SQL Business Case : Target

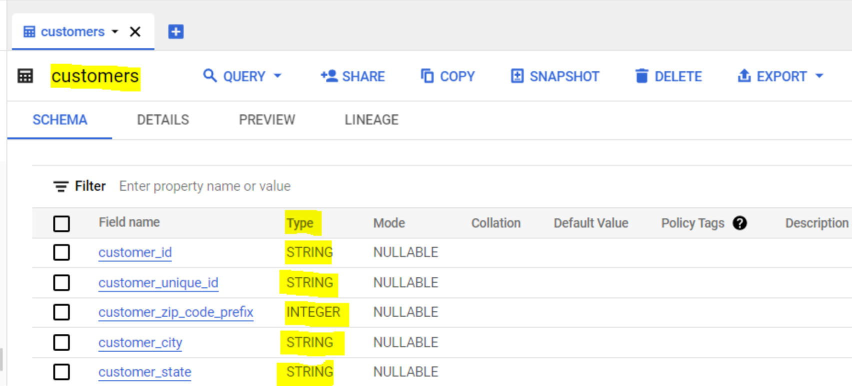
Name : Sayan Pal
Mail ID : psayan1998@gmail.com
Phone No. : 7001792963

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

- Data imported on Big Query

A. Data type of columns in a table

➢ Click on the Added table and user able to see the Data types of the columns of the selected table. (Consist of String, Integer, Char, VARCHAR types)

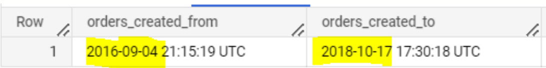


	Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
□	customer_id	STRING	NULLABLE				
□	customer_unique_id	STRING	NULLABLE				
□	customer_zip_code_prefix	INTEGER	NULLABLE				
□	customer_city	STRING	NULLABLE				
□	customer_state	STRING	NULLABLE				

B. Time period for which the data is given

➢ Refer to the below queries for time period for which the data is given (The time period is 2016 to 2018)

```
SELECT *
FROM `scaler-dsml-sql-381008.Target.orders`;

SELECT min(order_purchase_timestamp) as orders_created_from,
max(order_purchase_timestamp) as orders_created_to
FROM `scaler-dsml-sql-381008.Target.orders`;
```



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

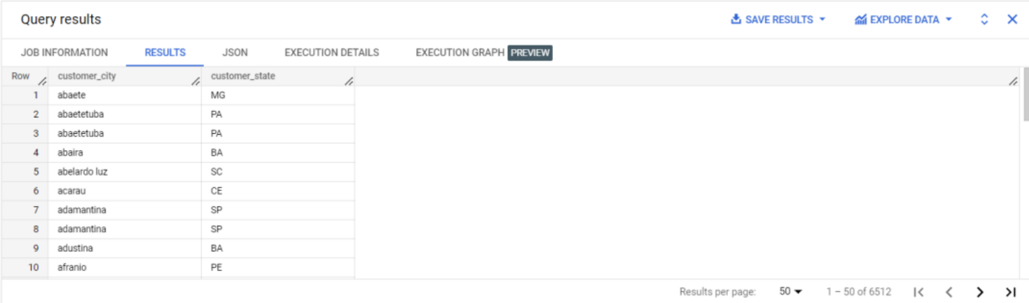
C. Cities and States of customers ordered during the given period

- Orders placed for year 2018 and month August
- During This period people of Brazil tends to order more

```
with ord_month_year as(
SELECT customer_id, order_month
FROM
(SELECT *, EXTRACT(month from order_purchase_timestamp) order_month,
EXTRACT(year from order_purchase_timestamp) as order_year
FROM `scaler-dsml-sql-381008.Target.orders`)
where order_month = 8 and order_year = 2018
)
SELECT
customer_city, customer_state
FROM `scaler-dsml-sql-381008.Target.customers`
where customer_id IN (SELECT customer_id FROM ord_month_year) and customer_id is not NULL
```

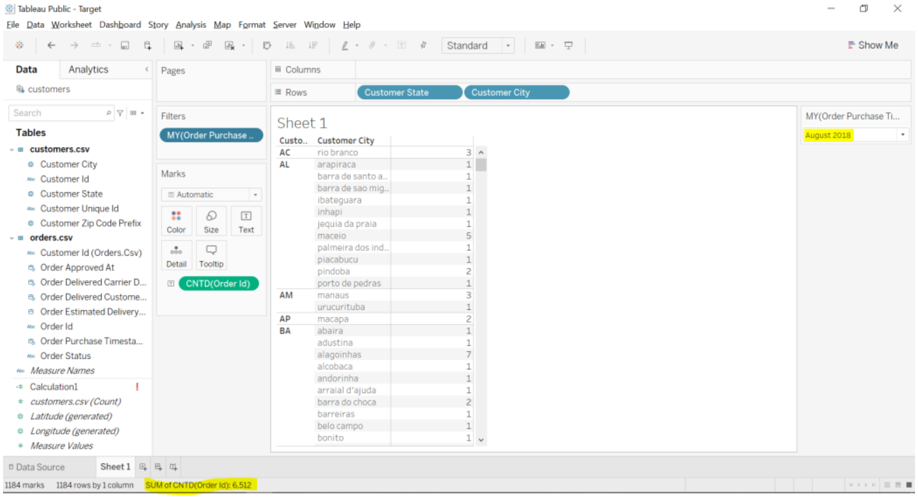
Alternative Code

```
SELECT
c.customer_city, c.customer_state
FROM
`scaler-dsml-sql-381008.Target.orders` o JOIN `scaler-dsml-sql-381008.Target.customers` c
ON o.customer_id = c.customer_id
WHERE EXTRACT(year from order_purchase_timestamp) = 2018 AND EXTRACT(month from order_purchase_timestamp) = 8
order by 1, 2;
```



Row	customer_city	customer_state
1	abaete	MG
2	abaetetuba	PA
3	abaetetuba	PA
4	abaira	BA
5	abelardo luz	SC
6	acarau	CE
7	adamantina	SP
8	adamantina	SP
9	adustina	BA
10	afranio	PE

Results per page: 50 1 – 50 of 6512



MY(Order Purchase Ti...: August 2018

Sheet 1

Custo..	Customer City	
AC	rio branco	3
AL	arapiraca	1
	barra de santo a..	1
	barra de sao mig..	1
	ibateguara	1
	inhapi	1
	jequia da praia	1
	maceio	5
	palmeira dos ind..	1
	piacabucu	1
	pindoba	2
	porto de pedras	1
AM	manaus	3
	urucurituba	1
AP	macapa	2
BA	abaira	1
	adustina	1
	alagoinhas	7
	alcobaca	1
	andorinha	1
	arraial d'ajuda	1
	barra do choca	2
	barreiras	1
	belo campo	1
	bonito	1

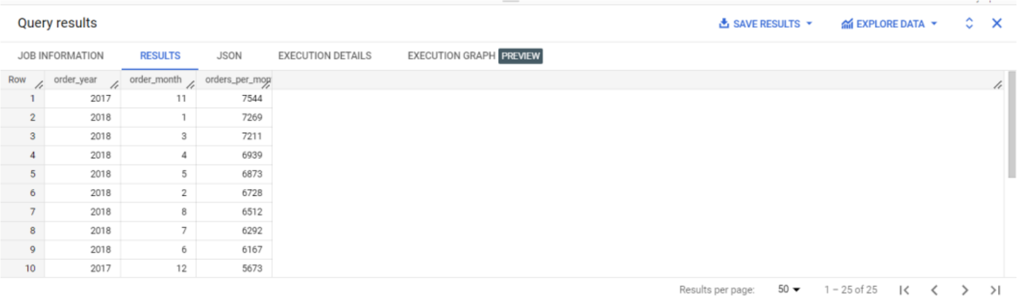
1184 marks 1184 rows by 1 column SUM of CNTD(Order Id): 6,512

2. In-depth Exploration:

A. 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?

- In Brazil, people mostly ordered during November to March (Perhaps there will be some carnival in Brazil during these months)
- Calculated highest orders per month and came to conclusion that during November to March are the highest ordered months.
- **Recommendations** - During this period All warehouse should focus of availability of the product, And marketing should be done as well

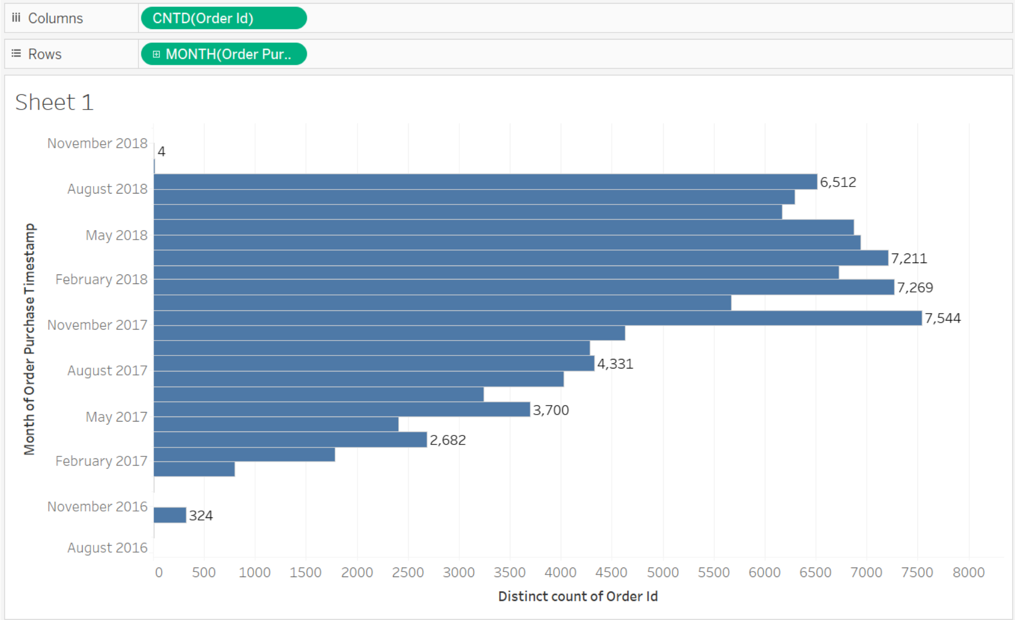
```
-- Top orders per month

SELECT distinct order_year, order_month,
count(distinct order_id) OVER(partition by order_month, order_year) as orders_per_month
FROM
(SELECT *, EXTRACT(month from order_purchase_timestamp) order_month,
EXTRACT(year from order_purchase_timestamp) as order_year
FROM `scaler-dsml-sql-381008.Target.orders`)
order by orders_per_month desc;
```



Query results

Row	order_year	order_month	orders_per_mon
1	2017	11	7544
2	2018	1	7269
3	2018	3	7211
4	2018	4	6939
5	2018	5	6873
6	2018	2	6728
7	2018	8	6512
8	2018	7	6292
9	2018	6	6167
10	2017	12	5673

Results per page: 50 1 – 25 of 25



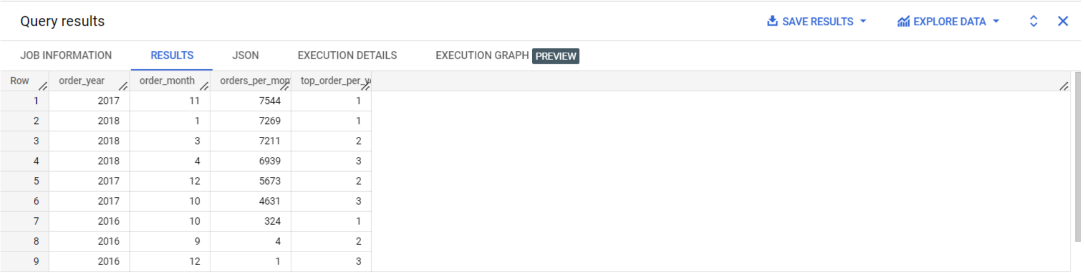
Columns CNTD(Order Id)

Rows MONTH(Order Pur..

Sheet 1

- I have also calculated highest ordered placed in single month for each year

```
-- Highest orders placed per month and for each year
with most_orders as(
SELECT distinct order_year, order_month,
count(order_id) OVER(partition by order_month, order_year) as orders_per_month
FROM
(SELECT *, EXTRACT(month from order_purchase_timestamp) order_month,
EXTRACT(year from order_purchase_timestamp) as order_year
FROM `scaler-dsml-sql-381008.Target.orders`)
order by orders_per_month desc
)
SELECT order_year,order_month, orders_per_month,top_order_per_year
FROM
(SELECT order_year,order_month, orders_per_month,
dense_rank() over(partition by order_year order by orders_per_month desc) as top_order_per_year
FROM most_orders)
where top_order_per_year <=3
order by orders_per_month desc;
```

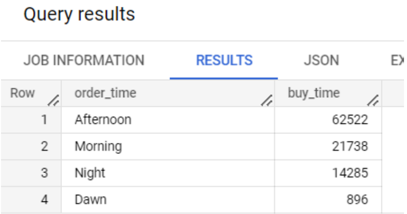


Row	order_year	order_month	orders_per_mon	top_order_per_y
1	2017	11	7544	1
2	2018	1	7269	1
3	2018	3	7211	2
4	2018	4	6939	3
5	2017	12	5673	2
6	2017	10	4631	3
7	2016	10	324	1
8	2016	9	4	2
9	2016	12	1	3

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

- ➢ Customers tend to buy at afternoon mostly.
- ➢ Probably traffic on website will be high during Afternoon time, that need to focused

```
with order_timing as(
SELECT customer_id,time_stamp,
CASE
WHEN time_stamp between '04:00'  and '06:59' Then 'Dawn'
WHEN time_stamp between '07:00'  and '11:59' Then 'Morning'
WHEN time_stamp between '12:00' and '16:59' Then 'Afternoon'
WHEN time_stamp between '17:00' and '21:59' Then 'Afternoon'
ELSE 'Night'
END as order_time
FROM
(SELECT *,
FORMAT_TIMESTAMP('%H:%M', order_purchase_timestamp) as time_stamp
FROM `scaler-dsml-sql-381008.Target.orders`)
)
SELECT distinct order_time,
count(customer_id) over(partition by order_time order by order_time) as buy_time
FROM order_timing
order by buy_time desc;
```



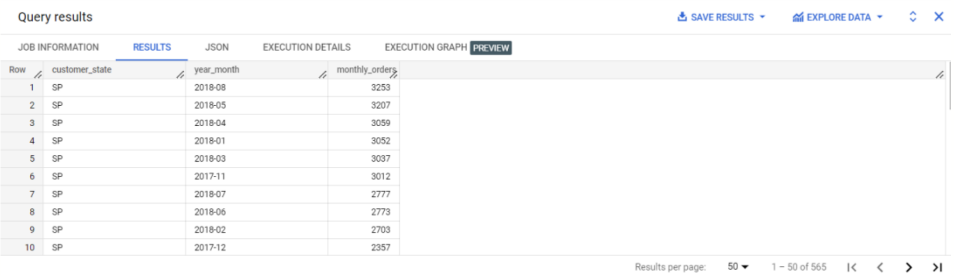
Row	order_time	buy_time
1	Afternoon	62522
2	Morning	21738
3	Night	14285
4	Dawn	896

Note - Instead of customer_id we can use order_id as well.

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

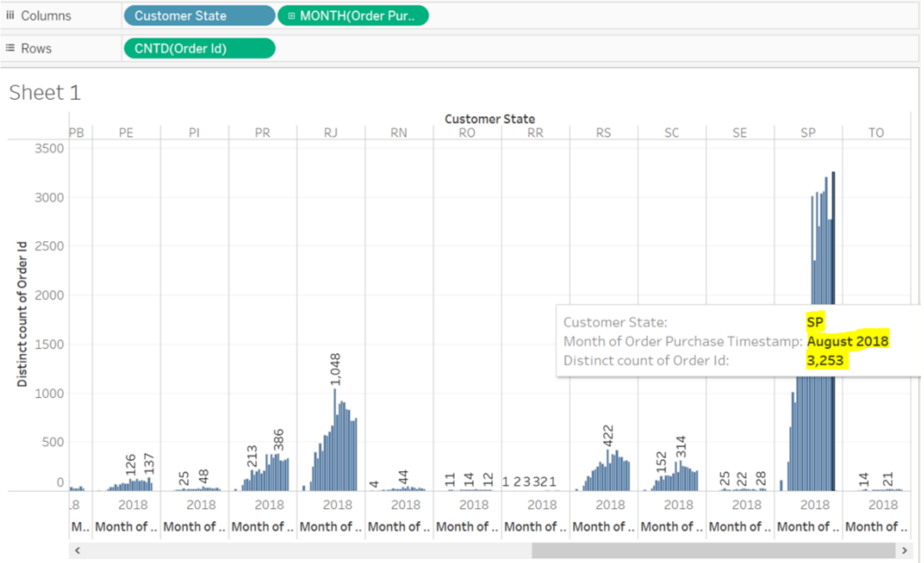
A. Get month on month orders by states

```sql
SELECT distinct customer_state, year_month,
count(order_id) monthly_orders
FROM
(SELECT cust.customer_state, od.order_id, FORMAT_DATE("%Y-%m", od.order_purchase_timestamp) AS year_month
FROM
`scaler-dsml-sql-381008.Target.orders` as od join `scaler-dsml-sql-381008.Target.customers` as cust
on od.customer_id = cust.customer_id)
GROUP BY year_month, customer_state
order by monthly_orders desc;
```



Row	customer_state	year_month	monthly_orders
1	SP	2018-08	3253
2	SP	2018-05	3207
3	SP	2018-04	3059
4	SP	2018-01	3052
5	SP	2018-03	3037
6	SP	2017-11	3012
7	SP	2018-07	2777
8	SP	2018-06	2773
9	SP	2018-02	2703
10	SP	2017-12	2357

Results per page: 50 1 – 50 of 565

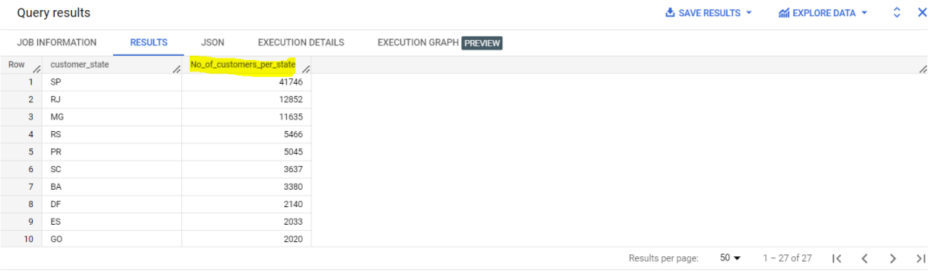


Customer State: SP
Month of Order Purchase Timestamp: August 2018
Distinct count of Order Id: 3,253

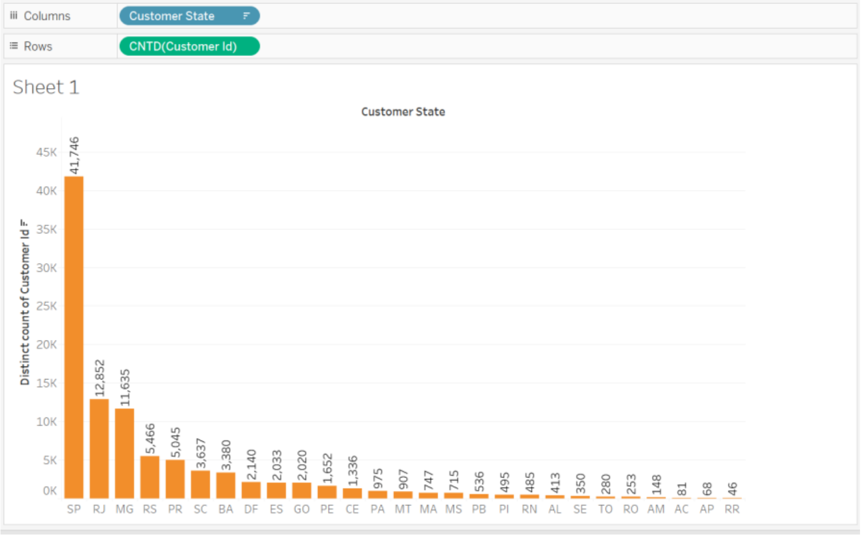
- From above 2 images we can validate that on August 2018, 3253 orders placed in Sao Paulo state.

B. Distribution of customers across the states in Brazil

```
SELECT distinct customer_state, count(customer_id) as No_of_customers_per_state
FROM `scaler-dsml-sql-381008.Target.customers`
GROUP BY customer_state
order by No_of_customers_per_state desc;
```



Row	customer_state	No_of_customers_per_state
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



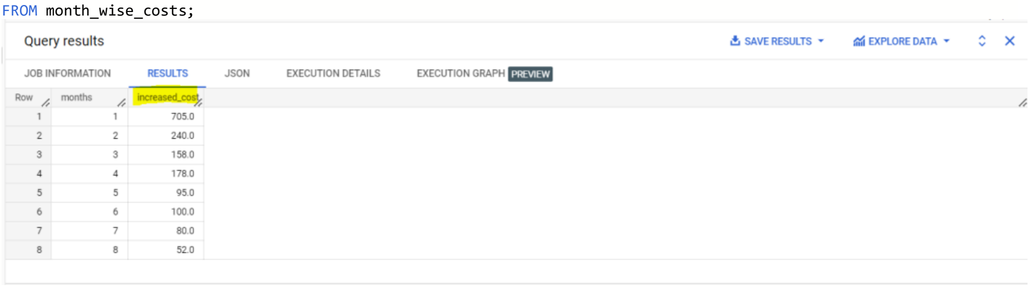
- Most Orders and Most customers in Brazil are from Sao Paulo, So company should focused on better facility and availability in the city as well as should focus on expanding to other cities for better sales growth.

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

A. 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

- Increased % per month(1 to 8) from 2017 to 2028 calculated

```
with month_wise_costs as (
SELECT months, SUM(CASE WHEN years = 2017 and months between 1 and 8 THEN payment_value ELSE 0 END) as cost_2017,
SUM(
  CASE
  WHEN years = 2018 and months between 1 and 8
  THEN payment_value
  ELSE 0
  END) as cost_2018
FROM
(SELECT *,
#SUM(payment_value) OVER(partition by years) as cost_of_orders,
EXTRACT(month FROM order_purchase_timestamp) as months
FROM
(SELECT od.order_id, od.order_purchase_timestamp, py.payment_value,
EXTRACT(year FROM od.order_purchase_timestamp) as years
FROM `scaler-dsml-sql-381008.Target.orders` od
INNER JOIN `scaler-dsml-sql-381008.Target.payments` py
ON od.order_id = py.order_id))
WHERE months between 1 and 8
GROUP BY months
ORDER BY months
)
SELECT months,
round((cost_2018 - cost_2017)/cost_2017*100) as increased_cost
FROM month_wise_costs;
```

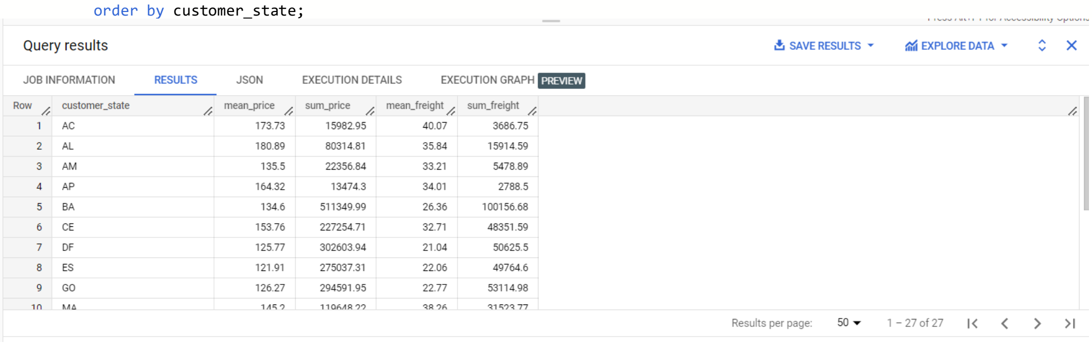


Query results

Row	months	increased_cost
1	1	705.0
2	2	240.0
3	3	158.0
4	4	178.0
5	5	95.0
6	6	100.0
7	7	80.0
8	8	52.0

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

```
SELECT customer_state, round(avg(price),2) as mean_price, round(sum(price),2) as sum_price,
round(avg(freight_value),2) as mean_freight,round(sum(freight_value),2) as sum_freight
FROM
(SELECT c.customer_id, c.customer_state, oi.order_id, oi.price, oi.freight_value
FROM
`scaler-dsml-sql-381008.Target.customers` c join `scaler-dsml-sql-381008.Target.orders` o
on o.customer_id = c.customer_id
join `scaler-dsml-sql-381008.Target.order_items` oi
on o.order_id = oi.order_id)
group by customer_state
order by customer_state;
```



Query results

Row	customer_state	mean_price	sum_price	mean_freight	sum_freight
1	AC	173.73	15982.95	40.07	3686.75
2	AL	180.89	80314.81	35.84	15914.59
3	AM	135.5	22356.84	33.21	5478.89
4	AP	164.32	13474.3	34.01	2788.5
5	BA	134.6	511349.99	26.36	100156.68
6	CE	153.76	227254.71	32.71	48351.59
7	DF	125.77	302603.94	21.04	50625.5
8	ES	121.91	275037.31	22.06	49764.6
9	GO	126.27	294591.95	22.77	53114.98
10	MA	145.2	119648.22	38.26	31523.77

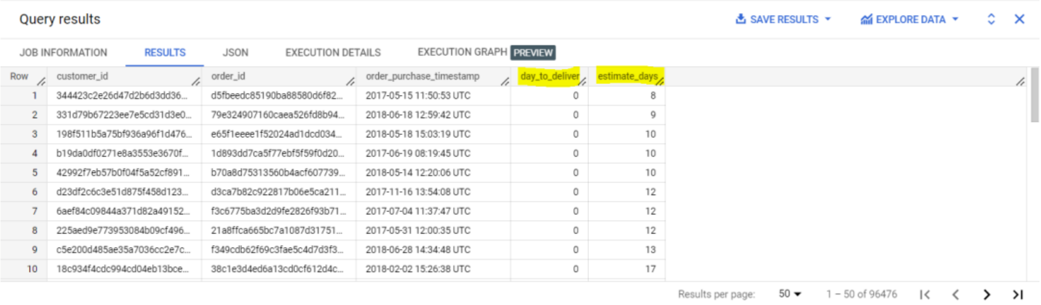
Results per page: 50 1 – 27 of 27

- ➢ As we can observe that, In some of the states freight value is high. This issue need to be considered as well.

5. Analysis on sales, freight and delivery time

A. Calculate days between purchasing, delivering and estimated delivery

```
SELECT customer_id, order_id, order_purchase_timestamp,day_to_deliver, estimate_days
FROM
(SELECT *,
DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, day) as day_to_deliver,
DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, day) as estimate_days
FROM `scaler-dsml-sql-381008.Target.orders`
order by day_to_deliver) as tbl1
Where day_to_deliver is not null and estimate_days is not null
order by day_to_deliver, estimate_days;
```



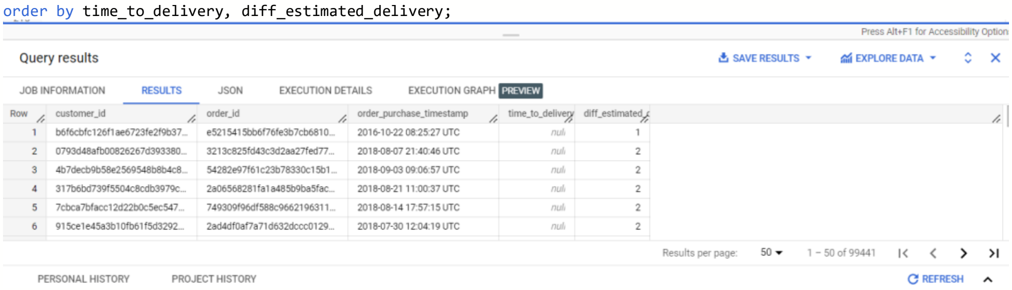
Row	customer_id	order_id	order_purchase_timestamp	day_to_deliver	estimate_days
1	344423c2e26d47d2b6d3dd36...	d5fbeedc85190ba88580d6f82...	2017-05-15 11:50:53 UTC	0	8
2	331d79b67223ee7e5cd31d3e0...	79e324907160caea526fd8b94...	2018-06-18 12:59:42 UTC	0	9
3	198f511b5a75bf936a96f1d476...	e65f1eeee1f52024ad1dcd034...	2018-05-18 15:03:19 UTC	0	10
4	b19da0df0271e8a3553e3670f...	1d893dd7ca5f77ebf5f59f0d20...	2017-06-19 08:19:45 UTC	0	10
5	42992f7eb57b0f04f5a52cf891...	b70a8d75313560b4acf607739...	2018-05-14 12:20:06 UTC	0	10
6	d23df2c6c3e51d875f458d123...	d3ca7b82c922817b06e5ca211...	2017-11-16 13:54:08 UTC	0	12
7	6aef84c09844a371d82a49152...	f3c6775ba3d2d9fe2826f93b71...	2017-07-04 11:37:47 UTC	0	12
8	225aed9e773953084b09cf496...	21a8ffca665bc7a1087d31751...	2017-05-31 12:00:35 UTC	0	12
9	c5e200d485ae35a7036cc2e7c...	f349cdb62f69c3fae5c4d7d3f3...	2018-06-28 14:34:48 UTC	0	13
10	18c934f4cdc994cd04eb13bce...	38c1e3d4ed6a13cd0cf612d4c...	2018-02-02 15:26:38 UTC	0	17

- ➢ We can observe that for some of the orders same day delivery also completed

B. Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:

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

```
SELECT customer_id, order_id, order_purchase_timestamp,time_to_delivery, diff_estimated_delivery
FROM
(SELECT *,
DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, day) as time_to_delivery,
DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, day) as diff_estimated_delivery
FROM `scaler-dsml-sql-381008.Target.orders`
order by time_to_delivery) as tbl1
order by time_to_delivery, diff_estimated_delivery;
```

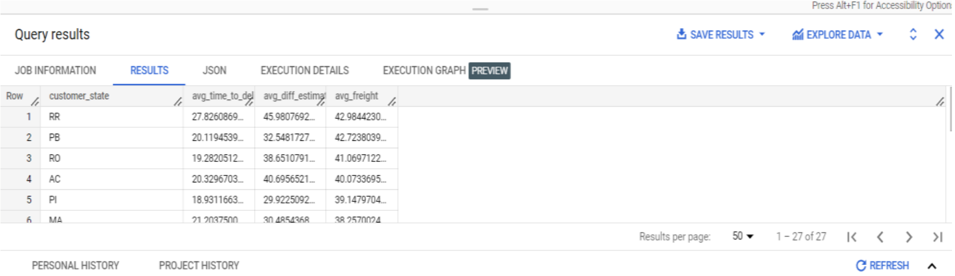


Row	customer_id	order_id	order_purchase_timestamp	time_to_delivery	diff_estimated_c
1	b6f6cbfc126f1ae6723fe2f9b37...	e5215415bb6f76fe3b7cb6810...	2016-10-22 08:25:27 UTC	null	1
2	0793d48afb00826267d393380...	3213c825fd43c3d2aa27fed77...	2018-08-07 21:40:46 UTC	null	2
3	4b7decb9b58e2569548b8b4c8...	54282e97f61c23b78330c15b1...	2018-09-03 09:06:57 UTC	null	2
4	317b6bd739f5504c8cdb3979c...	2a06568281fa1a485b9ba5fac...	2018-08-21 11:00:37 UTC	null	2
5	7cbca7bfacc12d22b0c5ec547...	749309f96df588c9662196311...	2018-08-14 17:57:15 UTC	null	2
6	915ce1e45a3b10fb61f5d3292...	2ad4df0af7a71d632dccc0129...	2018-07-30 12:04:19 UTC	null	2

- ➢ For some of the orders we do not have delivery date.

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

```
SELECT distinct customer_state, avg(time_to_delivery) as avg_time_to_delivery,
avg(diff_estimated_delivery) as avg_diff_estimated_delivery, avg(freight_value) as avg_freight
FROM
(SELECT *,
DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, day) as time_to_delivery,
DATE_DIFF(o.order_estimated_delivery_date, o.order_purchase_timestamp, day) as diff_estimated_delivery
FROM
`scaler-dsml-sql-381008.Target.customers` c join `scaler-dsml-sql-381008.Target.orders` o
on o.customer_id = c.customer_id
join `scaler-dsml-sql-381008.Target.order_items` oi
on o.order_id = oi.order_id)
GROUP BY 1
order by avg_freight desc,  avg_time_to_delivery desc, avg_diff_estimated_delivery desc;
```

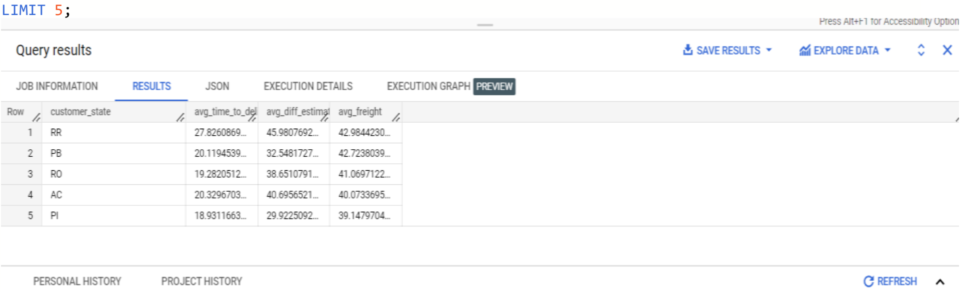


Row	customer_state	avg_time_to_del	avg_diff_estimat	avg_freight
1	RR	27.8260869...	45.9807692...	42.9844230...
2	PB	20.1194539...	32.5481727...	42.7238039...
3	RO	19.2820512...	38.6510791...	41.0697122...
4	AC	20.3296703...	40.6956521...	40.0733695...
5	PI	18.9311663...	29.9225092...	39.1479704...
6	MA	21.2037500	30.4854368	38.2570024

D. Sort the data to get the following:

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

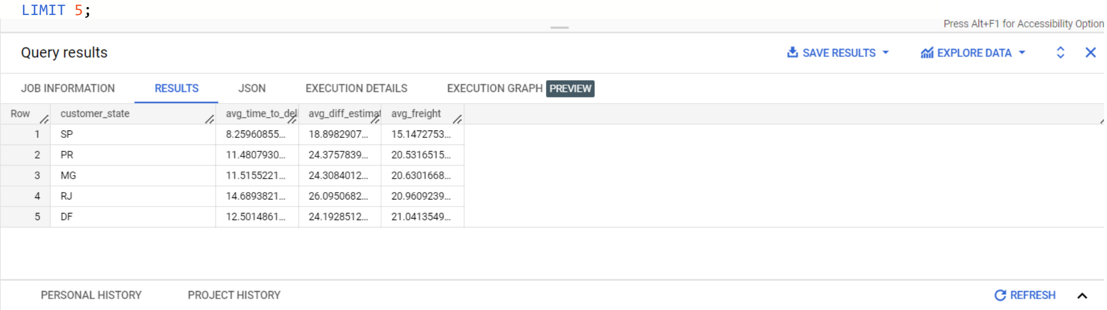
```
SELECT distinct customer_state, avg(time_to_delivery) as avg_time_to_delivery,
avg(diff_estimated_delivery) as avg_diff_estimated_delivery, avg(freight_value) as avg_freight
FROM
(SELECT *,
DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, day) as time_to_delivery,
DATE_DIFF(o.order_estimated_delivery_date, o.order_purchase_timestamp, day) as diff_estimated_delivery
FROM
`scaler-dsml-sql-381008.Target.customers` c join `scaler-dsml-sql-381008.Target.orders` o
on o.customer_id = c.customer_id
join `scaler-dsml-sql-381008.Target.order_items` oi
on o.order_id = oi.order_id)
GROUP BY 1
order by avg_freight desc
LIMIT 5;
```



Row	customer_state	avg_time_to_del	avg_diff_estimat	avg_freight
1	RR	27.8260869...	45.9807692...	42.9844230...
2	PB	20.1194539...	32.5481727...	42.7238039...
3	RO	19.2820512...	38.6510791...	41.0697122...
4	AC	20.3296703...	40.6956521...	40.0733695...
5	PI	18.9311663...	29.9225092...	39.1479704...

- Top 5 state with lowest average freight value

```
SELECT distinct customer_state, avg(time_to_delivery) as avg_time_to_delivery,
avg(diff_estimated_delivery) as avg_diff_estimated_delivery, avg(freight_value) as avg_freight
FROM
(SELECT *,
DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, day) as time_to_delivery,
DATE_DIFF(o.order_estimated_delivery_date, o.order_purchase_timestamp, day) as diff_estimated_delivery
FROM
`scaler-dsml-sql-381008.Target.customers` c join `scaler-dsml-sql-381008.Target.orders` o
on o.customer_id = c.customer_id
join `scaler-dsml-sql-381008.Target.order_items` oi
on o.order_id = oi.order_id)
GROUP BY 1
order by avg_freight asc
LIMIT 5;
```

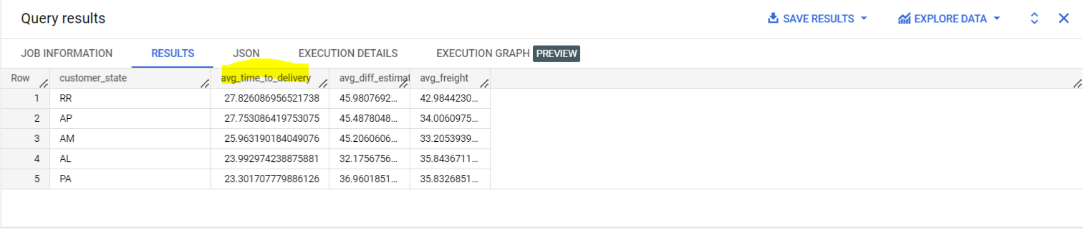


Row	customer_state	avg_time_to_deli	avg_diff_estimat	avg_freight
1	SP	8.25960855...	18.8982907...	15.1472753...
2	PR	11.4807930...	24.3757839...	20.5316515...
3	MG	11.5155221...	24.3084012...	20.6301668...
4	RJ	14.6893821...	26.0950682...	20.9609239...
5	DF	12.5014861...	24.1928512...	21.0413549...

- Top 5 states with highest/lowest average time to delivery

- Top 5 states with highest average time to delivery

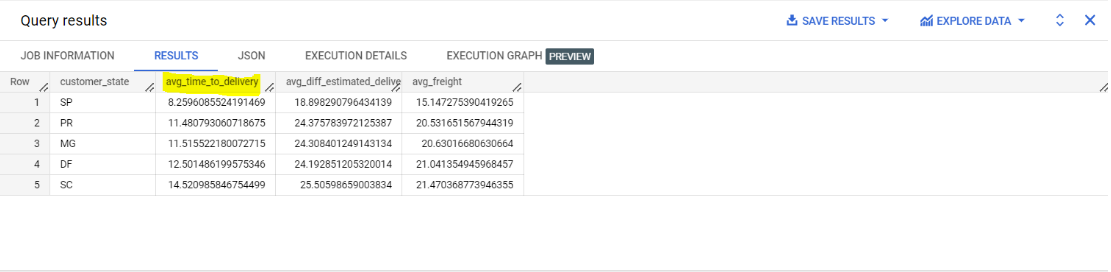
```
SELECT distinct customer_state, avg(time_to_delivery) as avg_time_to_delivery,
avg(diff_estimated_delivery) as avg_diff_estimated_delivery, avg(freight_value) as avg_freight
FROM
(SELECT *,
DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, day) as time_to_delivery,
DATE_DIFF(o.order_estimated_delivery_date, o.order_purchase_timestamp, day) as diff_estimated_delivery
FROM
`scaler-dsml-sql-381008.Target.customers` c join `scaler-dsml-sql-381008.Target.orders` o
on o.customer_id = c.customer_id
join `scaler-dsml-sql-381008.Target.order_items` oi
on o.order_id = oi.order_id)
GROUP BY 1
order by avg_time_to_delivery desc
LIMIT 5;
```



Row	customer_state	avg_time_to_delivery	avg_diff_estimat	avg_freight
1	RR	27.826086956521738	45.9807692...	42.9844230...
2	AP	27.753086419753075	45.4878048...	34.0060975...
3	AM	25.963190184049076	45.2060606...	33.2053939...
4	AL	23.992974238875881	32.1756756...	35.8436711...
5	PA	23.301707779886126	36.9601851...	35.8326851...

- Top 5 states with lowest average time to delivery

```
SELECT distinct customer_state, avg(time_to_delivery) as avg_time_to_delivery,
avg(diff_estimated_delivery) as avg_diff_estimated_delivery, avg(freight_value) as avg_freight
FROM
(SELECT *,
DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, day) as time_to_delivery,
DATE_DIFF(o.order_estimated_delivery_date, o.order_purchase_timestamp, day) as diff_estimated_delivery
FROM
`scaler-dsml-sql-381008.Target.customers` c join `scaler-dsml-sql-381008.Target.orders` o
on o.customer_id = c.customer_id
join `scaler-dsml-sql-381008.Target.order_items` oi
on o.order_id = oi.order_id)
GROUP BY 1
order by avg_time_to_delivery asc
LIMIT 5;
```



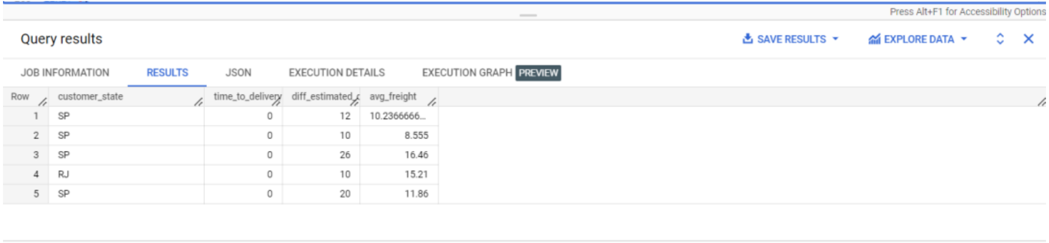
Query results

Row	customer_state	avg_time_to_delivery	avg_diff_estimated_delive	avg_freight
1	SP	8.2596085524191469	18.898290796434139	15.147275390419265
2	PR	11.480793060718675	24.375783972125387	20.531651567944319
3	MG	11.515522180072715	24.308401249143134	20.63016680630664
4	DF	12.501486199575346	24.192851205320014	21.041354945968457
5	SC	14.520985846754499	25.50598659003834	21.470368773946355

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

  - Top 5 state where same day delivery done

```
SELECT distinct customer_state, time_to_delivery,
diff_estimated_delivery, avg(freight_value) as avg_freight
FROM
(SELECT *,
DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, day) as time_to_delivery,
DATE_DIFF(o.order_estimated_delivery_date, o.order_purchase_timestamp, day) as diff_estimated_delivery
FROM
`scaler-dsml-sql-381008.Target.customers` c join `scaler-dsml-sql-381008.Target.orders` o
on o.customer_id = c.customer_id
join `scaler-dsml-sql-381008.Target.order_items` oi
on o.order_id = oi.order_id)
where time_to_delivery is not null
GROUP BY 1, 2, 3
order by time_to_delivery asc
LIMIT 5;
```

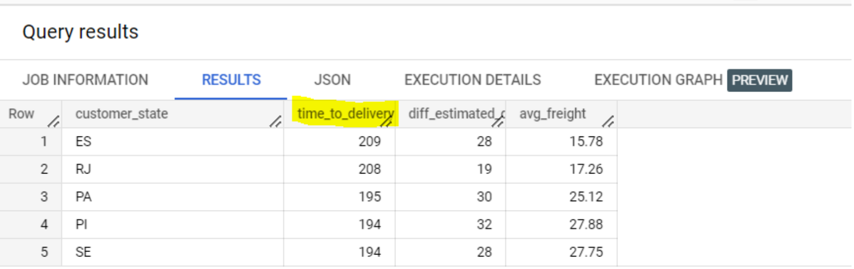


Query results

Row	customer_state	time_to_delivery	diff_estimated_c	avg_freight
1	SP	0	12	10.2366666...
2	SP	0	10	8.555
3	SP	0	26	16.46
4	RJ	0	10	15.21
5	SP	0	20	11.86

- Top 5 State where delivery is not fast compare to estimate date

```
SELECT distinct customer_state, time_to_delivery,
diff_estimated_delivery, avg(freight_value) as avg_freight
FROM
(SELECT *,
DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, day) as time_to_delivery,
DATE_DIFF(o.order_estimated_delivery_date, o.order_purchase_timestamp, day) as diff_estimated_delivery
FROM
`scaler-dsml-sql-381008.Target.customers` c join `scaler-dsml-sql-381008.Target.orders` o
on o.customer_id = c.customer_id
join `scaler-dsml-sql-381008.Target.order_items` oi
on o.order_id = oi.order_id)
where time_to_delivery is not null
GROUP BY 1, 2, 3
order by time_to_delivery desc
LIMIT 5;
```

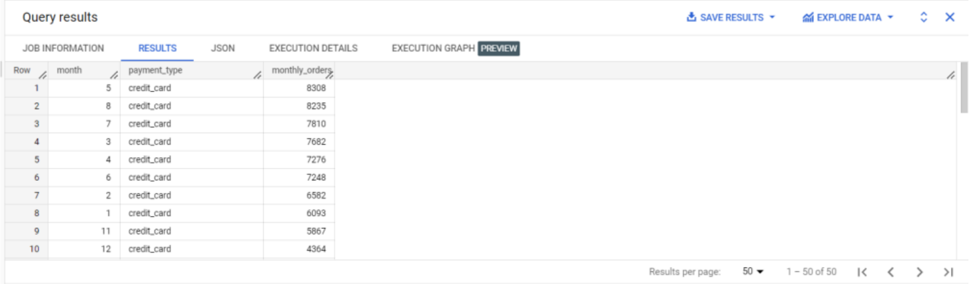


Row	customer_state	time_to_delivery	diff_estimated_c	avg_freight
1	ES	209	28	15.78
2	RJ	208	19	17.26
3	PA	195	30	25.12
4	PI	194	32	27.88
5	SE	194	28	27.75

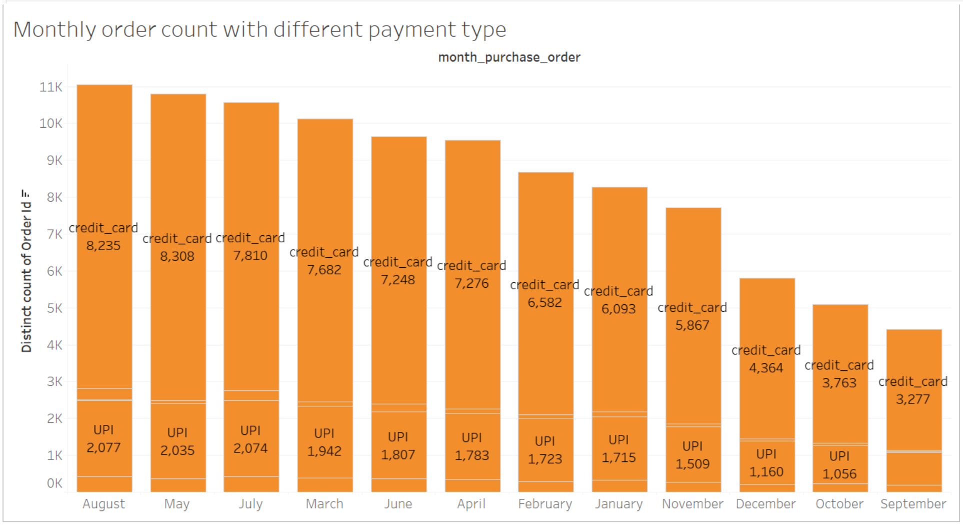
6. Payment type analysis:

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

```
SELECT distinct month, payment_type,
count(distinct order_id) as monthly_orders
FROM
(SELECT od.order_id, pay.payment_type, EXTRACT(month from od.order_purchase_timestamp) as month,
FORMAT_DATE("%Y-%m", od.order_purchase_timestamp) AS year_month
FROM
`scaler-dsml-sql-381008.Target.orders` as od join `scaler-dsml-sql-381008.Target.payments` as pay
on od.order_id = pay.order_id)
GROUP BY month, payment_type
order by monthly_orders desc;
```

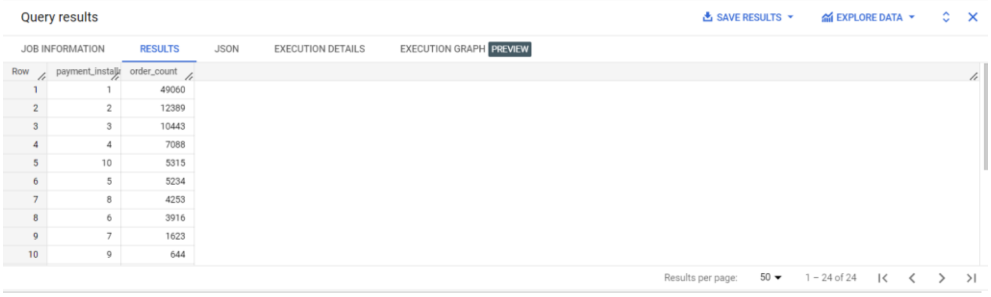


Row	month	payment_type	monthly_orders
1	5	credit_card	8308
2	8	credit_card	8235
3	7	credit_card	7810
4	3	credit_card	7682
5	4	credit_card	7276
6	6	credit_card	7248
7	2	credit_card	6582
8	1	credit_card	6093
9	11	credit_card	5867
10	12	credit_card	4364



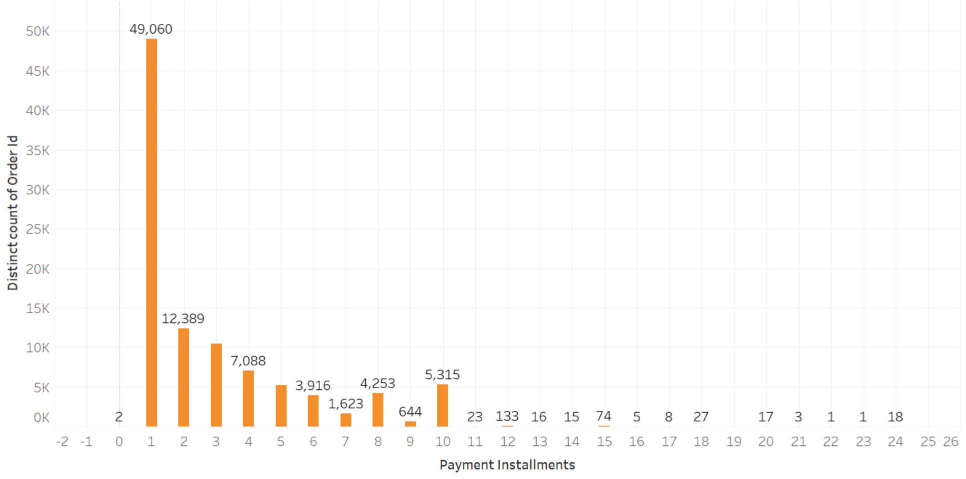
B. Count of orders based on the no. of payment installments

```
SELECT distinct payment_installments,
count(distinct order_id) as order_count
FROM
(SELECT od.order_id, pay.payment_installments
FROM
`scaler-dsml-sql-381008.Target.orders` as od join `scaler-dsml-sql-381008.Target.payments` as pay
on od.order_id = pay.order_id)

GROUP BY payment_installments
order by order_count desc;
```



Row	payment_installi	order_count
1	1	49060
2	2	12389
3	3	10443
4	4	7088
5	10	5315
6	5	5234
7	8	4253
8	6	3916
9	7	1623
10	9	644

Orders Counts based on the no. of payment installments



- **Actionable Insights:**
  - ➢ People mostly ordered from November to March every year (Perhaps there will be some carnival in Brazil during these months)
  - ➢ Calculated the highest orders per month and came to the conclusion that November, January, and March are the highest ordered months.
  - ➢ Growth of sales during peak months is increased by a factor of two in 2017 and 2018 when compared to 2016.
  - ➢ Customers tend to buy in the afternoon mostly.
  - ➢ Probably traffic on website will be high during Afternoon time, that need to focused Most Orders and Most customers in Brazil are from Sao Paulo, So company should focused on better facility and availability in the city as well as should focus on expanding to other cities for better sales growth.
  - ➢ As we can observe, In some of the states freight value is high. This issue needs to be considered as well.
  - ➢ For some states and some orders Delivery time period is too high, That needs to be taken care of.
  - ➢ Most of the orders placed using Credit Cards as a payment option
  - ➢ People tend to pay at one go mostly (1 installment)

- **Recommendations:**
  - ➢ During peak selling period, I.e. November to March every year, All warehouse should focus of availability of the product, And marketing should be done as well.
  - ➢ For some of the cities Freight value is too high this need to focused and need to be reduced as well
  - ➢ Some some cities delivery orders are too high, This must be addressed as well.
  - ➢ Payments mostly done by Credit Cards. So, Company should focused on Offers related Credit Cards. It will assist in attracting more customers and increasing sales.