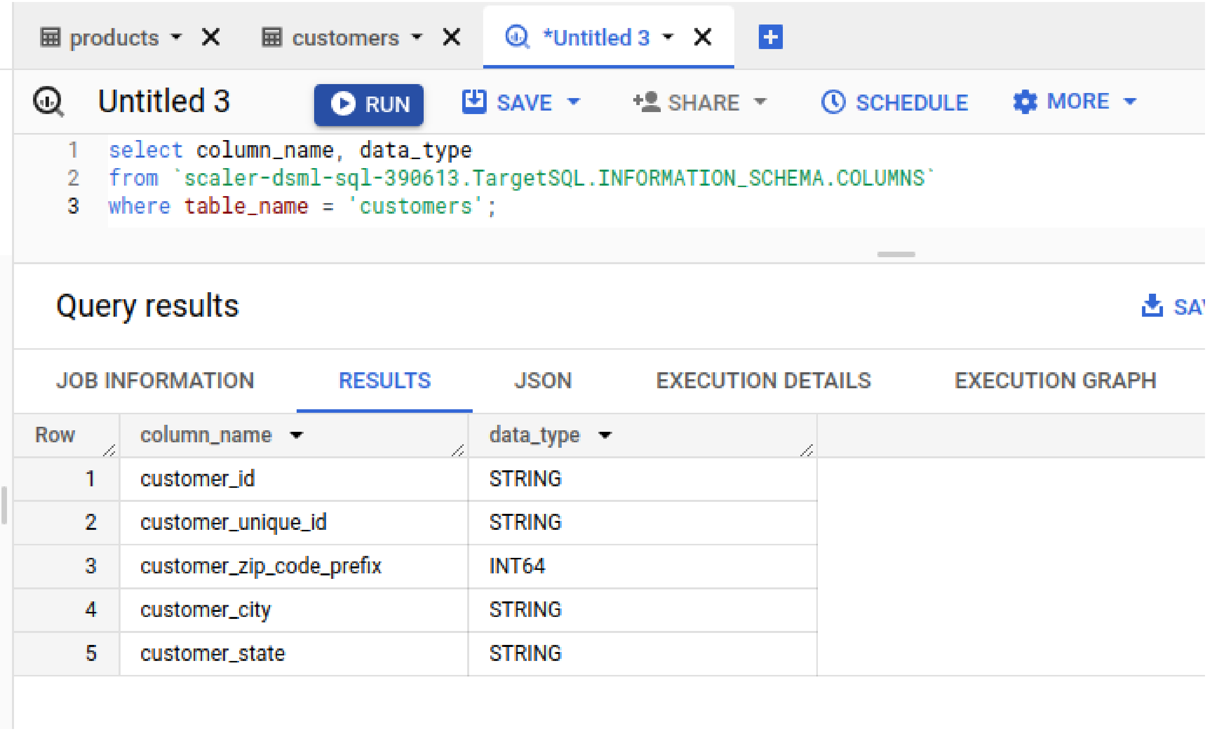
Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:

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

Query -

```
 select column_name, data_type
from `scaler-dsml-sql-390613.TargetSQL.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'customers';
```

Output -



```
select column_name, data_type
from `scaler-dsml-sql-390613.TargetSQL.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'customers';
```

Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

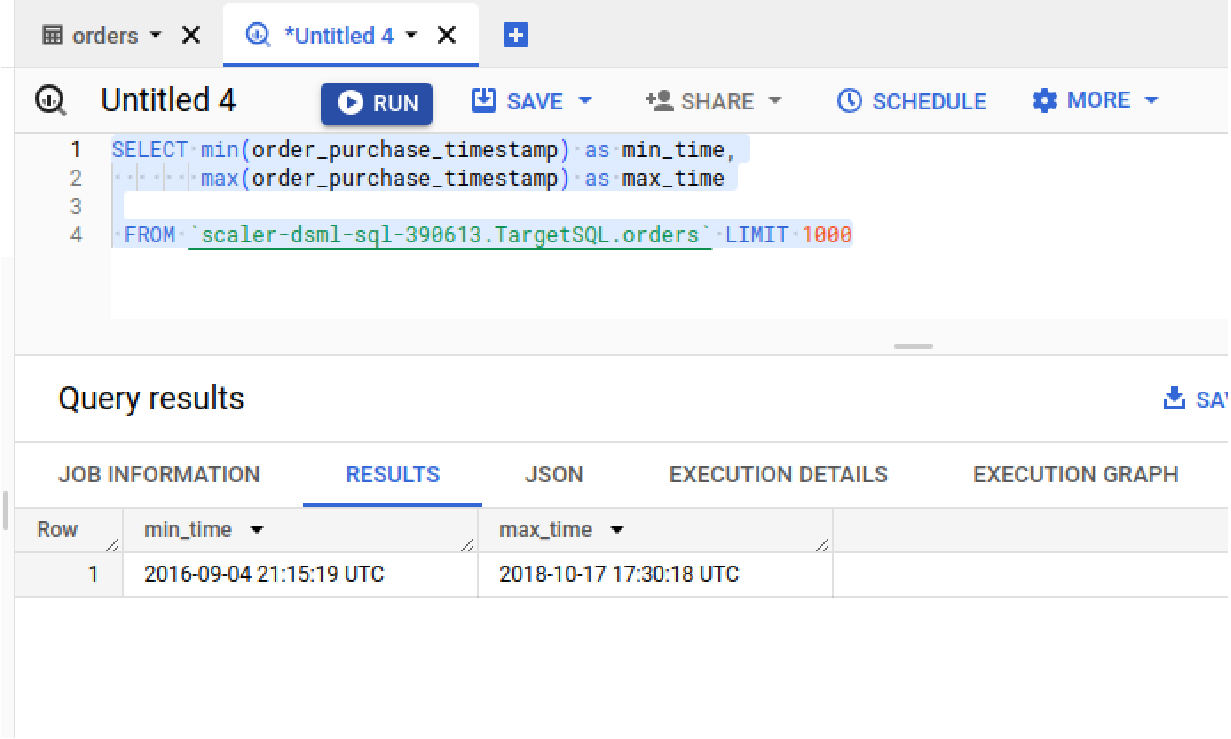
--------------------------------Please navigate to next page------------------------------

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

Query -

```
SELECT min(order_purchase_timestamp) as min_time,
max(order_purchase_timestamp) as max_time

FROM `scaler-dsml-sql-390613.TargetSQL.orders` LIMIT 1000
```

Output -



```
SELECT min(order_purchase_timestamp) as min_time,
      max(order_purchase_timestamp) as max_time

 FROM `scaler-dsml-sql-390613.TargetSQL.orders` LIMIT 1000
```

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

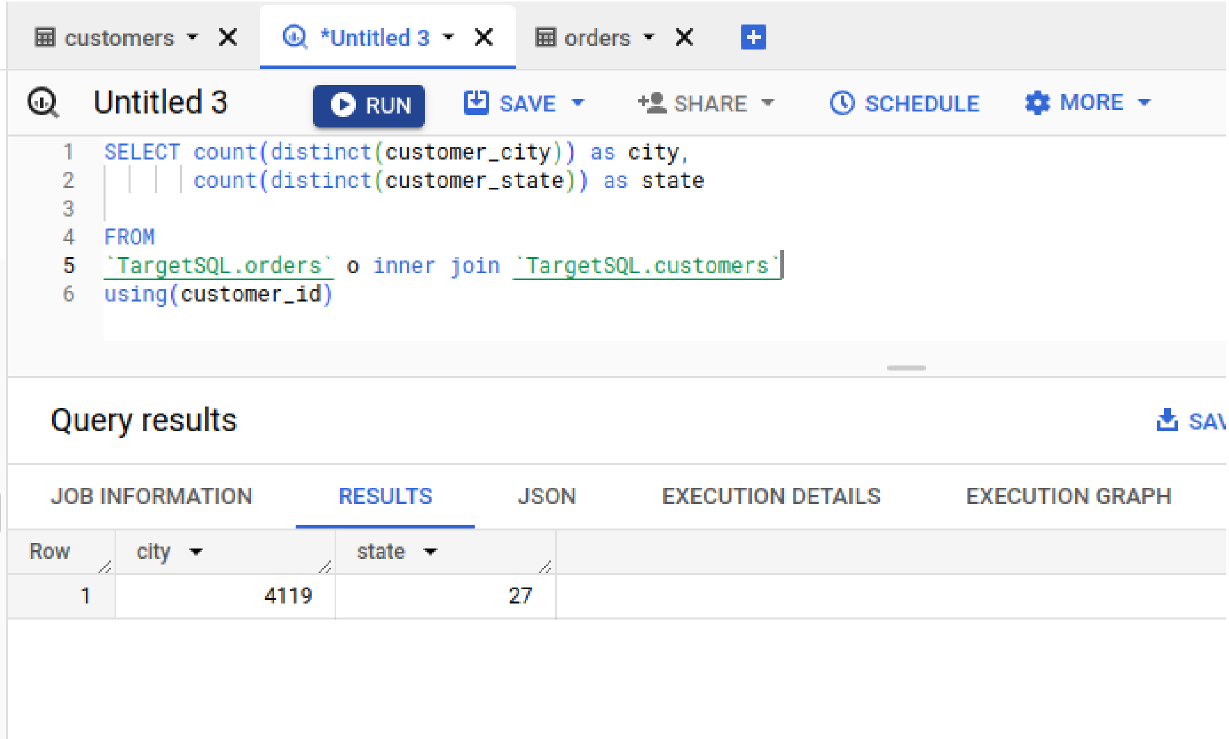
--------------------------------Please navigate to next page------------------------------

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

Query -

```
SELECT count(distinct(customer_city)) as city,
count(distinct(customer_state)) as state

FROM
`TargetSQL.orders` o inner join `TargetSQL.customers`
using(customer_id)
```

Output -



```
SELECT count(distinct(customer_city)) as city,
       count(distinct(customer_state)) as state

FROM
`TargetSQL.orders` o inner join `TargetSQL.customers`
using(customer_id)
```

Row	city	state
1	4119	27

Total number of city - 4119
Total number of states - 27

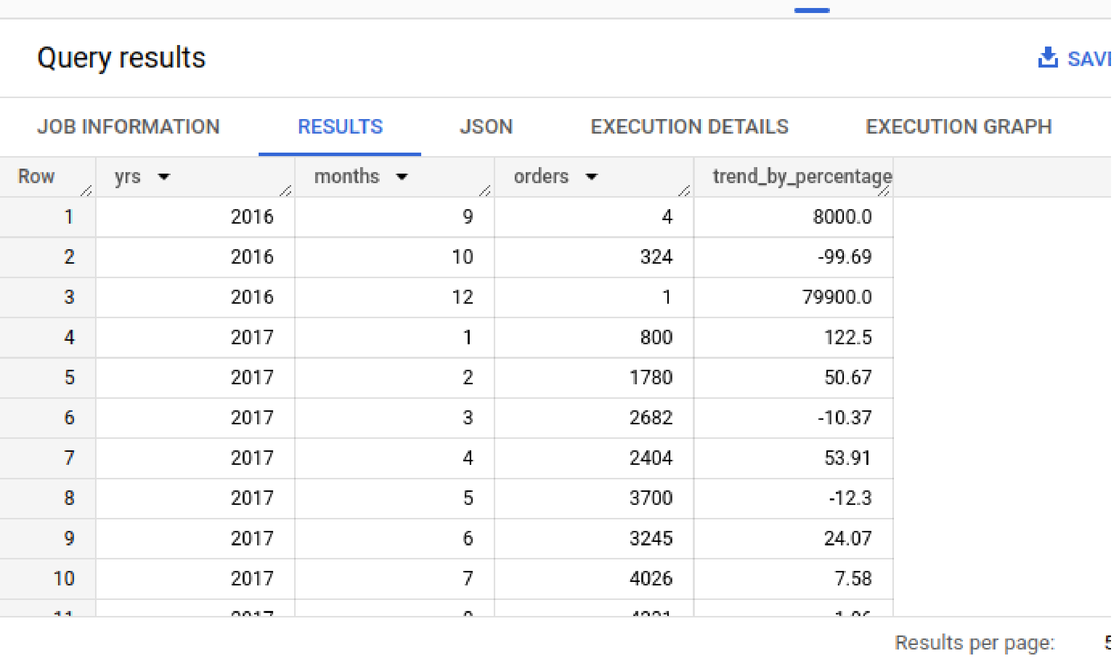
--------------------------------Please navigate to next page------------------------------

In-depth Exploration:

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

Query-

```
with cte as
(SELECT extract(year from order_purchase_timestamp) as yrs,
extract(month from order_purchase_timestamp) as months,
count(order_id) as orders

FROM `scaler-dsml-sql-390613.TargetSQL.orders`

group by yrs,months
order by yrs,months
)

select yrs,months,orders,
round(sum(((next-orders)/orders)*100) over(partition by yrs,months),2) as
trend_by_percentage
from(
select *,lead(orders) over(order by yrs,months) as next
from cte
order by yrs,months
)
order by yrs,months
```



Row	yrs	months	orders	trend_by_percentage
1	2016	9	4	8000.0
2	2016	10	324	-99.69
3	2016	12	1	79900.0
4	2017	1	800	122.5
5	2017	2	1780	50.67
6	2017	3	2682	-10.37
7	2017	4	2404	53.91
8	2017	5	3700	-12.3
9	2017	6	3245	24.07
10	2017	7	4026	7.58
11	2017	8	4331	1.06

We can identify tread by seeing order count increasing and decreasing or trend_by_ percentage shows us increase or decrease in number of orders per month.

By this trend we have observed that at initial stages when it was just started selling then we have a good upward trend in number of orders ,
Few months sales were up and down due to different festival activities and then at the end of the year we observed a drop in sales due to big holidays around the country.

---------------------------------Please navigate to next page-------------------------------

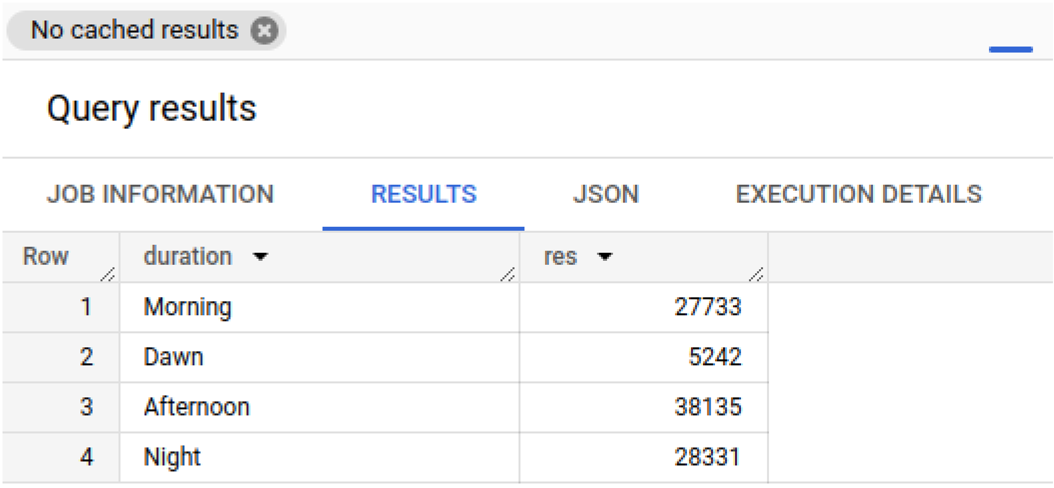
3. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

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

Query -

```
with cte as (
SELECT customer_id,order_id,
TIME(order_purchase_timestamp) as tym

FROM `scaler-dsml-sql-390613.TargetSQL.orders`
)
select duration,count(duration) as res
from(
select tym,
case
when tym between '00:00:00' and '06:59:59' then 'Dawn'
when tym between '07:00:00' and '12:59:59' then 'Morning'
when tym between '13:00:00' and '18:59:59' then 'Afternoon'
else 'Night'
end as duration
from cte)

group by duration
```

Output -



Row	duration	res
1	Morning	27733
2	Dawn	5242
3	Afternoon	38135
4	Night	28331

-------------------------------Please navigate to next page-----------------------------

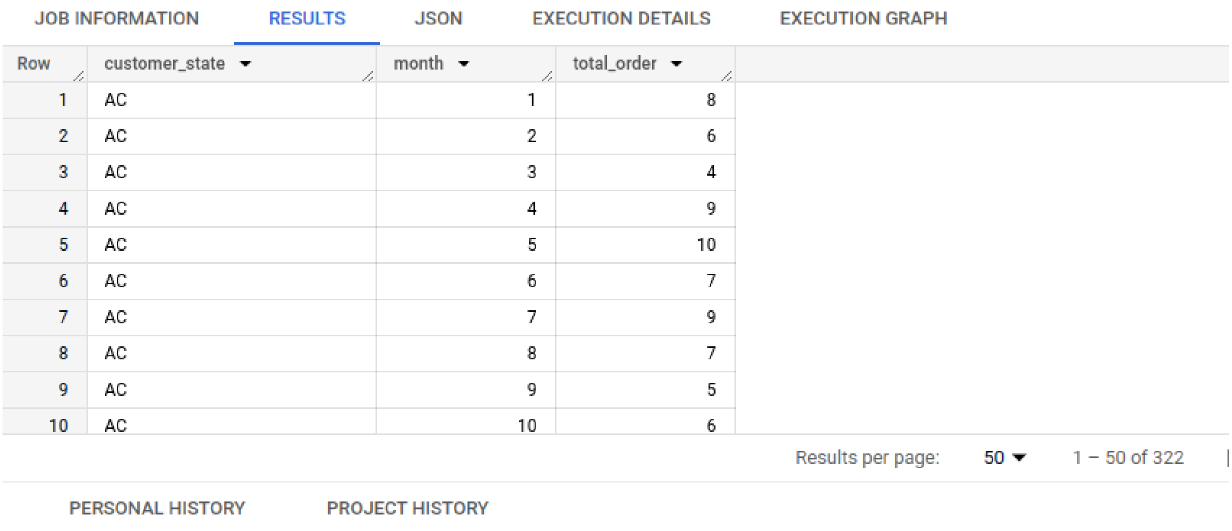
Evolution of E-commerce orders in the Brazil region

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

Query -

```
with cte as (SELECT c.customer_id,
                    c.customer_state,
                    o.order_id,
extract(month from o.order_purchase_timestamp) as month
FROM `scaler-dsml-sql-390613.TargetSQL.customers` c
join
`scaler-dsml-sql-390613.TargetSQL.orders` o
using (customer_id)

order by customer_state,month
)
select cte.customer_state,cte.month,count(cte.order_id) as total_order
from cte
group by cte.customer_state,cte.month
order by customer_state,month
```

Output -



Row	customer_state	month	total_order
1	AC	1	8
2	AC	2	6
3	AC	3	4
4	AC	4	9
5	AC	5	10
6	AC	6	7
7	AC	7	9
8	AC	8	7
9	AC	9	5
10	AC	10	6

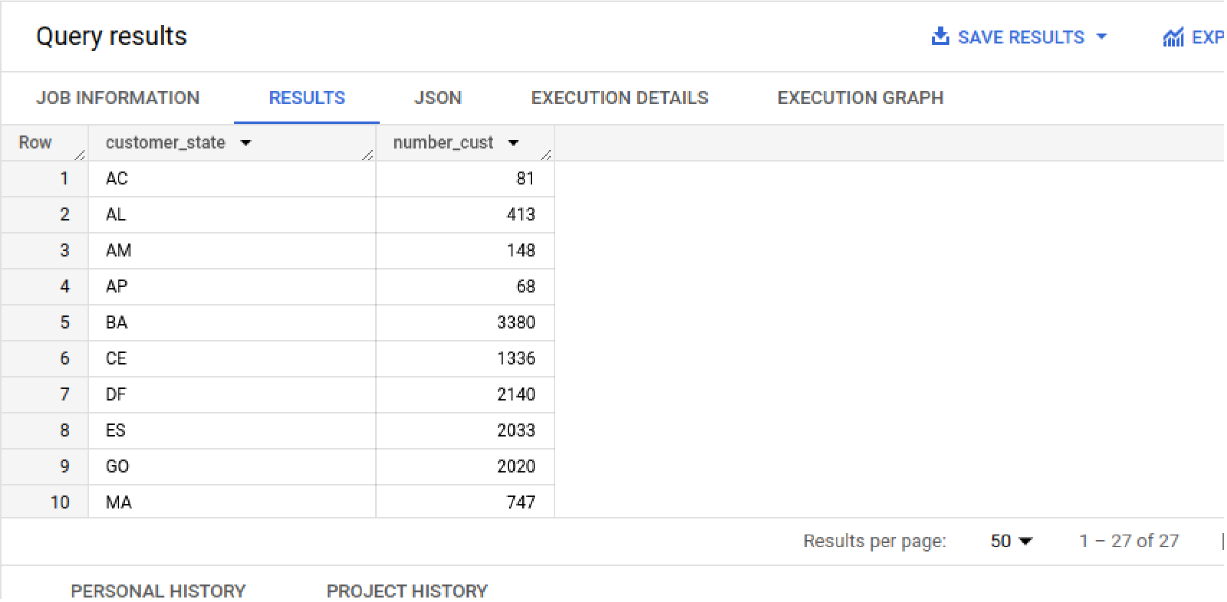
------------------------------Please navigate to next page------------------------------

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

Query -

```
SELECT customer_state,count(distinct(customer_id)) as number_cust
FROM `scaler-dsml-sql-390613.TargetSQL.customers`
group by customer_state
order by customer_state
```

Output -



Row	customer_state	number_cust
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

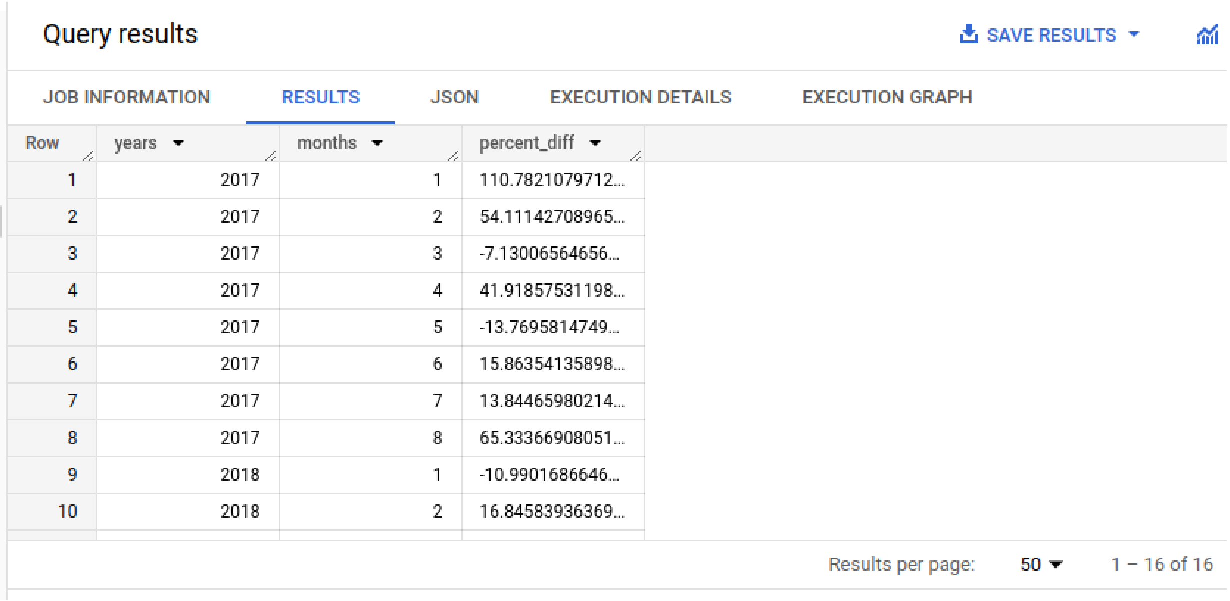
--------------------------------Please navigate to next page------------------------------

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 –

```
with cte as(
SELECT
p.payment_value,
extract(month from o.order_purchase_timestamp) as months,
extract(year from o.order_purchase_timestamp) as years
FROM `scaler-dsml-sql-390613.TargetSQL.orders` o join
`scaler-dsml-sql-390613.TargetSQL.payments` p
using(order_id)
order by years,months
)
select years,months,
sum(((nxt-total_cost)/total_cost)*100) over(partition by years,months
order by years,months) as percent_diff
from(
select *,
lead(total_cost) over(order by years,months) as nxt
from(
select years,months,
round(sum(payment_value),2) as total_cost #over(partition by
years,months order by years,months) as sum_order_by_month
from cte
where years IN(2017,2018) and months >0 and months <9
group by years,months
order by years,months
)
order by years,months
)
order by years,months
```



Row	years	months	percent_diff
1	2017	1	110.7821079712...
2	2017	2	54.11142708965...
3	2017	3	-7.13006564656...
4	2017	4	41.91857531198...
5	2017	5	-13.7695814749...
6	2017	6	15.86354135898...
7	2017	7	13.84465980214...
8	2017	8	65.33366908051...
9	2018	1	-10.9901686646...
10	2018	2	16.84583936369...

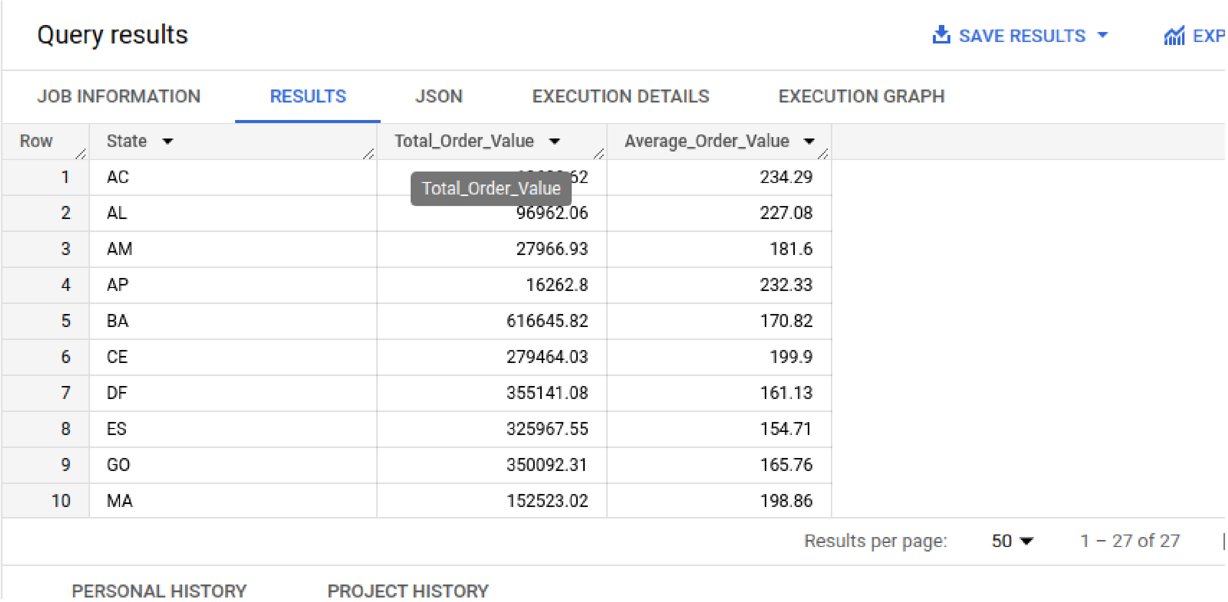
Results per page: 50 1 – 16 of 16

--------------------------------Please navigate to next page------------------------------

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

Query-

```
SELECT
c.customer_state as State,
round(sum(p.payment_value),2) as Total_Order_Value,
round(avg(p.payment_value),2) as Average_Order_Value

FROM `scaler-dsml-sql-390613.TargetSQL.customers` c
join
`scaler-dsml-sql-390613.TargetSQL.orders` o
using(customer_id)
join
`scaler-dsml-sql-390613.TargetSQL.payments` p
using(order_id)

group by c.customer_state
order by c.customer_state
```



Row	State	Total_Order_Value	Average_Order_Value
1	AC	[illegible]62	234.29
2	AL	96962.06	227.08
3	AM	27966.93	181.6
4	AP	16262.8	232.33
5	BA	616645.82	170.82
6	CE	279464.03	199.9
7	DF	355141.08	161.13
8	ES	325967.55	154.71
9	GO	350092.31	165.76
10	MA	152523.02	198.86

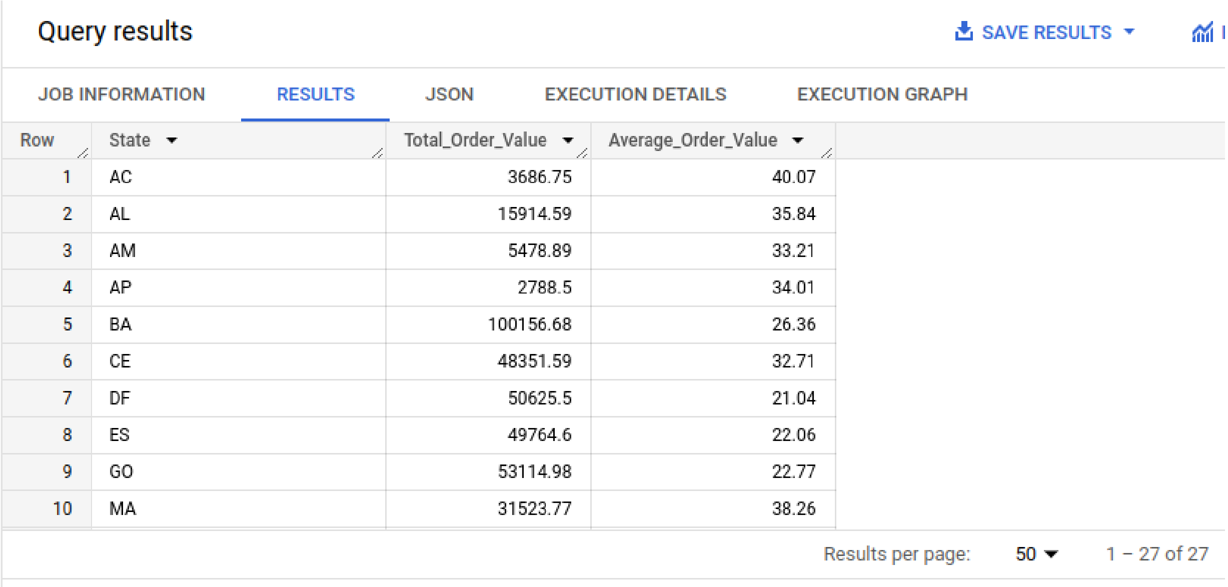
Results per page: 50 1 – 27 of 27

--------------------------------Please navigate to next page------------------------------

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

Query -

```
SELECT
c.customer_state as State,
round(sum(ot.freight_value),2) as Total_Order_Value,
round(avg(ot.freight_value),2) as Average_Order_Value


FROM `scaler-dsml-sql-390613.TargetSQL.customers` c
join
`scaler-dsml-sql-390613.TargetSQL.orders` o
using(customer_id)
join
`scaler-dsml-sql-390613.TargetSQL.order_items` ot
using(order_id)


group by c.customer_state
order by c.customer_state
```



Row	State	Total_Order_Value	Average_Order_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

--------------------------------Please navigate to next page------------------------------

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.

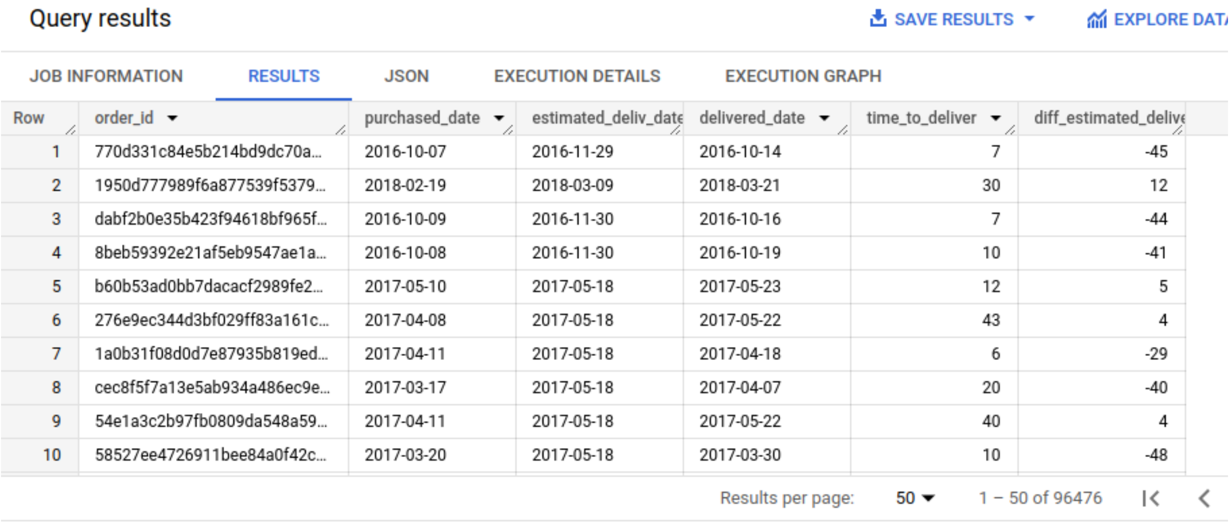
   You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

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

Query

```
SELECT order_id,

-- CASE

-- WHEN order_delivered_customer_date IS NULL THEN
'order_not_delivered'

-- ELSE order_status

-- END AS status,

extract(DATE from order_purchase_timestamp) as purchased_date,

extract(DATE from order_estimated_delivery_date) as
estimated_deliv_date,

extract(DATE from order_delivered_customer_date) as delivered_date,

TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp,
day) AS time_to_deliver,

TIMESTAMP_DIFF(order_delivered_customer_date,order_estimated_delivery_d
ate,day) AS diff_estimated_delivery
```

```
FROM `scaler-dsml-sql-390613.TargetSQL.orders`

where order_delivered_customer_date is not null
```



Query results

Row	order_id	purchased_date	estimated_deliv_date	delivered_date	time_to_deliver	diff_estimated_delive
1	770d331c84e5b214bd9dc70a...	2016-10-07	2016-11-29	2016-10-14	7	-45
2	1950d777989f6a877539f5379...	2018-02-19	2018-03-09	2018-03-21	30	12
3	dabf2b0e35b423f94618bf965f...	2016-10-09	2016-11-30	2016-10-16	7	-44
4	8beb59392e21af5eb9547ae1a...	2016-10-08	2016-11-30	2016-10-19	10	-41
5	b60b53ad0bb7dacacf2989fe2...	2017-05-10	2017-05-18	2017-05-23	12	5
6	276e9ec344d3bf029ff83a161c...	2017-04-08	2017-05-18	2017-05-22	43	4
7	1a0b31f08d0d7e87935b819ed...	2017-04-11	2017-05-18	2017-04-18	6	-29
8	cec8f5f7a13e5ab934a486ec9e...	2017-03-17	2017-05-18	2017-04-07	20	-40
9	54e1a3c2b97fb0809da548a59...	2017-04-11	2017-05-18	2017-05-22	40	4
10	58527ee4726911bee84a0f42c...	2017-03-20	2017-05-18	2017-03-30	10	-48

Here negative diff_estimated_delivery_date shows the order has been delivered earlier than estimated date and positive value indicates that order has been delayed by a number of days.

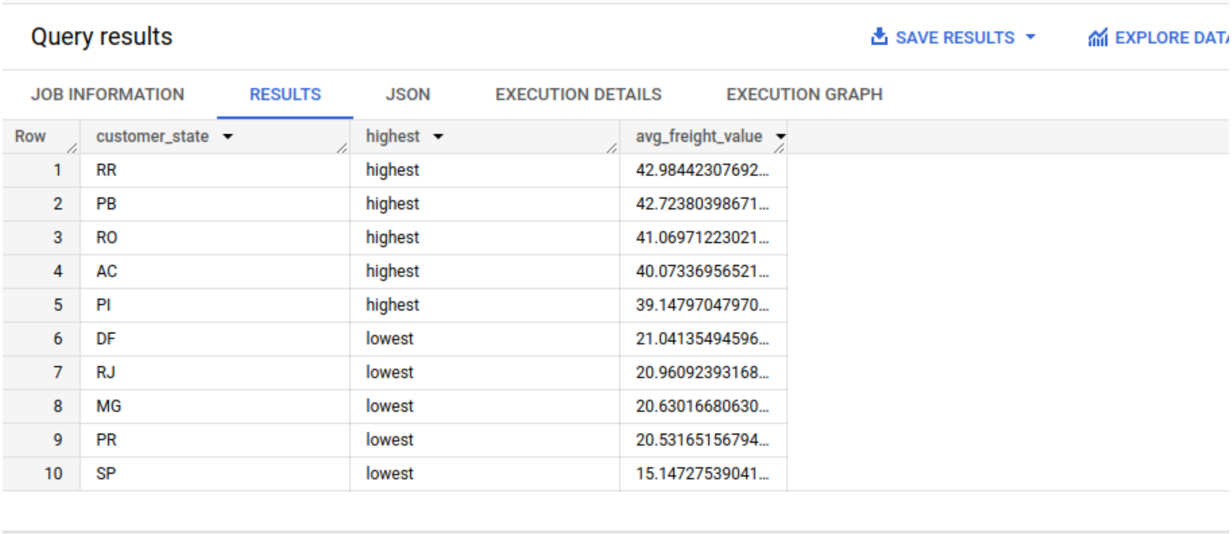
We could have just multiplied by -1 to get the positive output but that wouldn’t have indicated that the delivery date was earlier.

---------------------------------Please navigate to next page-------------------------------

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

Query-

```
with cte as (
SELECT
C.customer_state,
avg(OI.freight_value) as avg_freight_value

FROM `scaler-dsml-sql-390613.TargetSQL.order_items` OI
join
`scaler-dsml-sql-390613.TargetSQL.orders` O
using( order_id )
join
`scaler-dsml-sql-390613.TargetSQL.customers` C
using(customer_id)
group by customer_state
)
(select customer_state,'highest' as highest,avg_freight_value from cte
order by avg_freight_value DESC
limit 5)
UNION ALL
(select customer_state,'lowest' as lowest,avg_freight_value from cte
order by avg_freight_value asc
limit 5)
order by avg_freight_value DESC
```



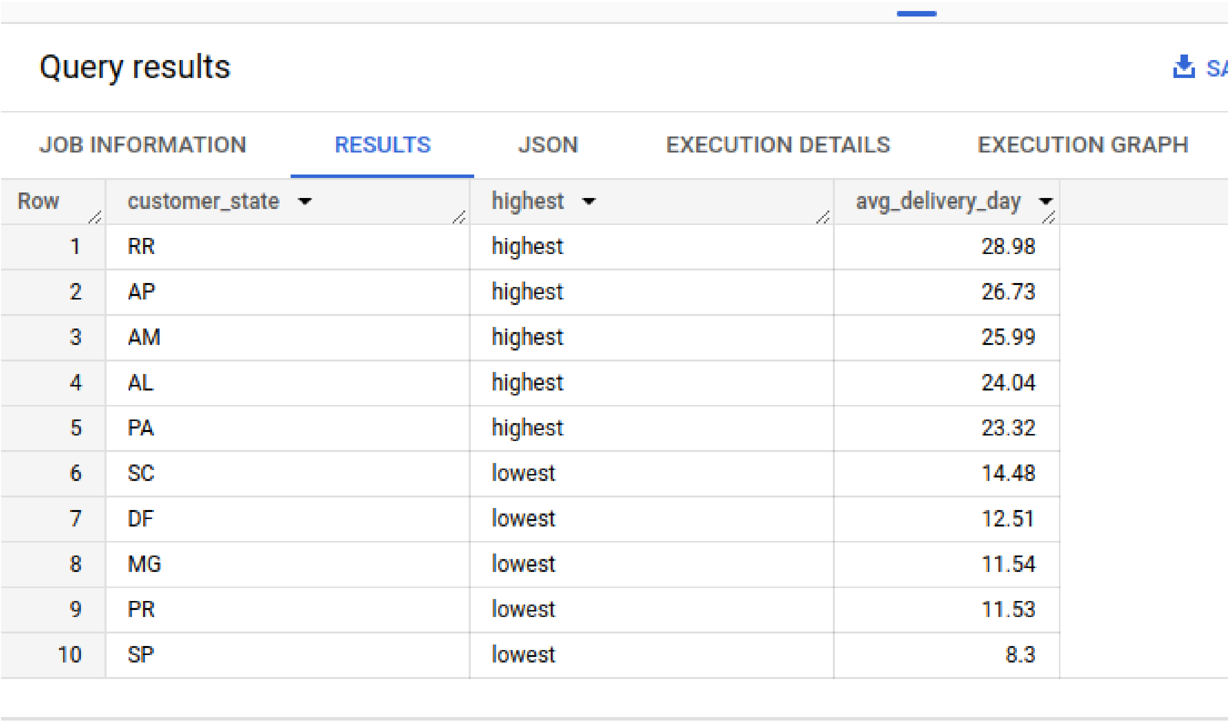
Row	customer_state	highest	avg_freight_value
1	RR	highest	42.98442307692...
2	PB	highest	42.72380398671...
3	RO	highest	41.06971223021...
4	AC	highest	40.07336956521...
5	PI	highest	39.14797047970...
6	DF	lowest	21.04135494596...
7	RJ	lowest	20.96092393168...
8	MG	lowest	20.63016680630...
9	PR	lowest	20.53165156794...
10	SP	lowest	15.14727539041...

--------------------------------Please navigate to next page------------------------------

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

Query -

```
with cte as(
SELECT order_id,customer_id,
TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, day)
AS time_to_deliver
FROM `scaler-dsml-sql-390613.TargetSQL.orders`
where order_delivered_customer_date is not null
)
(
select cu.customer_state,'highest' as highest,
round(avg(ct.time_to_deliver),2) as avg_delivery_day
from cte ct join `scaler-dsml-sql-390613.TargetSQL.customers` cu
using(customer_id)
group by cu.customer_state
order by avg_delivery_day desc
limit 5
)
UNION ALL
(
select cu.customer_state,'lowest' as lowest,
round(avg(ct.time_to_deliver),2) as avg_delivery_day
from cte ct join `scaler-dsml-sql-390613.TargetSQL.customers` cu
using(customer_id)
group by cu.customer_state
order by avg_delivery_day asc
limit 5
)
order by avg_delivery_day desc
```



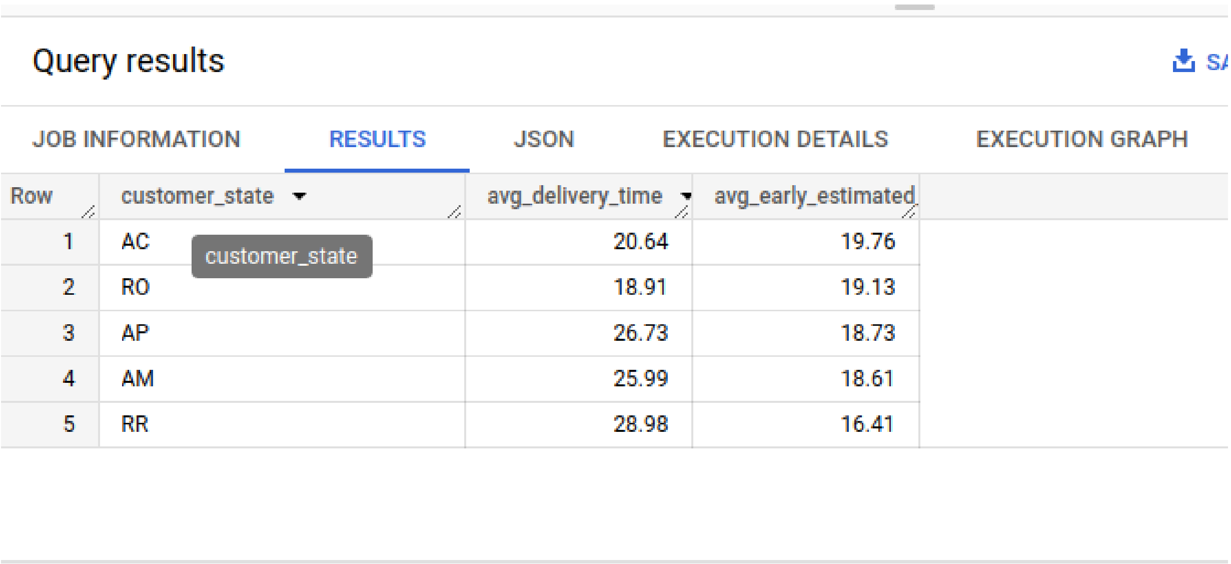
Row	customer_state	highest	avg_delivery_day
1	RR	highest	28.98
2	AP	highest	26.73
3	AM	highest	25.99
4	AL	highest	24.04
5	PA	highest	23.32
6	SC	lowest	14.48
7	DF	lowest	12.51
8	MG	lowest	11.54
9	PR	lowest	11.53
10	SP	lowest	8.3

--------------------------------Please navigate to next page------------------------------

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 -

```
with cte as(
SELECT order_id,customer_id,
TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp,
day) AS time_to_deliver,
TIMESTAMP_DIFF(order_delivered_customer_date,order_estimated_delivery_d
ate,day) AS diff_estimated_delivery
FROM `scaler-dsml-sql-390613.TargetSQL.orders`
where order_delivered_customer_date is not null
)

select cu.customer_state,
round(avg(ct.time_to_deliver),2) as avg_delivery_time,
round(avg(ct.diff_estimated_delivery)*-1,2) as
avg_early_estimated_delivery

from cte ct join `scaler-dsml-sql-390613.TargetSQL.customers` cu
using(customer_id)
group by cu.customer_state
order by avg_early_estimated_delivery desc
limit 5
```



Row	customer_state	avg_delivery_time	avg_early_estimated
1	AC	20.64	19.76
2	RO	18.91	19.13
3	AP	26.73	18.73
4	AM	25.99	18.61
5	RR	28.98	16.41

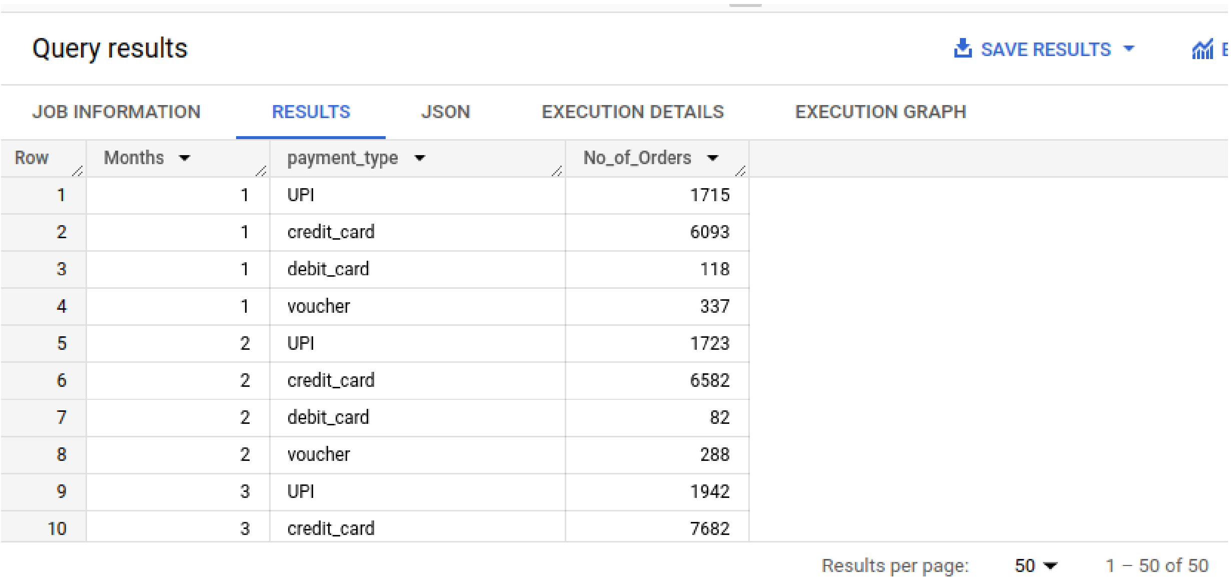
-------------------------------Please navigate to next page-----------------------------

Analysis based on the payments:

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

Query -

```
with CTE as(
SELECT o.order_id,
p.payment_type,
extract(month from order_purchase_timestamp) as Months

FROM `scaler-dsml-sql-390613.TargetSQL.orders` o
join
`scaler-dsml-sql-390613.TargetSQL.payments` p
using(order_id)
order by Months
)
select Months,
payment_type,
count(distinct(order_id)) as No_of_Orders
from CTE
group by CTE.Months,CTE.payment_type
order by Months,payment_type
```



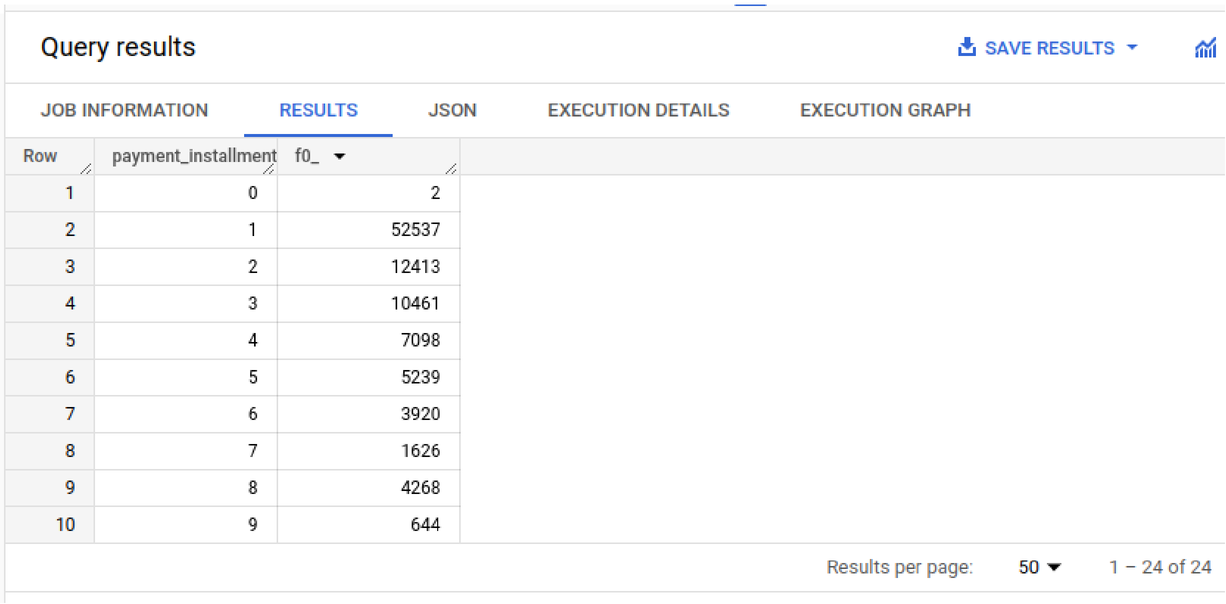
Row	Months	payment_type	No_of_Orders
1	1	UPI	1715
2	1	credit_card	6093
3	1	debit_card	118
4	1	voucher	337
5	2	UPI	1723
6	2	credit_card	6582
7	2	debit_card	82
8	2	voucher	288
9	3	UPI	1942
10	3	credit_card	7682

Results per page: 50 1 – 50 of 50

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

Query -

```
SELECT payment_installments,
       count(order_id)
FROM `scaler-dsml-sql-390613.TargetSQL.payments`
where payment_value is not null and payment_value >0
group by payment_installments
order by payment_installments
```



Row	payment_installment	f0_
1	0	2
2	1	52537
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626
9	8	4268
10	9	644