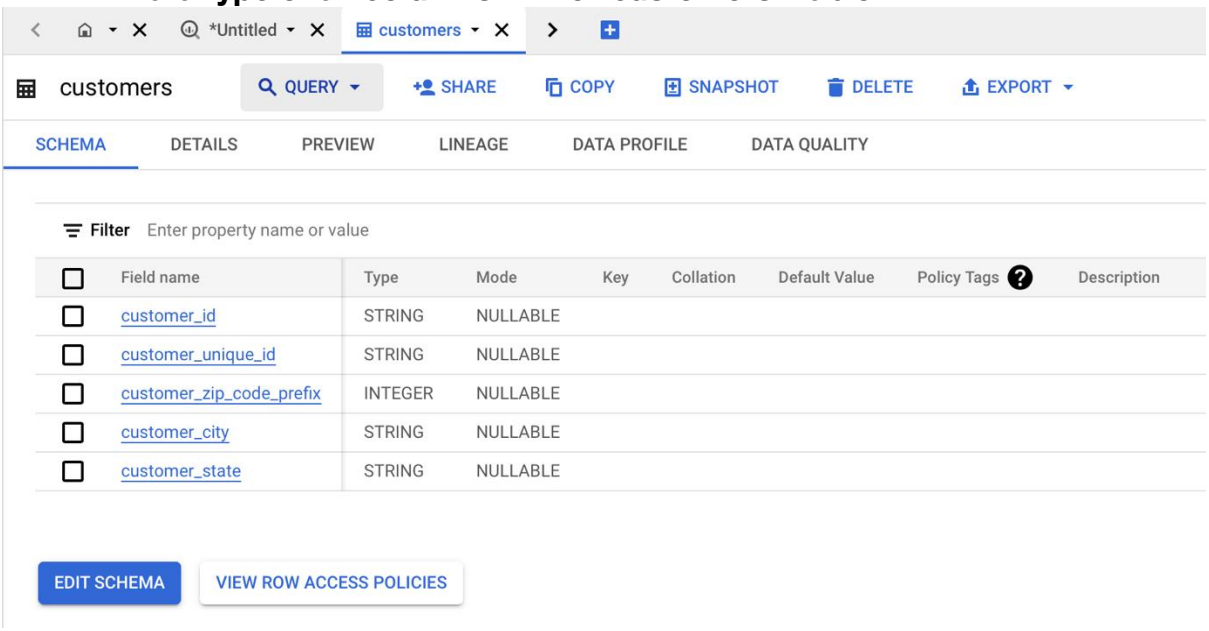
1.Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:

1.Data type of all columns in the "customers" table.

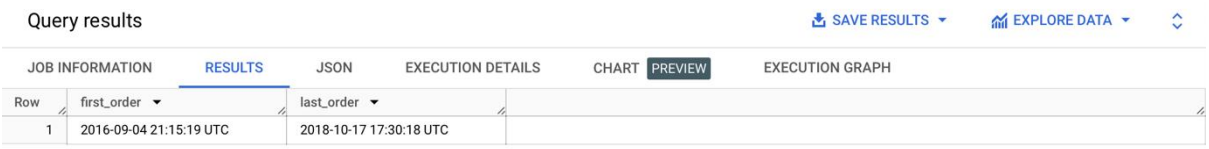


These are the Data types of all columns in the customers table

2.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 `dsml-sql-396708.case_study.orders`
```

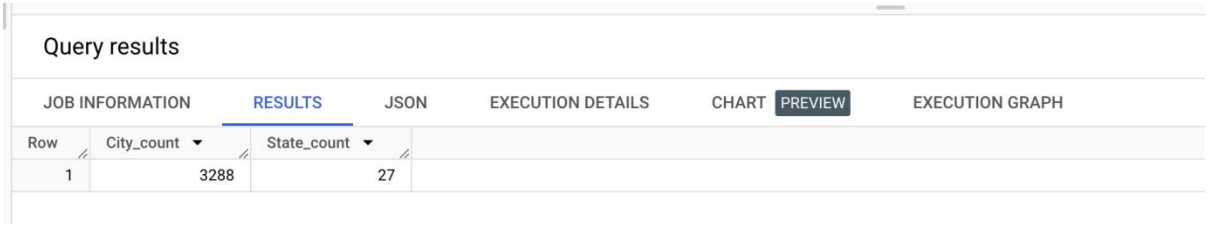
Output:



The first order was placed on 4th sept 2016 and last order was placed on 17th oct 2018

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

```
select count(distinct(customer_city)) as
City_count,count(distinct(customer_state)) as State_count from
`dsml-sql-396708.case_study.customers` c
join `dsml-sql-396708.case_study.orders` o using(customer_id)
where extract(year from order_approved_at) = 2017
```

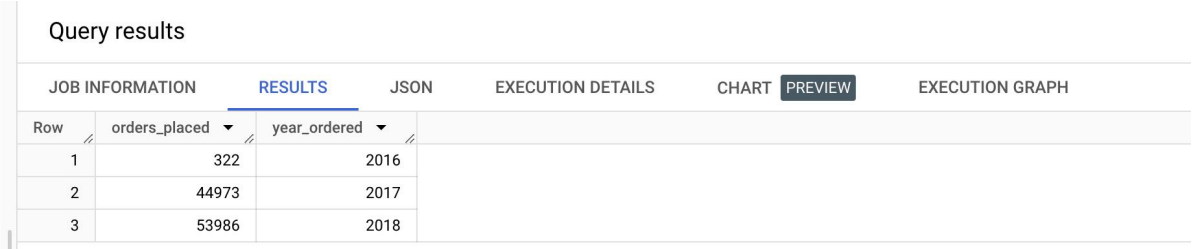


City and state count of oders place in the year 2017

2. In-depth Exploration:

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

```
SELECT count(order_id) as orders_placed,extract(year from
order_approved_at) as year_ordered FROM `dsml-sql-
396708.case_study.orders`
where extract(year from order_approved_at) is not null
group by year_ordered
order by year_ordered
```



There is not much growth in the no of orders places in the years 2017,2018
theer are low orders places in 2016 as the orders were placed from seept 2016

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

```
select orders_placed,
case
when month_ordered = 1 then 'January'
when month_ordered = 2 then 'February'
when month_ordered = 3 then 'March'
when month_ordered = 4 then 'April'
when month_ordered = 5 then 'May'
when month_ordered = 6 then 'June'
when month_ordered = 7 then 'July'
when month_ordered = 8 then 'August'
when month_ordered = 9 then 'September'
when month_ordered = 10 then 'October'
when month_ordered = 11 then 'November'
when month_ordered = 12 then 'December'
end as month_ordered
from (SELECT count(order_id) as orders_placed,extract(month
from order_approved_at) as month_ordered FROM `dsml-sql-
396708.case_study.orders`
where extract(month from order_approved_at) is not null
group by month_ordered order by month_ordered ) t
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART PREVIEW | EXECUTION GRAPH

Row	orders_placed	month_ordered
1	7947	January
2	8471	February
3	9977	March
4	9152	April
5	10759	May
6	9416	June
7	10150	July
8	10968	August
9	4303	September
10	4910	October
11	7395	November
12	5833	December

Highest no of the orders are placed between the months May to August and then there is a drastic drop in the no of orders placed while the least orders were placed in the months September and October.

2. **3.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 sum(orders_placed),
case
when Hour_ordered between 0 and 6 then 'Dawn'
when Hour_ordered between 7 and 12 then 'Mornings'
when Hour_ordered between 13 and 18 then 'Afternoon'
when Hour_ordered between 19 and 23 then 'Night'
end as Time_of_Day
from (select count(order_id) as orders_placed,extract(hour
from order_approved_at) as Hour_ordered from `dsml-sql-
396708.case_study.orders`
where extract(hour from order_approved_at) is not null
group by Hour_ordered
order by Hour_ordered) t
group by Time_of_Day
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART PREVIEW | EXECUTION GRAPH

Row	f0_	Time_of_Day
1	22312	Mornings
2	32667	Afternoon
3	21413	Dawn
4	22889	Night

Maximum no of orders were placed in the Afternoon.

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

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

```
select c.customer_state, count(order_id) as
orders_placed,extract(month from order_approved_at) as
month_ordered from `dsml-sql-396708.case_study.customers` c
join `dsml-sql-396708.case_study.orders` o using(customer_id)
where extract(month from order_approved_at) is not null
group by month_ordered,customer_state order by
customer_state,month_ordered
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART PREVIEW | EXECUTION GRAPH

Row	customer_state	orders_placed	month_ordered
1	AC	8	1
2	AC	6	2
3	AC	4	3
4	AC	9	4
5	AC	10	5
6	AC	7	6
7	AC	9	7
8	AC	7	8
9	AC	5	9
10	AC	6	10
11	AC	5	11
12	AC	5	12

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

```
select count(customer_id) as customer_count,customer_state
from dsml-sql-396708.case_study.customers
group by customer_state
order by customer_count
```

Query results SAVE RESU

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART PREVIEW | EXECUTION GRAPH

Row	customer_count	customer_state
1	46	RR
2	68	AP
3	81	AC
4	148	AM
5	253	RO
6	280	TO
7	350	SE
8	413	AL
9	485	RN
10	495	PI
11	536	PB
12	715	MS
13	747	MA
14	907	MT
15	975	PA
16	1336	CE
17	1652	PE
18	2020	GO

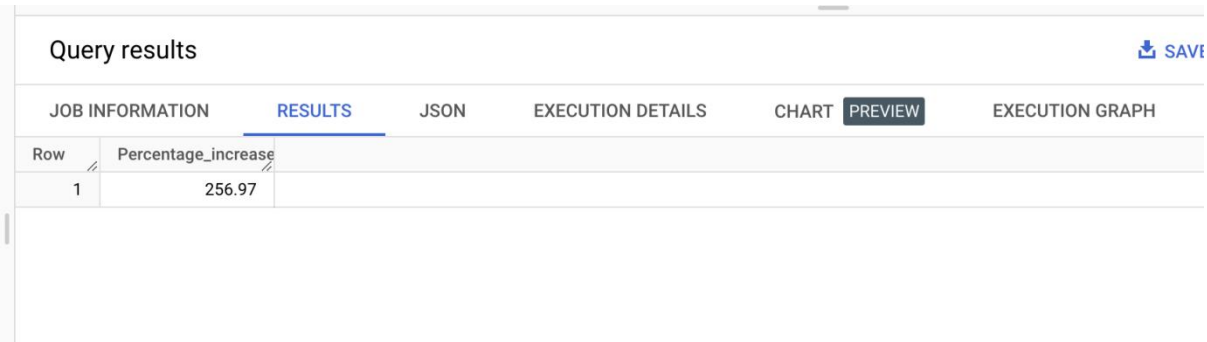
Results per page: 50

The Highest no of customers are from the state SP and the least are from RR

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

1.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.

```
select round(Payment_value/ LAG(Payment_value) OVER ( ORDER BY
year ) *100,2) AS Percentage_increase from(select
round(sum(payment_value),2) as Payment_value, extract(year
from order_approved_at) as Year from dsml-sql-
396708.case_study.payments p join dsml-sql-
396708.case_study.orders o using(order_id)
where extract(year from order_approved_at) = 2017 and
extract(month from order_approved_at) between 1 and 7 or
extract(year from order_approved_at) = 2018 and extract(month
from order_approved_at) between 1 and 7
group by Year
order by Year)
limit 1
```



The percentage increase of payments from 2017 to 2018 for only the months Jan to Aug is 256.97

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

```
select round(sum(payment_value)) as
Sum_of_orders,round(Avg(payment_value)) as avg_of_order,
customer_state from dsml-sql-396708.case_study.payments p
join dsml-sql-396708.case_study.orders o using(order_id)
join dsml-sql-396708.case_study.customers c using(customer_id)
group by c.customer_state
order by Sum_of_orders,avg_of_order
```

Query results SAVE

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART PREVIEW | EXECUTION GRAPH

Row	Sum_of_orders	avg_of_order	customer_state
1	10065.0	219.0	RR
2	16263.0	232.0	AP
3	19681.0	234.0	AC
4	27967.0	182.0	AM
5	60866.0	233.0	RO
6	61485.0	204.0	TO
7	75246.0	208.0	SE
8	96962.0	227.0	AL
9	102718.0	197.0	RN
10	108524.0	207.0	PI
11	137535.0	187.0	MS
12	141546.0	248.0	PB
13	152523.0	199.0	MA
14	187029.0	195.0	MT

The sate RR has the least Sum of order value where as the state SP has the highest sum of order value.

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

```
SELECT round(sum(freight_value)) as
Sum_of_freight_value,round(Avg(freight_value)) as
avg_of_freight_value,customer_state FROM `dsml-sql-
396708.case_study.order_items` join dsml-sql-
396708.case_study.orders o using(order_id) join dsml-sql-
396708.case_study.customers c using(customer_id)
group by c.customer_state
order by Sum_of_freight_value
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART PREVIEW | EXECUTION GRAPH

Row	Sum_of_freight_valu	avg_of_freight_value	customer_state
1	2235.0	43.0	RR
2	2789.0	34.0	AP
3	3687.0	40.0	AC
4	5479.0	33.0	AM
5	11417.0	41.0	RO
6	11733.0	37.0	TO
7	14111.0	37.0	SE
8	15915.0	36.0	AL
9	18860.0	36.0	RN
10	19144.0	23.0	MS
11	21218.0	39.0	PI
12	25720.0	43.0	PB
13	29715.0	28.0	MT
14	31524.0	38.0	MA

The sate RR has the least Sum of order freight value where as the state SP has the highest sum of order freight value.

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

1. 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.

```
select
date_diff(order_delivered_customer_date,order_purchase_timesta
mp,day) as
time_to_deliver,date_diff(order_estimated_delivery_date,order_
delivered_customer_date,day) as diff_estimated_delivery from
dsml-sql-396708.case_study.orders
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART PREVIEW | EXECUTION GRAPH

Row	time_to_deliver	diff_estimated_deliv
1	30	-12
2	30	28
3	35	16
4	30	1
5	32	0
6	29	1
7	43	-4
8	40	-4
9	37	-1
10	33	-5
11	38	-6
12	36	-2
13	34	0
14	42	-11

There is a negative in diff_estimated_delivery as the order has taken more time to deliver than the estimated date

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

```
SELECT Top_5_states_with_highest_avg_freight_value,
Top_5_states_with_lowest_avg_freight_value
FROM (select
Top_5_states_with_highest_avg_freight_value,row_number() over()
as top_rank from(SELECT round(avg(freight_value)) as
freight_value,customer_state as
Top_5_states_with_highest_avg_freight_value FROM `dsml-sql-
396708.case_study.order_items` join dsml-sql-
396708.case_study.orders o using(order_id) join dsml-sql-
396708.case_study.customers c using(customer_id)
group by customer_state
ORDER BY freight_value desc
LIMIT 5)) AS top_states
JOIN ( select
Top_5_states_with_lowest_avg_freight_value,row_number() over()
as bottom_rank from(SELECT round(avg(freight_value)) as
freight_value,customer_state as
Top_5_states_with_lowest_avg_freight_value FROM `dsml-sql-
396708.case_study.order_items` join dsml-sql-
396708.case_study.orders o using(order_id) join dsml-sql-
396708.case_study.customers c using(customer_id)
group by customer_state
ORDER BY freight_value asc
LIMIT 5)) AS bottom_states
ON top_states.top_rank = bottom_states.bottom_rank;
```

Query results SAVE

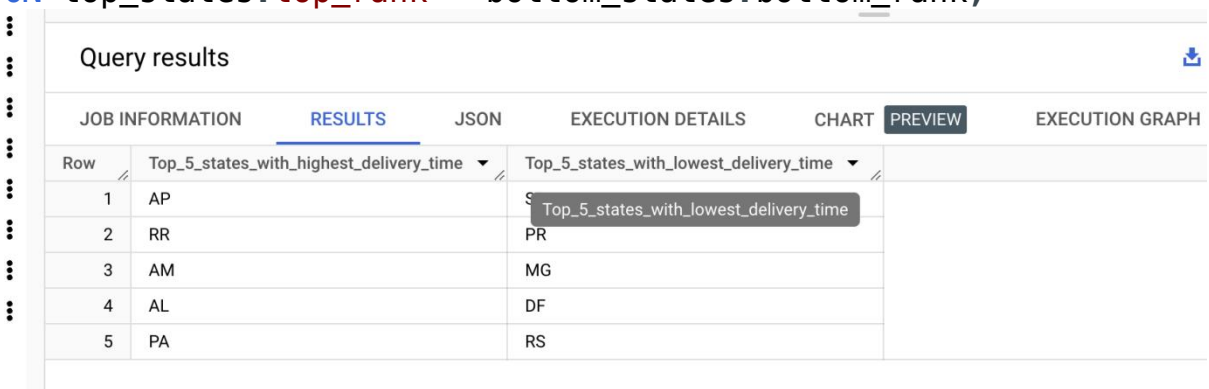
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART PREVIEW | EXECUTION GRAPH

Row	Top_5_states_with_highest_avg_freight_value	Top_5_states_with_lowest_avg_freight_value
1	PB	SP
2	RR	PR
3	RO	RJ
4	AC	DF
5	PI	MG

Here are the list of Top 5 states with the highest & lowest average freight value

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

```
SELECT Top_5_states_with_highest_delivery_time,
Top_5_states_with_lowest_delivery_time
FROM (select
Top_5_states_with_highest_delivery_time,row_number() over() as
top_rank from(SELECT
round(avg(date_diff(order_delivered_customer_date,order_purcha
se_timestamp,day))) as time_to_deliver,customer_state as
Top_5_states_with_highest_delivery_time FROM `dsml-sql-
396708.case_study.order_items` join dsml-sql-
396708.case_study.orders o using(order_id) join dsml-sql-
396708.case_study.customers c using(customer_id)
group by customer_state
ORDER BY time_to_deliver desc
LIMIT 5)) AS top_states
JOIN ( select
Top_5_states_with_lowest_delivery_time,row_number() over() as
bottom_rank from(SELECT
round(avg(date_diff(order_delivered_customer_date,order_purcha
se_timestamp,day))) as time_to_deliver,customer_state as
Top_5_states_with_lowest_delivery_time FROM `dsml-sql-
396708.case_study.order_items` join dsml-sql-
396708.case_study.orders o using(order_id) join dsml-sql-
396708.case_study.customers c using(customer_id)
group by customer_state
ORDER BY time_to_deliver asc
LIMIT 5)) AS bottom_states
ON top_states.top_rank = bottom_states.bottom_rank;
```



Here are the list of Top 5 states with the highest & lowest average delivery time

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

```
select
round(avg(date_diff(order_delivered_customer_date,order_estima
ted_delivery_date,day))) as
diff_between_delivered_and_estimated_date,customer_state from
dsml-sql-396708.case_study.orders o join dsml-sql-
396708.case_study.customers c using(customer_id)
group by customer_state
order by diff_between_delivered_and_estimated_date
limit 5
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART PREVIEW | EXECUTION GRA

Row	diff_between_deliver	customer_state
1	-20.0	AC
2	-19.0	RO
3	-19.0	AM
4	-19.0	AP
5	-16.0	RR

These are the top 5 states where the customer gets thier orders fast when compared to the estimated delivery time

6 . Analysis based on the payments:

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

```
select count(order_id) as No_of_orders,payment_type,
extract(month from order_purchase_timestamp) as Month FROM
`dsml-sql-396708.case_study.orders` o join dsml-sql-
396708.case_study.payments p using (order_id)
group by Month,payment_type order by Month
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART PREVIEW | EXECUTION GRA

Row	No_of_orders	payment_type	Month
1	6103	credit_card	1
2	1715	UPI	1
3	477	voucher	1
4	118	debit_card	1
5	1723	UPI	2
6	6609	credit_card	2
7	424	voucher	2
8	82	debit_card	2
9	7707	credit_card	3
10	1942	UPI	3

This is the Month on month no. of orders across different payment types.

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

```
SELECT count(order_id) as No_of_orders,payment_installments
FROM `dsml-sql-396708.case_study.payments`
group by payment_installments
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART PREVIEW | EXECUTION GRAP

Row	No_of_orders	payment_installment
1	2	0
2	52546	1
3	12413	2
4	10461	3
5	7098	4
6	5239	5
7	3920	6
8	1626	7
9	4268	8
10	644	9

These are the No of orders that are being paid on the basis of installments that have bee paid