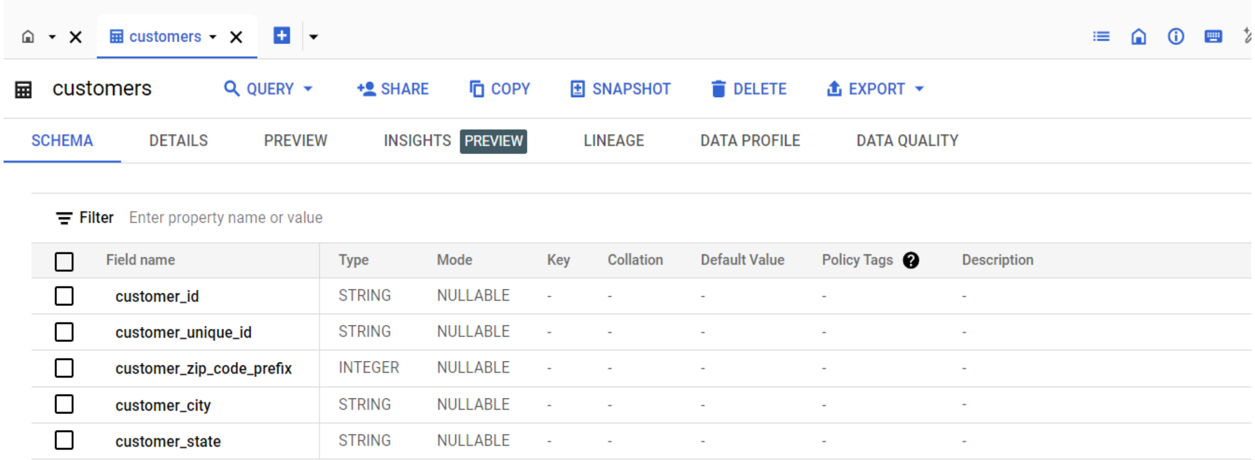
I. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset.

A. Data type of all columns in the "customers" table:



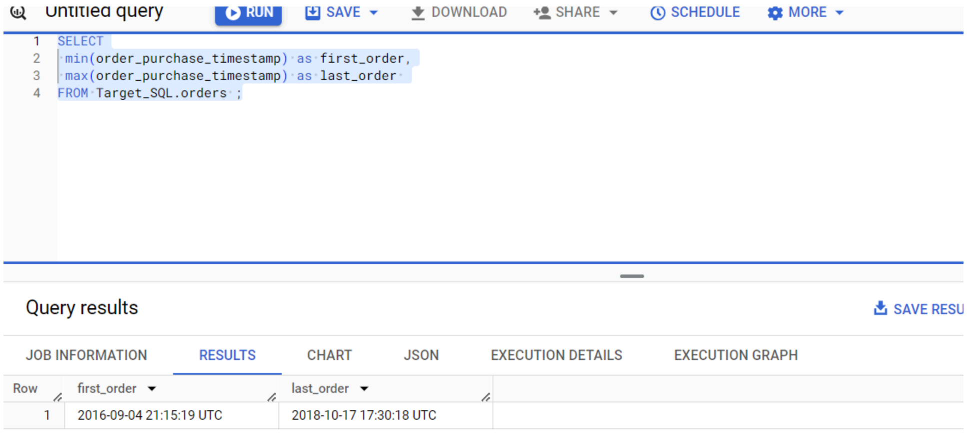
□	Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Description
□	customer_id	STRING	NULLABLE	-	-	-	-	-
□	customer_unique_id	STRING	NULLABLE	-	-	-	-	-
□	customer_zip_code_prefix	INTEGER	NULLABLE	-	-	-	-	-
□	customer_city	STRING	NULLABLE	-	-	-	-	-
□	customer_state	STRING	NULLABLE	-	-	-	-	-

Insight: **I have created Target_SQL dataset where customers is a table and in this screenshot I am showing all the column names of customers table along with its Data Types.**

—--

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

```
SELECT
 min(order_purchase_timestamp) as first_order,
 max(order_purchase_timestamp) as last_order
FROM Target_SQL.orders ;
```



```
SELECT
 min(order_purchase_timestamp) as first_order,
 max(order_purchase_timestamp) as last_order
FROM Target_SQL.orders ;
```

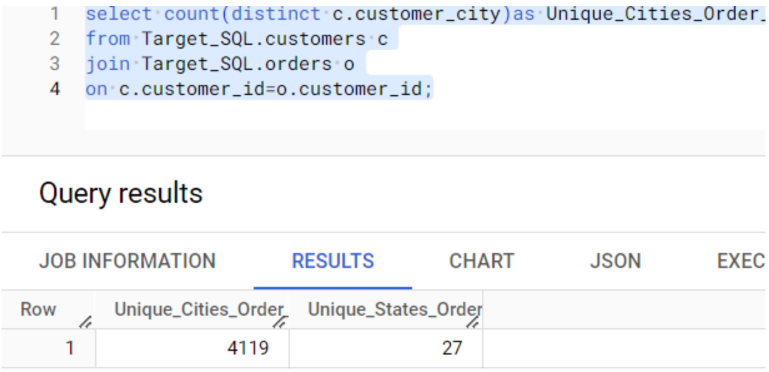
Row	first_order	last_order
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

Insight:Here we want to show the time range between first and last order being purchased by customers hence min(order_purchase_timestamp) is used to give the first order purchase date and max(order_purchase_timestamp) is used to get the last date of order purchased.

—--

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

```
select count(distinct c.customer_city)as
Unique_Cities_Order_Placed,count(distinct c.customer_state) as
Unique_States_Order_Placed
from Target_SQL.customers c
join Target_SQL.orders o
on c.customer_id=o.customer_id;
```



```
select count(distinct c.customer_city)as Unique_Cities_Order_
from Target_SQL.customers c
join Target_SQL.orders o
on c.customer_id=o.customer_id;
```

Query results

Row	Unique_Cities_Order_	Unique_States_Order
1	4119	27

Insights:Inorder to find the unique number of cities and states where orders were placed by the customers count(distinct columnname) function is used. Since the time range was not given, I tried to find the unique cities and states from the entire table.

Inorder to find unique cities and states between specified dates then where clause would have been added.

Query:

```
select count(distinct c.customer_city)as Unique_Cities_Order_Placed,
count(distinct c.customer_state) as Unique_States_Order_Placed
from Target_SQL.customers c
left join Target_SQL.orders o
on c.customer_id=o.customer_id
```

```
where order_purchase_timestamp between '2016-01-01' and
'2016-12-31';
```

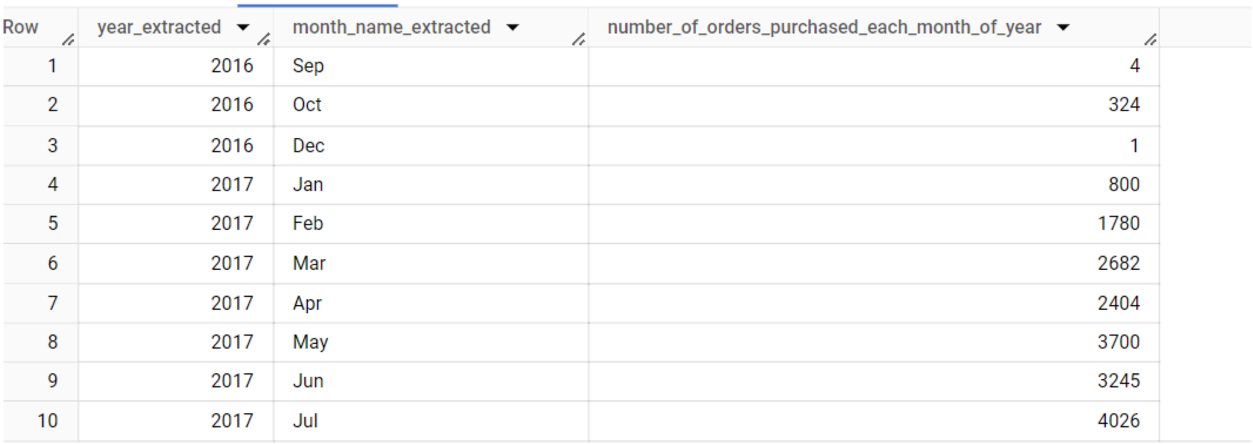
Above query is to find unique cities and states of customers who ordered between 1st Jan 2016 to 31st DEC 2016.

–---

II. In-depth Exploration:

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

```
select
year_extracted,month_name_extracted,number_of_orders_purchased_each_month
_of_year  from (
select
Extract (year from order_purchase_timestamp) as year_extracted,
Extract (month from order_purchase_timestamp) as month_extracted,
FORMAT_DATE('%b',order_purchase_timestamp) as month_name_extracted,
count(order_id)  as number_of_orders_purchased_each_month_of_year from
Target_SQL.orders
group by month_name_extracted,month_extracted,year_extracted
order by year_extracted, month_extracted)as t
limit 10;
```

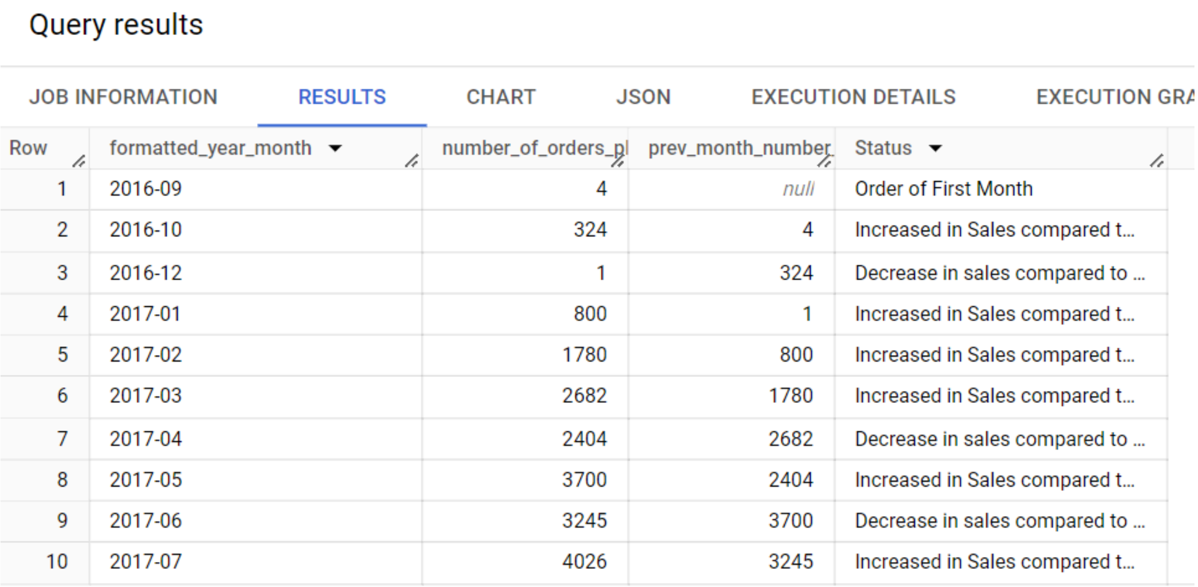


Row	year_extracted	month_name_extracted	number_of_orders_purchased_each_month_of_year
1	2016	Sep	4
2	2016	Oct	324
3	2016	Dec	1
4	2017	Jan	800
5	2017	Feb	1780
6	2017	Mar	2682
7	2017	Apr	2404
8	2017	May	3700
9	2017	Jun	3245
10	2017	Jul	4026

Insights:Here in order to find out if no. of orders placed has increased gradually in each month, over the past years-> count(order_id) function is used which will help us to get the number of orders placed in each month year wise. Extract (month from o.order_purchase_timestamp) is used to extract number of each month,FORMAT_DATE('%b',o.order_purchase_timestamp) is used to extract starting First 3 letters of month from order_purchase_timestamp column,Extract (year from o.order_purchase_timestamp) will extract years.I have grouped month_name_extracted,month_extracted,year_extracted in order to get grouped data and at last sorted using year,month number in ascending order.

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

```
select *,case when number_of_orders_placed_each_month_of_year >=
prev_month_number_orders_placed then
'Increased in Sales compared to previous month'
when  number_of_orders_placed_each_month_of_year <
prev_month_number_orders_placed then  'Decrease in sales compared to previous
month'
else 'Order of First Month' end as Status from (
select * ,lag(number_of_orders_placed_each_month_of_year) over(order by
formatted_year_month ) as prev_month_number_orders_placed from (
select FORMAT_TIMESTAMP('%Y-%m', order_purchase_timestamp)as
formatted_year_month,count(order_id)  as
number_of_orders_placed_each_month_of_year from Target_SQL.orders
group by  formatted_year_month
 ) as t) as y order by formatted_year_month asc limit 10;
```



Row	formatted_year_month	number_of_orders_pl	prev_month_number	Status
1	2016-09	4	null	Order of First Month
2	2016-10	324	4	Increased in Sales compared t...
3	2016-12	1	324	Decrease in sales compared to ...
4	2017-01	800	1	Increased in Sales compared t...
5	2017-02	1780	800	Increased in Sales compared t...
6	2017-03	2682	1780	Increased in Sales compared t...
7	2017-04	2404	2682	Decrease in sales compared to ...
8	2017-05	3700	2404	Increased in Sales compared t...
9	2017-06	3245	3700	Decrease in sales compared to ...
10	2017-07	4026	3245	Increased in Sales compared t...

Insight:Here in order to find out if the no. of orders placed are at peak during certain months. -> count(order_id) function is used which will help us to get the number of orders placed each month year wise.

Status column will help us to find the comparison of current month count with the previous month count which will help us to know in which month order count was high in comparison to previous month.

lag() is used to know the previous month count .

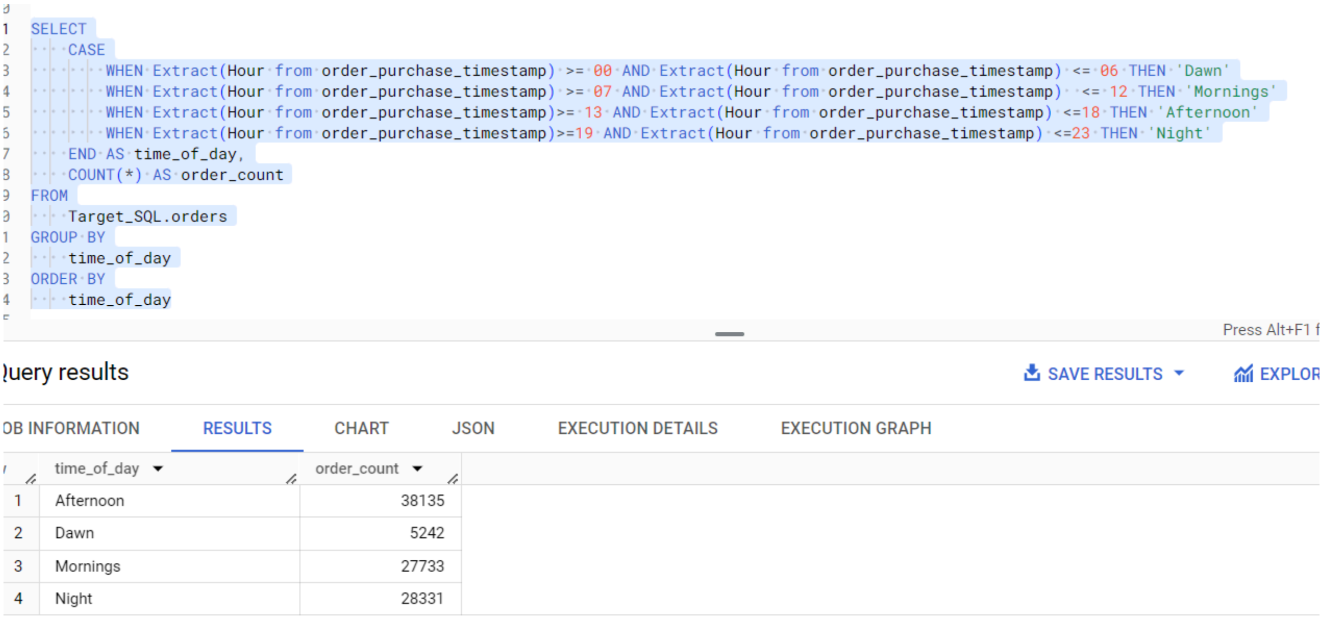
FORMAT_TIMESTAMP('%Y-%m', order_purchase_timestamp) is used to format order_purchase_timestamp into year-month number wise.(i.e. 2016-09).

Lastly I have grouped formatted_year_month column in order to get grouped data and at last sorted using formatted_year_month to get year wise month data in ascending order.

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

```
SELECT
    CASE
        WHEN Extract(Hour from order_purchase_timestamp) >= 00 AND Extract(Hour
from order_purchase_timestamp) <= 06 THEN 'Dawn'
        WHEN Extract(Hour from order_purchase_timestamp) >= 07 AND Extract(Hour
from order_purchase_timestamp)  <= 12 THEN 'Mornings'
        WHEN Extract(Hour from order_purchase_timestamp)>= 13 AND Extract(Hour
from order_purchase_timestamp) <=18 THEN 'Afternoon'
        WHEN Extract(Hour from order_purchase_timestamp)>=19 AND Extract(Hour
from order_purchase_timestamp) <=23 THEN 'Night'
    END AS time_of_day,
    COUNT(*) AS order_count
FROM
    Target_SQL.orders
GROUP BY
    time_of_day
ORDER BY
    Time_of_day;
```



```
SELECT
    CASE
        WHEN Extract(Hour from order_purchase_timestamp) >= 00 AND Extract(Hour from order_purchase_timestamp) <= 06 THEN 'Dawn'
        WHEN Extract(Hour from order_purchase_timestamp) >= 07 AND Extract(Hour from order_purchase_timestamp)  <= 12 THEN 'Mornings'
        WHEN Extract(Hour from order_purchase_timestamp)>= 13 AND Extract(Hour from order_purchase_timestamp) <=18 THEN 'Afternoon'
        WHEN Extract(Hour from order_purchase_timestamp)>=19 AND Extract(Hour from order_purchase_timestamp) <=23 THEN 'Night'
    END AS time_of_day,
    COUNT(*) AS order_count
FROM
    Target_SQL.orders
GROUP BY
    time_of_day
ORDER BY
    time_of_day
```

	time_of_day	order_count
1	Afternoon	38135
2	Dawn	5242
3	Mornings	27733
4	Night	28331

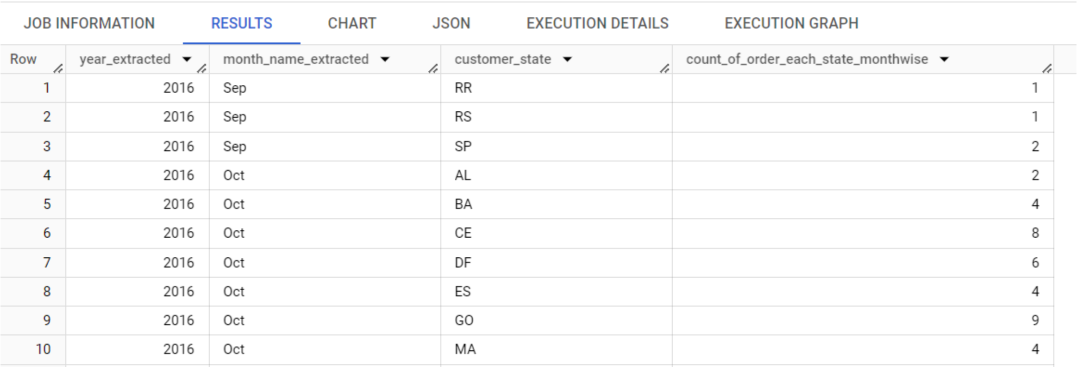
Insight: Since,order_purchase_timestamp is of timestamp Datatype, hence by using Extract(Hour from order_purchase_timestamp) function I have extracted hours from each column of order_purchase_timestamp and post that I have Categorized the hours of a day into the given time brackets intervals and used count(*) function to find out during which intervals the Brazilian customers usually order the most.

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

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

```
select
year_extracted,month_name_extracted,customer_state,count_of_order_each_st
ate_monthwise  from (


select
Extract (year from o.order_purchase_timestamp) as year_extracted,
Extract (month from o.order_purchase_timestamp) as month_extracted,
FORMAT_DATE('%b',o.order_purchase_timestamp) as month_name_extracted,
c.customer_state customer_state ,
count(order_id) count_of_order_each_state_monthwise
from Target_SQL.customers c
join Target_SQL.orders o
on c.customer_id=o.customer_id
group by
c.customer_state,month_name_extracted,month_extracted,year_extracted) as
t
order by year_extracted, month_extracted,customer_state ;
```

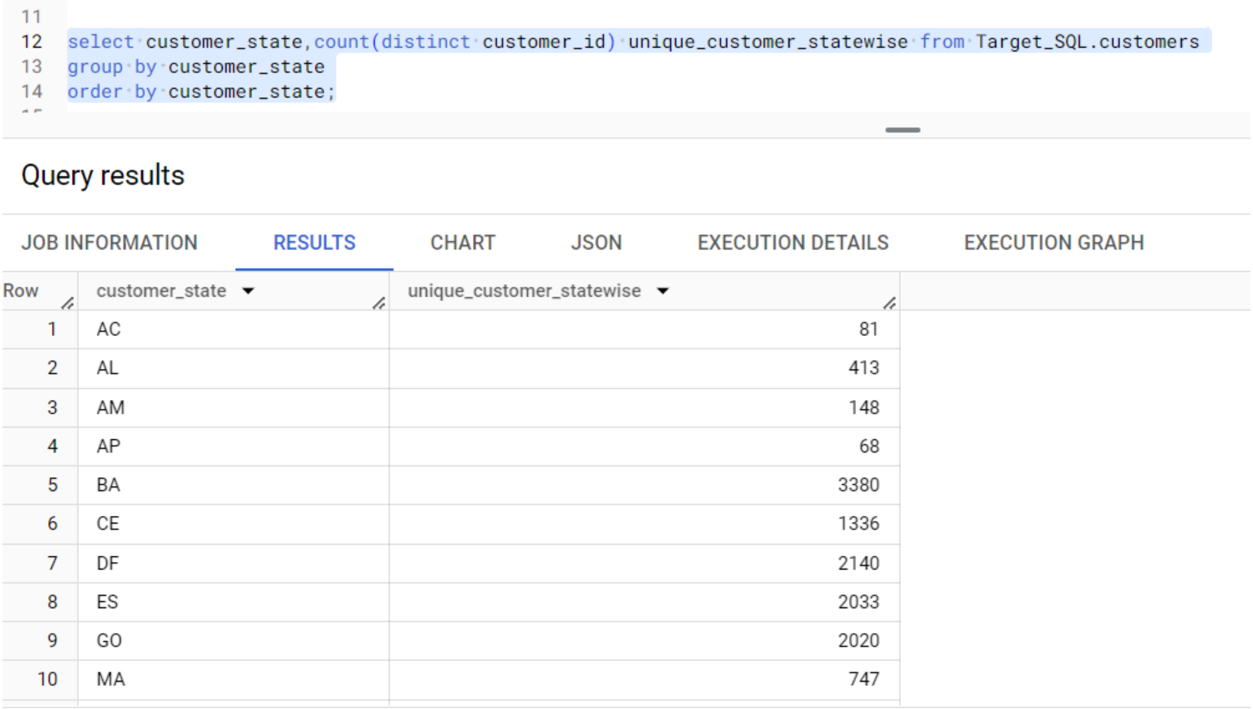


Row	year_extracted	month_name_extracted	customer_state	count_of_order_each_state_monthwise
1	2016	Sep	RR	1
2	2016	Sep	RS	1
3	2016	Sep	SP	2
4	2016	Oct	AL	2
5	2016	Oct	BA	4
6	2016	Oct	CE	8
7	2016	Oct	DF	6
8	2016	Oct	ES	4
9	2016	Oct	GO	9
10	2016	Oct	MA	4

Insights: Here in order to find the count of order placed for each state above query is used. Extract (month from o.order_purchase_timestamp) is used to extract number of each month,FORMAT_DATE('%b',o.order_purchase_timestamp) is used to extract starting First 3 letters of month from order_purchase_timestamp column,Extract (year from o.order_purchase_timestamp) will extract years, count(order_id) is used to display number of counts placed for each state in each month.I have grouped customer_state,month and year in order to get grouped data and at last sorted using year,month and state.

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

```
select customer_state,count(distinct customer_id) unique_customer_statewise
from Target_SQL.customers
group by customer_state
order by customer_state;
```



```
select customer_state,count(distinct customer_id) unique_customer_statewise from Target_SQL.customers
group by customer_state
order by customer_state;
```

Query results

Row	customer_state	unique_customer_statewise
1	AC	81
2	AL	413
3	AM	148
4	AP	68
5	BA	3380
6	CE	1336
7	DF	2140
8	ES	2033
9	GO	2020
10	MA	747

Insights: Inorder to find number of unique customers in each state, `count(distinct customer_id)`
Is used to give the unique count of customers grouped by customer_state to give count as per customer's state and post that sorting the data by customer_state in ascending order.

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

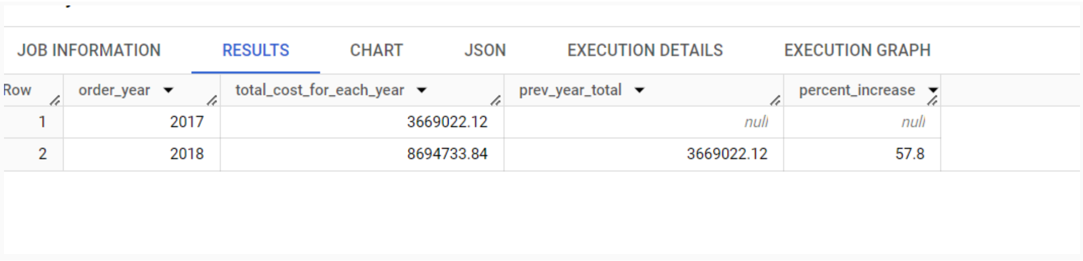
A. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only):

```
WITH YearlyCost AS (
    SELECT
        Extract (Year from o.order_purchase_timestamp) AS order_year,

        SUM(p.payment_value) AS total_cost_for_each_year
   from Target_SQL.payments p
join Target_SQL.orders o
on p.order_id=o.order_id
    WHERE
        Extract (Year from o.order_purchase_timestamp) IN (2017, 2018)
        AND Extract (month from o.order_purchase_timestamp) BETWEEN 1 AND 8
    GROUP BY
        Extract (Year from o.order_purchase_timestamp)


)
SELECT

    order_year,
round(total_cost_for_each_year,2)total_cost_for_each_year,round(lag(total_cost_
for_each_year) OVER (ORDER BY order_year),2) prev_year_total,
    round((total_cost_for_each_year-LAG(total_cost_for_each_year) OVER (ORDER
BY order_year) ) / total_cost_for_each_year * 100,2) AS percent_increase
FROM
    YearlyCost
    order by order_year
```



Row	order_year	total_cost_for_each_year	prev_year_total	percent_increase
1	2017	3669022.12	null	null
2	2018	8694733.84	3669022.12	57.8

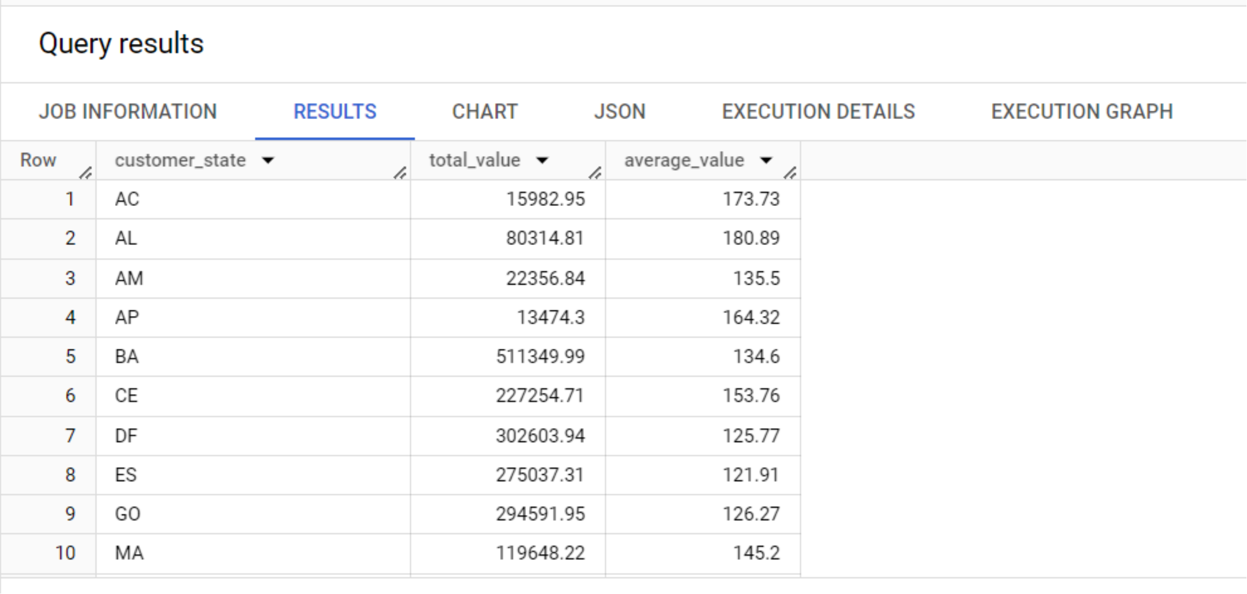
Insights: Here I used Common table Expression as a virtual table inorder to fetch the year and sum of cost of orders for each year 2017 and 2018 between Jan to Aug only.

Post that I used lag function just to fetch sum of cost of orders of 2017 in the same row as sum of cost of orders of 2018 inorder to get the % increase in the cost of orders from year 2017 to 2018 data is displayed in column:percent_increase.Rounded each column upto 2 decimal places.

```
-----------------------------------------------------------------------------
```

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

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

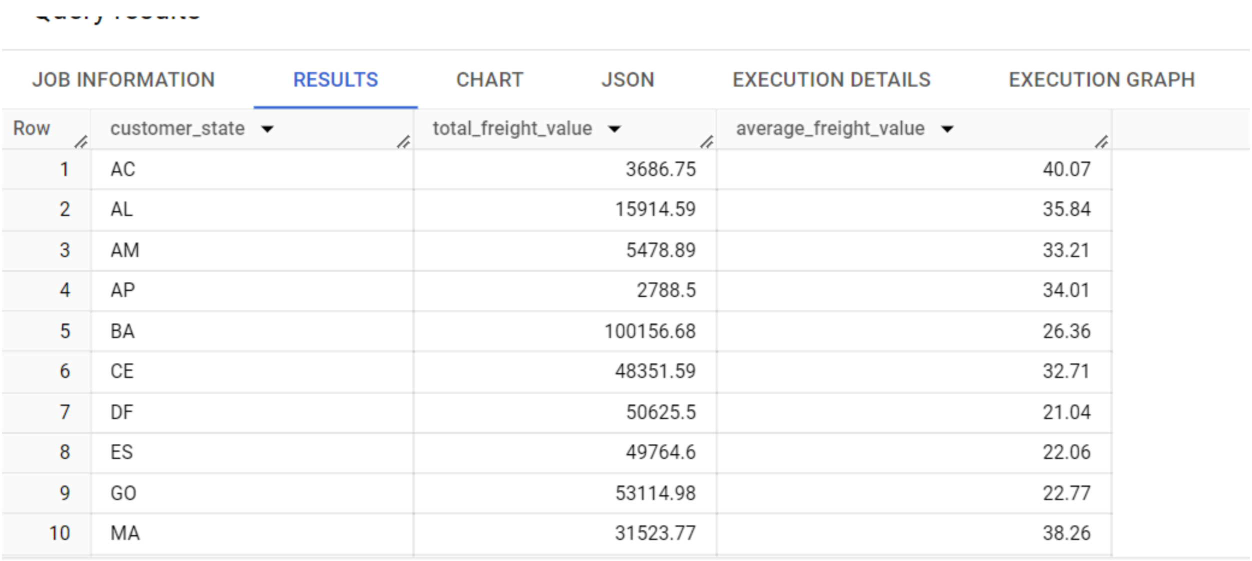


Row	customer_state	total_value	average_value
1	AC	15982.95	173.73
2	AL	80314.81	180.89
3	AM	22356.84	135.5
4	AP	13474.3	164.32
5	BA	511349.99	134.6
6	CE	227254.71	153.76
7	DF	302603.94	125.77
8	ES	275037.31	121.91
9	GO	294591.95	126.27
10	MA	119648.22	145.2

Insight: Above query is used to find the total price of orders and average price of orders from each state of customer purchased.sum() is used to sum all the price value by grouping state of customer and avg() is used to provide average value of price by grouping the state of customer. Round(column name,2) is used to round the output of sum and average value of price by 2.

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

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



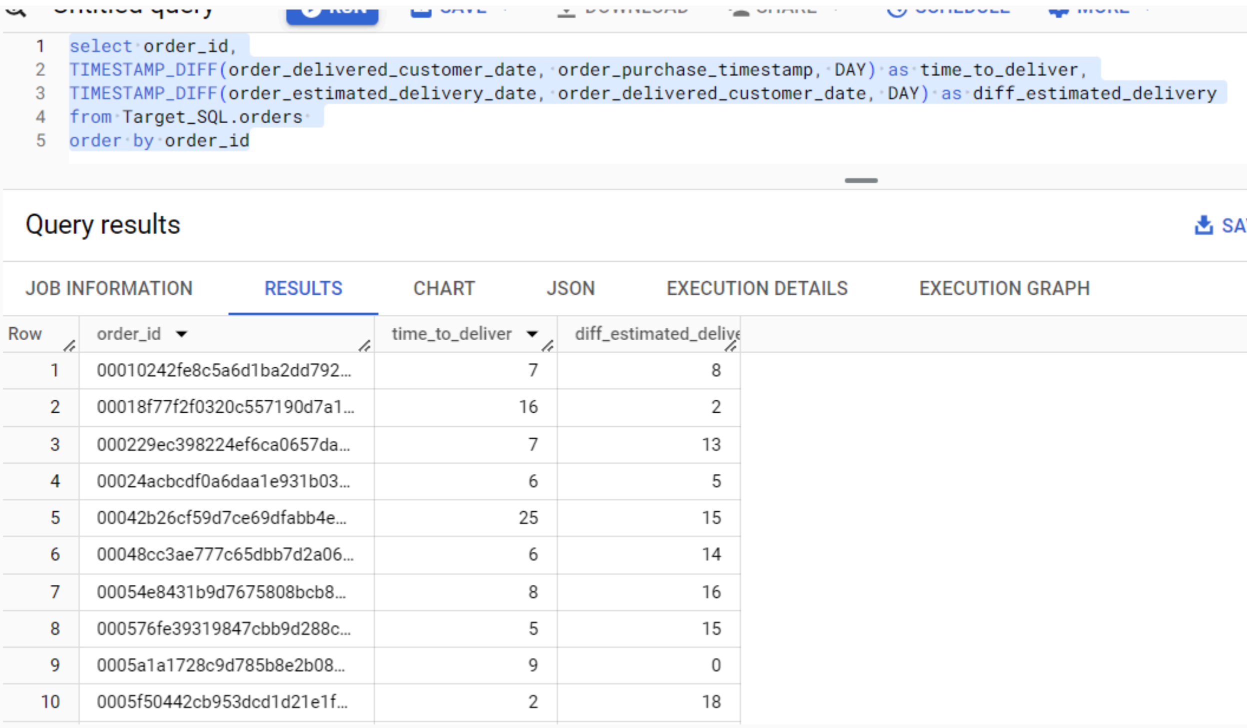
Row	customer_state	total_freight_value	average_freight_value
1	AC	3686.75	40.07
2	AL	15914.59	35.84
3	AM	5478.89	33.21
4	AP	2788.5	34.01
5	BA	100156.68	26.36
6	CE	48351.59	32.71
7	DF	50625.5	21.04
8	ES	49764.6	22.06
9	GO	53114.98	22.77
10	MA	31523.77	38.26

Insight: Above query is used to find the Total & Average value of order freight for each state.sum() is used to sum all the freight value by grouping state of customer and avg() is used to provide average value of freight value by grouping the state of customer. Round(column name,2) is used to round the output of sum and average value of freight value by 2.

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:

```
select order_id,
TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp,
DAY) as time_to_deliver,
TIMESTAMP_DIFF(order_estimated_delivery_date,
order_delivered_customer_date, DAY) as diff_estimated_delivery
from Target_SQL.orders
order by order_id;
```



```
select order_id,
TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) as time_to_deliver,
TIMESTAMP_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY) as diff_estimated_delivery
from Target_SQL.orders
order by order_id
```

Row	order_id	time_to_deliver	diff_estimated_delive
1	00010242fe8c5a6d1ba2dd792...	7	8
2	00018f77f2f0320c557190d7a1...	16	2
3	000229ec398224ef6ca0657da...	7	13
4	00024acbcdf0a6daa1e931b03...	6	5
5	00042b26cf59d7ce69dfabb4e...	25	15
6	00048cc3ae777c65dbb7d2a06...	6	14
7	00054e8431b9d7675808bcb8...	8	16
8	000576fe39319847cbb9d288c...	5	15
9	0005a1a1728c9d785b8e2b08...	9	0
10	0005f50442cb953dcd1d21e1f...	2	18

Insight:TIMESTAMP_DIFF() is used to find the difference in number of days.
1)TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) helps us to get us the number of days within order get deliver.

2)TIMESTAMP_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY) helps us to get us the number of in how many order got delivered as per estimated day. IF result is positive means order got delivered before estimated delivery date and if the result is in negative means order got delayed to deliver with that number of days.

TIMESTAMP_DIFF() is used as datatype of above used columns are TimeStamp,order by will sort data by order_id in ascending order.

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

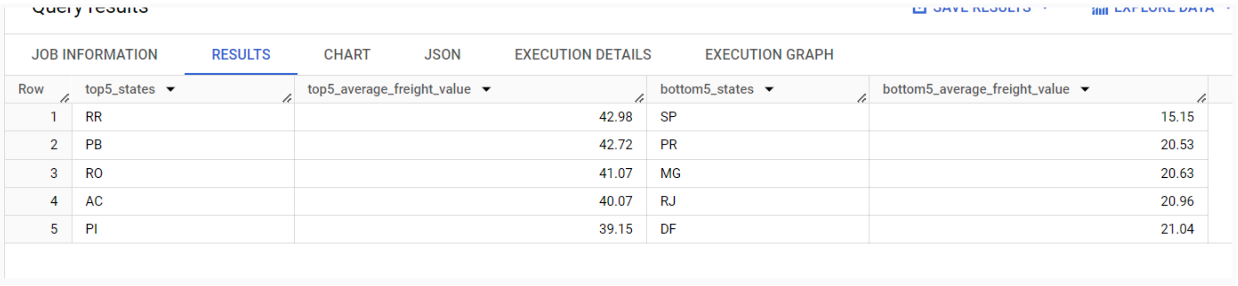
```
with cte_top as(
select top_5_state,top5_average_freight_value,row_num_top5 from (
select top_5_state,top5_average_freight_value,dense_rank() over(order by
top5_average_freight_value desc) as row_num_top5 from (
select c.customer_state top_5_state,avg(oi.freight_value) as
top5_average_freight_value from Target_SQL.order_items oi
join Target_SQL.orders o
on oi.order_id=o.order_id
join  Target_SQL.customers c
on c.customer_id=o.customer_id
group by  c.customer_state) as t) as z
where row_num_top5 <=5


) ,

 cte_last as(
select last_5_state,bottom5_average_freight_value,row_num_last5 from (
select last_5_state,bottom5_average_freight_value,dense_rank() over(order by
bottom5_average_freight_value ) as row_num_last5 from (
select c.customer_state last_5_state,avg(oi.freight_value) as
bottom5_average_freight_value from Target_SQL.order_items oi
join Target_SQL.orders o
on oi.order_id=o.order_id
join  Target_SQL.customers c
on c.customer_id=o.customer_id
group by  c.customer_state) as t) as z
where row_num_last5<=5
)

select a.top_5_state top5_states,round(a.top5_average_freight_value,2)
top5_average_freight_value,b.last_5_state
```

```
bottom5_states,round(b.bottom5_average_freight_value,2)
bottom5_average_freight_value
from cte_top a join cte_last b
on a.row_num_top5=b.row_num_last5
order by top5_average_freight_value desc, bottom5_average_freight_value
```



Row	top5_states	top5_average_freight_value	bottom5_states	bottom5_average_freight_value
1	RR	42.98	SP	15.15
2	PB	42.72	PR	20.53
3	RO	41.07	MG	20.63
4	AC	40.07	RJ	20.96
5	PI	39.15	DF	21.04

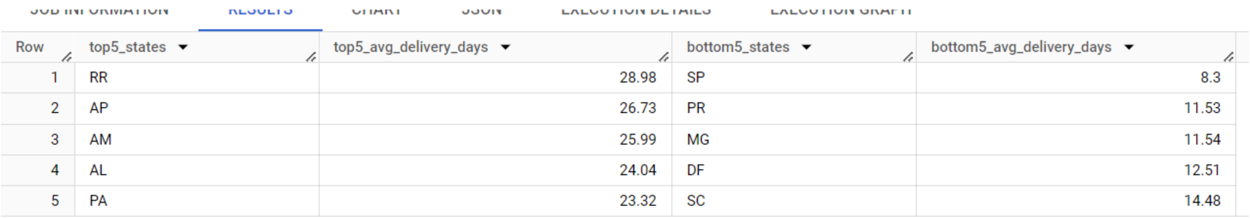
Insight:I have created 2 CTEs to find the average freight value. Cte_top: will top 5 states average freight values and cte_last : will find bottom 5 states average freight values using dense_rank() Lastly select statement is used to extract top5 states name and their corresponding average freight values and also bottom5 states and theirs corresponding average freight values using both the ctes(i.e. cte_top and cte_last) lastly I have sorted the result top5_average_freight_value in descending and bottom5_average_freight_value in ascending order.

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

```
with cte_top as (
select top_5_state,top5_avg_delivery_days,row_num_top5 from (
select top_5_state,top5_avg_delivery_days,DENSE_RANK() over(order by
top5_avg_delivery_days desc) as row_num_top5 from (
select c.customer_state top_5_state,
ROUND(AVG(TIMESTAMP_DIFF(order_delivered_customer_date,
order_purchase_timestamp, DAY)), 2) top5_avg_delivery_days
from  Target_SQL.orders o
join  Target_SQL.customers c
on c.customer_id=o.customer_id
group by  c.customer_state) as t) as z
where row_num_top5 <=5 )  ,

cte_last as(
select last_5_state,bottom5_avg_delivery_days,row_num_last5 from (
select last_5_state,bottom5_avg_delivery_days,dense_rank() over(order by
bottom5_avg_delivery_days ) as row_num_last5 from (
select c.customer_state
last_5_state,ROUND(AVG(TIMESTAMP_DIFF(order_delivered_customer_date,
order_purchase_timestamp, DAY)), 2) AS bottom5_avg_delivery_days
from  Target_SQL.orders o
join  Target_SQL.customers c
on c.customer_id=o.customer_id
group by  c.customer_state) as t) as z
where row_num_last5<=5)

select a.top_5_state top5_states,a.top5_avg_delivery_days,b.last_5_state
bottom5_states,b.bottom5_avg_delivery_days
from cte_top a join cte_last b
on a.row_num_top5=b.row_num_last5
order by a.top5_avg_delivery_days desc ,b.bottom5_avg_delivery_days
```



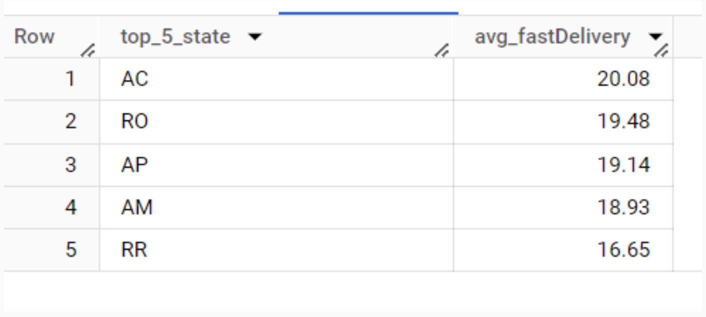
Row	top5_states	top5_avg_delivery_days	bottom5_states	bottom5_avg_delivery_days
1	RR	28.98	SP	8.3
2	AP	26.73	PR	11.53
3	AM	25.99	MG	11.54
4	AL	24.04	DF	12.51
5	PA	23.32	SC	14.48

Insight: I have created 2 CTEs to find the average delivery time.
Cte_top: will top 5 states average delivery time and cte_last : will find bottom 5 states average delivery time using dense_rank()

Lastly select statement is used to extract top5 states name and their corresponding average delivery time and also bottom5 states and theirs corresponding average delivery time using both the ctes(i.e. cte_top and cte_last) lastly I have sorted the result top5_avg_delivery_days in descending and bottom5_avg_delivery_days in ascending order.

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

```
select top_5_state,avg_fastDelivery from (
select top_5_state,dense_rank() over(order by avg_fastDelivery desc) as
dense_rn,avg_fastDelivery from (
select top_5_state,Round((avg_estimated_delivery-avg_actual_delivery),2) as
avg_fastDelivery
from
(
select  c.customer_state top_5_state,
ROUND(AVG(TIMESTAMP_DIFF(order_delivered_customer_date,
order_purchase_timestamp, DAY)),2) avg_actual_delivery,
Round(AVG(TIMESTAMP_DIFF(order_estimated_delivery_date,
order_purchase_timestamp, DAY)),2) avg_estimated_delivery
from Target_SQL.orders o
join  Target_SQL.customers c
on c.customer_id=o.customer_id
Where lower(o.order_status)='delivered'
group by c.customer_state) as t) as a)as z
where dense_rn <=5
order by dense_rn;
```



Row	top_5_state	avg_fastDelivery
1	AC	20.08
2	RO	19.48
3	AP	19.14
4	AM	18.93
5	RR	16.65

Insight:TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) will help us to get actual delivery date from the day of purchase in DAY and
TIMESTAMP_DIFF(order_estimated_delivery_date, order_purchase_timestamp, DAY) will help us to calculate the actual estimated date from the day of purchase in DAY.

Post that I have calculated AVG for above extracted actual delivery date and actual estimated date.

Avg_estimated_delivery-avg_actual_delivery → will provide insights how fast the delivery was made then mentioned actual estimated days.Round() is used to round the result upto 2 decimal.
As in question it was mentioned that Include only the orders that are already delivered so used →Where lower(o.order_status)='delivered'
dense_rank() over(order by avg_fastDelivery desc) helped to find the rank average of fast delivery and finally used dense_rn <=5 to find top 5 states only.

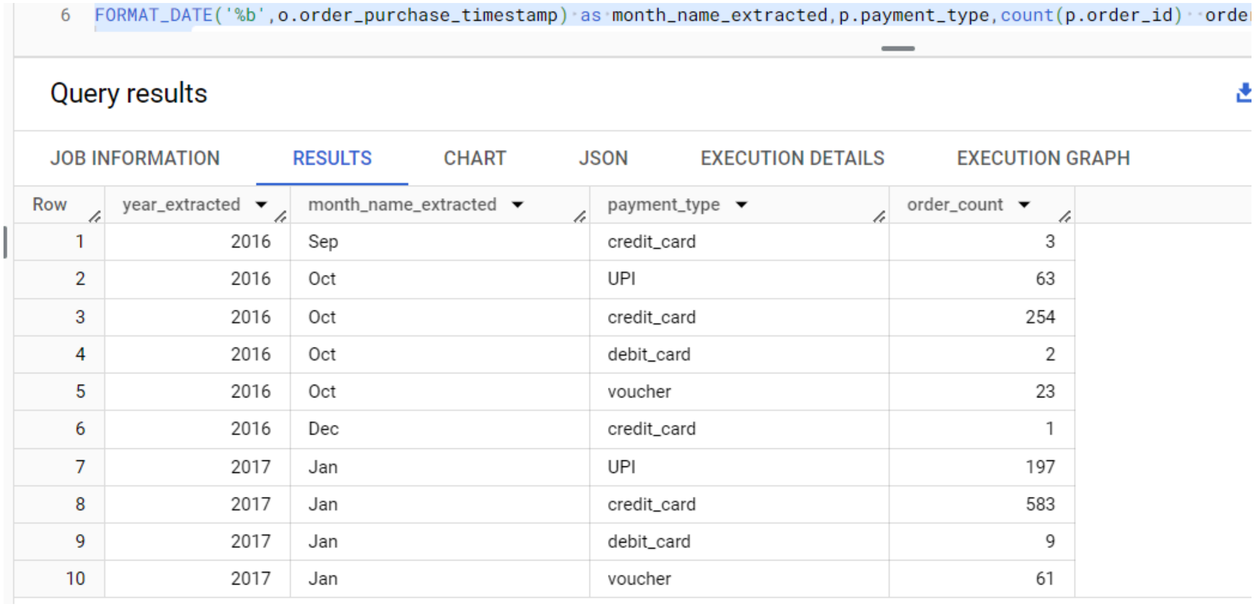
VI. Analysis based on the payments:

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

```
select year_extracted,month_name_extracted,payment_type,order_count  from (
select
Extract (year from o.order_purchase_timestamp) as year_extracted,
Extract (month from o.order_purchase_timestamp) as month_extracted,
FORMAT_DATE('%b',o.order_purchase_timestamp) as
month_name_extracted,p.payment_type,count(p.order_id)  order_count
```

```
   from Target_SQL.payments p
join Target_SQL.orders o
on p.order_id=o.order_id
group by payment_type,month_name_extracted,month_extracted,year_extracted) as t

order by year_extracted, month_extracted,payment_type;
```



```
FORMAT_DATE('%b',o.order_purchase_timestamp) as month_name_extracted,p.payment_type,count(p.order_id)  orde
```

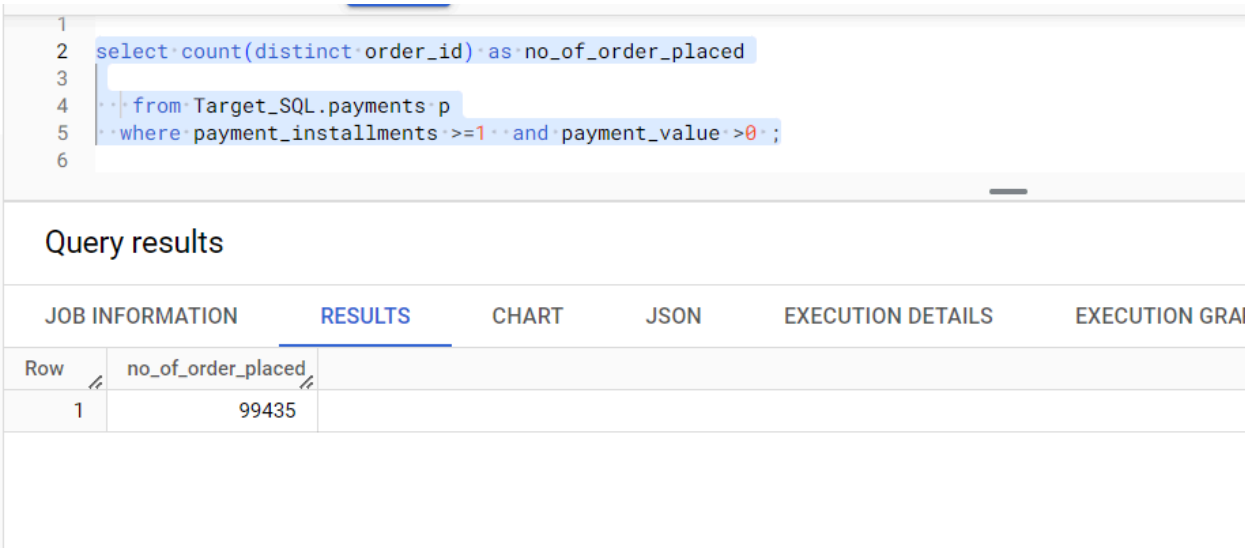
Row	year_extracted	month_name_extracted	payment_type	order_count
1	2016	Sep	credit_card	3
2	2016	Oct	UPI	63
3	2016	Oct	credit_card	254
4	2016	Oct	debit_card	2
5	2016	Oct	voucher	23
6	2016	Dec	credit_card	1
7	2017	Jan	UPI	197
8	2017	Jan	credit_card	583
9	2017	Jan	debit_card	9
10	2017	Jan	voucher	61

Insight:Inorder to find month on month no. of orders placed using different payment types, I have extracted year,month and year-month format using Date and Time function as sorted year,month and payment type in ascending order.

Count(p.order_id) function is used to find different number of order placed using different payment terms. Duplicates and null both are included as in question unique count was not mentioned.

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

```
select count(distinct order_id) as no_of_order_placed
from Target_SQL.payments p
where payment_installments >=1 and payment_value >0 ;
```



```
select count(distinct order_id) as no_of_order_placed

   from Target_SQL.payments p
  where payment_installments >=1  and payment_value >0 ;
```

Row	no_of_order_placed
1	99435

Insights: In this query ; where payment_installments >=1 and payment_value >0 condition is used to find payment installments that have been paid with payment_value not equal to zero.

Count(distinct order_id) gives unique order number