) as estimated_days_for_delivery_avg,
round(avg(freight_value),2) as freight_value_avg
from
(select base2.*, cast(items.["freight_value"] as float) as freight_value
from
(select
base.*,customers.["customer_city"],customers.["customer_state"]
from
(select ["customer_id"],["order_id"],
["order_purchase_timestamp"],["order_delivered_customer_date"],["order_estimated_delivery_date"],
DATEDIFF(DD, ["order_purchase_timestamp"],["order_delivered_customer_date"]) as time_to_delivery,
DATEDIFF(DD, ["order_delivered_customer_date"],["order_estimated_delivery_date"]) as diff_estimated_delivery,
DATEDIFF(DD, ["order_purchase_timestamp"],["order_estimated_delivery_date"]) as estimated_days_for_delivery
```

```
from TargetCustomerAnalysis.dbo.orders
where (["order_status"] = 'delivered'))
base
join TargetCustomerAnalysis.dbo.customers as customers
on base.["customer_id"] = customers.["customer_id"]
where time_to_delivery >= 0)
base2
                                        join TargetCustomerAnalysis.dbo.order_items as items
on items.["order_id"] = base2.["order_id"])
main
GROUP BY main.["customer_state"])x
order by time_to_delivery_avg asc
```

Query Output :

	"customer_state"	time_to_delivery_avg	diff_estimated_delivery_avg	estimated_days_for_delivery_avg	freight_value_avg
1	SP	8	11	19	15.11
2	PR	11	13	25	20.47
3	MG	11	13	25	20.63
4	DF	12	12	25	21.07
5	SC	14	11	26	21.51

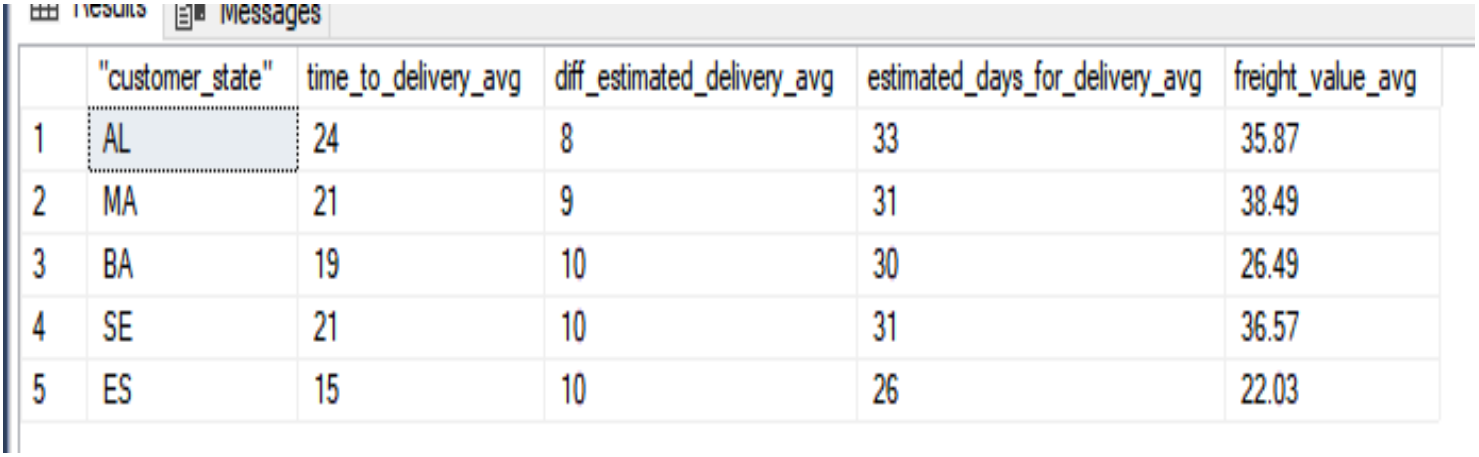
- **Top 5 highest time to delivery with respect to estimated delivery**

SQL Server Query :

```
select top(5) x.*
from
(select main.["customer_state"], avg(time_to_delivery) as time_to_delivery_avg, avg(diff_estimated_delivery) as
diff_estimated_delivery_avg, avg(estimated_days_for_delivery) as estimated_days_for_delivery_avg,
round(avg(freight_value),2) as freight_value_avg
from
(select base2.*, cast(items.["freight_value"] as float) as freight_value
from
(select
base.*,customers.["customer_city"],customers.["customer_state"]
from
(select ["customer_id"],["order_id"],
["order_purchase_timestamp"],["order_delivered_customer_date"],["order_estimated_delivery_date"],
DATEDIFF(DD, ["order_purchase_timestamp"],["order_delivered_customer_date"]) as time_to_delivery,
DATEDIFF(DD, ["order_delivered_customer_date"],["order_estimated_delivery_date"]) as diff_estimated_delivery,
DATEDIFF(DD, ["order_purchase_timestamp"],["order_estimated_delivery_date"]) as estimated_days_for_delivery
```

```
from TargetCustomerAnalysis.dbo.orders
where (["order_status"] = 'delivered'))
base
join TargetCustomerAnalysis.dbo.customers as customers
on base.["customer_id"] = customers.["customer_id"]
where time_to_delivery >= 0)
base2
join TargetCustomerAnalysis.dbo.order_items as items
on items.["order_id"] = base2.["order_id"])
main
GROUP BY main.["customer_state"])x
order by diff_estimated_delivery_avg asc
```

Query Output :



	"customer_state"	time_to_delivery_avg	diff_estimated_delivery_avg	estimated_days_for_delivery_avg	freight_value_avg
1	AL	24	8	33	35.87
2	MA	21	9	31	38.49
3	BA	19	10	30	26.49
4	SE	21	10	31	36.57
5	ES	15	10	26	22.03

'diff_estimated_delivery_avg' is low means it took max amount of time with respect to the estimated delivery date ie. It took almost the days which was basically the days, estimated for delivery.

- **Top 5 states with lowest time_to_delivery with respect to estimated delivery date**

SQL Server Query :

```
select top(5) x.*
from
(select main.["customer_state"], avg(time_to_delivery) as time_to_delivery_avg, avg(diff_estimated_delivery) as
diff_estimated_delivery_avg, avg(estimated_days_for_delivery) as estimated_days_for_delivery_avg,
round(avg(freight_value),2) as freight_value_avg
from
(select base2.*, cast(items.["freight_value"] as float) as freight_value
from
(select
base.*,customers.["customer_city"],customers.["customer_state"]
from
(select ["customer_id"],["order_id"],
["order_purchase_timestamp"],["order_delivered_customer_date"],["order_estimated_delivery_date"],
DATEDIFF(DD, ["order_purchase_timestamp"],["order_delivered_customer_date"]) as time_to_delivery,
```

```
DATEDIFF(DD, ["order_delivered_customer_date"],["order_estimated_delivery_date"]) as diff_estimated_delivery,
DATEDIFF(DD, ["order_purchase_timestamp"],["order_estimated_delivery_date"]) as estimated_days_for_delivery
```

```
from TargetCustomerAnalysis.dbo.orders
where (["order_status"] = 'delivered'))
base
join TargetCustomerAnalysis.dbo.customers as customers
on base.["customer_id"] = customers.["customer_id"]
where time_to_delivery >= 0)
base2
join TargetCustomerAnalysis.dbo.order_items as items
on items.["order_id"] = base2.["order_id"])
main
GROUP BY main.["customer_state"])x
order by diff_estimated_delivery_avg desc
```

Query Output :

	"customer_state"	time_to_delivery_avg	diff_estimated_delivery_avg	estimated_days_for_delivery_avg	freight_value_avg
1	AC	20	20	41	40.05
2	RO	19	20	39	41.33
3	AM	26	19	46	33.31
4	RR	28	18	46	43.09
5	AP	28	18	46	34.16

Max ‘diff_estimated delivery_avg’ means it tas the fastest delivery after the order I splaced with respect to the declared estimated delivery.

- **PAYMENT TYPE EVOLUTION OVER THE YEARS**

SQL Server Query:

```
with base as
(select
payments.order_id,orders.["order_purchase_timestamp"],
DATENAME(year,orders.["order_purchase_timestamp"]) as purchase_year,
DATENAME(MONTH,orders.["order_purchase_timestamp"]) as purchase_month,
payments.payment_type,
payments.payment_installments
from TargetCustomerAnalysis.dbo.payments
join TargetCustomerAnalysis.dbo.orders
on orders.["order_id"] = payments.order_id)
```

select base.purchase_year, base.purchase_month, base.payment_type, count(base.order_id) as order_count
from base
group by base.purchase_year, base.purchase_month, base.payment_type;

Query Output :

	purchase_year	purchase_month	payment_type	order_count
1	2016	December	credit_card	1
2	2016	October	credit_card	88
3	2016	October	debit_card	1
4	2016	October	UPI	28
5	2016	October	voucher	4
6	2016	September	credit_card	1
7	2017	April	credit_card	698
8	2017	April	debit_card	12
9	2017	April	UPI	191
10	2017	April	voucher	94
11	2017	August	credit_card	1174
12	2017	August	debit_card	9
13	2017	August	UPI	322
14	2017	August	voucher	101
15	2017	December	credit_card	1678

From the table this may be not clear but a basic visualization on the growth will clear the insights.

- **PAYMENT INSTALLMENTS DISTRIBUTION OVER THE YEARS**

SQL Query :

select
DATENAME(year,orders.["order_purchase_timestamp"]) as purchase_year,
DATENAME(MONTH,orders.["order_purchase_timestamp"]) as purchase_month,
payment_installments, count(order_id) as order_counts
from TargetCustomerAnalysis.dbo.payments
join TargetCustomerAnalysis.dbo.orders
on orders.["order_id"] = payments.order_id

group by
DATENAME(year,orders.["order_purchase_timestamp"]),DATENAME(MONTH,orders.["order_purchase_timestamp"]),payment_installments

Query Output :

	purchase_year	purchase_month	payment_installments	order_counts
4	2016	October	3	16
5	2016	October	4	5
6	2016	October	5	6
7	2016	October	6	7
8	2016	October	7	6
9	2016	October	8	1
10	2016	October	9	1
11	2016	October	10	18
12	2016	September	3	1
13	2017	April	1	433
14	2017	April	2	96
15	2017	April	3	107
16	2017	April	4	75
17	2017	April	5	51
18	2017	April	6	39
19	2017	April	7	15

Its more of a in detailed evolution of the installments distribution over granularity like months level.
Let's see more broad spectrum like how are the installments are distributed ove rthe years.

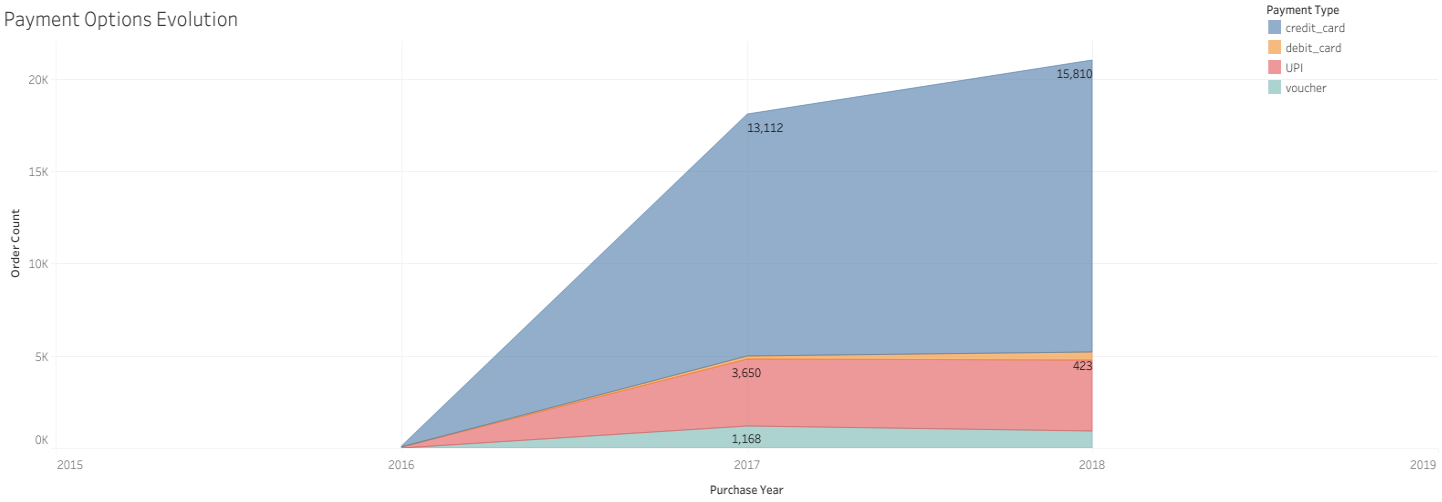
SQL Query :

```
select
DATENAME(year,orders.["order_purchase_timestamp"]) as purchase_year,
payment_installments, count(distinct order_id) as order_counts
from TargetCustomerAnalysis.dbo.payments
join TargetCustomerAnalysis.dbo.orders
on orders.["order_id"] = payments.order_id
group by DATENAME(year,orders.["order_purchase_timestamp"]),payment_installments
order by purchase_year
```

Query Output :

	purchase_year	payment_installments	order_counts
1	2016	1	47
2	2016	2	12
3	2016	3	17
4	2016	4	5
5	2016	5	6
6	2016	6	7
7	2016	7	6
8	2016	8	1
9	2016	9	1
10	2016	10	18
11	2017	1	8155
12	2017	2	2047
13	2017	3	1904
14	2017	4	1224
15	2017	5	996
16	2017	6	728

So we can see one thing from this that, Numbers of available installments have increased over the years. But people prefers more to use less installments which is expected due to increasing interest with more installments.



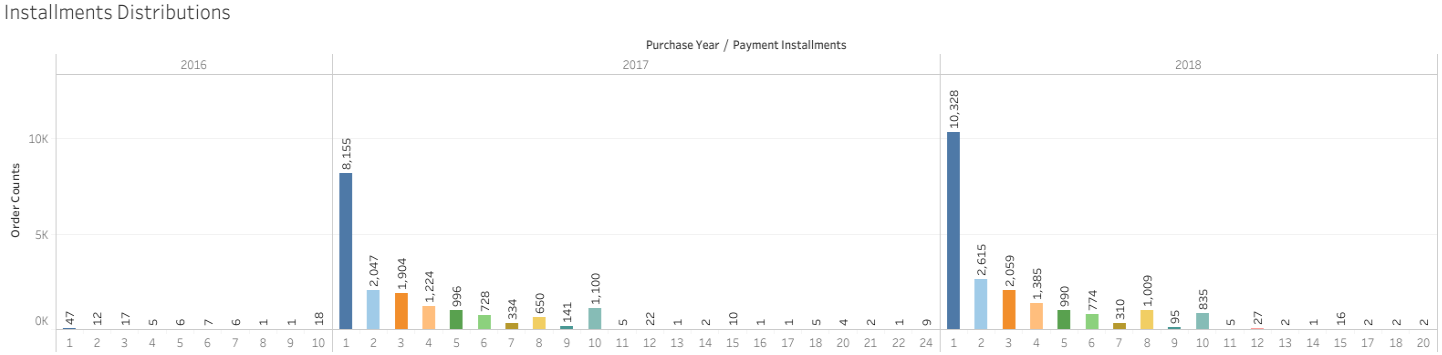


Tableau Dashboard :
https://public.tableau.com/app/profile/writabrata.dey/viz/PaymentAnalysis_16607546366710/Dashboard2

Thus from the visualization its clear that credit card usage has grown a lot. Whether voucher usages has dropped. And also its being shown in the lower plot that a large amount of people are basically preferring to go for one go, not with installments. As a by product also its getting clear again that how amazingly the company has grown since 2017.