Target SQL Project - Data Analysis

August 8

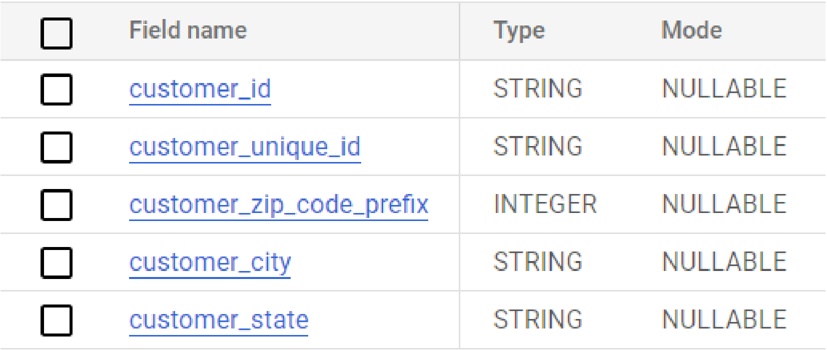
2023

Analyzing the given data set to extract valuable insights and provide actionable recommendations.

Mohammad Ahmed Khan

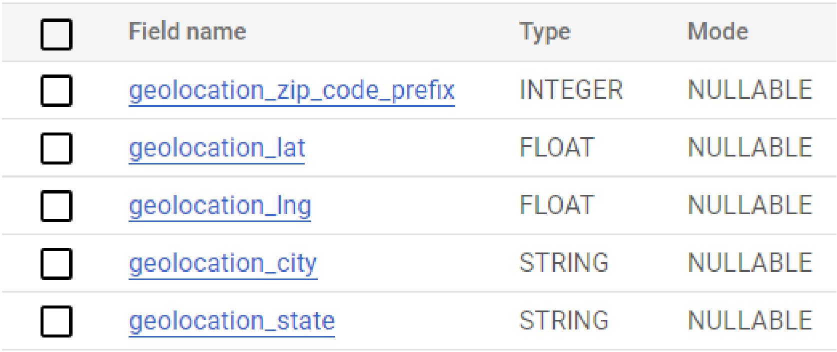
Given Data

I. Customer.csv



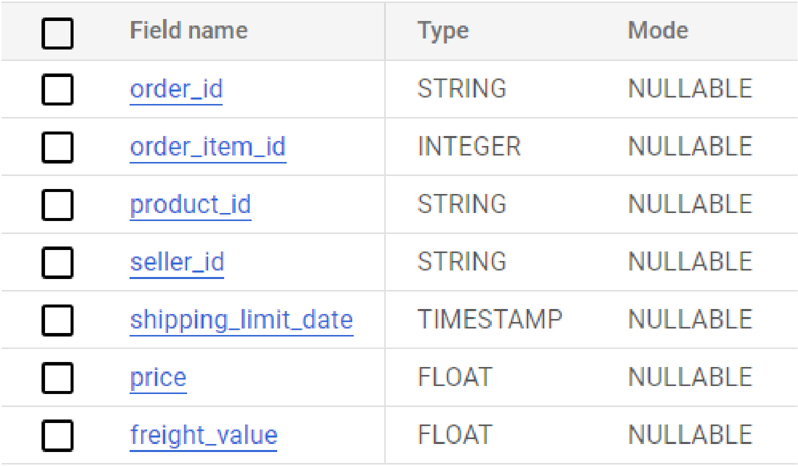
□	Field name	Type	Mode
□	customer_id	STRING	NULLABLE
□	customer_unique_id	STRING	NULLABLE
□	customer_zip_code_prefix	INTEGER	NULLABLE
□	customer_city	STRING	NULLABLE
□	customer_state	STRING	NULLABLE

II. Geolocation.csv



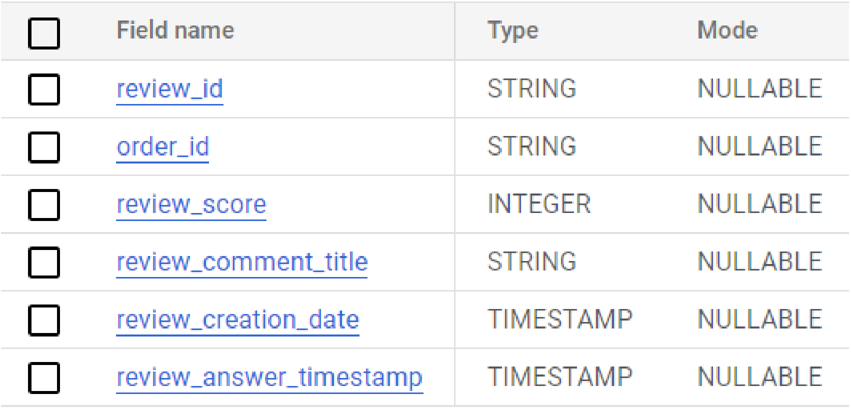
□	Field name	Type	Mode
□	geolocation_zip_code_prefix	INTEGER	NULLABLE
□	geolocation_lat	FLOAT	NULLABLE
□	geolocation_lng	FLOAT	NULLABLE
□	geolocation_city	STRING	NULLABLE
□	geolocation_state	STRING	NULLABLE

III. Order_items.csv



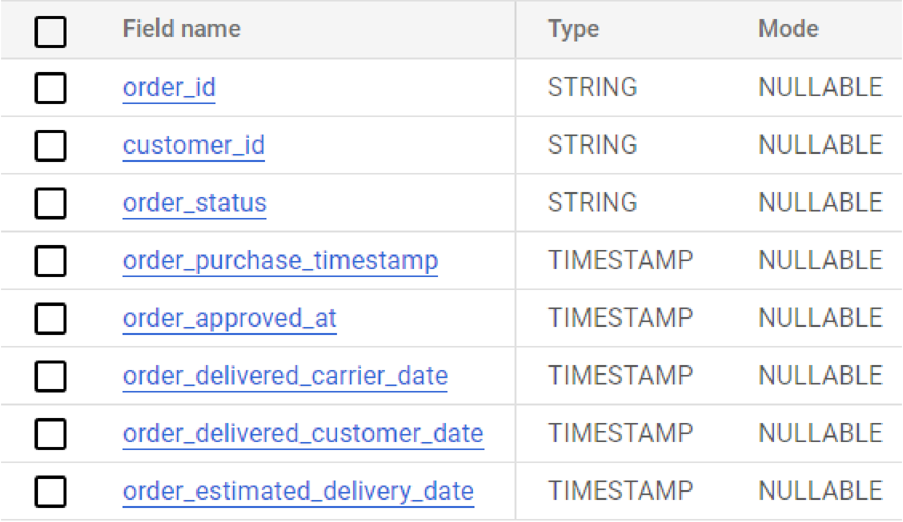
□	Field name	Type	Mode
□	order_id	STRING	NULLABLE
□	order_item_id	INTEGER	NULLABLE
□	product_id	STRING	NULLABLE
□	seller_id	STRING	NULLABLE
□	shipping_limit_date	TIMESTAMP	NULLABLE
□	price	FLOAT	NULLABLE
□	freight_value	FLOAT	NULLABLE

IV. Order_reviews.csv



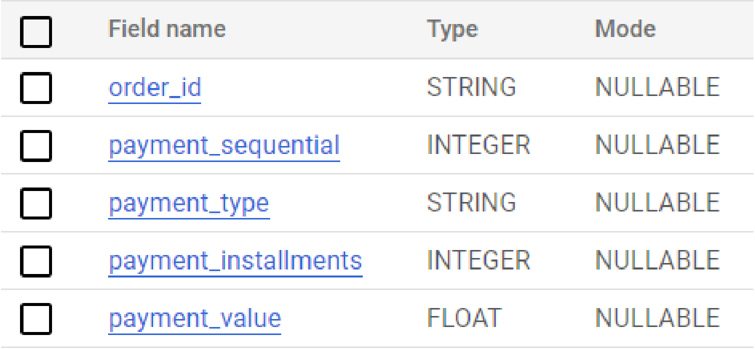
□	Field name	Type	Mode
□	review_id	STRING	NULLABLE
□	order_id	STRING	NULLABLE
□	review_score	INTEGER	NULLABLE
□	review_comment_title	STRING	NULLABLE
□	review_creation_date	TIMESTAMP	NULLABLE
□	review_answer_timestamp	TIMESTAMP	NULLABLE

V. Orders.csv



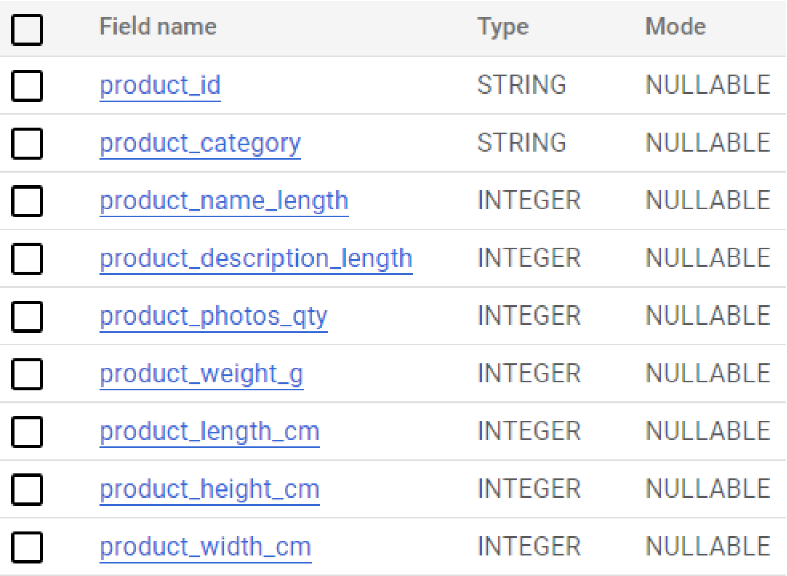
□	Field name	Type	Mode
□	order_id	STRING	NULLABLE
□	customer_id	STRING	NULLABLE
□	order_status	STRING	NULLABLE
□	order_purchase_timestamp	TIMESTAMP	NULLABLE
□	order_approved_at	TIMESTAMP	NULLABLE
□	order_delivered_carrier_date	TIMESTAMP	NULLABLE
□	order_delivered_customer_date	TIMESTAMP	NULLABLE
□	order_estimated_delivery_date	TIMESTAMP	NULLABLE

VI. Payments.csv



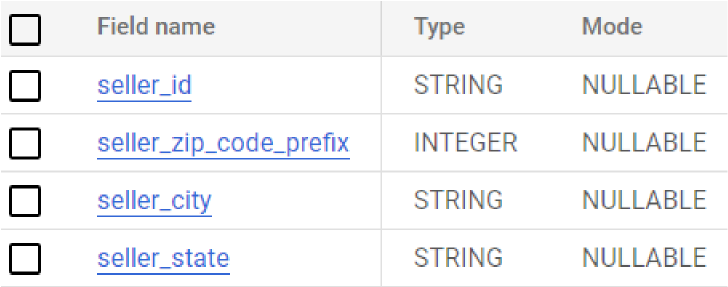
□	Field name	Type	Mode
□	order_id	STRING	NULLABLE
□	payment_sequential	INTEGER	NULLABLE
□	payment_type	STRING	NULLABLE
□	payment_installments	INTEGER	NULLABLE
□	payment_value	FLOAT	NULLABLE

VII. Products.csv



□	Field name	Type	Mode
□	product_id	STRING	NULLABLE
□	product_category	STRING	NULLABLE
□	product_name_length	INTEGER	NULLABLE
□	product_description_length	INTEGER	NULLABLE
□	product_photos_qty	INTEGER	NULLABLE
□	product_weight_g	INTEGER	NULLABLE
□	product_length_cm	INTEGER	NULLABLE
□	product_height_cm	INTEGER	NULLABLE
□	product_width_cm	INTEGER	NULLABLE

VIII. Sellers.csv

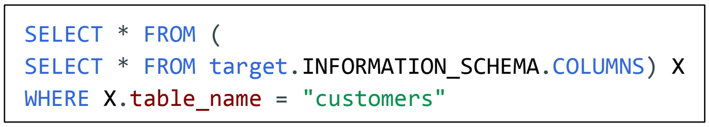


□	Field name	Type	Mode
□	seller_id	STRING	NULLABLE
□	seller_zip_code_prefix	INTEGER	NULLABLE
□	seller_city	STRING	NULLABLE
□	seller_state	STRING	NULLABLE

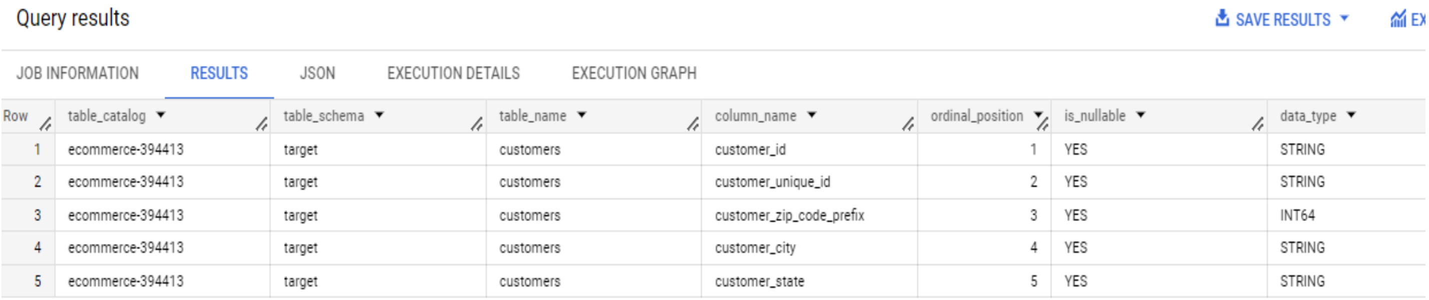
1-Usual exploratory analysis steps like checking the structure & characteristics of the data set.

1A:-Data type of all columns in the “customers” table

SQL QUERY-



```
SELECT * FROM (
SELECT * FROM target.INFORMATION_SCHEMA.COLUMNS) X
WHERE X.table_name = "customers"
```



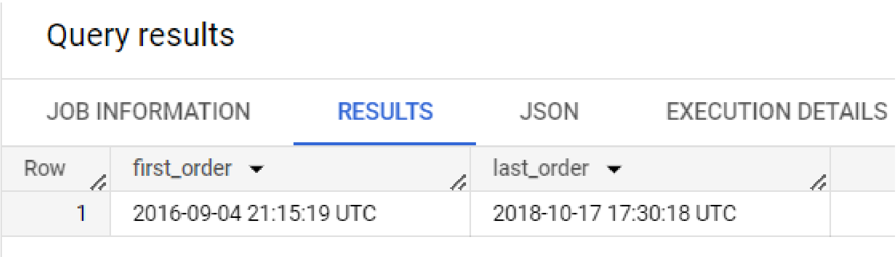
Row	table_catalog	table_schema	table_name	column_name	ordinal_position	is_nullable	data_type
1	ecommerce-394413	target	customers	customer_id	1	YES	STRING
2	ecommerce-394413	target	customers	customer_unique_id	2	YES	STRING
3	ecommerce-394413	target	customers	customer_zip_code_prefix	3	YES	INT64
4	ecommerce-394413	target	customers	customer_city	4	YES	STRING
5	ecommerce-394413	target	customers	customer_state	5	YES	STRING

We can observe that customer table has 5 columns and in which columns customer_id, customer_unique_id, customer_city and customer_state are “STRING” Type and customer_zip_code is “INT” type

1B: Get the time range between which the orders were placed

SQL QUERY-

```
select min(order_purchase_timestamp) as first_order,
max(order_purchase_timestamp) as last_order
from `target.orders`
```



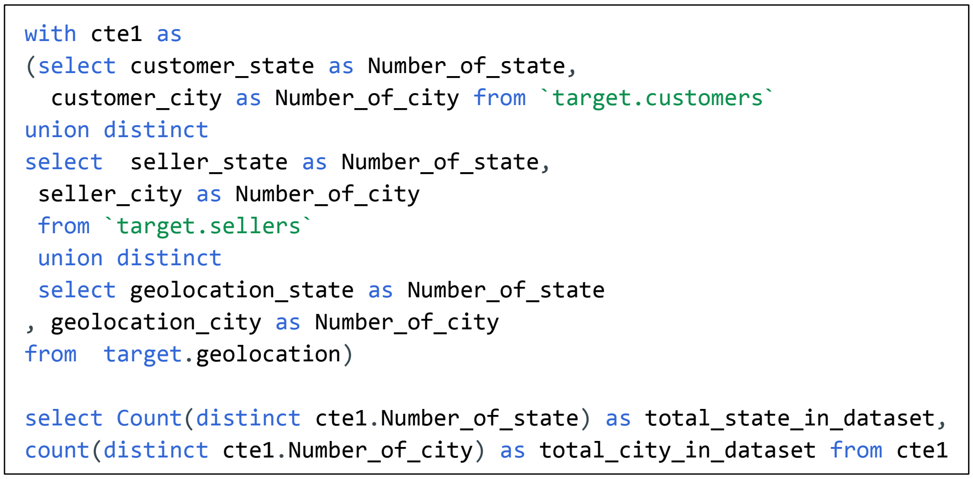
Row	first_order	last_order
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

And the range between very first order and last order “2016-09-04” & “2018-10-17”

1C: Count the number of Cities and States in our dataset.

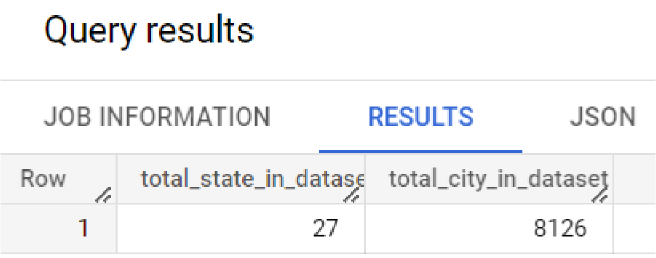
As we have three table customers, sellers and geolocation which contains information about city and state, merged three tables by union distinct, then counted , resulted total distinct state 27 and total distinct city 8126

SQL QUERY-



```
with cte1 as
(select customer_state as Number_of_state,
  customer_city as Number_of_city from `target.customers`
union distinct
select  seller_state as Number_of_state,
 seller_city as Number_of_city
 from `target.sellers`
 union distinct
 select geolocation_state as Number_of_state
, geolocation_city as Number_of_city
from  target.geolocation)

select Count(distinct cte1.Number_of_state) as total_state_in_dataset,
count(distinct cte1.Number_of_city) as total_city_in_dataset from cte1
```



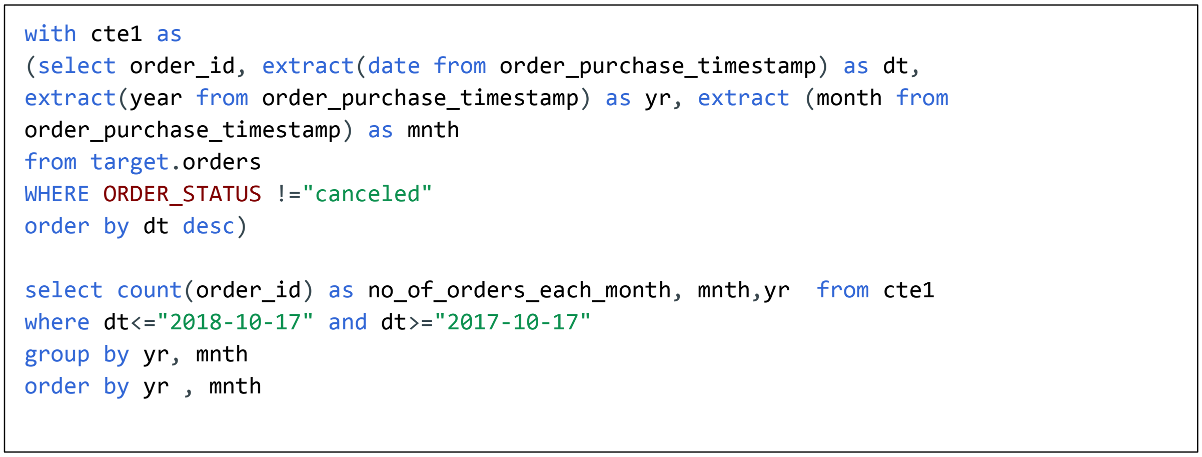
Row	total_state_in_datase	total_city_in_dataset
1	27	8126

2-In-depth Exploration:

2A: Is there a growing trend in the no.of orders placed over the past years?

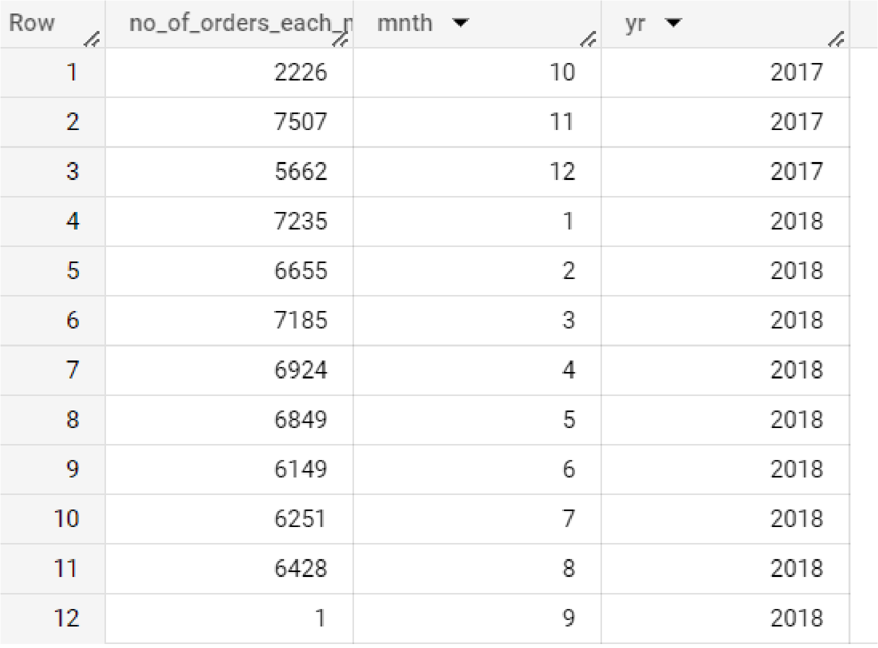
SQL QUERY- (12 months Data)

Data from over the past year “2018-10-17” to “2017-10-17”



```
with cte1 as
(select order_id, extract(date from order_purchase_timestamp) as dt,
extract(year from order_purchase_timestamp) as yr, extract (month from
order_purchase_timestamp) as mnth
from target.orders
WHERE ORDER_STATUS !="canceled"
order by dt desc)

select count(order_id) as no_of_orders_each_month, mnth,yr  from cte1
where dt<="2018-10-17" and dt>="2017-10-17"
group by yr, mnth
order by yr , mnth
```



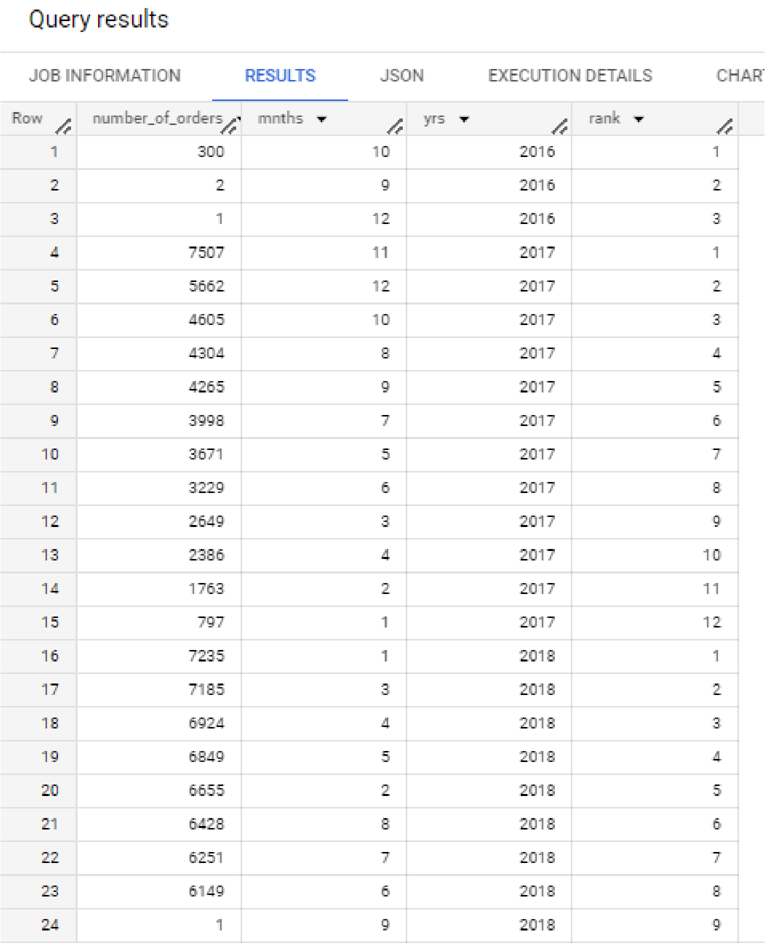
Row	no_of_orders_each_n	mnth	yr
1	2226	10	2017
2	7507	11	2017
3	5662	12	2017
4	7235	1	2018
5	6655	2	2018
6	7185	3	2018
7	6924	4	2018
8	6849	5	2018
9	6149	6	2018
10	6251	7	2018
11	6428	8	2018
12	1	9	2018

As per above results we can observe that orders in year 2018 except month September and October the number of receiving order is consistent throughout the year, in Nov 2017 order is all time high,

2B: Can we see some kind of monthly seasonality in terms of the no.of orders being placed?

SQL QUERY-

```
select count(order_id) as number_of_orders, mnths, yrs,dense_rank() over(partition by yrs
order by count(order_id) desc) as rank from (
select order_id,order_status, extract(year from order_purchase_timestamp ) as yrs,
extract(month from order_purchase_timestamp) as mnths
from target.orders) x
where x.order_status !="canceled"
group by yrs, mnths
order by yrs, rank, mnths
```



Row	number_of_orders	mnths	yrs	rank
1	300	10	2016	1
2	2	9	2016	2
3	1	12	2016	3
4	7507	11	2017	1
5	5662	12	2017	2
6	4605	10	2017	3
7	4304	8	2017	4
8	4265	9	2017	5
9	3998	7	2017	6
10	3671	5	2017	7
11	3229	6	2017	8
12	2649	3	2017	9
13	2386	4	2017	10
14	1763	2	2017	11
15	797	1	2017	12
16	7235	1	2018	1
17	7185	3	2018	2
18	6924	4	2018	3
19	6849	5	2018	4
20	6655	2	2018	5
21	6428	8	2018	6
22	6251	7	2018	7
23	6149	6	2018	8
24	1	9	2018	9

Actionable insights:

Company has highest orders in November in 2017 and in 2018 a consistent order till 8 months, in 2018 month September and October and 2016 when the company started has least orders compare to other months of years

2C: During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night) ● 0-6hrs: Dawn ● 7-12hrs: Mornings ● 13-18hrs: Afternoon ● 19-23hrs: Night

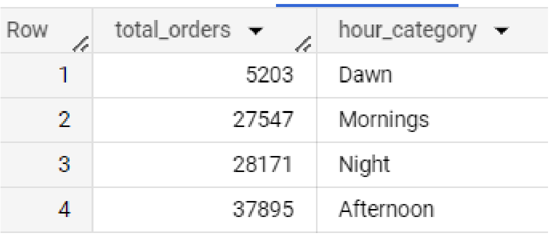
SQL QUERY-

From below results We can conclude that maximum order received in afternoon and minimum order received during dawn, morning and night have around similar order count

```
with cte1 as
(select count(order_id) as orders_count, extract(hour from order_purchase_timestamp) as
order_time from target.orders where order_status != 'canceled' group by order_time order by
order_time),

cte2 as
(select orders_count, order_time, case when order_time between 0 and 6 then "Dawn" when
order_time between 7 and 12 then "Mornings" when order_time between 13 and 18 then
"Afternoon" when order_time between 19 and 23 then "Night" end as hour_category
from cte1)


select sum(orders_count) as total_orders, hour_category
from cte2
group by hour_category
order by total_orders
```



Row	total_orders	hour_category
1	5203	Dawn
2	27547	Mornings
3	28171	Night
4	37895	Afternoon

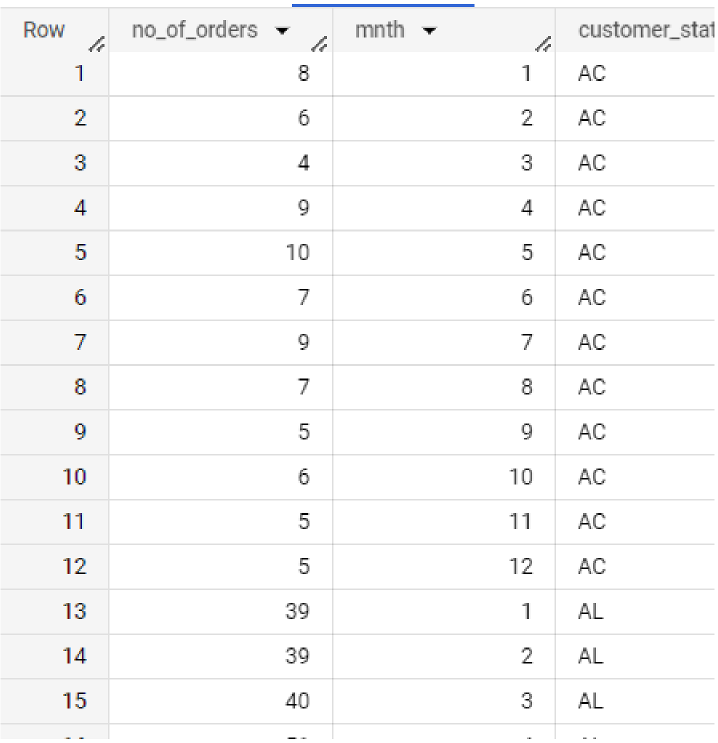
3-Evolution of E-commerce orders in the Brazil region:

3A: Get the month on month no.of orders placed in each state.

SQL QUERY-

This is the monthly order received in from each state irrespective of year

```
select count(o.order_id) no_of_orders,extract(month from o.order_purchase_timestamp) mnth,
c.customer_state from `target.orders` o
left join
target.customers c on o.customer_id=c.customer_id where o.order_status != "canceled"
group by mnth,c.customer_state order by  customer_state , mnth
```



Row	no_of_orders	mnth	customer_stat
1	8	1	AC
2	6	2	AC
3	4	3	AC
4	9	4	AC
5	10	5	AC
6	7	6	AC
7	9	7	AC
8	7	8	AC
9	5	9	AC
10	6	10	AC
11	5	11	AC
12	5	12	AC
13	39	1	AL
14	39	2	AL
15	40	3	AL

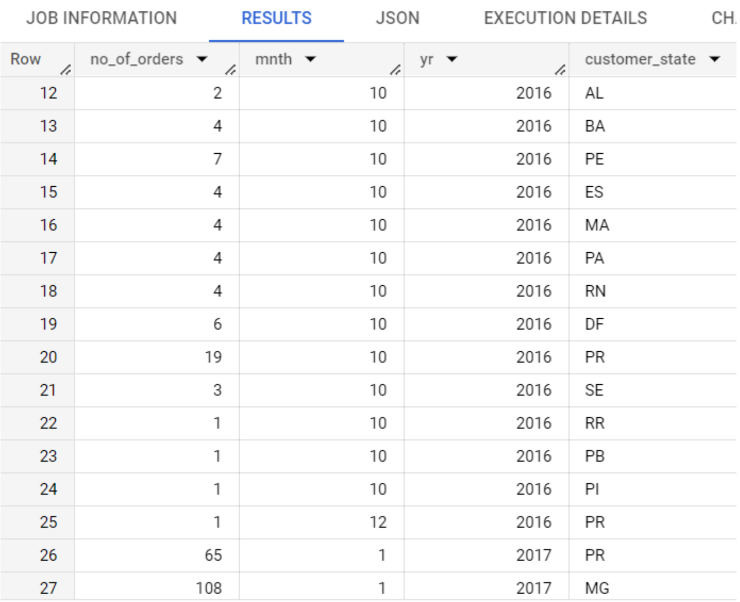
→Get the month on month no.of orders placed in each state every year

SQL QUERY-

```
select count(o.order_id) no_of_orders ,extract(month from o.order_purchase_timestamp) mnth
,extract(year from o.order_purchase_timestamp) yr, c.customer_state

from `target.orders` o left join  target.customers c on o.customer_id=c.customer_id where
o.order_status != "canceled"
group by yr, mnth,c.customer_state order by customer_state,yr, mnth
```

This result gives us information about order placed in each month from each year and each state



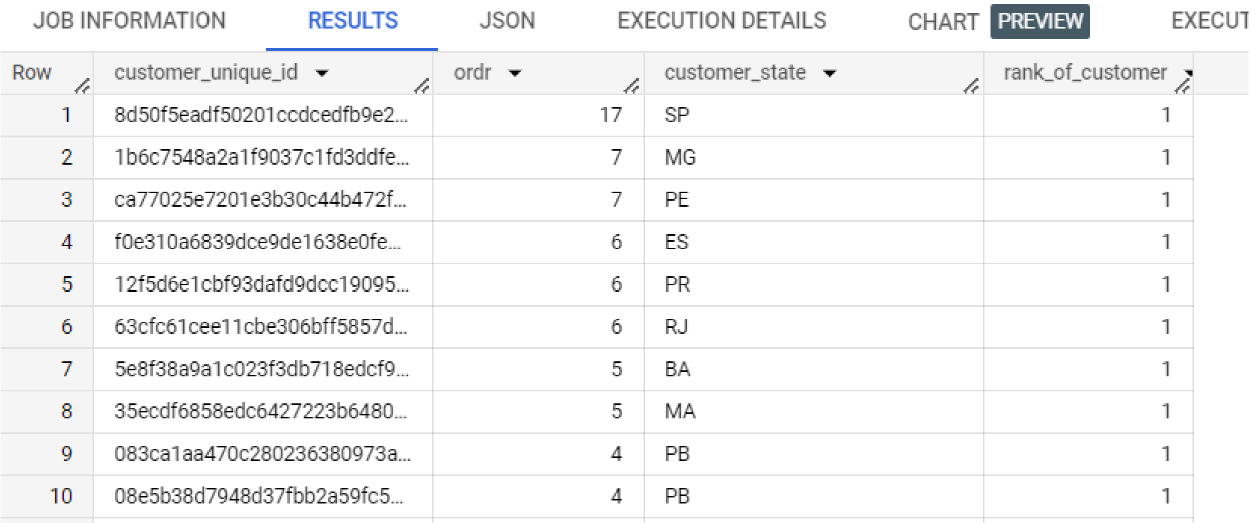
Row	no_of_orders	mnth	yr	customer_state
12	2	10	2016	AL
13	4	10	2016	BA
14	7	10	2016	PE
15	4	10	2016	ES
16	4	10	2016	MA
17	4	10	2016	PA
18	4	10	2016	RN
19	6	10	2016	DF
20	19	10	2016	PR
21	3	10	2016	SE
22	1	10	2016	RR
23	1	10	2016	PB
24	1	10	2016	PI
25	1	12	2016	PR
26	65	1	2017	PR
27	108	1	2017	MG

3B: How are the customers distributed across all the states?

SQL QUERY-

```
select * from (select c.customer_unique_id, count(o.order_id) ordr, c.customer_state,rank()
over (partition by customer_state order by count(o.order_id) desc) as rank_of_customer from
`target.orders` o RIGHT JOIN  target.customers c on o.customer_id=c.customer_id where
o.order_status != "canceled"
group by c.customer_unique_id, c.customer_state) x where rank_of_customer=1 order by x.ordr
desc
```

I have selected those customer who is unique in terms of placing order than any other customer in each state (in this results it shows the customers who placed more than one order)

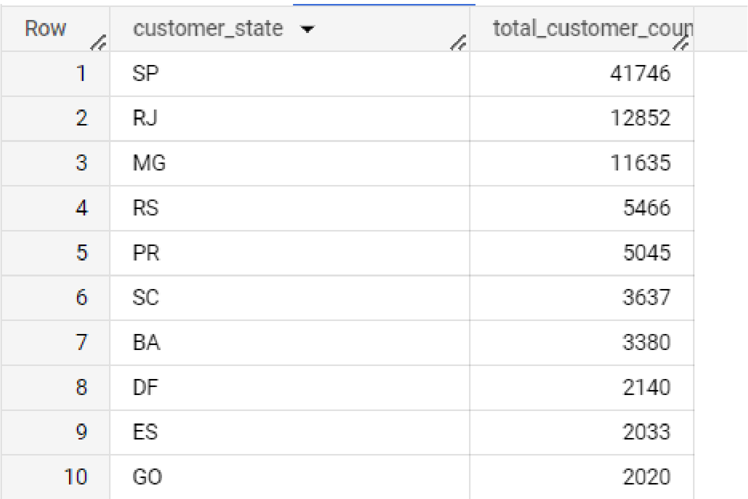


Row	customer_unique_id	ordr	customer_state	rank_of_customer
1	8d50f5eadf50201ccdcedfb9e2...	17	SP	1
2	1b6c7548a2a1f9037c1fd3ddfe...	7	MG	1
3	ca77025e7201e3b30c44b472f...	7	PE	1
4	f0e310a6839dce9de1638e0fe...	6	ES	1
5	12f5d6e1cbf93dafd9dcc19095...	6	PR	1
6	63cfc61cee11cbe306bff5857d...	6	RJ	1
7	5e8f38a9a1c023f3db718edcf9...	5	BA	1
8	35ecdf6858edc6427223b6480...	5	MA	1
9	083ca1aa470c280236380973a...	4	PB	1
10	08e5b38d7948d37fbb2a59fc5...	4	PB	1

→Distribution of customers across the states in Brazil

SQL QUERY-

```
select customer_state,count(customer_id) as total_customer_count
from `target.customers`
group by customer_state
order by total_customer_count desc;
```



Row	customer_state	total_customer_coun
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

Actionable insights:

State SP has highest customer count

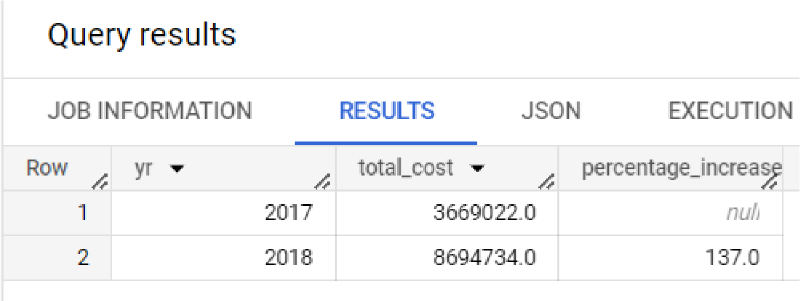
4- Impact on Economy

4A: Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

```
with cte1 as  (select p.order_id, p.payment_value, extract(year from
o.order_purchase_timestamp) yr,
extract(month from o.order_purchase_timestamp) mnth
from `target.payments` p left join target.orders o
on  p.order_id=o.order_id where extract(month from o.order_purchase_timestamp) between 1
and 8 and  extract(year from o.order_purchase_timestamp) between 2017 and 2018 order by
yr),
cte2 as
(select yr, round(sum(payment_value)) as total_cost, lag(round(sum(payment_value)), 1)
over(order by yr) as prev_value from cte1 group by yr)

select yr, total_cost, round(((total_cost-prev_value)/prev_value)*100) as
percentage_increase from cte2 order by yr
```

SQL QUERY-



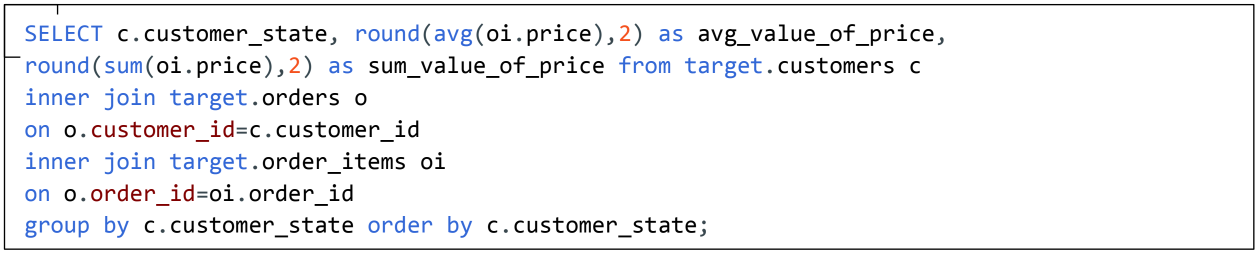
Row	yr	total_cost	percentage_increase
1	2017	3669022.0	null
2	2018	8694734.0	137.0

Actionable insights:

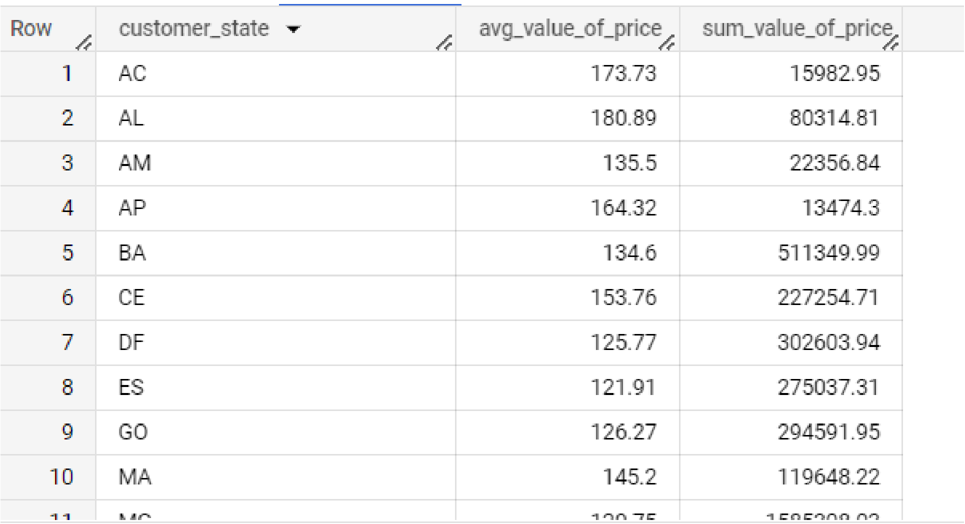
From year 2017 to 2018 137% total cost incresed

4B: Calculate the Total & Average value of order price for each state.

SQL QUERY-



```
SELECT c.customer_state, round(avg(oi.price),2) as avg_value_of_price,
round(sum(oi.price),2) as sum_value_of_price from target.customers c
inner join target.orders o
on o.customer_id=c.customer_id
inner join target.order_items oi
on o.order_id=oi.order_id
group by c.customer_state order by c.customer_state;
```

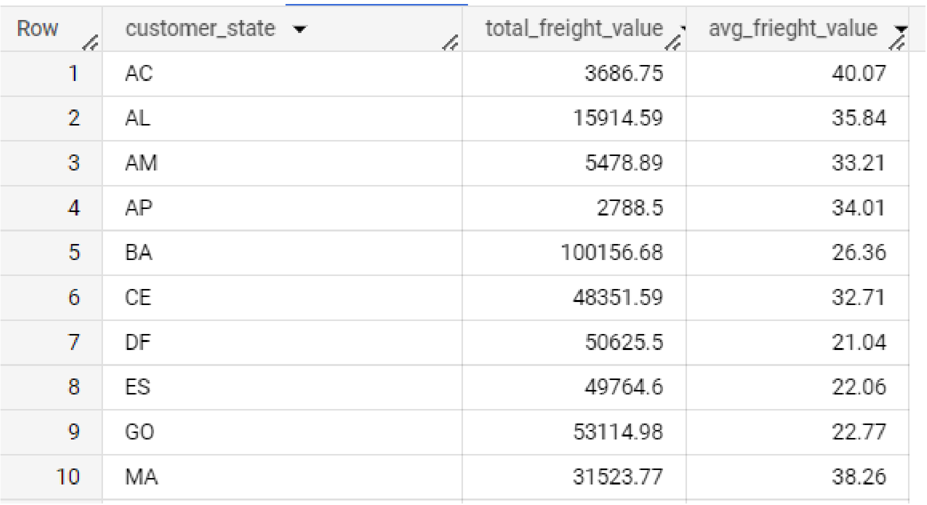


Row	customer_state	avg_value_of_price	sum_value_of_price
1	AC	173.73	15982.95
2	AL	180.89	80314.81
3	AM	135.5	22356.84
4	AP	164.32	13474.3
5	BA	134.6	511349.99
6	CE	153.76	227254.71
7	DF	125.77	302603.94
8	ES	121.91	275037.31
9	GO	126.27	294591.95
10	MA	145.2	119648.22
11	MG	120.75	1585308.03

4C: Calculate the Total & Average value of order freight for each state.

SQL QUERY-

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



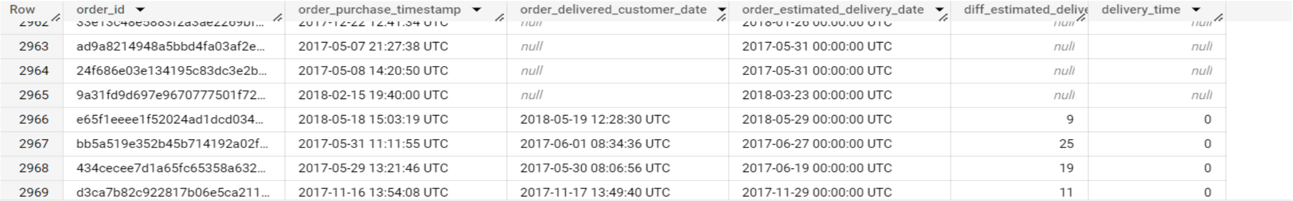
Row	customer_state	total_freight_value	avg_frieght_value
1	AC	3686.75	40.07
2	AL	15914.59	35.84
3	AM	5478.89	33.21
4	AP	2788.5	34.01
5	BA	100156.68	26.36
6	CE	48351.59	32.71
7	DF	50625.5	21.04
8	ES	49764.6	22.06
9	GO	53114.98	22.77
10	MA	31523.77	38.26

5- Analysis based on sales, freight and delivery time.

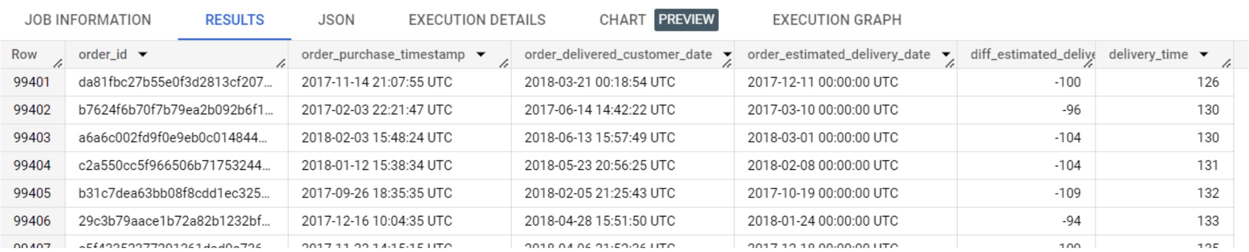
5A: Find the no.of days taken to deliver each order from the order’s purchase date as delivery time. Also, calculate the difference (in days) between the estimated & actual delivery date of an order. Do this in a single query.

SQL Query-

```
select
order_id,order_purchase_timestamp,order_delivered_customer_date,order_estimated_delivery_da
te,
date_diff(order_estimated_delivery_date,order_delivered_customer_date, day) as
diff_estimated_delivery,
date_diff(order_delivered_customer_date,order_purchase_timestamp , day) as delivery_time
from target.orders
order by delivery_time
```



Row	order_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	diff_estimated_delive	delivery_time
2962	33e13c48e5883f2a3ae2269bf...	2017-12-22 12:41:34 UTC	null	2018-01-26 00:00:00 UTC	null	null
2963	ad9a8214948a5bbd4fa03af2e...	2017-05-07 21:27:38 UTC	null	2017-05-31 00:00:00 UTC	null	null
2964	24f686e03e134195c83dc3e2b...	2017-05-08 14:20:50 UTC	null	2017-05-31 00:00:00 UTC	null	null
2965	9a31fd9d697e9670777501f72...	2018-02-15 19:40:00 UTC	null	2018-03-23 00:00:00 UTC	null	null
2966	e65f1eeee1f52024ad1dcd034...	2018-05-18 15:03:19 UTC	2018-05-19 12:28:30 UTC	2018-05-29 00:00:00 UTC	9	0
2967	bb5a519e352b45b714192a02f...	2017-05-31 11:11:55 UTC	2017-06-01 08:34:36 UTC	2017-06-27 00:00:00 UTC	25	0
2968	434cecee7d1a65fc65358a632...	2017-05-29 13:21:46 UTC	2017-05-30 08:06:56 UTC	2017-06-19 00:00:00 UTC	19	0
2969	d3ca7b82c922817b06e5ca211...	2017-11-16 13:54:08 UTC	2017-11-17 13:49:40 UTC	2017-11-29 00:00:00 UTC	11	0



JOB INFORMATION RESULTS JSON EXECUTION DETAILS CHART PREVIEW EXECUTION GRAPH

Row	order_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	diff_estimated_delive	delivery_time
99401	da81fbc27b55e0f3d2813cf207...	2017-11-14 21:07:55 UTC	2018-03-21 00:18:54 UTC	2017-12-11 00:00:00 UTC	-100	126
99402	b7624f6b70f7b79ea2b092b6f1...	2017-02-03 22:21:47 UTC	2017-06-14 14:42:22 UTC	2017-03-10 00:00:00 UTC	-96	130
99403	a6a6c002fd9f0e9eb0c014844...	2018-02-03 15:48:24 UTC	2018-06-13 15:57:49 UTC	2018-03-01 00:00:00 UTC	-104	130
99404	c2a550cc5f966506b71753244...	2018-01-12 15:38:34 UTC	2018-05-23 20:56:25 UTC	2018-02-08 00:00:00 UTC	-104	131
99405	b31c7dea63bb08f8cdd1ec325...	2017-09-26 18:35:35 UTC	2018-02-05 21:25:43 UTC	2017-10-19 00:00:00 UTC	-109	132
99406	29c3b79aace1b72a82b1232bf...	2017-12-16 10:04:35 UTC	2018-04-28 15:51:50 UTC	2018-01-24 00:00:00 UTC	-94	133
99407	[illegible]	2017-11-22 14:15:15 UTC	2018-04-06 21:52:36 UTC	2017-12-18 00:00:00 UTC	-109	135

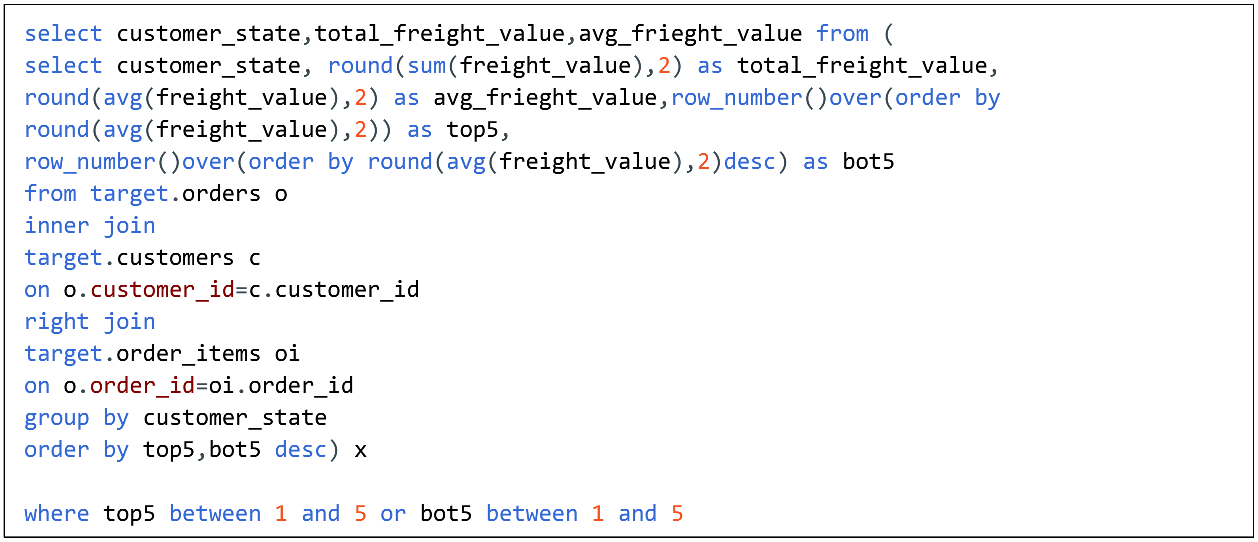
Actionable insights:

- By observing above results we can conclude that if delivery time is 0 it means order delivery at same day (within 24hrs)

- If diff_estimated_delivery_date 9 or 25 it means order delivered before 9 or 25 days before estimated delivery date (in other words customer received order before estimated delivery date)

- If diff_estimated_delivery_date is a negative value suppose ‘-n’ it means customer received order n days late than estimated date

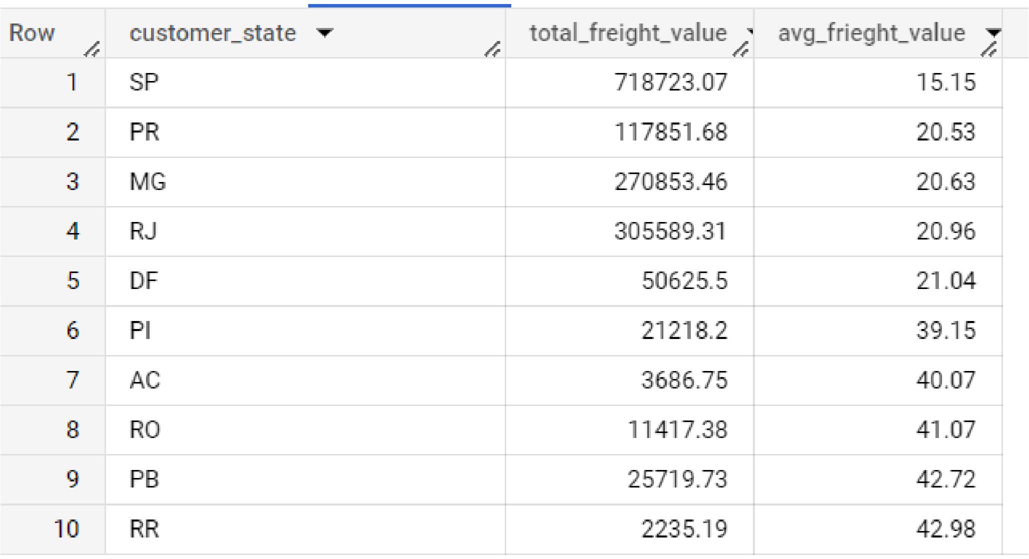
5B: Find out the top 5 states with the highest & lowest average freight value.



```
select customer_state,total_freight_value,avg_frieght_value from (
select customer_state, round(sum(freight_value),2) as total_freight_value,
round(avg(freight_value),2) as avg_frieght_value,row_number()over(order by
round(avg(freight_value),2)) as top5,
row_number()over(order by round(avg(freight_value),2)desc) as bot5
from target.orders o
inner join
target.customers c
on o.customer_id=c.customer_id
right join
target.order_items oi
on o.order_id=oi.order_id
group by customer_state
order by top5,bot5 desc) x

where top5 between 1 and 5 or bot5 between 1 and 5
```

SQL QUERY-



Row	customer_state	total_freight_value	avg_frieght_value
1	SP	718723.07	15.15
2	PR	117851.68	20.53
3	MG	270853.46	20.63
4	RJ	305589.31	20.96
5	DF	50625.5	21.04
6	PI	21218.2	39.15
7	AC	3686.75	40.07
8	RO	11417.38	41.07
9	PB	25719.73	42.72
10	RR	2235.19	42.98

Actionable insights:

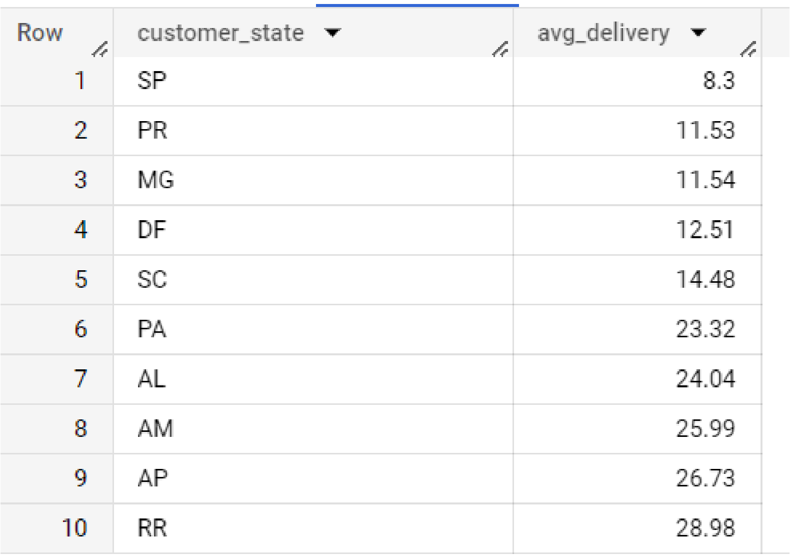
1 to 5 are the state with top 5 lowest avg freight value, and 5 to 10 are the state with top 5 highest avg freight value (both are in increasing order)

5-C: Find out the top5 states with the highest & lowest average delivery time.

SQL QUERY-

```
with cte1 as
(select customer_state,
date_diff(order_delivered_customer_date,order_purchase_timestamp , day) as delivery_time
from target.orders o
left join
target.customers c
on o.customer_id=c.customer_id
where date_diff(order_delivered_customer_date,order_purchase_timestamp , day) is not null
order by delivery_time),
cte2 as
(select customer_state, round(avg(delivery_time),2) as avg_delivery,
row_number() over (order by round(avg(delivery_time),2)) as top5,
row_number() over (order by round(avg(delivery_time),2) desc) as bot5 from  cte1
group by customer_state
order by avg_delivery)

select customer_state,avg_delivery from cte2
where top5 between 1 and 5 or bot5 between 1 and 5
```



Row	customer_state	avg_delivery
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48
6	PA	23.32
7	AL	24.04
8	AM	25.99
9	AP	26.73
10	RR	28.98

Actionable insights:

- Top 5 (1 to5) state are with lowest avg_delivery time in day
- Bot 5 (6 to10) are the state with highest avg_delivery time in day (both are in increasing order)

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

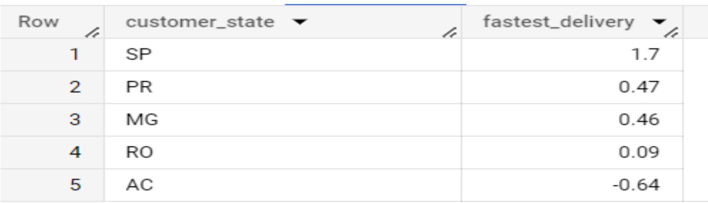
Hint–You can use the difference between the averages of actual & estimated delivery date *to figure out how fast the delivery was for each state.*

SQL QUERY-

In below query first calculated the difference of days between estimated delivery and actual delivery, and the difference of days between order purchase day and order delivered date to customer, then find the averages of both, after finding the averages of them subtract both as per question hints difference between the averages of actual and estimated delivery .

```
with cte1 as
(select
customer_state,date_diff(order_estimated_delivery_date,order_delivered_customer_date, day)
as estimated_delivery,
date_diff(order_delivered_customer_date,order_purchase_timestamp , day) as delivery_time
from target.orders o
left join
target.customers c
on o.customer_id=c.customer_id
where date_diff(order_delivered_customer_date,order_purchase_timestamp , day) is not null
order by delivery_time,estimated_delivery),
cte2 as
(select customer_state, round(avg(delivery_time),2) as
actual_delivery_date,round(avg(estimated_delivery)) as avg_estimated_delivery from cte1
group by customer_state)

select customer_state, round((avg_estimated_delivery-actual_delivery_date),2) as
fastest_delivery from cte2
order by fastest_delivery desc
limit 5
```



Row	customer_state	fastest_delivery
1	SP	1.7
2	PR	0.47
3	MG	0.46
4	RO	0.09
5	AC	-0.64

Actionable insights:

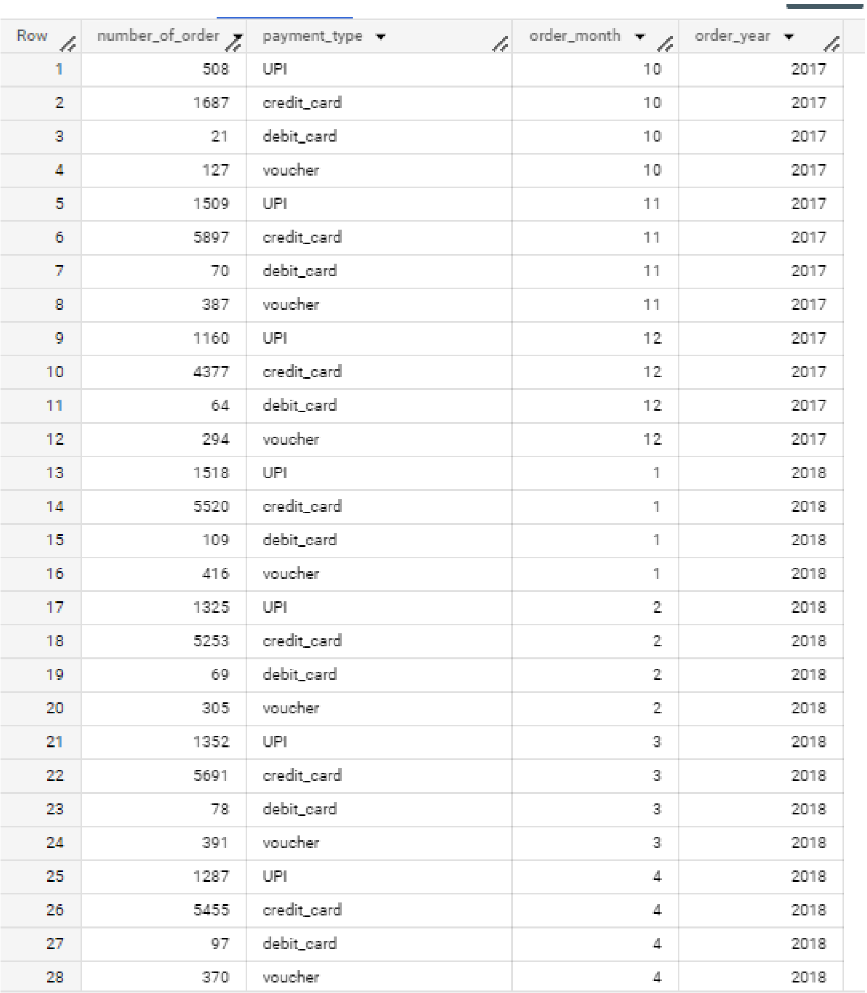
- SP state has fastest rate of delivery
- AC state has **negative value** that means in this state order received late than estimated delivery

6-A: Find the month on month no.of orders placed using different payment types over the past year.

SQL QUERY-

```
with cte1 as
(select
o.order_id, p.payment_type,extract(date from o.order_purchase_timestamp) as
order_date,extract(year from o.order_purchase_timestamp) as order_year,
extract(month from o.order_purchase_timestamp) as order_month
from target.payments p
left join
target.orders o
on p.order_id=o.order_id
order by order_date desc)

select count(order_id) number_of_order, payment_type, order_month, order_year from cte1
where order_date<='2018-10-17' and order_date>='2017-10-17'
group by payment_type, order_month, order_year
order by order_year, order_month, payment_type
```



Row	number_of_order	payment_type	order_month	order_year
1	508	UPI	10	2017
2	1687	credit_card	10	2017
3	21	debit_card	10	2017
4	127	voucher	10	2017
5	1509	UPI	11	2017
6	5897	credit_card	11	2017
7	70	debit_card	11	2017
8	387	voucher	11	2017
9	1160	UPI	12	2017
10	4377	credit_card	12	2017
11	64	debit_card	12	2017
12	294	voucher	12	2017
13	1518	UPI	1	2018
14	5520	credit_card	1	2018
15	109	debit_card	1	2018
16	416	voucher	1	2018
17	1325	UPI	2	2018
18	5253	credit_card	2	2018
19	69	debit_card	2	2018
20	305	voucher	2	2018
21	1352	UPI	3	2018
22	5691	credit_card	3	2018
23	78	debit_card	3	2018
24	391	voucher	3	2018
25	1287	UPI	4	2018
26	5455	credit_card	4	2018
27	97	debit_card	4	2018
28	370	voucher	4	2018

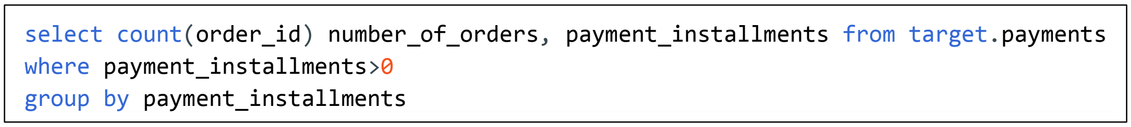
Actionable insights:

- Total 4 types of payment and by above results we can conclude that over the past year (between “2018-10-17” to “2017-10-17”) month on month basis using different types of payment mode how many order were placed.

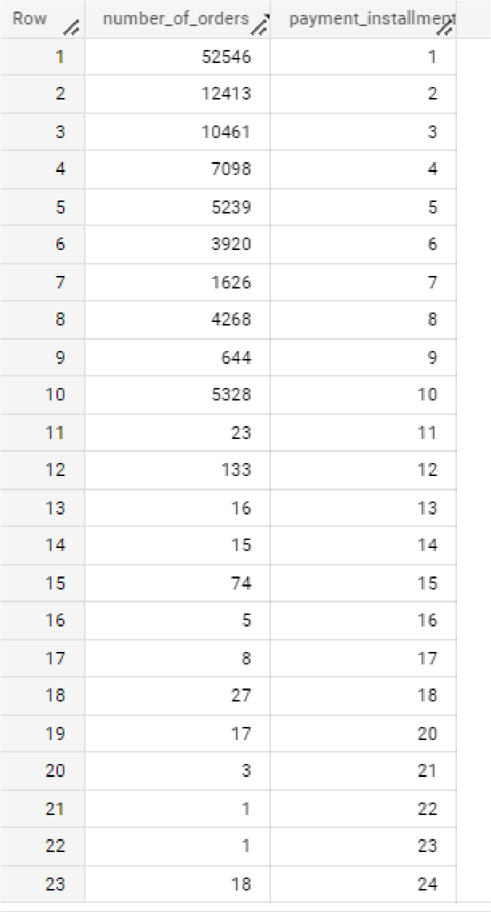
- Customers mostly placed order using credit card and least order placed through debit card.

6-B: Find the no.of orders placed on the basis of the payment installments that have been paid.

SQL Query-



```
select count(order_id) number_of_orders, payment_installments from target.payments
where payment_installments>0
group by payment_installments
```



Row	number_of_orders	payment_installme
1	52546	1
2	12413	2
3	10461	3
4	7098	4
5	5239	5
6	3920	6
7	1626	7
8	4268	8
9	644	9
10	5328	10
11	23	11
12	133	12
13	16	13
14	15	14
15	74	15
16	5	16
17	8	17
18	27	18
19	17	20
20	3	21
21	1	22
22	1	23
23	18	24

Recommendations:

- **In the cities and states where we have less number of customers and orders**

Required root cause analysis for those states where we have very less no. of customers. Check the connectivity issue, social networking site availability of people. If required, plan some kind of advertisement

- **What time Brazilians do orders**

For instant delivery in afternoon and evening, we need more no. of delivery persons

- **The Average value and Sum of order price and freight value for each state**

It is recommended that If freight value is high for low cost product then accept only when order quantity is in bulk

- **State with highest freight value**

Focus on those states where average freight value and time delivery are high (RR state). Check nearest retail shop and check whether all required products are available or not

- **Month to Month count of orders for different payment types**

Provide some discount to credit card customers on particular banks (need more analysis)

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

Focus on those states where average time delivery is very high (RR state). Check logistics issue, no. of retail shops, connectivity, inventory.