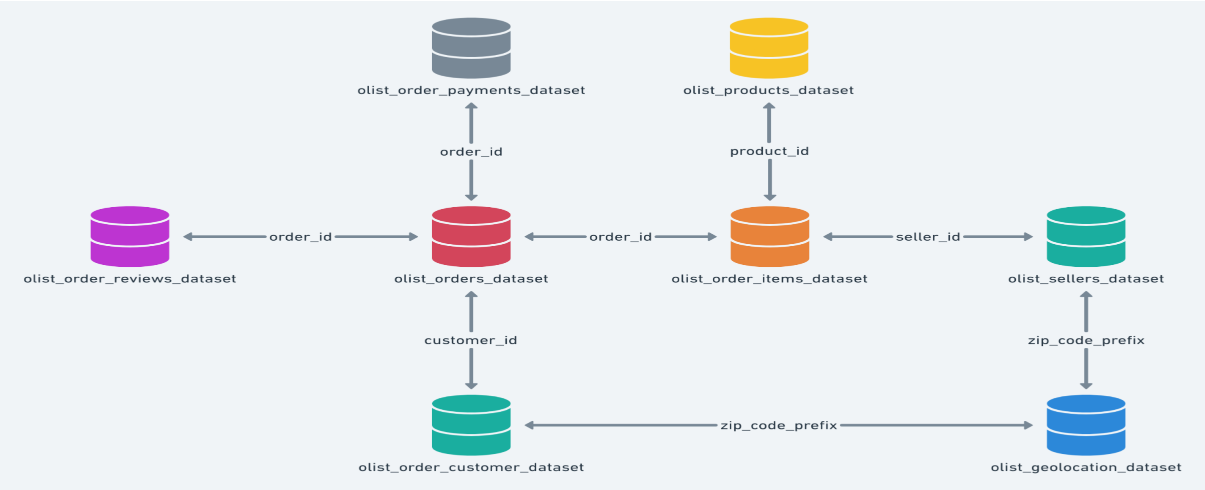
Target

Business Case

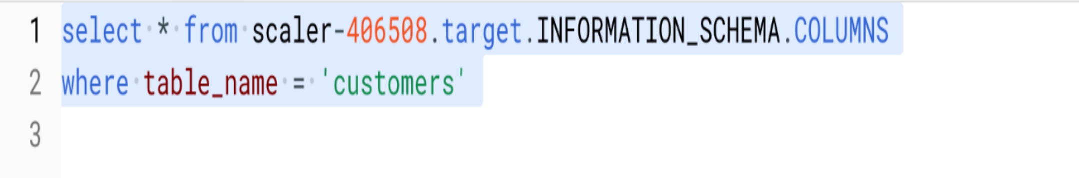
Scaler



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

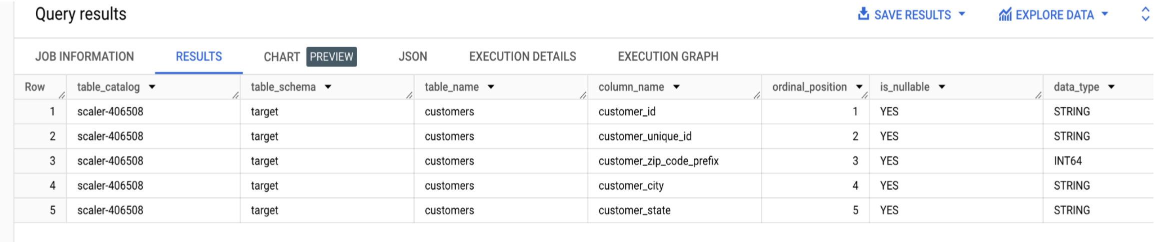
A. Data type of all columns in the “customers” table.

Query:



```
select * from scaler-406508.target.INFORMATION_SCHEMA.COLUMNS
where table_name = 'customers'

```

Results:

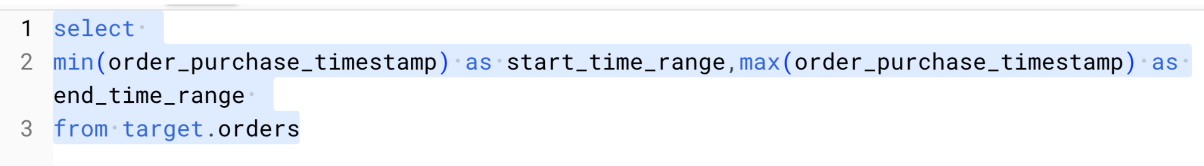


Row	table_catalog	table_schema	table_name	column_name	ordinal_position	is_nullable	data_type
1	scaler-406508	target	customers	customer_id	1	YES	STRING
2	scaler-406508	target	customers	customer_unique_id	2	YES	STRING
3	scaler-406508	target	customers	customer_zip_code_prefix	3	YES	INT64
4	scaler-406508	target	customers	customer_city	4	YES	STRING
5	scaler-406508	target	customers	customer_state	5	YES	STRING

Insights: *Data_type of all the columns from the customer table, most of the rows are of STRING type and only customer zip_code is of INT type.*

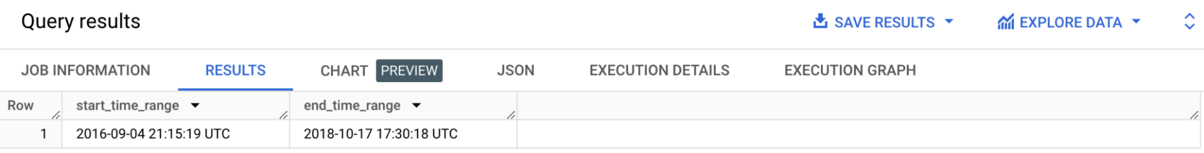
B. Get the time range between which the orders were placed.

Query:



```
select
min(order_purchase_timestamp) as start_time_range,max(order_purchase_timestamp) as end_time_range
from target.orders
```

Results:



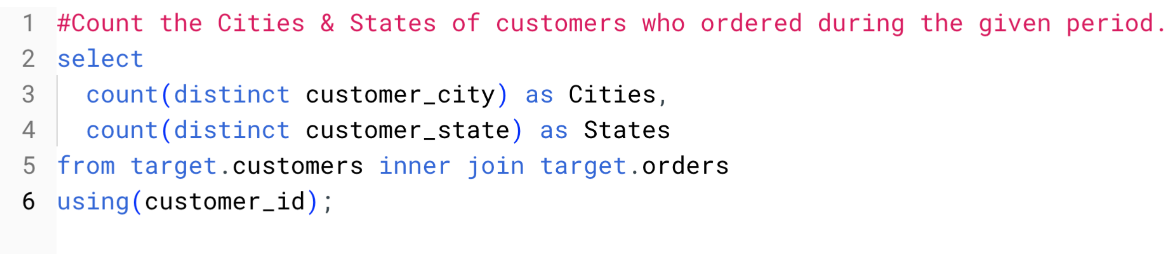
Query results

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

Insights: ***The orders were placed between ‘2016-09-04 21:15:19 utc’ and ‘2018-10-17 17:30:18 utc’***

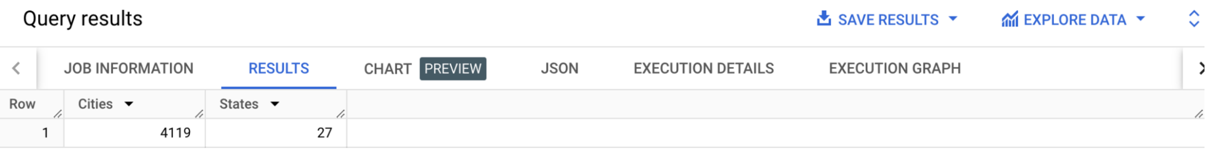
C. Count the Cities & States of customers who ordered during the given period.

Query:



```
#Count the Cities & States of customers who ordered during the given period.
select
  count(distinct customer_city) as Cities,
  count(distinct customer_state) as States
from target.customers inner join target.orders
using(customer_id);
```

Results:



Query results

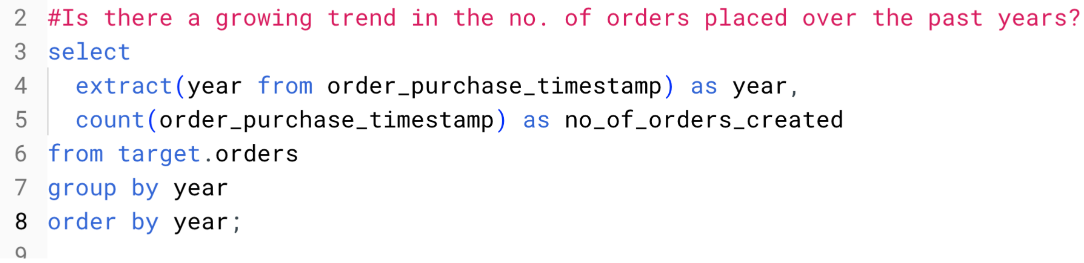
Row	Cities	States
1	4119	27

Insights: ***Total no of cities is 4119 & states is 27 from where orders were placed during the time period ‘2016-09-04 21:15:19 utc’ and ‘2018-10-17 17:30:18 utc’***

II. In-depth Exploration:

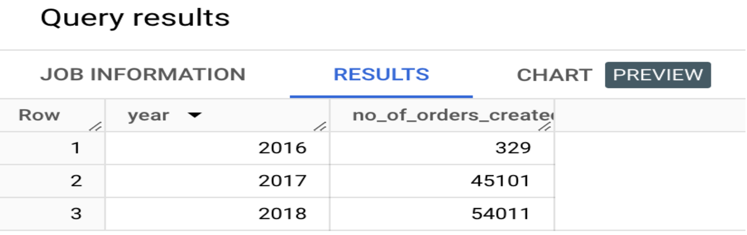
A. Is there a growing trend in the no. of orders placed over the past years?

Query:



```
#Is there a growing trend in the no. of orders placed over the past years?
select
  extract(year from order_purchase_timestamp) as year,
  count(order_purchase_timestamp) as no_of_orders_created
from target.orders
group by year
order by year;
```

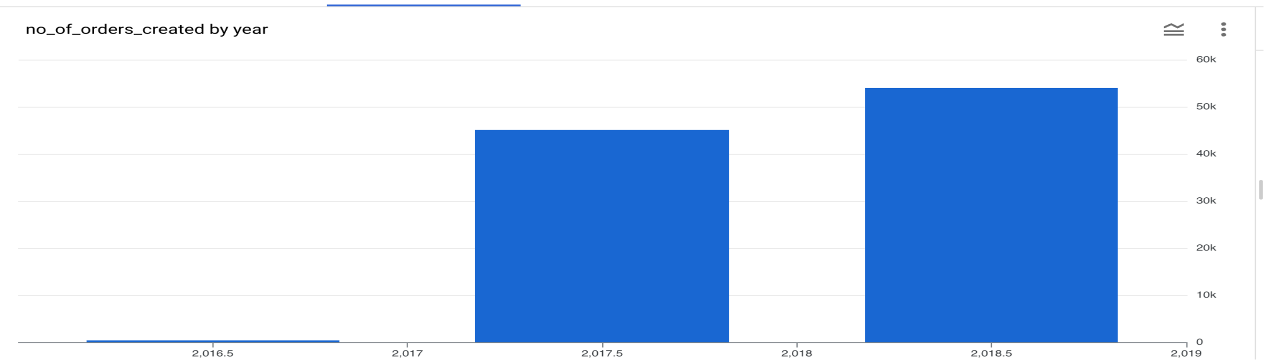
Results:



Query results

Row	year	no_of_orders_create
1	2016	329
2	2017	45101
3	2018	54011

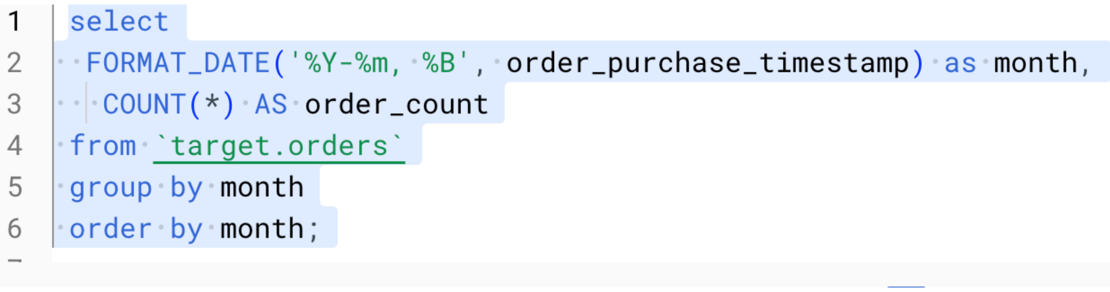
Charts:



Insights: *Over the years the orders placed have skyrocketed. In the year 2016 orders was 329 and in the year 2017 and 2018 orders climbed to 45101 to 54011*

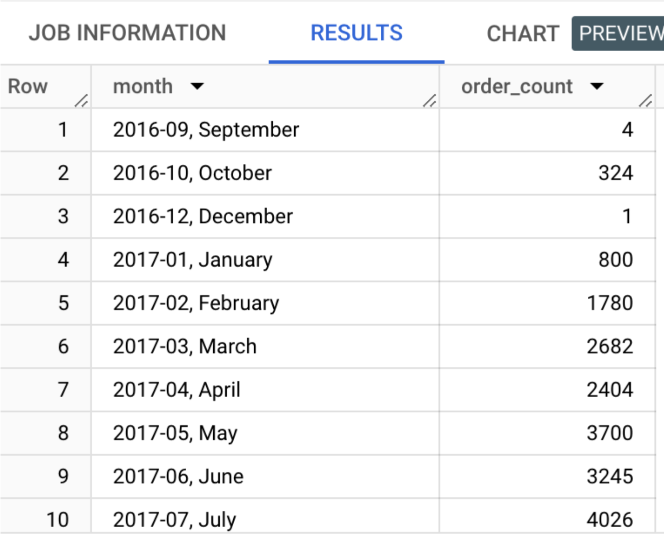
B. Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

Query:



```
select
  FORMAT_DATE('%Y-%m, %B', order_purchase_timestamp) as month,
   COUNT(*) AS order_count
from `target.orders`
group by month
order by month;
```

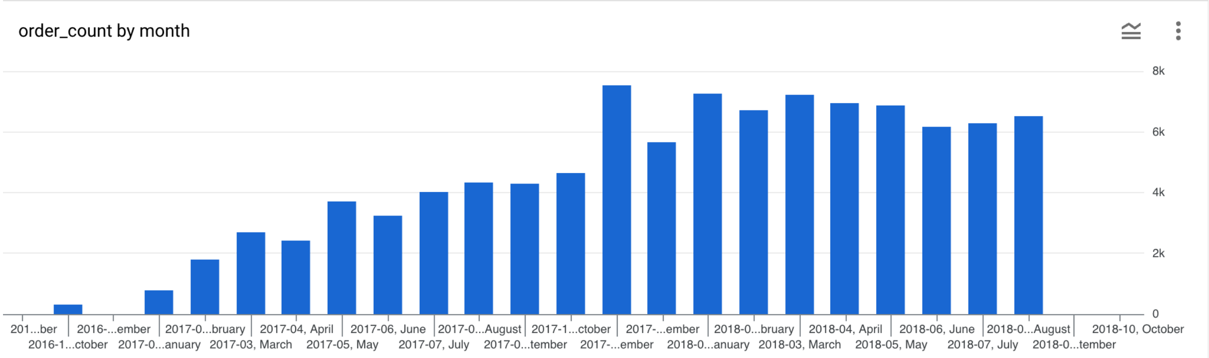
Results:



JOB INFORMATION | RESULTS | CHART PREVIEW

Row	month	order_count
1	2016-09, September	4
2	2016-10, October	324
3	2016-12, December	1
4	2017-01, January	800
5	2017-02, February	1780
6	2017-03, March	2682
7	2017-04, April	2404
8	2017-05, May	3700
9	2017-06, June	3245
10	2017-07, July	4026

Charts:

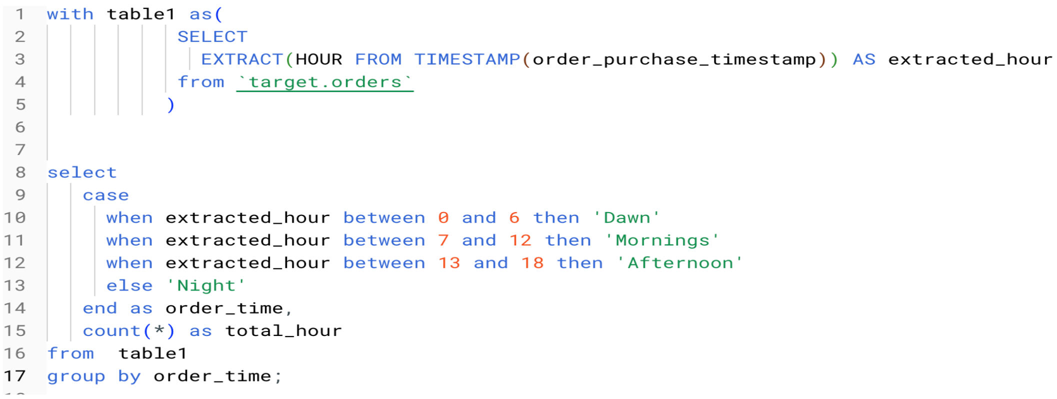


Insights: ***Yes we can see that over the months the orders have increased gradually and for the seasonality we can see that the second half of the year has always a linear increase in no of orders placed.***

C. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

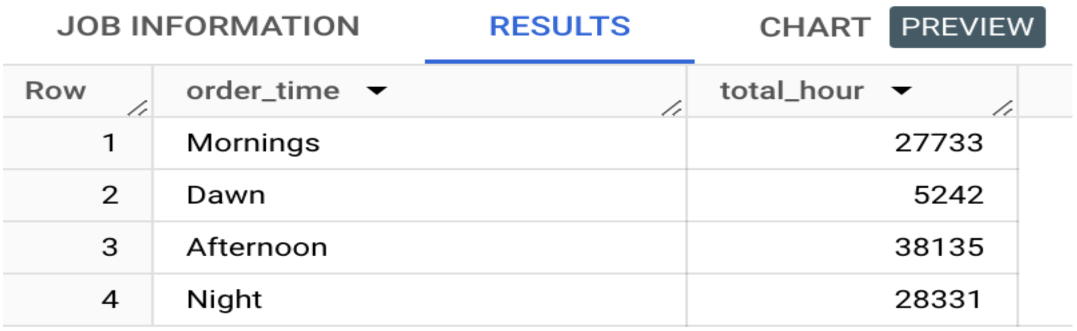
- 0-6 hrs : Dawn
- 7-12 hrs : Mornings
- 13-18 hrs : Afternoon
- 19-23 hrs : Night

Query:



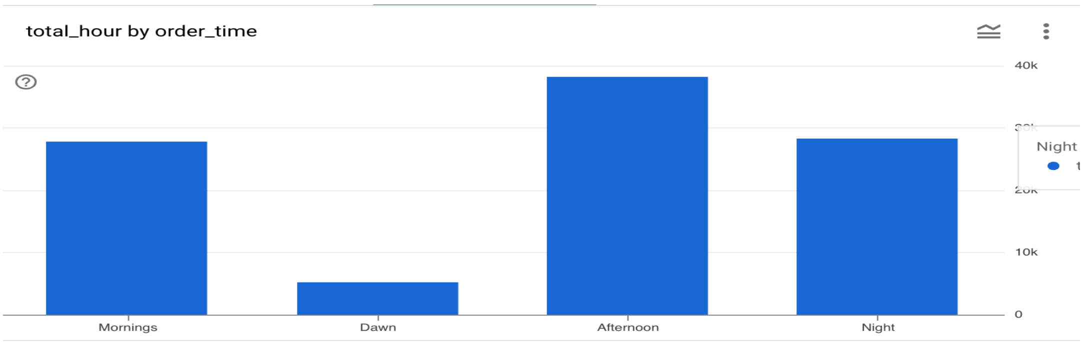
```
with table1 as(
                SELECT
                  EXTRACT(HOUR FROM TIMESTAMP(order_purchase_timestamp)) AS extracted_hour
                from `target.orders`
               )


select
   case
     when extracted_hour between 0 and 6 then 'Dawn'
     when extracted_hour between 7 and 12 then 'Mornings'
     when extracted_hour between 13 and 18 then 'Afternoon'
     else 'Night'
   end as order_time,
   count(*) as total_hour
from  table1
group by order_time;
```

Results:



Row	order_time	total_hour
1	Mornings	27733
2	Dawn	5242
3	Afternoon	38135
4	Night	28331

Charts:



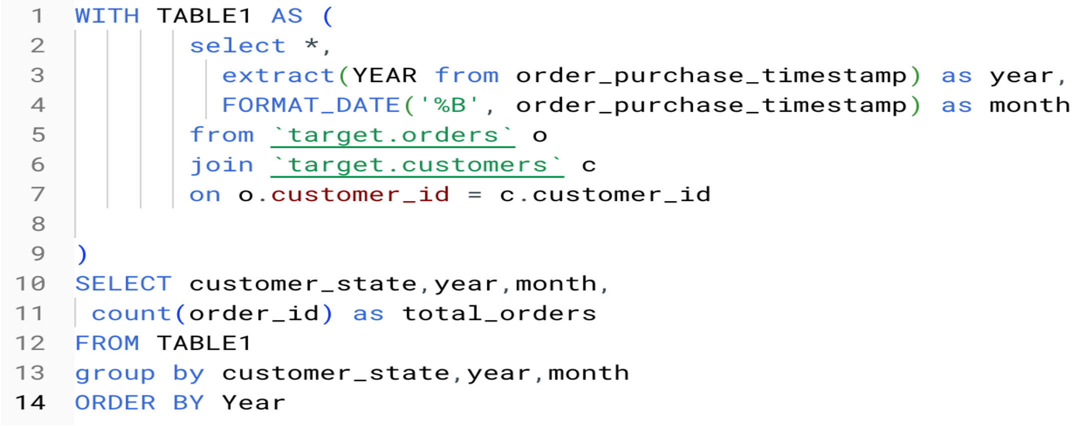
***Insights:** From the DATA we can see that Most of the Brazillians like to do shopping in the afternoon.*

***Recommendations:** Target can put up offers like the hourly sales in the afternoon as more customers are active during that point of the day.*

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

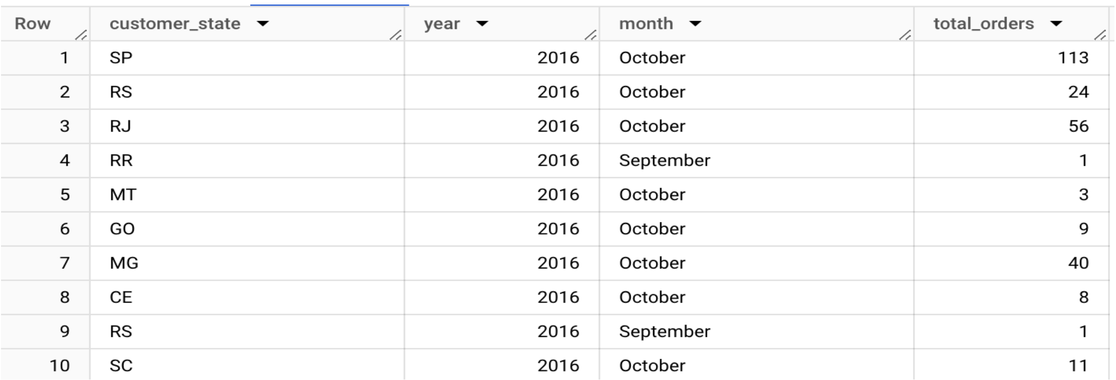
A. Get the month on month no. of orders placed in each state.

Query:



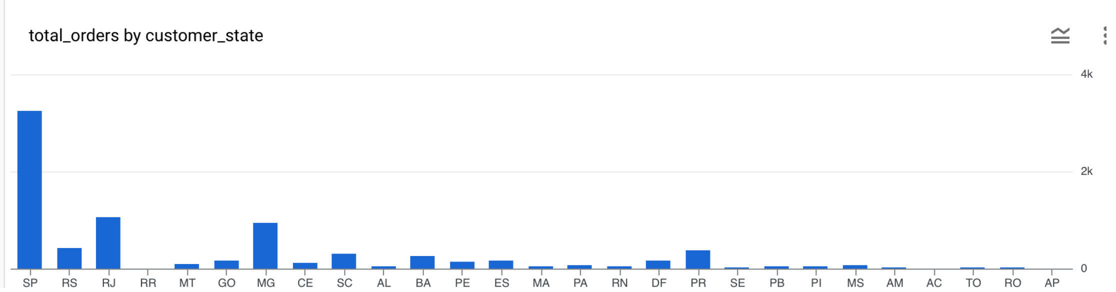
```
WITH TABLE1 AS (
       select *,
         extract(YEAR from order_purchase_timestamp) as year,
         FORMAT_DATE('%B', order_purchase_timestamp) as month
       from `target.orders` o
       join `target.customers` c
       on o.customer_id = c.customer_id

)
SELECT customer_state,year,month,
 count(order_id) as total_orders
FROM TABLE1
group by customer_state,year,month
ORDER BY Year
```

Results:



Row	customer_state	year	month	total_orders
1	SP	2016	October	113
2	RS	2016	October	24
3	RJ	2016	October	56
4	RR	2016	September	1
5	MT	2016	October	3
6	GO	2016	October	9
7	MG	2016	October	40
8	CE	2016	October	8
9	RS	2016	September	1
10	SC	2016	October	11

Charts:

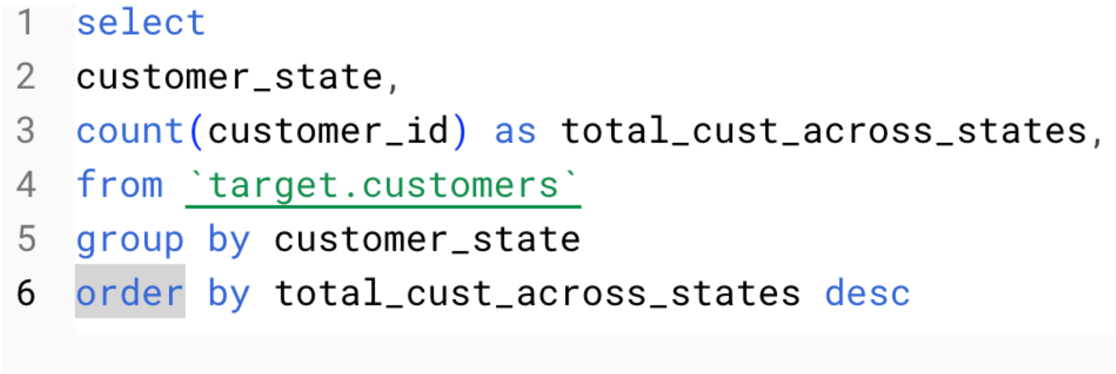


Insights: *From here we can see that most no of orders are pouring out from the states of SP RJ and MG*

Recommendations: *from here we can also check why not other states are not placing a good amount of orders and check on the situations .*

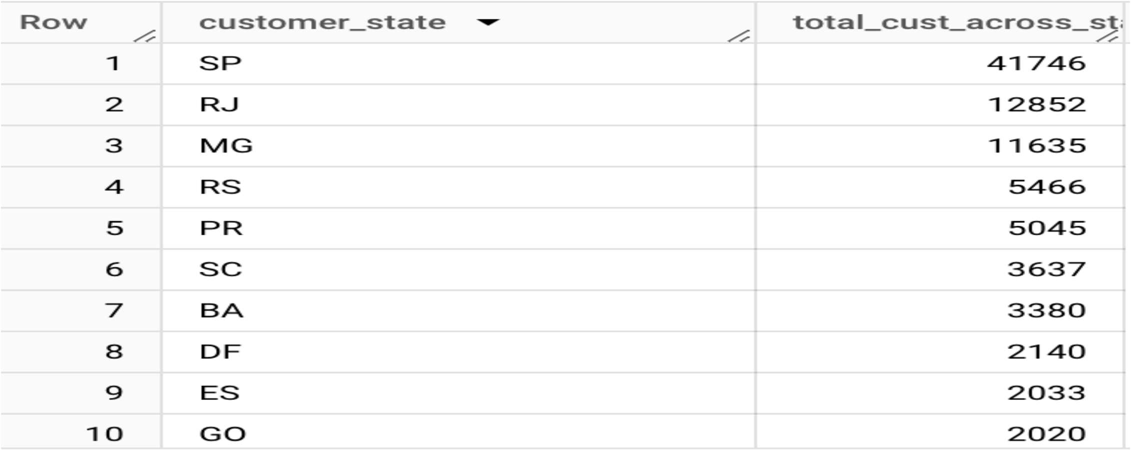
B. How are the customers distributed across all the states?

Query:



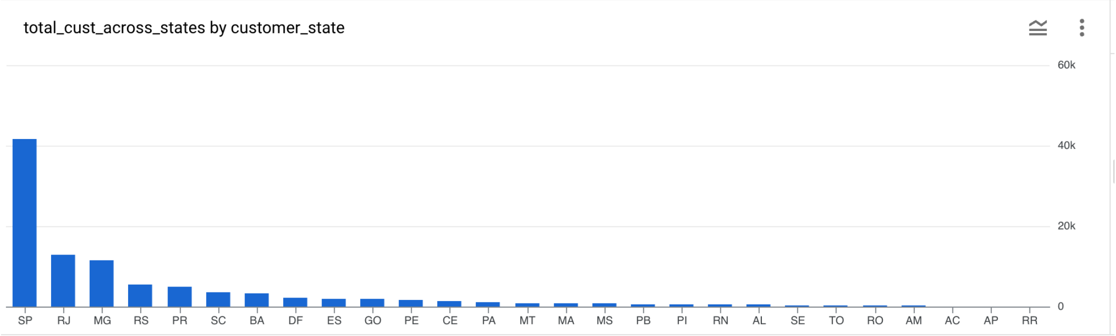
```
select
customer_state,
count(customer_id) as total_cust_across_states,
from `target.customers`
group by customer_state
order by total_cust_across_states desc
```

Results:



Row	customer_state	total_cust_across_st
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

Charts:



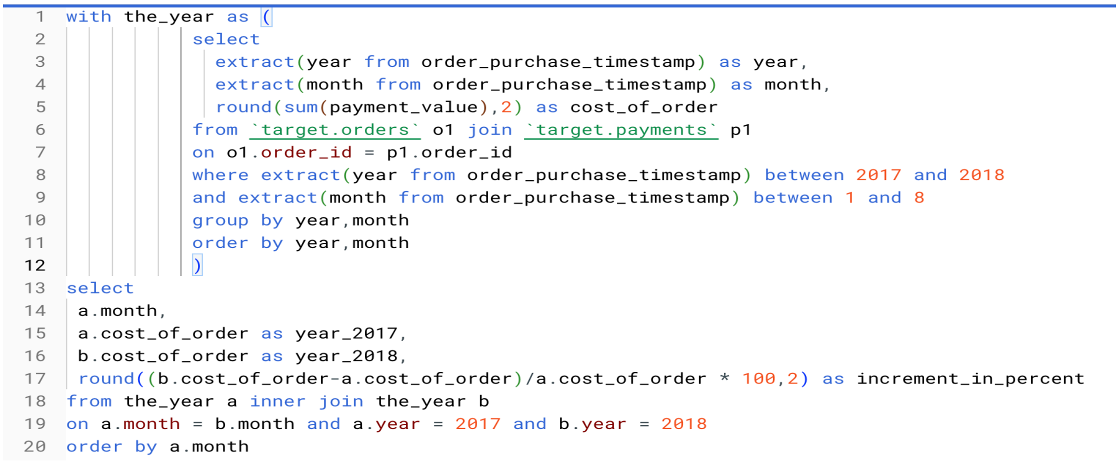
Insights: *From the chart you can see that 40k of the customers are in SP and almost all the states have less than 20k customers.*

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

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

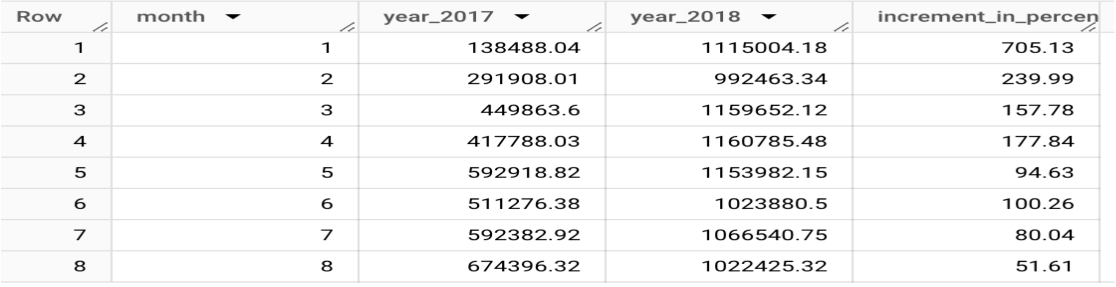
You can use the “payment_value” column in the payments table to get the cost of orders.

Query:



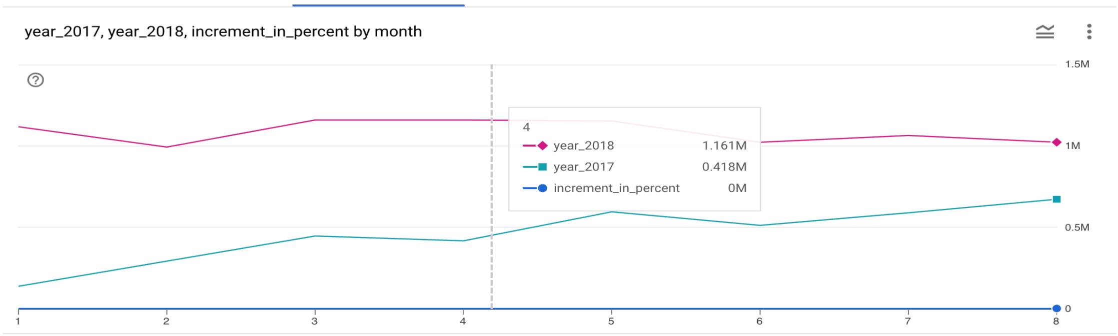
```
with the_year as (
           select
             extract(year from order_purchase_timestamp) as year,
             extract(month from order_purchase_timestamp) as month,
             round(sum(payment_value),2) as cost_of_order
           from `target.orders` o1 join `target.payments` p1
           on o1.order_id = p1.order_id
           where extract(year from order_purchase_timestamp) between 2017 and 2018
           and extract(month from order_purchase_timestamp) between 1 and 8
           group by year,month
           order by year,month
           )
select
 a.month,
 a.cost_of_order as year_2017,
 b.cost_of_order as year_2018,
 round((b.cost_of_order-a.cost_of_order)/a.cost_of_order * 100,2) as increment_in_percent
from the_year a inner join the_year b
on a.month = b.month and a.year = 2017 and b.year = 2018
order by a.month
```

Results:



Row	month	year_2017	year_2018	increment_in_percen
1	1	138488.04	1115004.18	705.13
2	2	291908.01	992463.34	239.99
3	3	449863.6	1159652.12	157.78
4	4	417788.03	1160785.48	177.84
5	5	592918.82	1153982.15	94.63
6	6	511276.38	1023880.5	100.26
7	7	592382.92	1066540.75	80.04
8	8	674396.32	1022425.32	51.61

Charts:

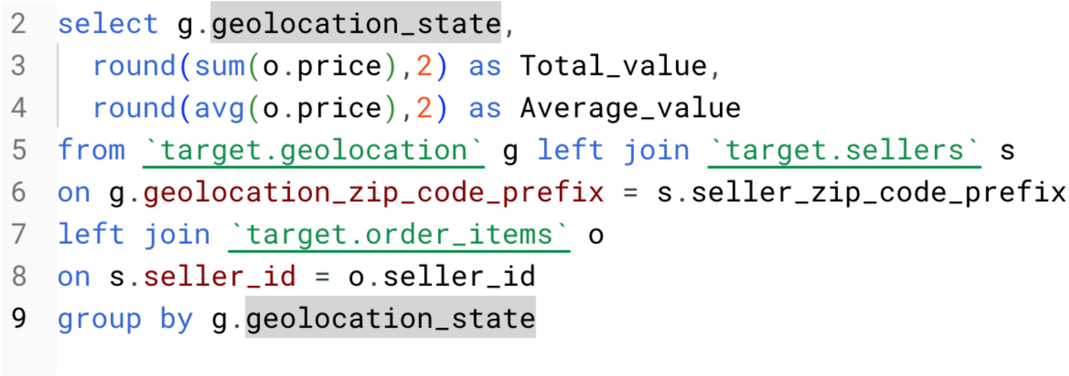


year_2017, year_2018, increment_in_percent by month

Insights: *There has been an increase in the percentage of orders over the month clearly from the graph.*

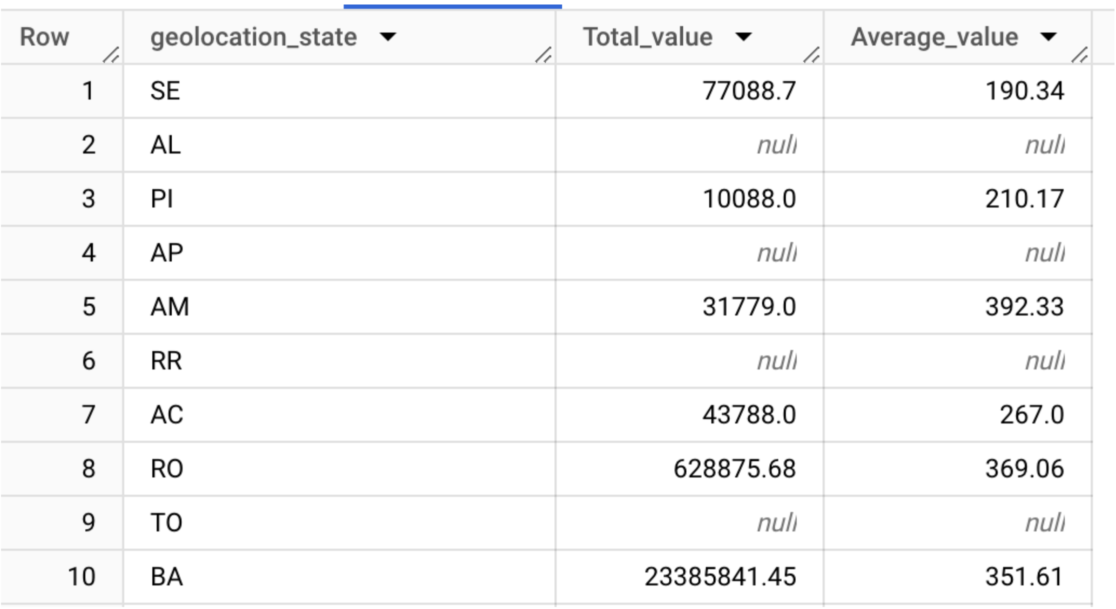
B. Calculate the Total & Average value of order price for each state.

Query:



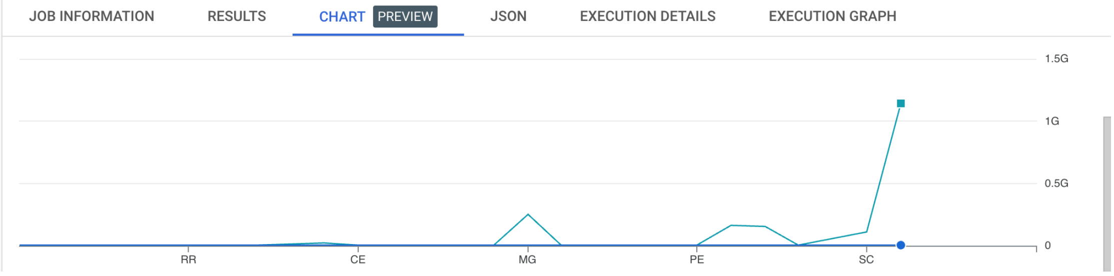
```
select g.geolocation_state,
  round(sum(o.price),2) as Total_value,
  round(avg(o.price),2) as Average_value
from `target.geolocation` g left join `target.sellers` s
on g.geolocation_zip_code_prefix = s.seller_zip_code_prefix
left join `target.order_items` o
on s.seller_id = o.seller_id
group by g.geolocation_state
```

Results:



Row	geolocation_state	Total_value	Average_value
1	SE	77088.7	190.34
2	AL	null	null
3	PI	10088.0	210.17
4	AP	null	null
5	AM	31779.0	392.33
6	RR	null	null
7	AC	43788.0	267.0
8	RO	628875.68	369.06
9	TO	null	null
10	BA	23385841.45	351.61

Charts:

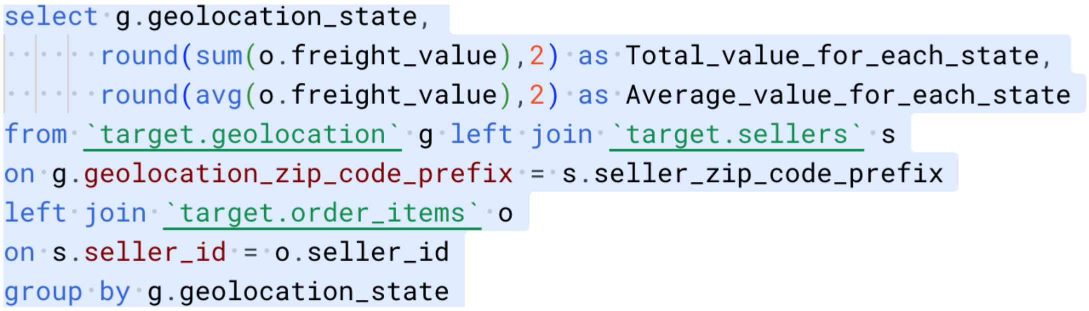


Insights:

We can see here the order price of each state, the total and average value .

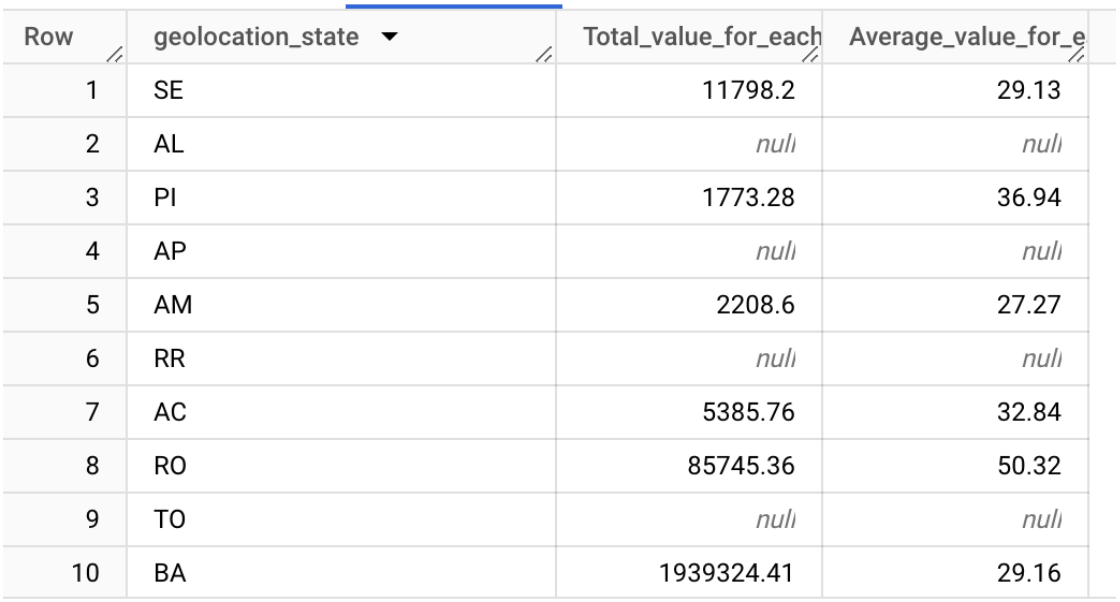
C. Calculate the Total & Average value of order freight for each state.

Query:



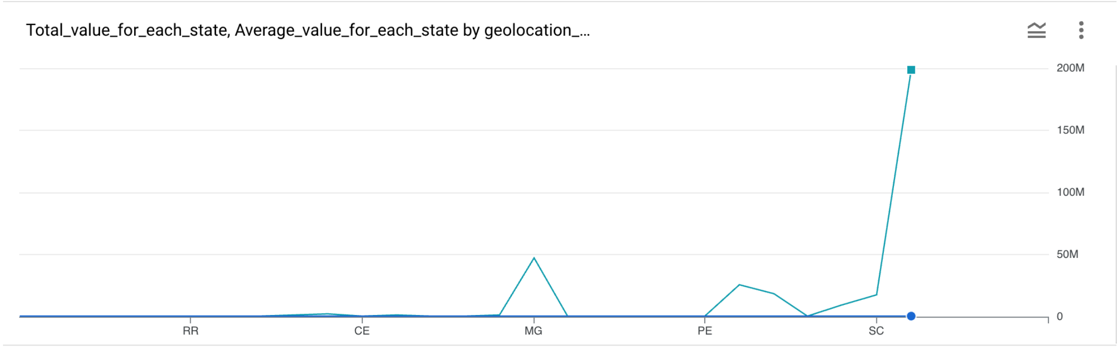
```
select g.geolocation_state,
      round(sum(o.freight_value),2) as Total_value_for_each_state,
      round(avg(o.freight_value),2) as Average_value_for_each_state
from `target.geolocation` g left join `target.sellers` s
on g.geolocation_zip_code_prefix = s.seller_zip_code_prefix
left join `target.order_items` o
on s.seller_id = o.seller_id
group by g.geolocation_state
```

Results:



Row	geolocation_state	Total_value_for_each	Average_value_for_e
1	SE	11798.2	29.13
2	AL	null	null
3	PI	1773.28	36.94
4	AP	null	null
5	AM	2208.6	27.27
6	RR	null	null
7	AC	5385.76	32.84
8	RO	85745.36	50.32
9	TO	null	null
10	BA	1939324.41	29.16

Charts:



Insights: ***The total and average freight_value for each state has been shown here.***

V. Analysis based on sales, freight and delivery time.

A. Find the no. of days taken to deliver each order from the order's purchase date as delivery time.

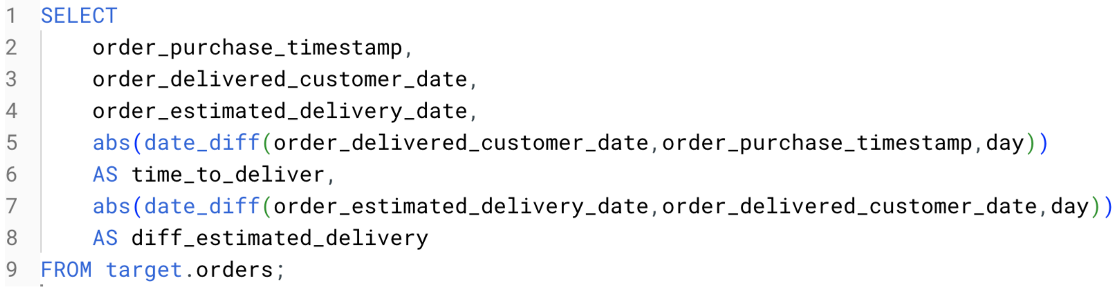
Also, calculate the difference (in days) between the estimated & actual delivery date of an order.

Do this in a single query.

You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

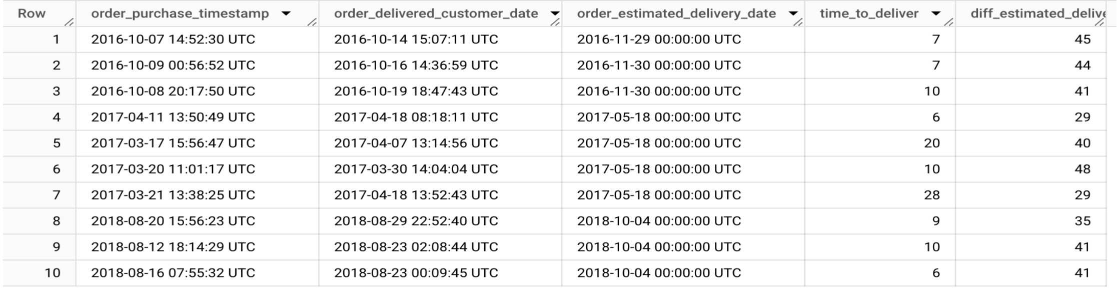
- **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
- **diff_estimated_delivery** = order_estimated_delivery_date - order_delivered_customer_date

Query:



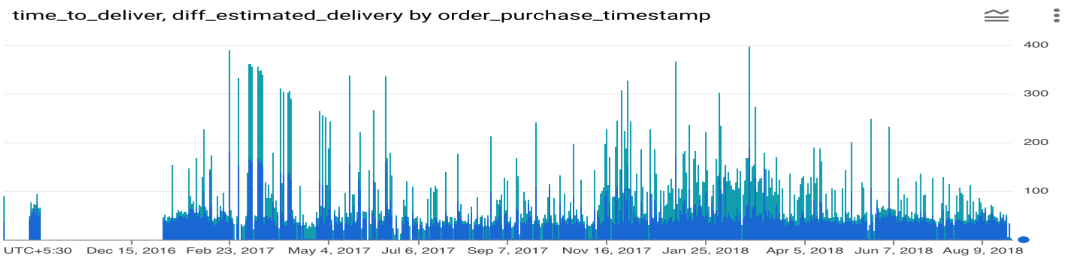
```sql
SELECT
    order_purchase_timestamp,
    order_delivered_customer_date,
    order_estimated_delivery_date,
    abs(date_diff(order_delivered_customer_date,order_purchase_timestamp,day))
    AS time_to_deliver,
    abs(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day))
    AS diff_estimated_delivery
FROM target.orders;
```

Results:



Row	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_delive
1	2016-10-07 14:52:30 UTC	2016-10-14 15:07:11 UTC	2016-11-29 00:00:00 UTC	7	45
2	2016-10-09 00:56:52 UTC	2016-10-16 14:36:59 UTC	2016-11-30 00:00:00 UTC	7	44
3	2016-10-08 20:17:50 UTC	2016-10-19 18:47:43 UTC	2016-11-30 00:00:00 UTC	10	41
4	2017-04-11 13:50:49 UTC	2017-04-18 08:18:11 UTC	2017-05-18 00:00:00 UTC	6	29
5	2017-03-17 15:56:47 UTC	2017-04-07 13:14:56 UTC	2017-05-18 00:00:00 UTC	20	40
6	2017-03-20 11:01:17 UTC	2017-03-30 14:04:04 UTC	2017-05-18 00:00:00 UTC	10	48
7	2017-03-21 13:38:25 UTC	2017-04-18 13:52:43 UTC	2017-05-18 00:00:00 UTC	28	29
8	2018-08-20 15:56:23 UTC	2018-08-29 22:52:40 UTC	2018-10-04 00:00:00 UTC	9	35
9	2018-08-12 18:14:29 UTC	2018-08-23 02:08:44 UTC	2018-10-04 00:00:00 UTC	10	41
10	2018-08-16 07:55:32 UTC	2018-08-23 00:09:45 UTC	2018-10-04 00:00:00 UTC	6	41

Charts:

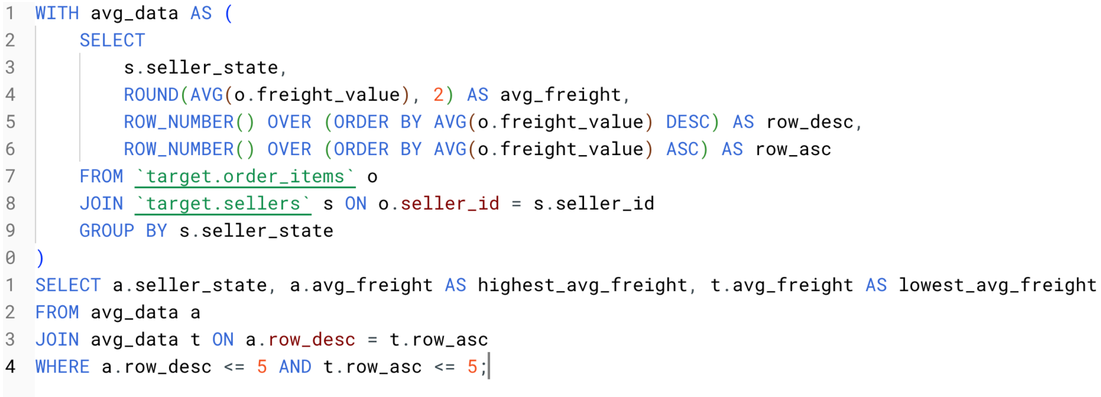


Insights: *Here we can see that actual delivery time is lesser than the estimated delivery time.*

Recommendations: *We can show the estimated time in the app or website nearly equal to the actual because many of the customer shops whether the delivery time is less or not if they see a less estimated delivery they won't switch to another Platform.*

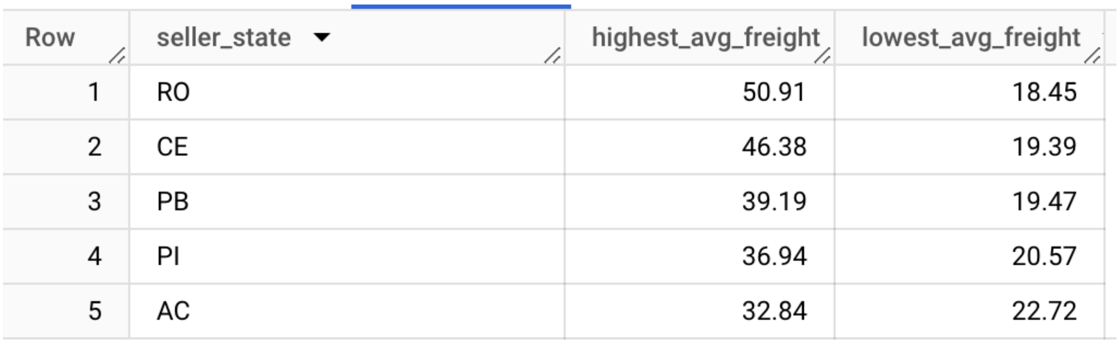
B. Find out the top 5 states with the highest & lowest average freight value.

Query:



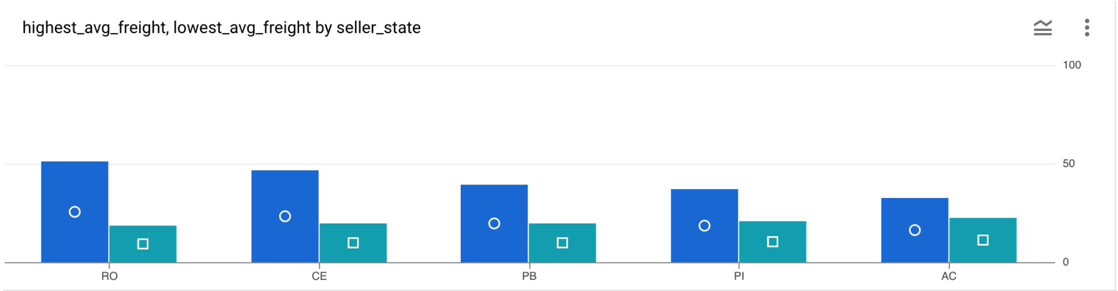
```
WITH avg_data AS (
    SELECT
        s.seller_state,
        ROUND(AVG(o.freight_value), 2) AS avg_freight,
        ROW_NUMBER() OVER (ORDER BY AVG(o.freight_value) DESC) AS row_desc,
        ROW_NUMBER() OVER (ORDER BY AVG(o.freight_value) ASC) AS row_asc
    FROM `target.order_items` o
    JOIN `target.sellers` s ON o.seller_id = s.seller_id
    GROUP BY s.seller_state
)
SELECT a.seller_state, a.avg_freight AS highest_avg_freight, t.avg_freight AS lowest_avg_freight
FROM avg_data a
JOIN avg_data t ON a.row_desc = t.row_asc
WHERE a.row_desc <= 5 AND t.row_asc <= 5;
```

Results:



Row	seller_state	highest_avg_freight	lowest_avg_freight
1	RO	50.91	18.45
2	CE	46.38	19.39
3	PB	39.19	19.47
4	PI	36.94	20.57
5	AC	32.84	22.72

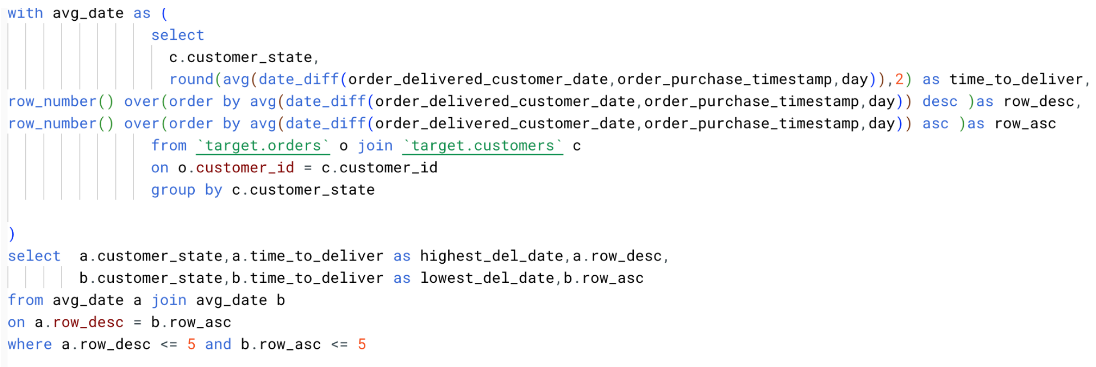
Charts:



Insights: ***Given here the top 5 states having highest_avg_freight and top 5 states having lowest_avg_freight.***

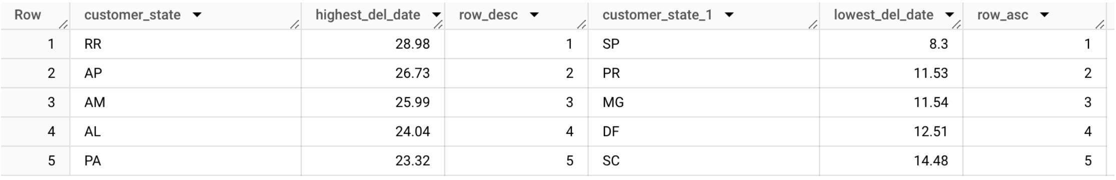
C. Find out the top 5 states with the highest & lowest average delivery time.

Query:



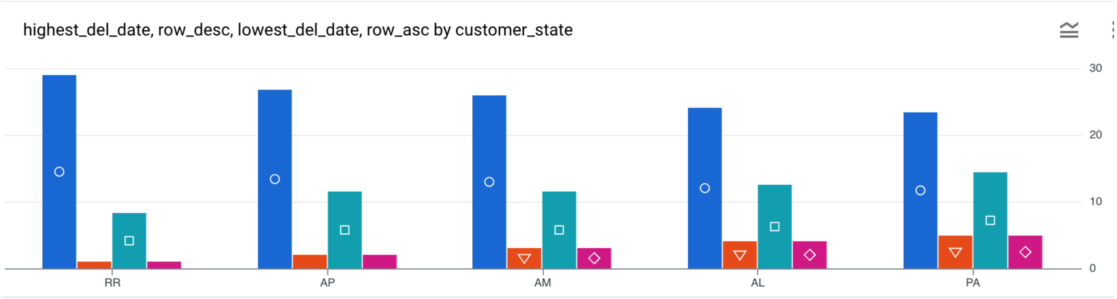
```
with avg_date as (
                select
                  c.customer_state,
                  round(avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)),2) as time_to_deliver,
row_number() over(order by avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)) desc )as row_desc,
row_number() over(order by avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)) asc )as row_asc
                from `target.orders` o join `target.customers` c
                on o.customer_id = c.customer_id
                group by c.customer_state

)
select  a.customer_state,a.time_to_deliver as highest_del_date,a.row_desc,
        b.customer_state,b.time_to_deliver as lowest_del_date,b.row_asc
from avg_date a join avg_date b
on a.row_desc = b.row_asc
where a.row_desc <= 5 and b.row_asc <= 5
```

Results:



Row	customer_state	highest_del_date	row_desc	customer_state_1	lowest_del_date	row_asc
1	RR	28.98	1	SP	8.3	1
2	AP	26.73	2	PR	11.53	2
3	AM	25.99	3	MG	11.54	3
4	AL	24.04	4	DF	12.51	4
5	PA	23.32	5	SC	14.48	5

Charts:

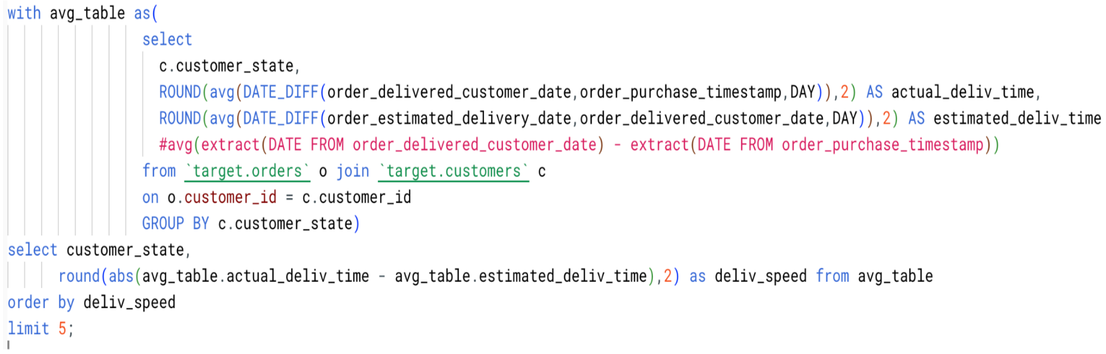


Insights: ***Here the results are showing the highest delivery time and the lowest delivery time for the customer state.***

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

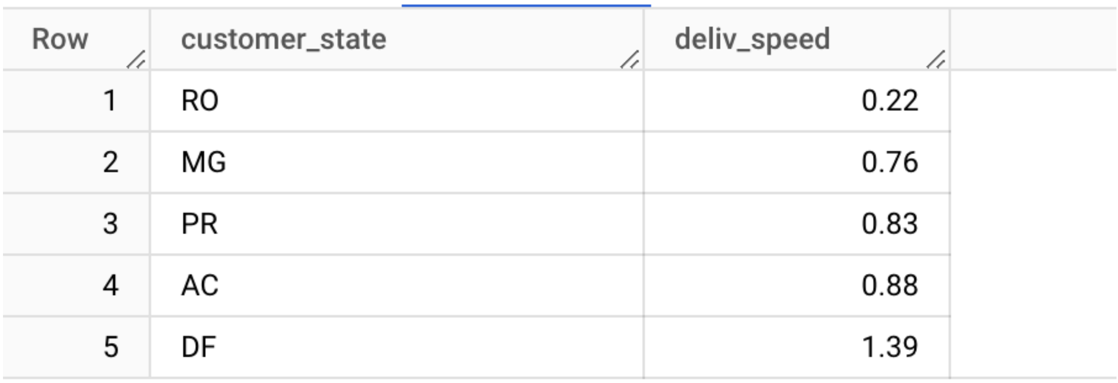
You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

Query:



```
with avg_table as(
                select
                  c.customer_state,
                  ROUND(avg(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY)),2) AS actual_deliv_time,
                  ROUND(avg(DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY)),2) AS estimated_deliv_time
                  #avg(extract(DATE FROM order_delivered_customer_date) - extract(DATE FROM order_purchase_timestamp))
                from `target.orders` o join `target.customers` c
                on o.customer_id = c.customer_id
                GROUP BY c.customer_state)
select customer_state,
      round(abs(avg_table.actual_deliv_time - avg_table.estimated_deliv_time),2) as deliv_speed from avg_table
order by deliv_speed
limit 5;
```

Results:



Row	customer_state	deliv_speed
1	RO	0.22
2	MG	0.76
3	PR	0.83
4	AC	0.88
5	DF	1.39

Insights: ***These are the top 5 states where the delivery time is very less.***

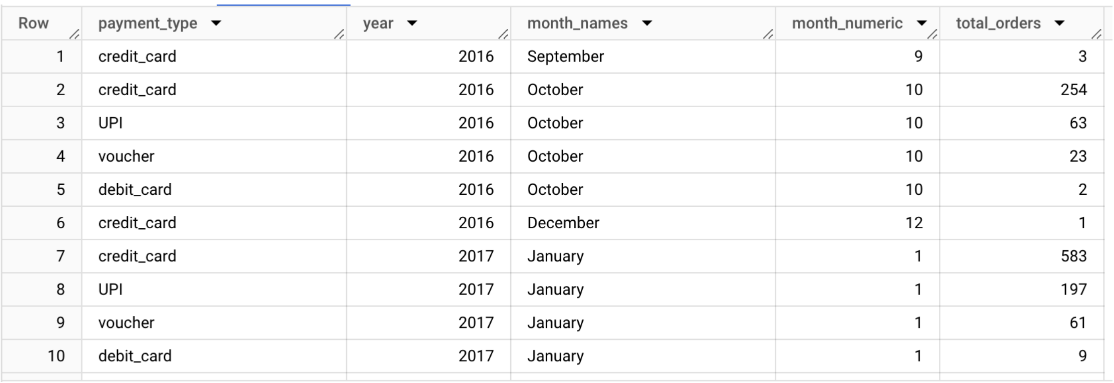
VI. Analysis based on the payments:

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

Query:

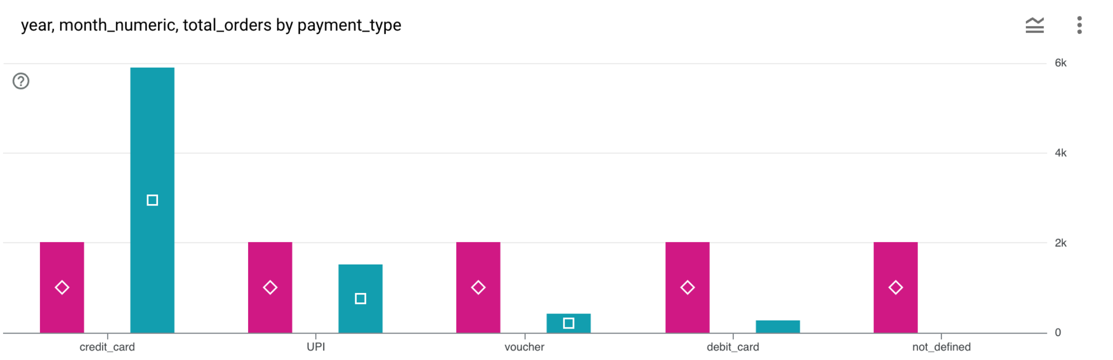
```
select p.payment_type,
  extract(year from order_purchase_timestamp) as year,
  format_date('%B',order_purchase_timestamp) as month_names,
  extract(month from order_purchase_timestamp) as month_numeric,
  count(*) as total_orders
from `target.orders` o inner join `target.payments` p
on o.order_id = p.order_id
group by p.payment_type,year,month_names,month_numeric
order by year,month_numeric
```

Results:



Row	payment_type	year	month_names	month_numeric	total_orders
1	credit_card	2016	September	9	3
2	credit_card	2016	October	10	254
3	UPI	2016	October	10	63
4	voucher	2016	October	10	23
5	debit_card	2016	October	10	2
6	credit_card	2016	December	12	1
7	credit_card	2017	January	1	583
8	UPI	2017	January	1	197
9	voucher	2017	January	1	61
10	debit_card	2017	January	1	9

Charts:

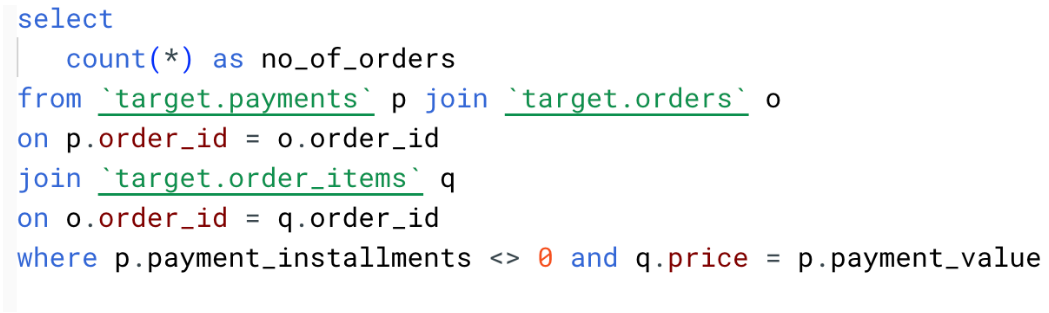


Insights: *Here we can see the payment_types that most of the customers have used to place their orders.*

Recommendations: ***Many of the customers have ordered using credit-card, we can rope in different credit card brands with offers for the customer and make a smooth experience for them.***

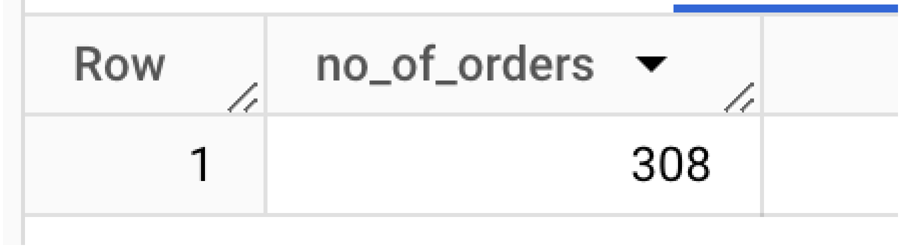
B. Find the no. of orders placed on the basis of the payment instalments that have been paid.

Query:



```
select
    count(*) as no_of_orders
from `target.payments` p join `target.orders` o
on p.order_id = o.order_id
join `target.order_items` q
on o.order_id = q.order_id
where p.payment_installments <> 0 and q.price = p.payment_value
```

Results:



Row	no_of_orders
1	308

Insights: *Total no. of orders that have been used to pay in instalments and that orders where instalments have been paid fully is 308.*