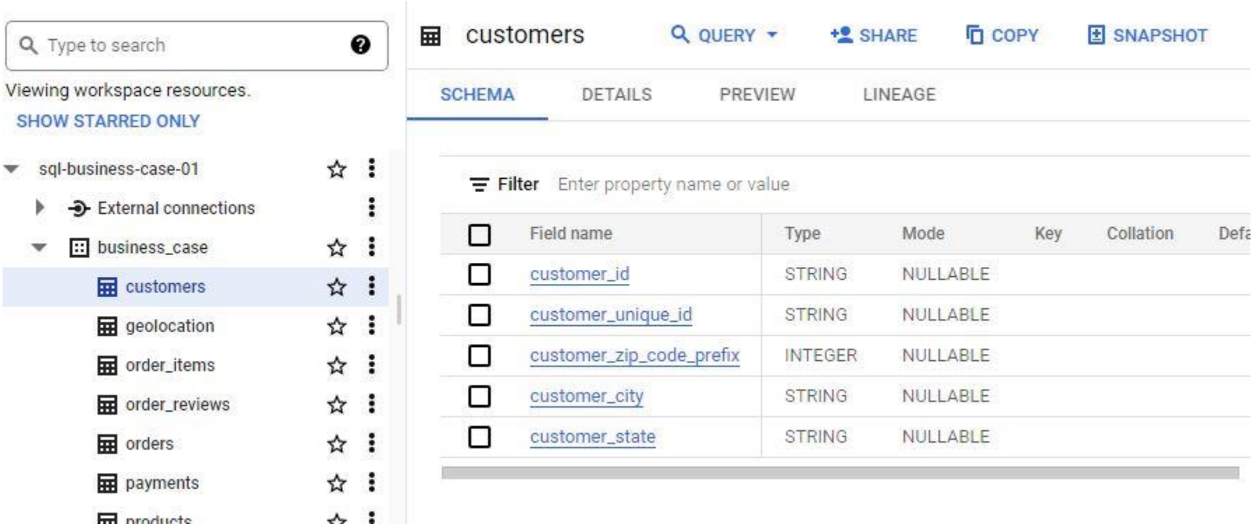
1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:

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



	Field name	Type	Mode	Key	Collation	Defa
☐	customer_id	STRING	NULLABLE			
☐	customer_unique_id	STRING	NULLABLE			
☐	customer_zip_code_prefix	INTEGER	NULLABLE			
☐	customer_city	STRING	NULLABLE			
☐	customer_state	STRING	NULLABLE			

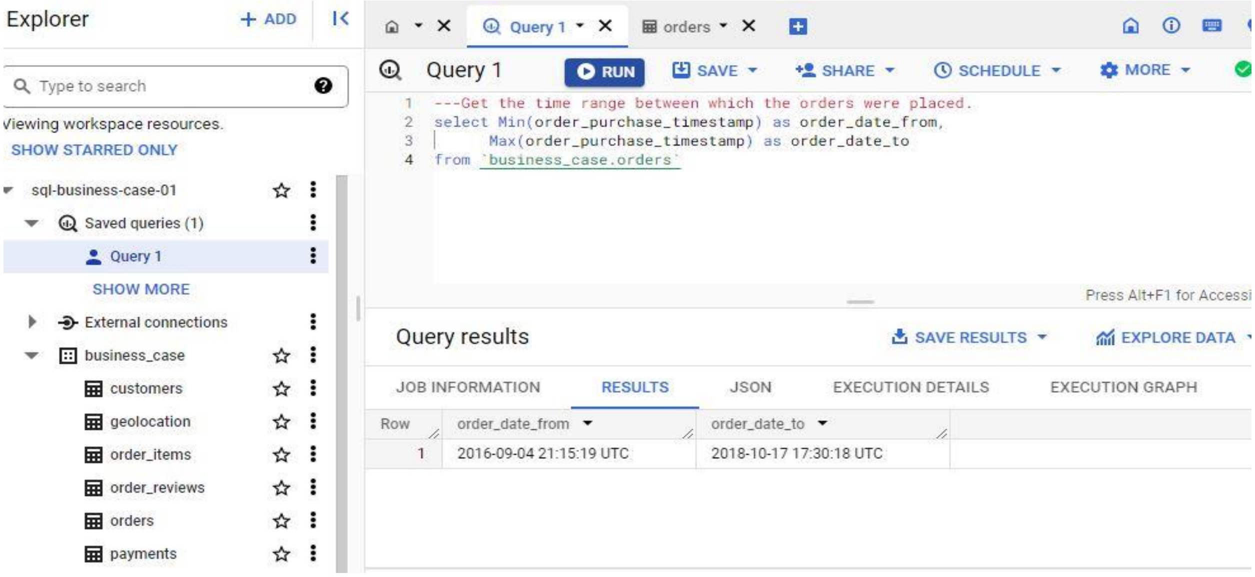
Observation:

- 5 columns are given
- 1 column consists of Numerical number and rest of 4 column are String

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

Query:

```
select Min(order_purchase_timestamp) as order_date_from,
      Max(order_purchase_timestamp) as order_date_to
from `business_case.orders`
```



```
---Get the time range between which the orders were placed.
select Min(order_purchase_timestamp) as order_date_from,
       Max(order_purchase_timestamp) as order_date_to
from `business_case.orders`
```

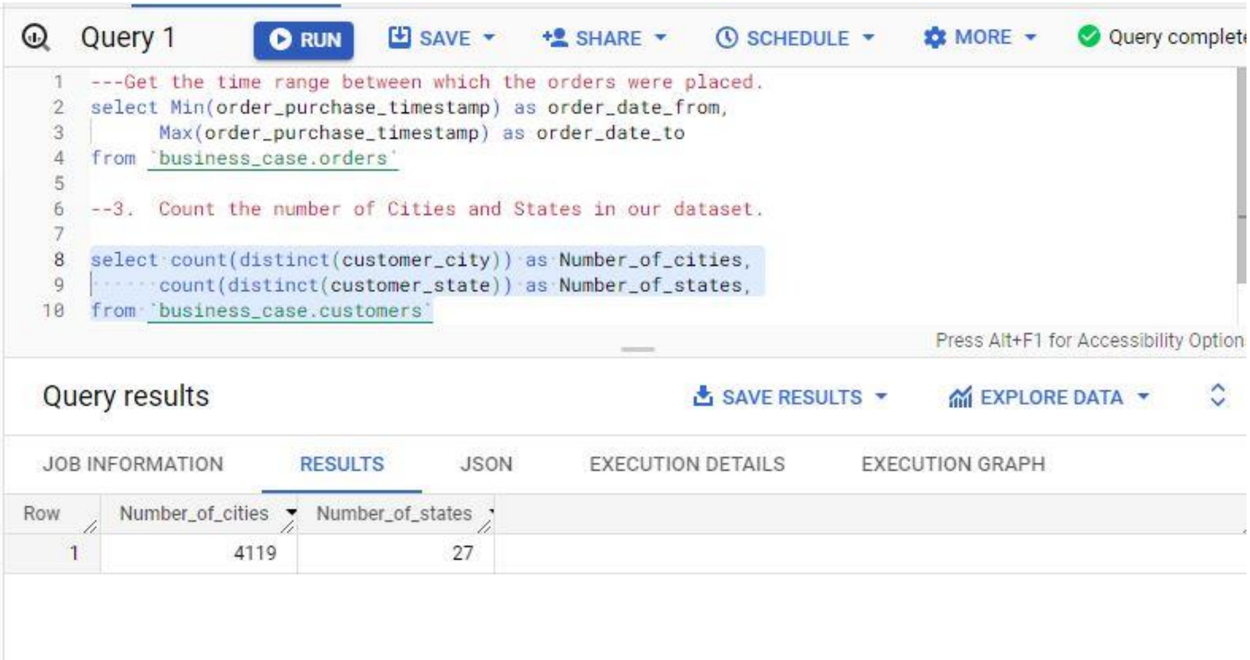
Row	order_date_from	order_date_to
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

Observation: Orders placed between from 2016-09-04 to 2018-10-17

3. Count the number of Cities and States in our dataset.

Query:

```
select count(distinct(customer_city)) as Number_of_cities,
      count(distinct(customer_state)) as Number_of_states,
from `business_case.customers`
```



```
---Get the time range between which the orders were placed.
select Min(order_purchase_timestamp) as order_date_from,
      Max(order_purchase_timestamp) as order_date_to
from `business_case.orders`

--3.  Count the number of Cities and States in our dataset.

select count(distinct(customer_city)) as Number_of_cities,
      count(distinct(customer_state)) as Number_of_states,
from `business_case.customers`
```

Row	Number_of_cities	Number_of_states
1	4119	27

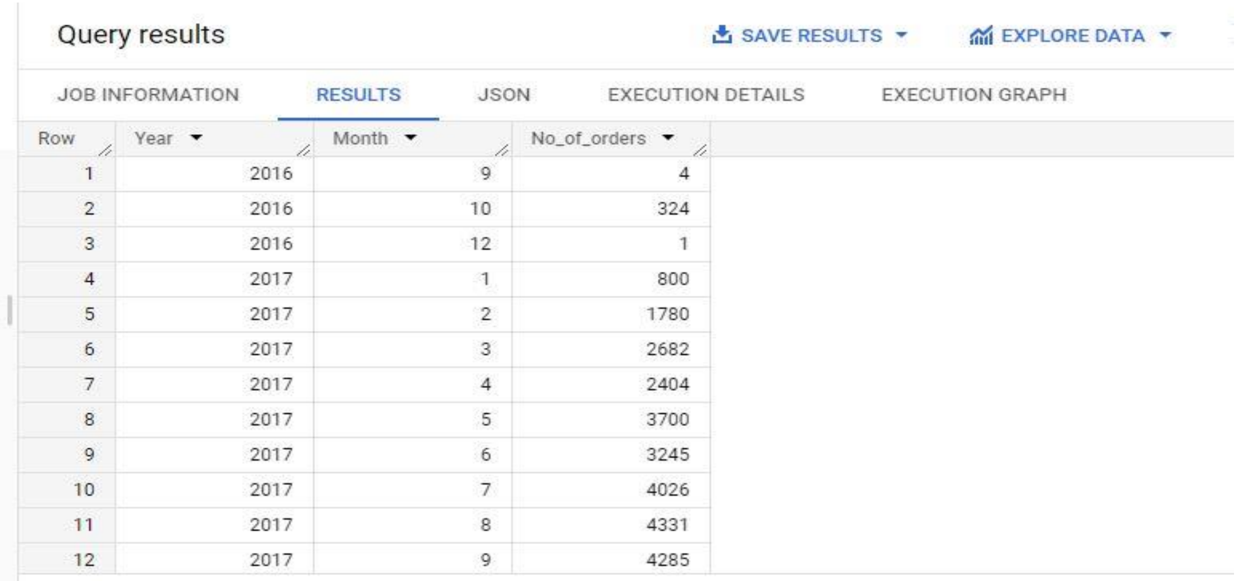
Observation: Number of cities are 4119 and Number of states are 27.

2. In-depth Exploration:

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

Query:

```
select extract(year from order_purchase_timestamp) as Year,
      extract(month from order_purchase_timestamp) as Month,
      count(order_id) as No_of_orders
from `business_case.orders`
group by 1,2
order by 1,2
```

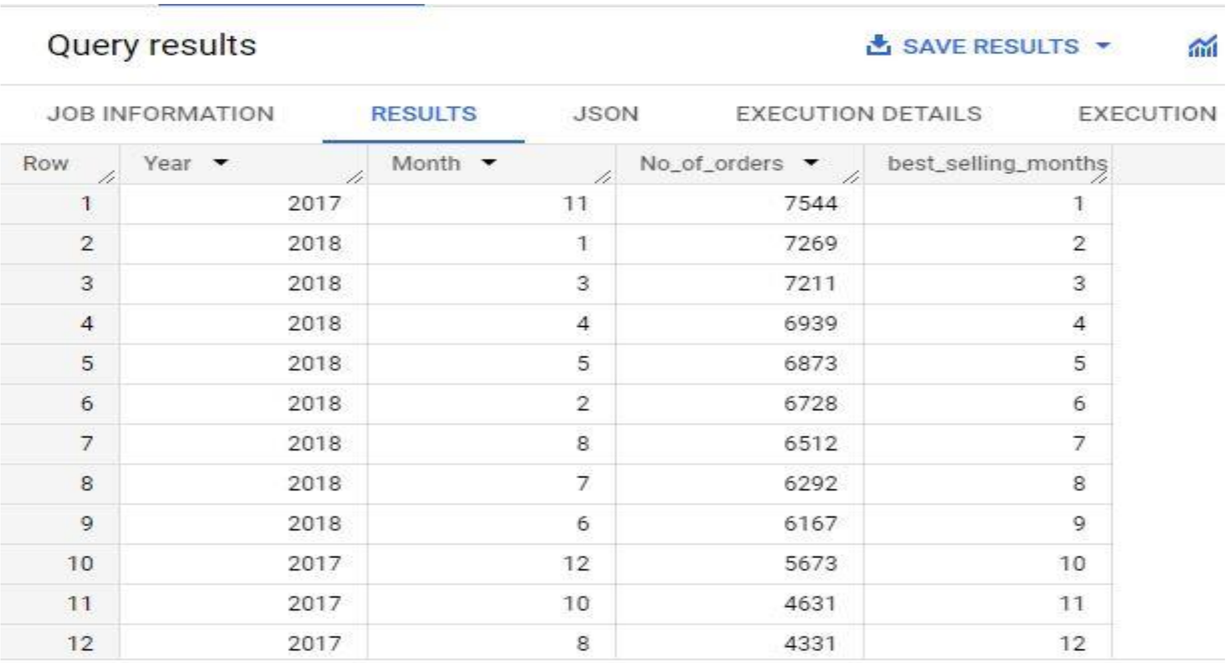


Row	Year	Month	No_of_orders
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026
11	2017	8	4331
12	2017	9	4285

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

Query:

```
select extract(year from order_purchase_timestamp) as Year,
      extract(month from order_purchase_timestamp) as Month,
      count(order_id) as No_of_orders,
      dense_rank() over(order by count(order_id) desc) as best_selling_months
from `business_case.orders`
group by 1,2
order by best_selling_months
```



Row	Year	Month	No_of_orders	best_selling_months
1	2017	11	7544	1
2	2018	1	7269	2
3	2018	3	7211	3
4	2018	4	6939	4
5	2018	5	6873	5
6	2018	2	6728	6
7	2018	8	6512	7
8	2018	7	6292	8
9	2018	6	6167	9
10	2017	12	5673	10
11	2017	10	4631	11
12	2017	8	4331	12

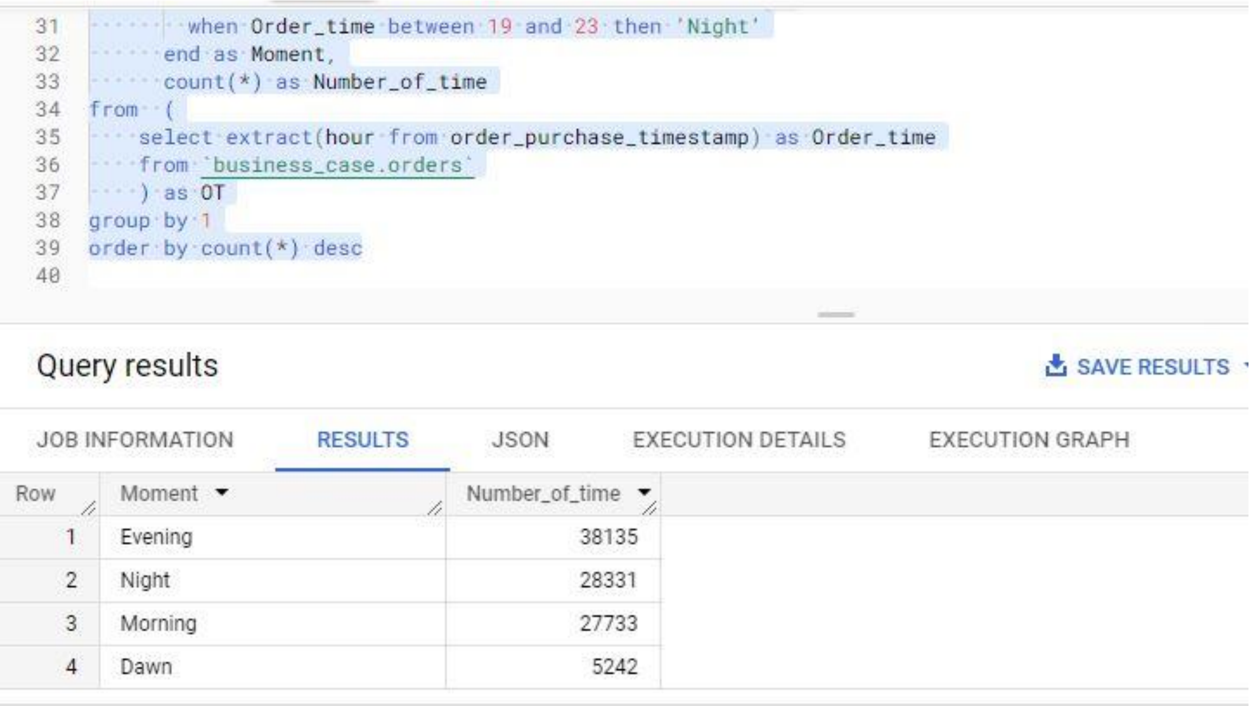
Observation:

- Highest order placed on the month of November’2017
- After that we can see a declination in order placing

3. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)
    i. 0-6 hrs : Dawn
    ii. 7-12 hrs : Mornings
    iii. 13-18 hrs : Afternoon
    iv. 19-23 hrs : Night

Query:

```
SELECT CASE
        WHEN Order_time between 0 and 6 THEN 'Dawn'
        WHEN Order_time between 7 and 12 THEN 'Morning'
        WHEN Order_time between 13 and 18 THEN 'Evening'
        WHEN Order_time between 19 and 23 THEN 'Night'
      END AS Moment,
      count(*) AS Number_of_time
FROM  (
    SELECT extract(hour from order_purchase_timestamp) AS Order_time
    FROM   `business_case.orders`
    ) as OT
GROUP  BY 1
ORDER  BY count(*) DESC
```



```
      when Order_time between 19 and 23 then 'Night'
    end as Moment,
    count(*) as Number_of_time
from  (
   select extract(hour from order_purchase_timestamp) as Order_time
   from `business_case.orders`
   ) as OT
group by 1
order by count(*) desc
```

Query results

Row	Moment	Number_of_time
1	Evening	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

Observation: We can observe that most of order placed by Brazilian customers on ‘Evening Time’

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

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

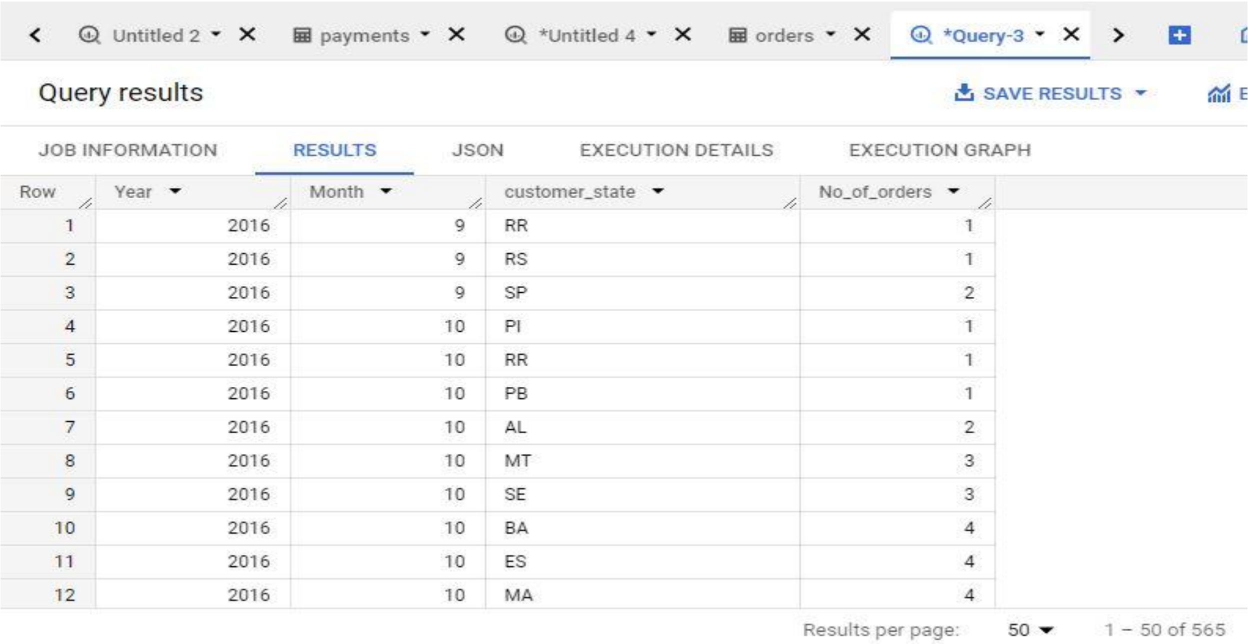
Query:

```
with order_state_tbl as
(select o.order_id, extract(year from o.order_purchase_timestamp) as Year,extract(month from
order_purchase_timestamp) as Month, c.customer_state
from `business_case.orders` as o
join `business_case.customers`as c
on o.customer_id=c.customer_id)


select distinct Year, Month,customer_state,
      count(order_id) over(partition by customer_state order by Year,Month) as No_of_orders
from order_state_tbl
order by 1,2,4 with order_state_tbl as

(select o.order_id, extract(year from o.order_purchase_timestamp) as Year,extract(month from
order_purchase_timestamp) as Month, c.customer_state
from `business_case.orders` as o
join `business_case.customers`as c
on o.customer_id=c.customer_id)

select Year,Month,customer_state,
      case
            when orders2 is null then orders1
            else (orders1-orders2)
      end as No_of_orders,
from(
select *,lag(orders1,1) over(partition by customer_state order by Year,Month) as orders2
from
(select distinct Year, Month,customer_state,
      count(order_id) over(partition by customer_state order by Year,Month) as orders1
from order_state_tbl))
order by 1,2,4
```



Query results

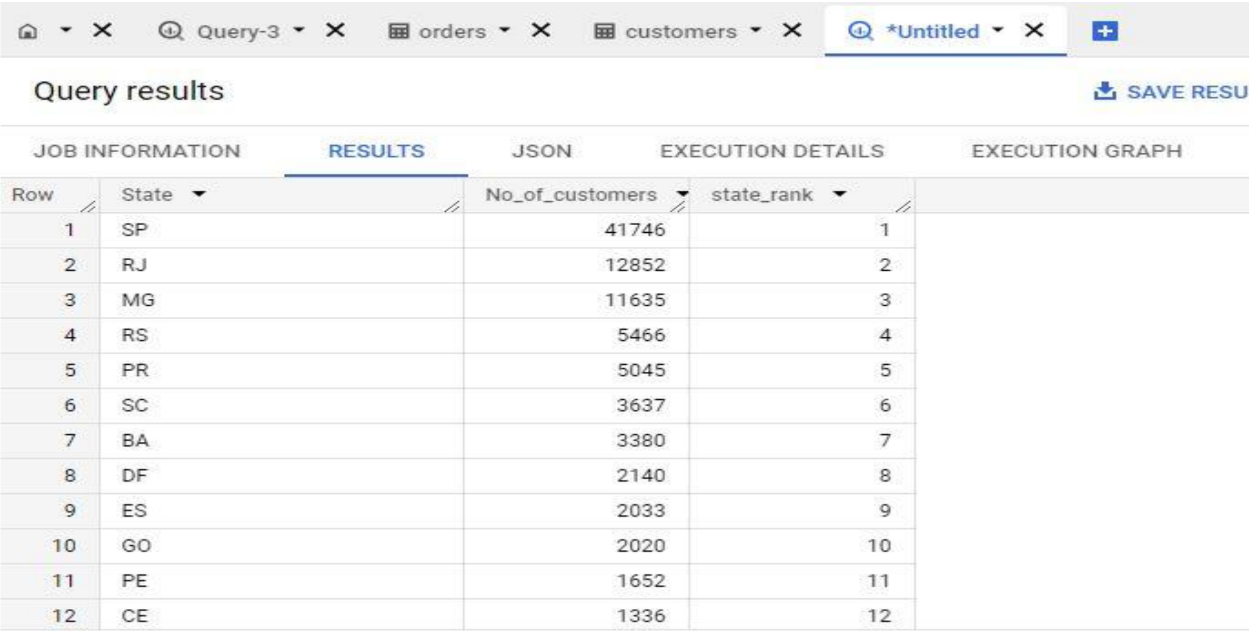
Row	Year	Month	customer_state	No_of_orders
1	2016	9	RR	1
2	2016	9	RS	1
3	2016	9	SP	2
4	2016	10	PI	1
5	2016	10	RR	1
6	2016	10	PB	1
7	2016	10	AL	2
8	2016	10	MT	3
9	2016	10	SE	3
10	2016	10	BA	4
11	2016	10	ES	4
12	2016	10	MA	4

Results per page: 50 1 – 50 of 565

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

Query:

```
select distinct c.customer_state as State, count(distinct c.customer_id) as No_of_customers,
      dense_rank() over(order by count(distinct c.customer_id) desc) as state_rank
from `business_case.customers` as c
join `business_case.orders` as o
on c.customer_id=o.customer_id
where o.order_id is not null
group by 1
order by 3
```



Query results

Row	State	No_of_customers	state_rank
1	SP	41746	1
2	RJ	12852	2
3	MG	11635	3
4	RS	5466	4
5	PR	5045	5
6	SC	3637	6
7	BA	3380	7
8	DF	2140	8
9	ES	2033	9
10	GO	2020	10
11	PE	1652	11
12	CE	1336	12

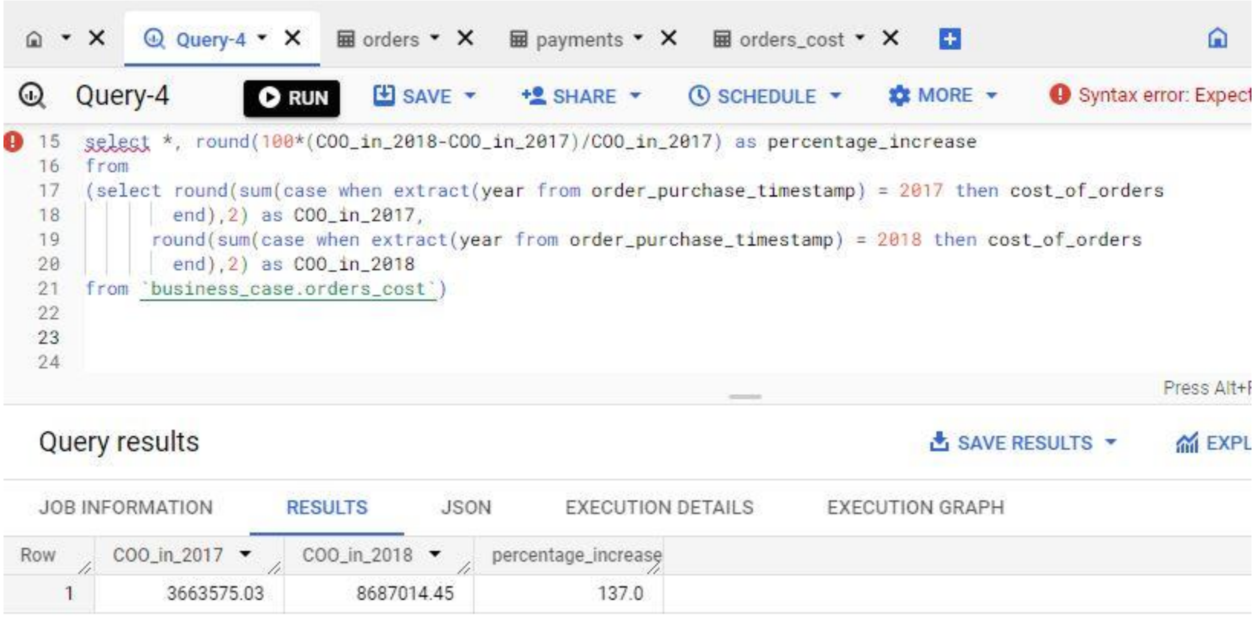
Observation: Highest number of customers, placed order from SP state.

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.

Query:

```
create or replace table `business_case.orders_cost` as
(
select distinct(o.order_id), o.order_purchase_timestamp, p.payment_value as cost_of_orders
from `business_case.orders`as o
join `business_case.payments`as p
on o.order_id=p.order_id
where (extract(year from order_purchase_timestamp) in (2017,2018))
      and (extract(month from order_purchase_timestamp) between 1 and 8)
)

select *, round(100*(COO_in_2018-COO_in_2017)/COO_in_2017) as percentage_increase
from
(select round(sum(case when extract(year from order_purchase_timestamp) = 2017 then
cost_of_orders
        end),2) as COO_in_2017,
      round(sum(case when extract(year from order_purchase_timestamp) = 2018 then
cost_of_orders
        end),2) as COO_in_2018
from `business_case.orders_cost`)
```



Syntax error: Expect

```
select *, round(100*(COO_in_2018-COO_in_2017)/COO_in_2017) as percentage_increase
from
(select round(sum(case when extract(year from order_purchase_timestamp) = 2017 then cost_of_orders
      end),2) as COO_in_2017,
    round(sum(case when extract(year from order_purchase_timestamp) = 2018 then cost_of_orders
      end),2) as COO_in_2018
from `business_case.orders_cost`)
```

Row	COO_in_2017	COO_in_2018	percentage_increase
1	3663575.03	8687014.45	137.0

Observation:

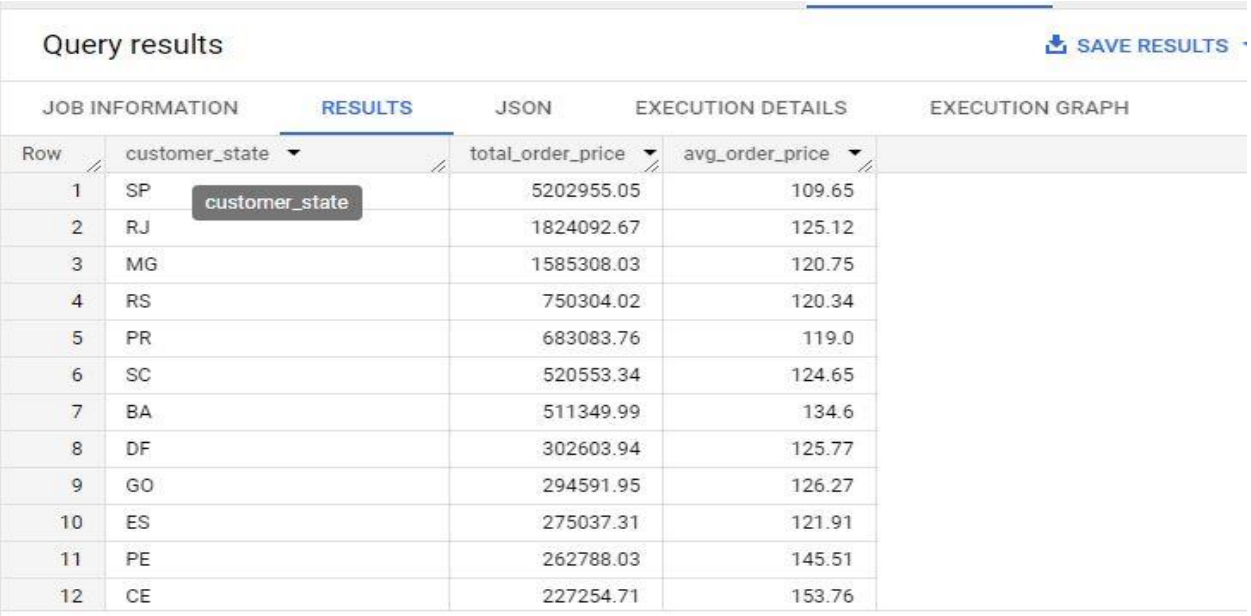
- The percentage increase in the ‘cost of orders’ from 2017 to 2018 is 137%.

- Note that, considered only month from ‘January’ to ‘August’.

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

Query:

```
select distinct(c.customer_state), round(sum(ot.price),2) as total_order_price,
round(avg(ot.price),2) as avg_order_price
from `business_case.order_items` ot
join
`business_case.orders` o
on ot.order_id=o.order_id
join
`business_case.customers` c
on o.customer_id=c.customer_id
group by 1
order by 2 desc
```



Row	customer_state	total_order_price	avg_order_price
1	SP	5202955.05	109.65
2	RJ	1824092.67	125.12
3	MG	1585308.03	120.75
4	RS	750304.02	120.34
5	PR	683083.76	119.0
6	SC	520553.34	124.65
7	BA	511349.99	134.6
8	DF	302603.94	125.77
9	GO	294591.95	126.27
10	ES	275037.31	121.91
11	PE	262788.03	145.51
12	CE	227254.71	153.76

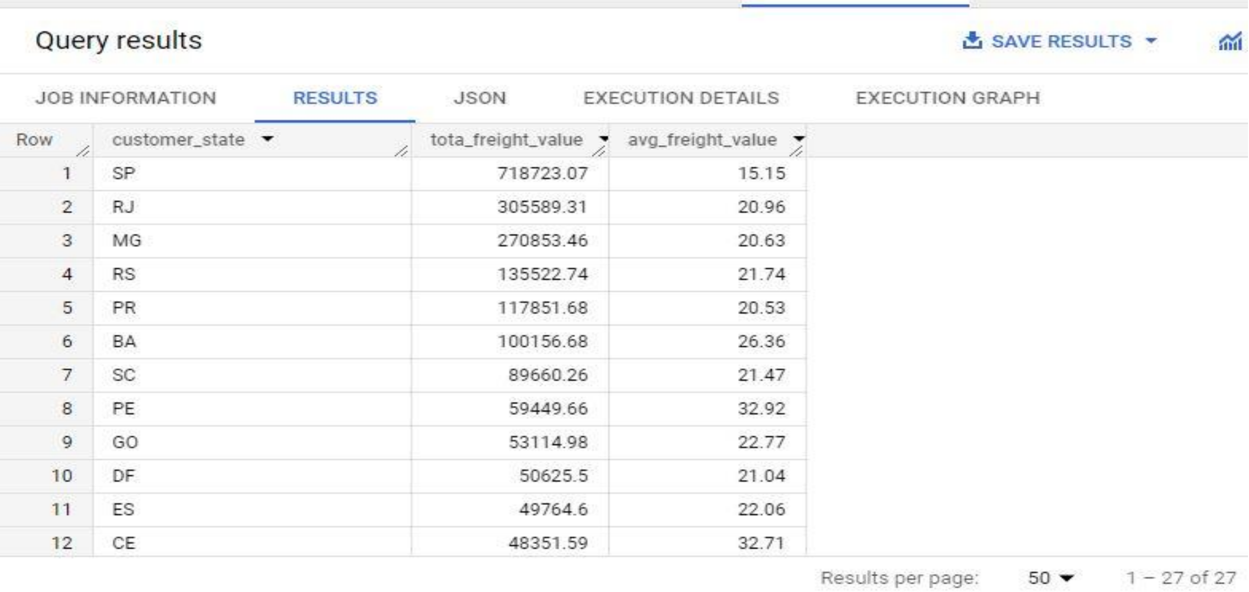
Observation: From this table, we can find out the total order price and avg order price for each state.

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

Query:

```
select distinct(c.customer_state), round(sum(ot.freight_value),2) as tota_freight_value,
round(avg(ot.freight_value),2) as avg_freight_value
from `business_case.order_items` ot
join
`business_case.orders` o
on ot.order_id=o.order_id
join
```

```
`business_case.customers` c
on o.customer_id=c.customer_id
group by 1
```



Row	customer_state	tota_freight_value	avg_freight_value
1	SP	718723.07	15.15
2	RJ	305589.31	20.96
3	MG	270853.46	20.63
4	RS	135522.74	21.74
5	PR	117851.68	20.53
6	BA	100156.68	26.36
7	SC	89660.26	21.47
8	PE	59449.66	32.92
9	GO	53114.98	22.77
10	DF	50625.5	21.04
11	ES	49764.6	22.06
12	CE	48351.59	32.71

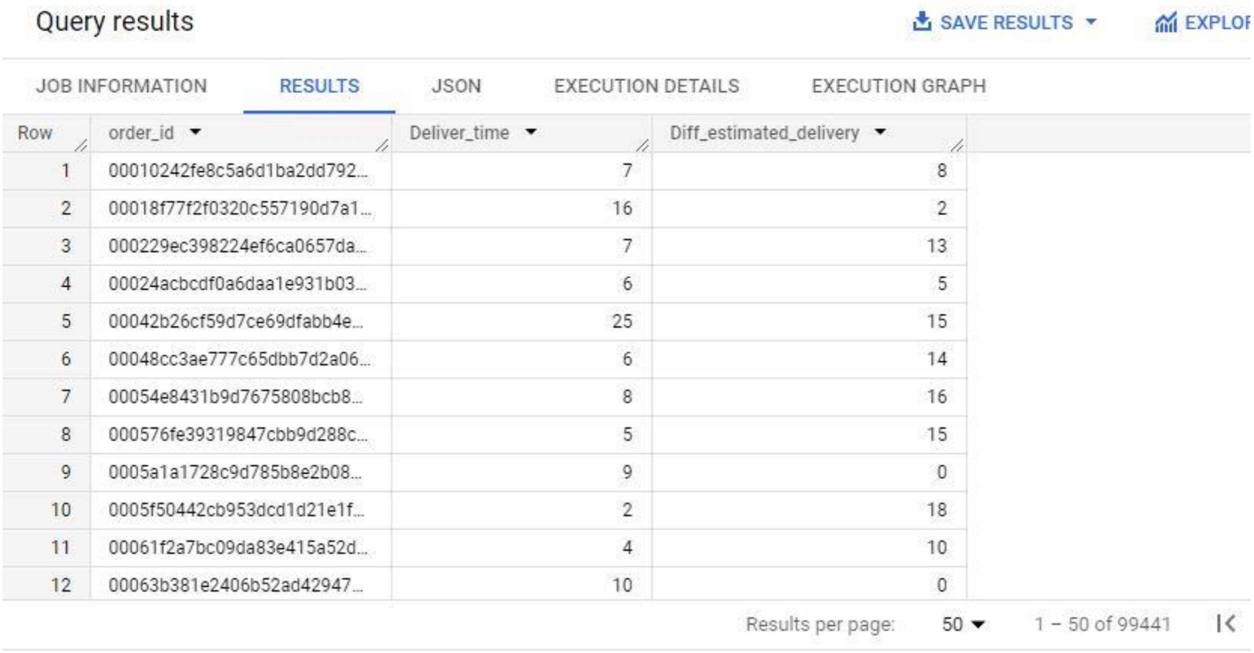
Results per page: 50 1 – 27 of 27

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.
       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:
        i. **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
        ii. **diff_estimated_delivery** = order_estimated_delivery_date - order_delivered_customer_date

Query:

```
select distinct order_id,
      date_diff(order_delivered_customer_date,order_purchase_timestamp, day) as Deliver_time,
      date_diff(order_estimated_delivery_date,order_delivered_customer_date, day) as
Diff_estimated_delivery
from `business_case.orders`
where order_id is not null
order by 1
```



Row	order_id	Deliver_time	Diff_estimated_delivery
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
11	00061f2a7bc09da83e415a52d...	4	10
12	00063b381e2406b52ad42947...	10	0

Results per page: 50 ▾ 1 – 50 of 99441

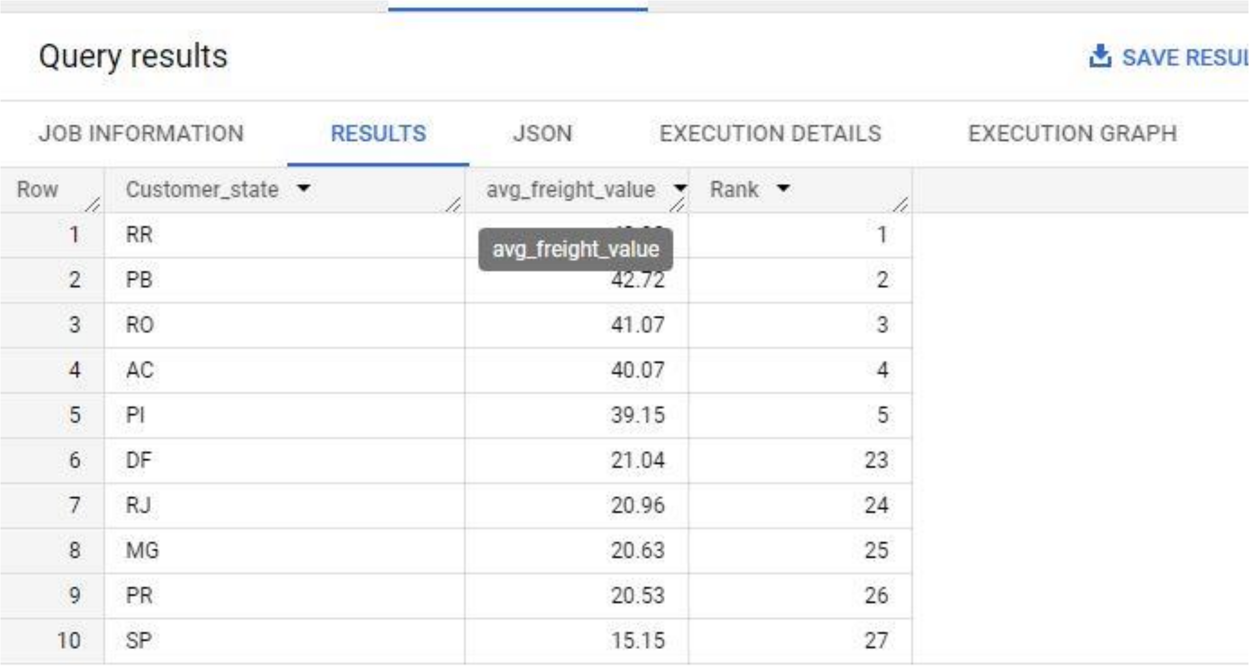
Observation:

- ‘Delivery_time’ column shows that the actual delivery time after placed the order.
- ‘Diff_estimated_delivery’ column shows that difference between the actual and estimated delivery date. +ve sign means the order is delivered before estimated delivery time and –ve sign denotes that the order takes more time to delivered as per scheduled estimated delivery time.

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

```
--2.  Find out the top 5 states with the highest & lowest average freight value.
with avg_freight_tbl as
(select distinct(c.customer_state) as Customer_state, round(avg(ot.freight_value),2) as
avg_freight_value
from `business_case.order_items` ot
join
`business_case.orders` o
on ot.order_id=o.order_id
join
`business_case.customers` c
on o.customer_id=c.customer_id
group by 1)
 ,
 Ranks as
(select *,
     row_number() over (order by  avg_freight_value desc) as TopFive,
     row_number() over(order by avg_freight_value) as BottomFive
from avg_freight_tbl)

select Customer_state, avg_freight_value, TopFive as Rank
from ranks
where TopFive <=5 or BottomFive <=5
order by 3
```



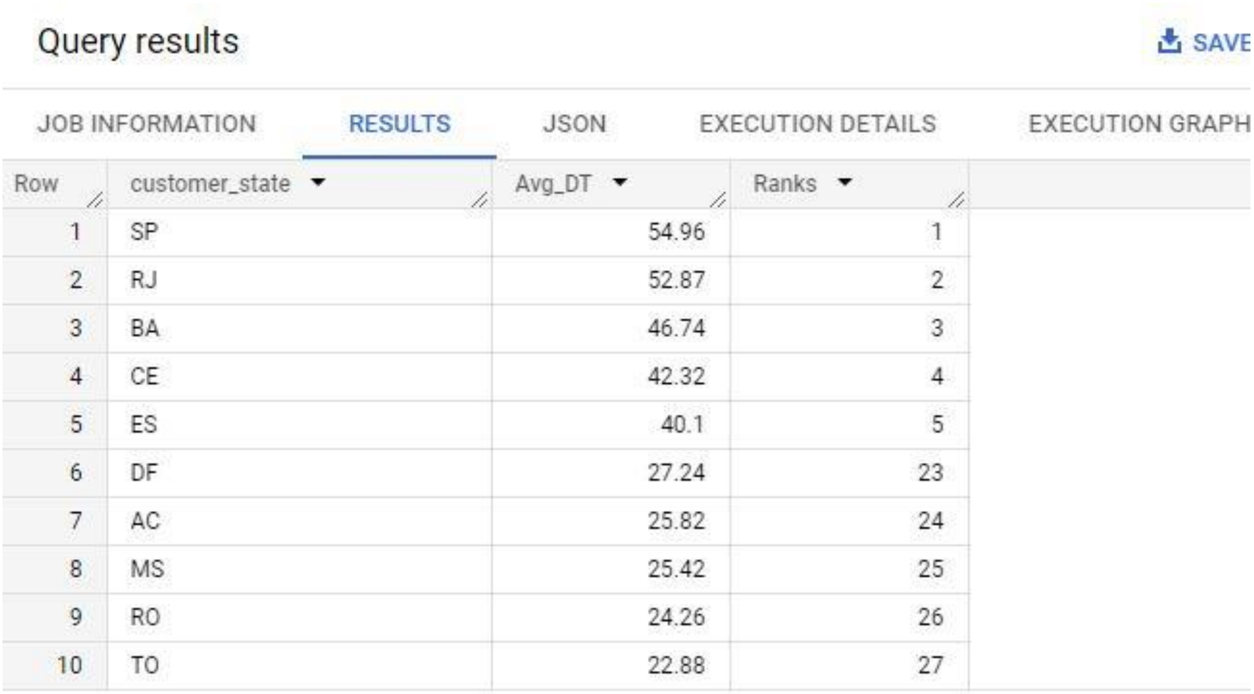
Row	Customer_state	avg_freight_value	Rank
1	RR	[illegible]	1
2	PB	42.72	2
3	RO	41.07	3
4	AC	40.07	4
5	PI	39.15	5
6	DF	21.04	23
7	RJ	20.96	24
8	MG	20.63	25
9	PR	20.53	26
10	SP	15.15	27

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

Query:

```
with avg_del_time_tbl as
(Select customer_state, round(avg(Deliver_time),2) as Avg_DT
from
(select distinct c.customer_state,
      date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp, day) as
Deliver_time,
from `business_case.orders` o
join `business_case.customers` c
on o.customer_id=c.customer_id
where o.order_delivered_customer_date is not null)
group by 1)
,
avg_del_rank as
(select *,
      row_number() over(order by Avg_DT desc) as Top5,
      row_number() over(order by Avg_DT) as Bottom5
from avg_del_time_tbl)

select customer_state, Avg_DT, Top5 as Ranks
from avg_del_rank
where Top5<=5 or Bottom5<=5
order by 3
```

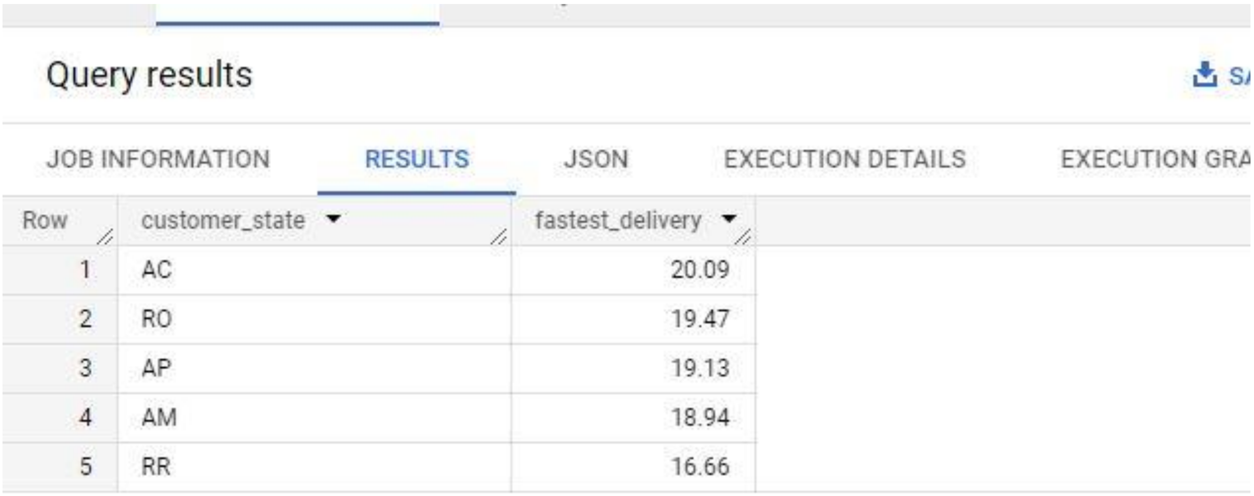


Row	customer_state	Avg_DT	Ranks
1	SP	54.96	1
2	RJ	52.87	2
3	BA	46.74	3
4	CE	42.32	4
5	ES	40.1	5
6	DF	27.24	23
7	AC	25.82	24
8	MS	25.42	25
9	RO	24.26	26
10	TO	22.88	27

4. 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:

```
select distinct c.customer_state, round(avg(o.estm_date) over(partition by customer_state)-
avg(o.actual_date) over(partition by customer_state),2) as fastest_delivery
from
(select distinct order_id, customer_id,
      date_diff(order_delivered_customer_date, order_purchase_timestamp,day) as actual_date,
      date_diff(order_estimated_delivery_date, order_purchase_timestamp, day) as estm_date
from `business_case.orders`
where order_delivered_customer_date is not null) as o
join
(select distinct customer_state, customer_id
from `business_case.customers`) c
on o.customer_id = c.customer_id
order by 2 desc
limit 5
```



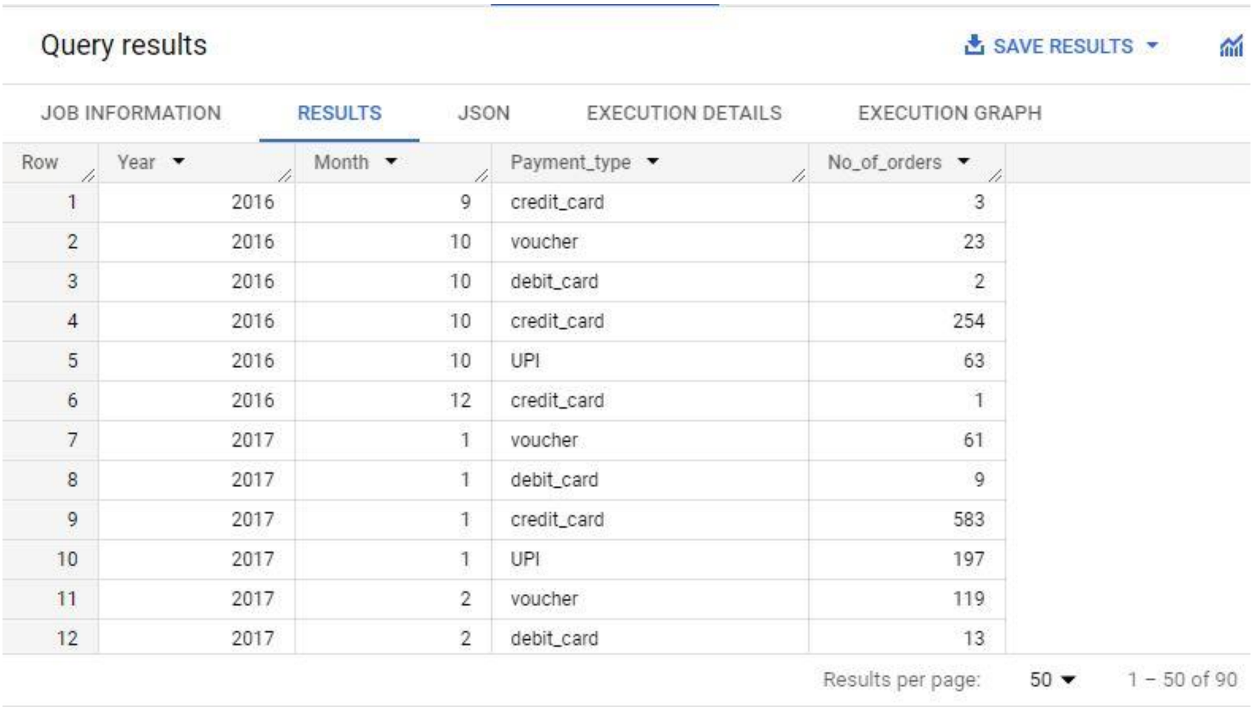
Row	customer_state	fastest_delivery
1	AC	20.09
2	RO	19.47
3	AP	19.13
4	AM	18.94
5	RR	16.66

6. Analysis based on the payments:

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

Query:

```
select Year,Month,Payment_type,
      case
            when noo2 is null then noo1
            else (noo1-noo2)
      end as No_of_orders,
from(
select *,lag(noo1,1) over(partition by Payment_type order by Year,Month) as noo2
from
(select distinct Year,Month,Payment_type, count(Payment_type) over(partition by Payment_type
order by Year,Month) as noo1
from
(select distinct order_id,
    extract(year from order_purchase_timestamp) as Year,
    extract(month from order_purchase_timestamp) as Month
from `business_case.orders`) o
join
(select order_id,
        payment_type as Payment_type,
from `business_case.payments`) p
on o.order_id=p.order_id))
order by 1,2
```



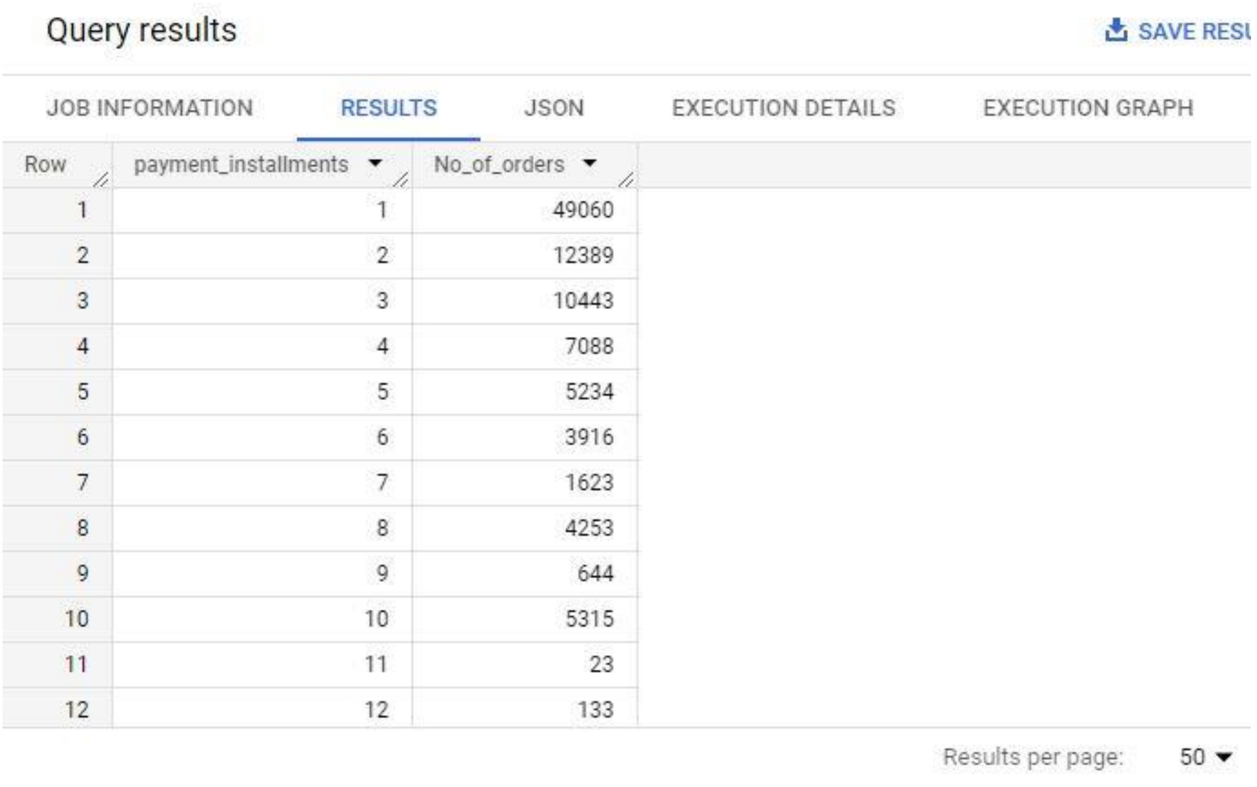
Row	Year	Month	Payment_type	No_of_orders
1	2016	9	credit_card	3
2	2016	10	voucher	23
3	2016	10	debit_card	2
4	2016	10	credit_card	254
5	2016	10	UPI	63
6	2016	12	credit_card	1
7	2017	1	voucher	61
8	2017	1	debit_card	9
9	2017	1	credit_card	583
10	2017	1	UPI	197
11	2017	2	voucher	119
12	2017	2	debit_card	13

Results per page: 50 1 – 50 of 90

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

Query:

```
select p.payment_installments, count(distinct o.order_id) as No_of_orders
from `business_case.payments` p
join `business_case.orders` o
on o.order_id=p.order_id
where o.order_id is not null
    and p.payment_installments >=1
group by 1
order by 1
```



Row	payment_installments	No_of_orders
1	1	49060
2	2	12389
3	3	10443
4	4	7088
5	5	5234
6	6	3916
7	7	1623
8	8	4253
9	9	644
10	10	5315
11	11	23
12	12	133

Results per page: 50