BUSINESS CASE STUDY_TARGET SQL

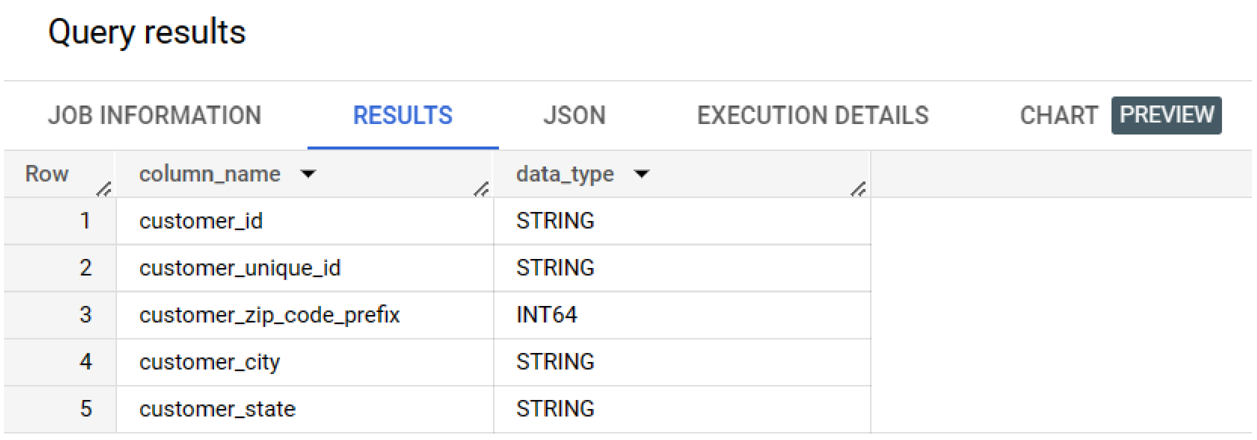
Submitted by: Chanchal Gupta
Date: 03 August 2023
Email ID: chanchalgupta1995@gmail.com

Question1: Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:

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

QUERY:

```
SELECT column_name, data_type
FROM `dsml-31072023.Target_SQL.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'customers';
```



Query results

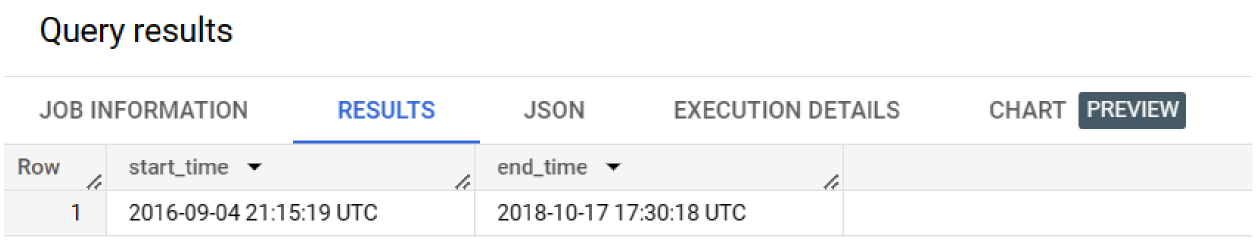
JOB INFORMATION RESULTS JSON EXECUTION DETAILS CHART PREVIEW

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

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

QUERY:

```
select
min(order_purchase_timestamp) as start_time,
max(order_purchase_timestamp) as end_time
from `Target_SQL.orders`;
```



Query results

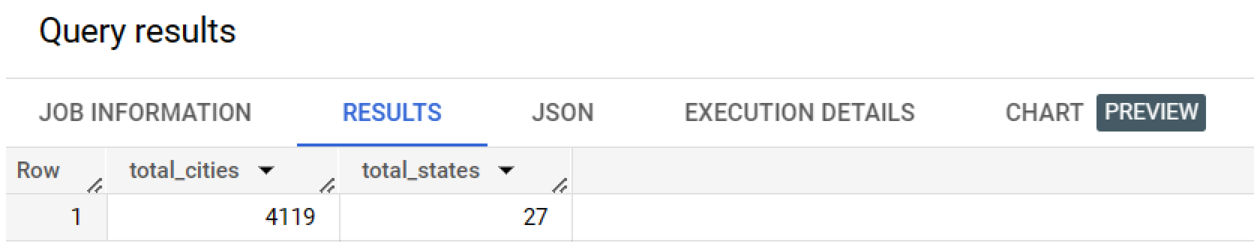
JOB INFORMATION RESULTS JSON EXECUTION DETAILS CHART PREVIEW

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

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

QUERY:

```
Select
count(distinct c.customer_city) as total_cities,
count(distinct c.customer_state) as total_states
from `Target_SQL.orders` o join `Target_SQL.customers` c using (customer_id);
```



Query results

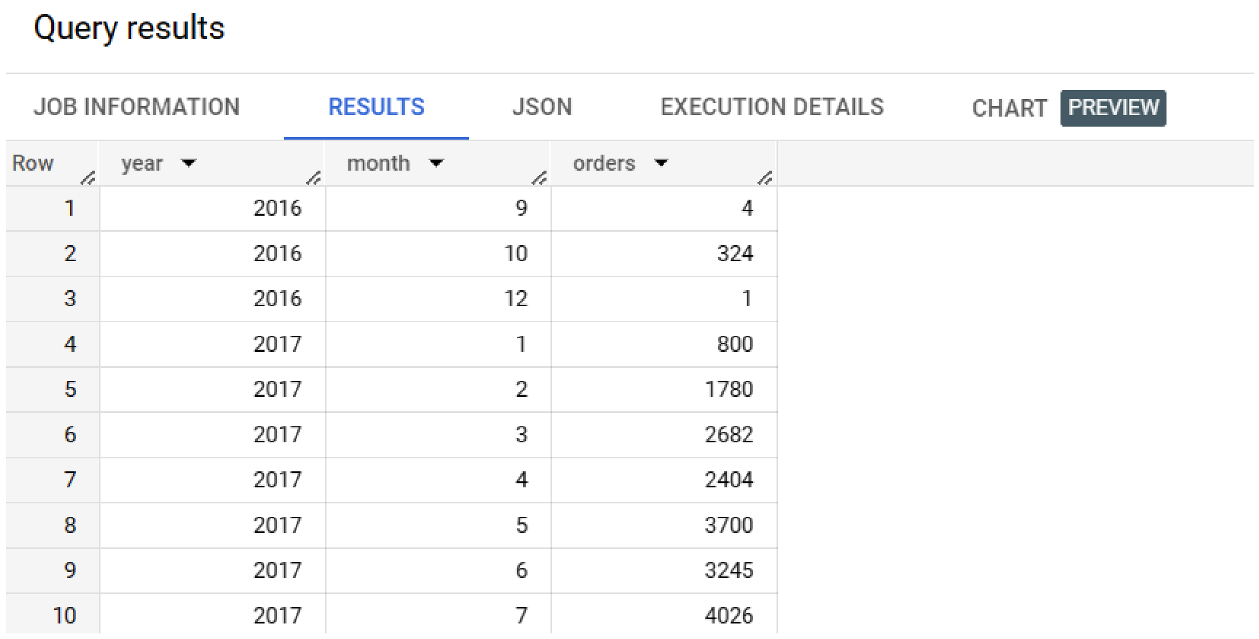
Row	total_cities	total_states
1	4119	27

Question 2: In-depth Exploration:

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

QUERY:

```sql
select extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month,
count(order_id) as orders
from `Target_SQL.orders`
group by 1, 2
order by 1, 2;
```



Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS CHART PREVIEW

Row	year	month	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

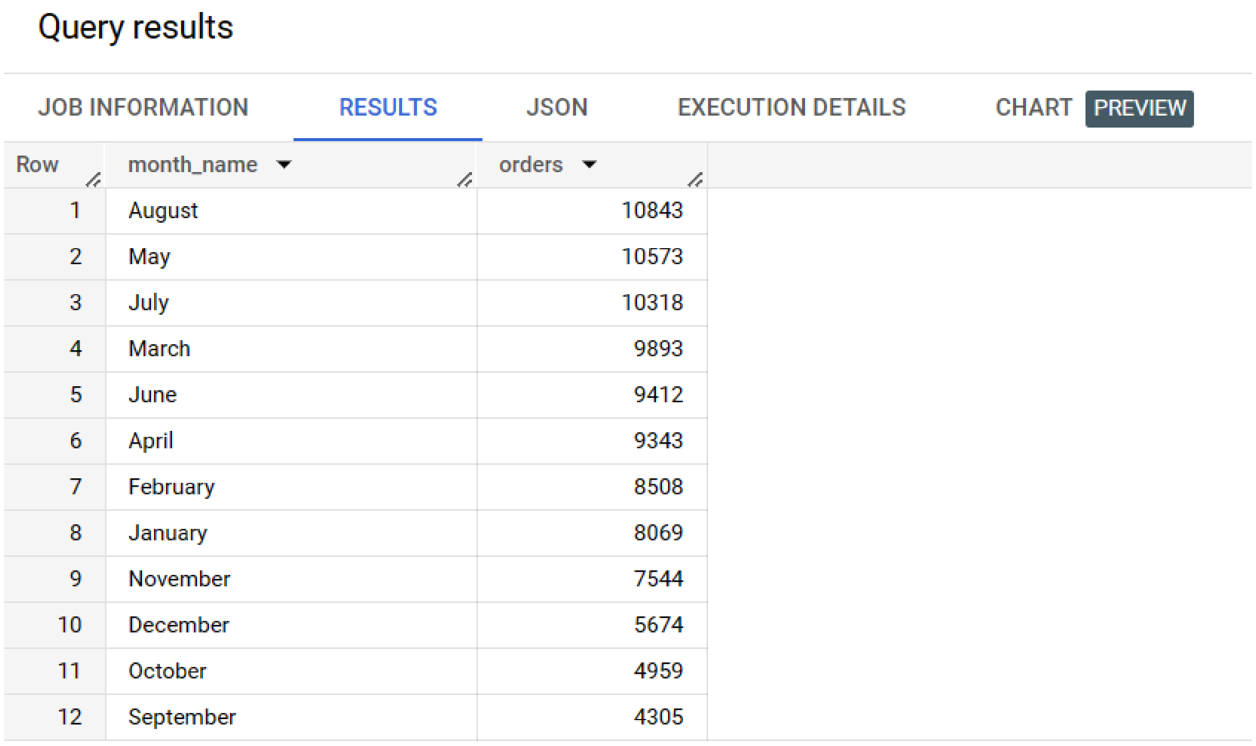
Actionable Insights: There was a growing trend w.r.t time, In the past years, if we look total orders w.r.t years, in year 2016, a total of 329 orders where placed which increased to 45101 orders in 2017 then 54011 orders in 2018

Additionally, We observed a growing trend in orders count throughout the month in 2017 with maximum in November 2017. But there was a sudden downfall in orders count from August 2018 to October 2018.

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

QUERY:

```
select
FORMAT_DATETIME("%B", DATETIME(order_purchase_timestamp)) as month_name,
count(order_id) as orders
from `Target_SQL.orders`
group by 1
order by 2 desc;
```


Query results

Row	month_name	orders
1	August	10843
2	May	10573
3	July	10318
4	March	9893
5	June	9412
6	April	9343
7	February	8508
8	January	8069
9	November	7544
10	December	5674
11	October	4959
12	September	4305

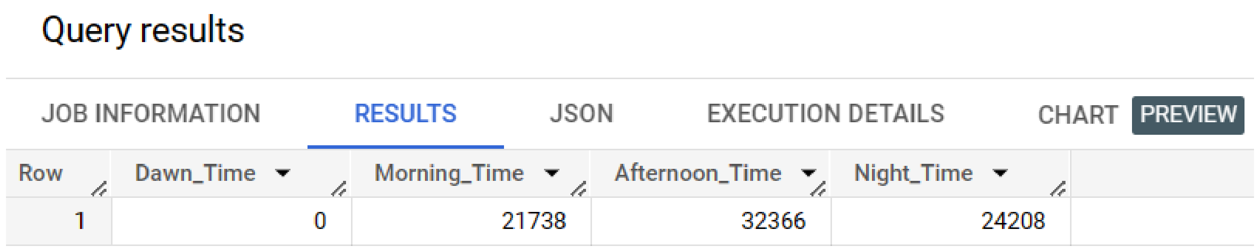
Actionable Insights: We have observed maximum orders in month of August and least in September

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

QUERY:

```
select
countif(t.time_Value>= '12:00:00' AND t.time_Value< '06:00:00') as Dawn_Time,
countif(t.time_Value>= '07:00:00' AND t.time_Value< '12:00:00') as Morning_Time,
countif(t.time_Value>= '13:00:00' AND t.time_Value< '18:00:00') as Afternoon_Time,
countif(t.time_Value>= '19:00:00' AND t.time_Value< '23:00:00') as Night_Time
from (select
extract(time from order_purchase_timestamp) as time_Value
from `Target_SQL.orders`)t
```



Query results

Row	Dawn_Time	Morning_Time	Afternoon_Time	Night_Time
1	0	21738	32366	24208

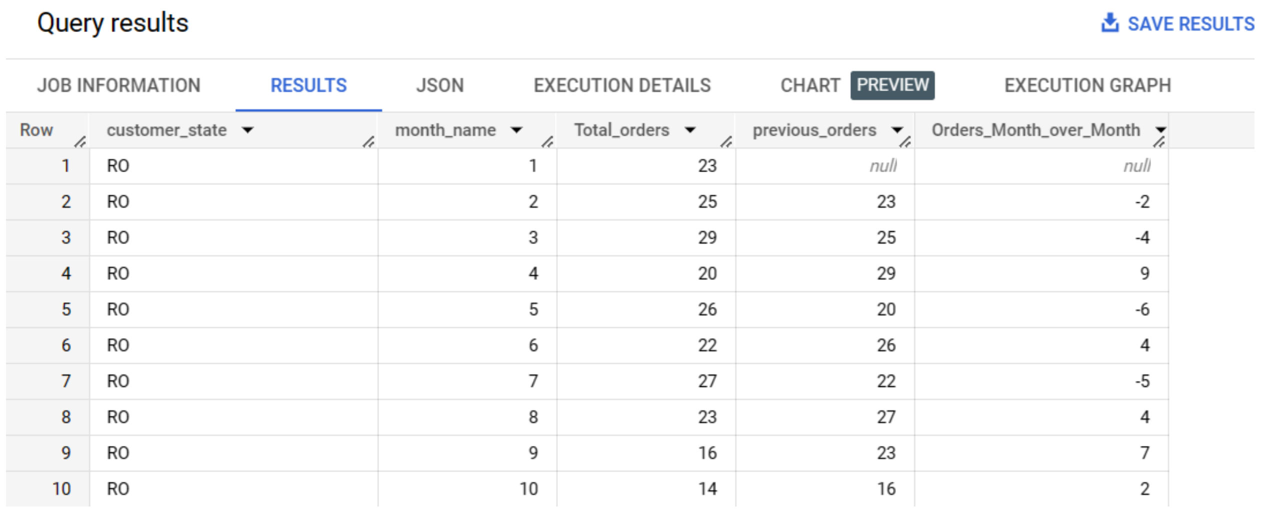
Actionable Insights: Maximum orders were placed during Afternoon timings i.e., from 1pm to 6pm.

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

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

QUERY:

```
SELECT
*,
FL.previous_orders - FL.Total_orders as Orders_Month_over_Month
from
(SELECT
*,
lag(t.Total_orders) over(partition by t.customer_state order by month_name) as
previous_orders
from
(select c.customer_state,
extract(month from o.order_purchase_timestamp) as month_name,
COUNT(o.order_id) as Total_orders
FROM `Target_SQL.customers` c
JOIN `Target_SQL.orders` o ON c.customer_id = o.customer_id
GROUP BY 1, 2) t ) FL
```

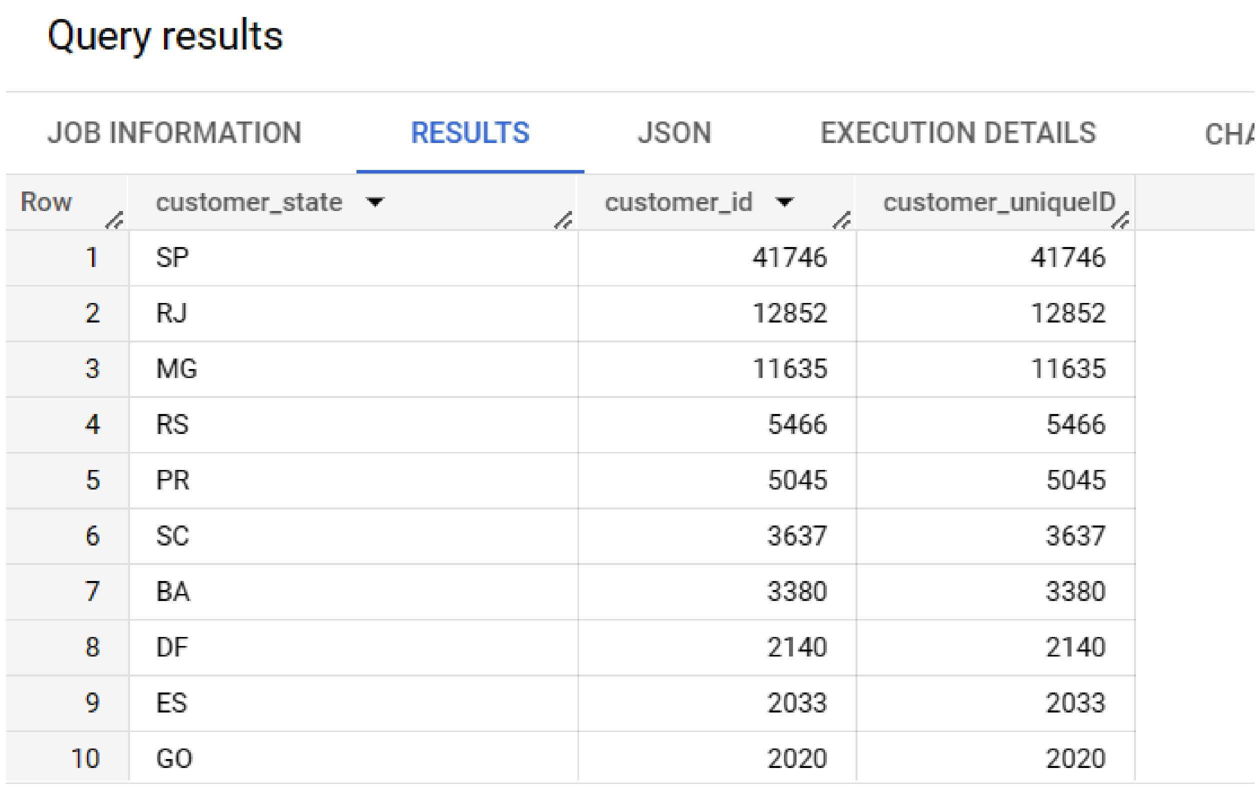


Row	customer_state	month_name	Total_orders	previous_orders	Orders_Month_over_Month
1	RO	1	23	null	null
2	RO	2	25	23	-2
3	RO	3	29	25	-4
4	RO	4	20	29	9
5	RO	5	26	20	-6
6	RO	6	22	26	4
7	RO	7	27	22	-5
8	RO	8	23	27	4
9	RO	9	16	23	7
10	RO	10	14	16	2

3.2 How are the customers distributed across all the states?

QUERY:

```
select customer_state, count(customer_id) as customer_id, count(customer_unique_id) as
customer_uniqueID
from `Target_SQL.customers`
group by customer_state
order by 2 desc, 3 desc
```


Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS CHA

Row	customer_state	customer_id	customer_uniqueID
1	SP	41746	41746
2	RJ	12852	12852
3	MG	11635	11635
4	RS	5466	5466
5	PR	5045	5045
6	SC	3637	3637
7	BA	3380	3380
8	DF	2140	2140
9	ES	2033	2033
10	GO	2020	2020

Actionable Insights: Customers are majorly distributed in SP i.e, Sau Paulo one of the most populous state in Brazil

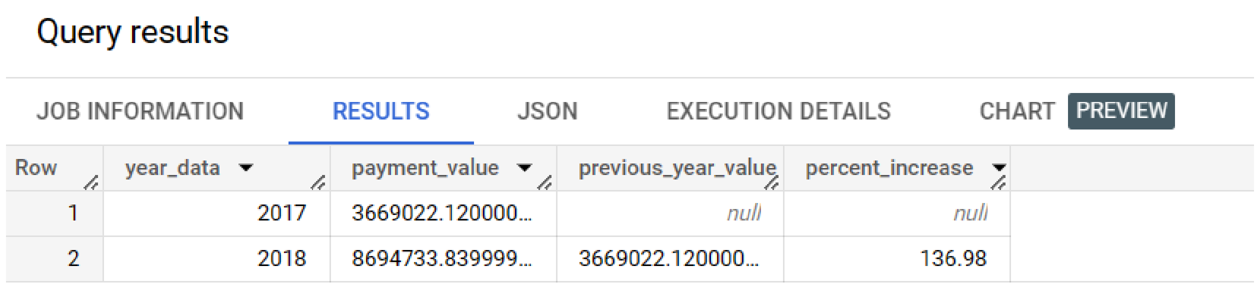
Question 4: Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

4.1 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:

```
select *,
round(((FL.payment_value - FL.previous_year_value)/FL.previous_year_value)*100, 2) as
percent_increase
from
(SELECT
t.year_data,
t.payment_value,
lag(t.payment_value) over(order by t.year_data) as previous_year_value
from
(SELECT
extract(year from order_purchase_timestamp) as year_data,
sum(p.payment_value) as payment_value
from `Target_SQL.payments` p join `Target_SQL.orders` o
on p.order_id = o.order_id
where (extract(month from order_purchase_timestamp) between 1 and 8) and
(extract(year from order_purchase_timestamp) between 2017 and 2018)
group by 1) t) FL
```

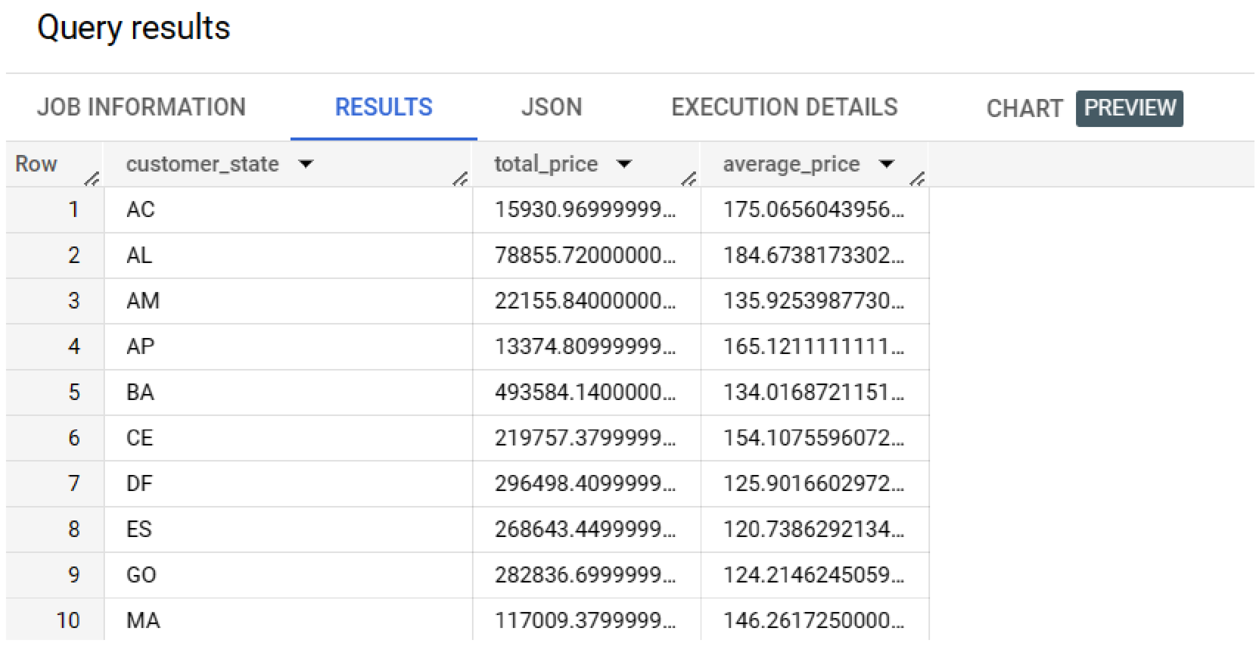


Row	year_data	payment_value	previous_year_value	percent_increase
1	2017	3669022.120000...	null	null
2	2018	8694733.839999...	3669022.120000...	136.98

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

QUERY:

```
select c.customer_state, sum(oi.price) as total_price, avg(oi.price) as average_price
from `Target_SQL.customers` c JOIN `Target_SQL.orders` o
on o.customer_id = c.customer_id JOIN `Target_SQL.order_items` oi on o.order_id =
oi.order_id
where o.order_status = 'delivered'
group by 1
order by 1
```



Query results

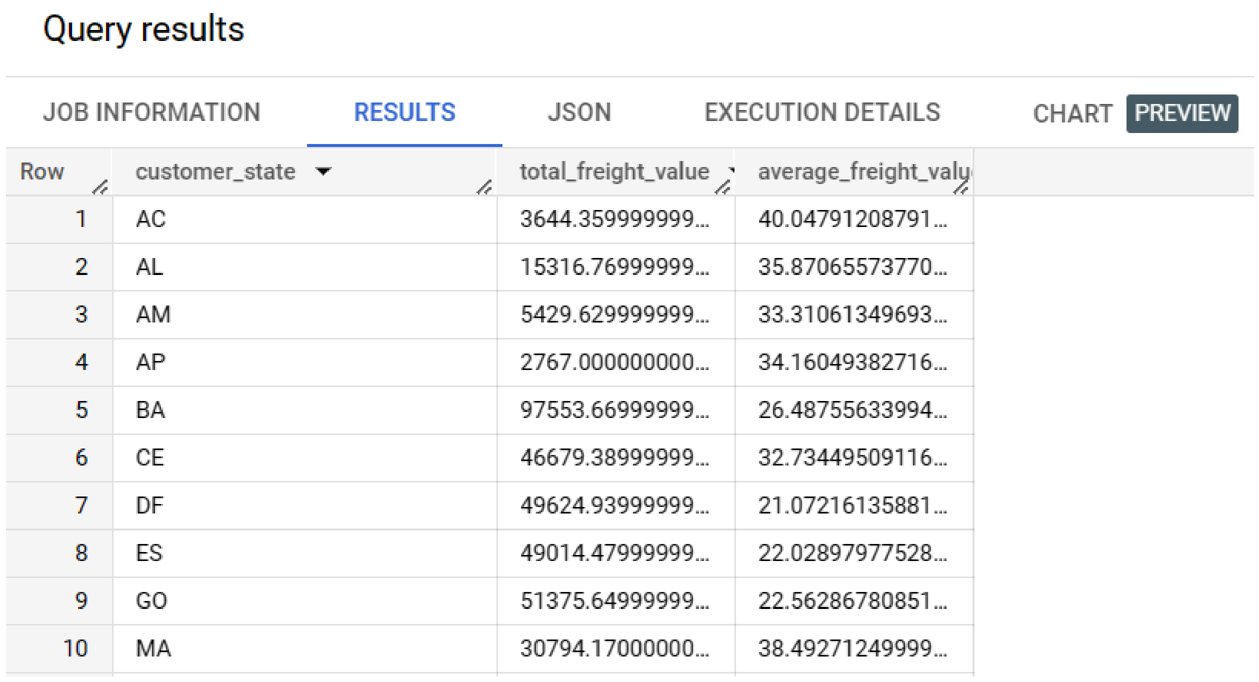
JOB INFORMATION RESULTS JSON EXECUTION DETAILS CHART PREVIEW

Row	customer_state	total_price	average_price
1	AC	15930.96999999...	175.0656043956...
2	AL	78855.72000000...	184.6738173302...
3	AM	22155.84000000...	135.9253987730...
4	AP	13374.80999999...	165.1211111111...
5	BA	493584.1400000...	134.0168721151...
6	CE	219757.3799999...	154.1075596072...
7	DF	296498.4099999...	125.9016602972...
8	ES	268643.4499999...	120.7386292134...
9	GO	282836.6999999...	124.2146245059...
10	MA	117009.3799999...	146.2617250000...

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

QUERY:

```
select c.customer_state, sum(oi.freight_value) as total_freight_value,
avg(oi.freight_value) as average_freight_value
from `Target_SQL.customers` c JOIN `Target_SQL.orders` o
on o.customer_id = c.customer_id JOIN `Target_SQL.order_items` oi on o.order_id =
oi.order_id
where o.order_status = 'delivered'
group by 1
order by 1
```



Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS CHART PREVIEW

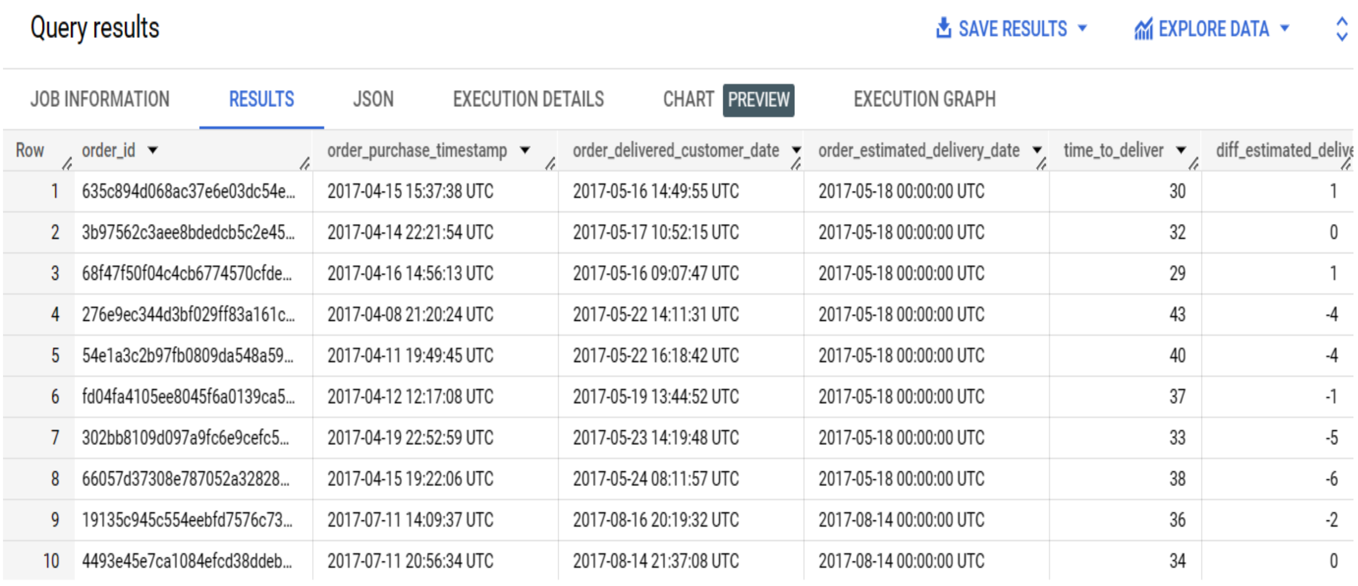
Row	customer_state	total_freight_value	average_freight_valu
1	AC	3644.359999999...	40.04791208791...
2	AL	15316.76999999...	35.87065573770...
3	AM	5429.629999999...	33.31061349693...
4	AP	2767.000000000...	34.16049382716...
5	BA	97553.66999999...	26.48755633994...
6	CE	46679.38999999...	32.73449509116...
7	DF	49624.93999999...	21.07216135881...
8	ES	49014.47999999...	22.02897977528...
9	GO	51375.64999999...	22.56286780851...
10	MA	30794.17000000...	38.49271249999...

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

5.1 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.

QUERY:

```
select
order_id,
order_purchase_timestamp,
order_delivered_customer_date,
order_estimated_delivery_date,
date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as
time_to_deliver,
date_diff(order_estimated_delivery_date,order_delivered_customer_date,day) as
diff_estimated_delivery
from `Target_SQL.orders`
where order_status = 'delivered'
```



Query results

SAVE RESULTS EXPLORE DATA

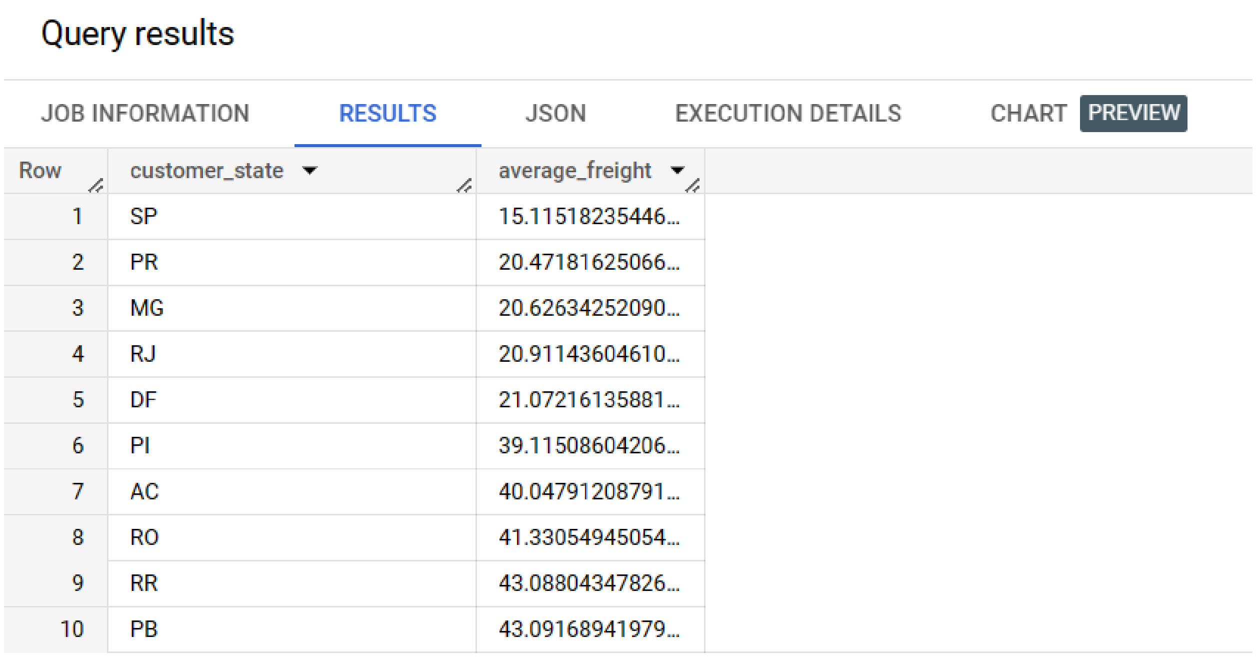
JOB INFORMATION RESULTS JSON EXECUTION DETAILS CHART PREVIEW EXECUTION GRAPH

Row	order_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_delive
1	635c894d068ac37e6e03dc54e...	2017-04-15 15:37:38 UTC	2017-05-16 14:49:55 UTC	2017-05-18 00:00:00 UTC	30	1
2	3b97562c3aee8bdedcb5c2e45...	2017-04-14 22:21:54 UTC	2017-05-17 10:52:15 UTC	2017-05-18 00:00:00 UTC	32	0
3	68f47f50f04c4cb6774570cfde...	2017-04-16 14:56:13 UTC	2017-05-16 09:07:47 UTC	2017-05-18 00:00:00 UTC	29	1
4	276e9ec344d3bf029ff83a161c...	2017-04-08 21:20:24 UTC	2017-05-22 14:11:31 UTC	2017-05-18 00:00:00 UTC	43	-4
5	54e1a3c2b97fb0809da548a59...	2017-04-11 19:49:45 UTC	2017-05-22 16:18:42 UTC	2017-05-18 00:00:00 UTC	40	-4
6	fd04fa4105ee8045f6a0139ca5...	2017-04-12 12:17:08 UTC	2017-05-19 13:44:52 UTC	2017-05-18 00:00:00 UTC	37	-1
7	302bb8109d097a9fc6e9cefc5...	2017-04-19 22:52:59 UTC	2017-05-23 14:19:48 UTC	2017-05-18 00:00:00 UTC	33	-5
8	66057d37308e787052a32828...	2017-04-15 19:22:06 UTC	2017-05-24 08:11:57 UTC	2017-05-18 00:00:00 UTC	38	-6
9	19135c945c554eebfd7576c73...	2017-07-11 14:09:37 UTC	2017-08-16 20:19:32 UTC	2017-08-14 00:00:00 UTC	36	-2
10	4493e45e7ca1084efcd38ddeb...	2017-07-11 20:56:34 UTC	2017-08-14 21:37:08 UTC	2017-08-14 00:00:00 UTC	34	0

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

QUERY:

```
select t.customer_state, t.average_freight from
(select c.customer_state, avg(oi.freight_value) as average_freight,
dense_rank() over(order by avg(oi.freight_value)) as rank_numb_aesc,
dense_rank() over(order by avg(oi.freight_value) desc) as rank_numb_desc
from `Target_SQL.customers` c JOIN `Target_SQL.orders` o
on o.customer_id = c.customer_id
JOIN `Target_SQL.order_items` oi
on o.order_id = oi.order_id
where o.order_status = 'delivered'
group by 1
order by 2) t
where (t.rank_numb_aesc<6) or (t.rank_numb_desc<6)
```

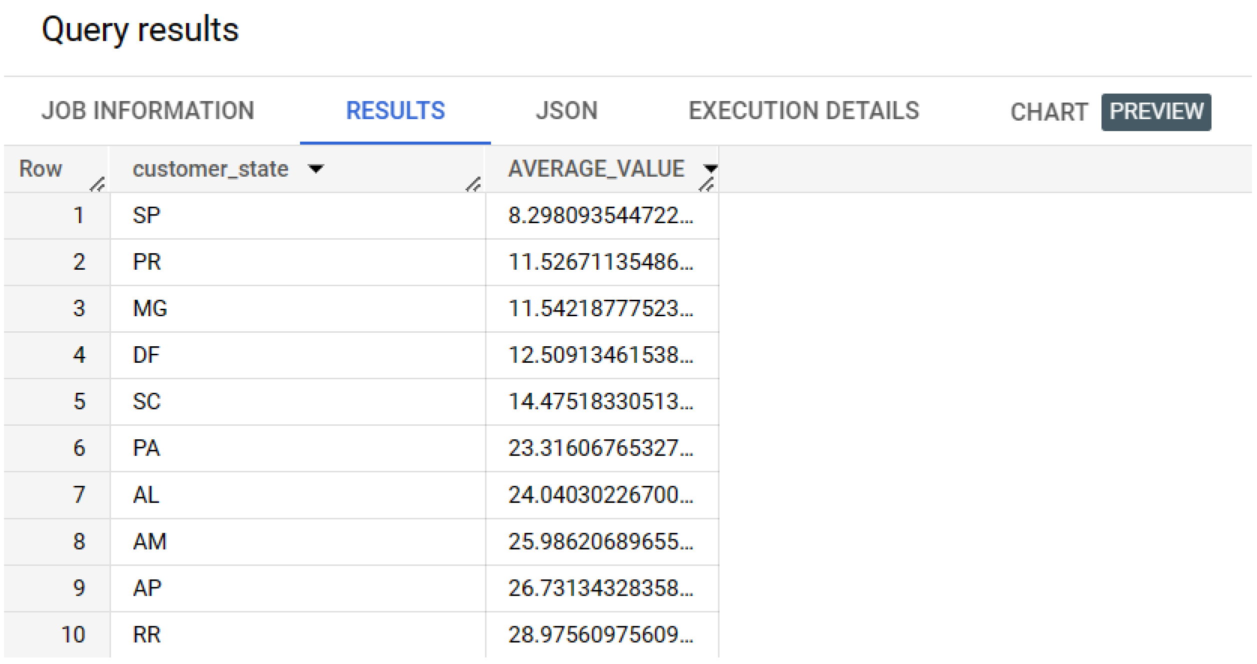


Row	customer_state	average_freight
1	SP	15.11518235446...
2	PR	20.47181625066...
3	MG	20.62634252090...
4	RJ	20.91143604610...
5	DF	21.07216135881...
6	PI	39.11508604206...
7	AC	40.04791208791...
8	RO	41.33054945054...
9	RR	43.08804347826...
10	PB	43.09168941979...

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

QUERY:

```
select t.customer_state, t.AVERAGE_VALUE from(
select
c.customer_state,
AVG(o.time_to_deliver) as AVERAGE_VALUE,
dense_rank() over(order by AVG(o.time_to_deliver)) as avg_aesc,
dense_rank() over(order by AVG(o.time_to_deliver) desc) as avg_desc
from
(select
*,
date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as
time_to_deliver,
from `Target_SQL.orders`
where order_status = 'delivered')
o join `Target_SQL.customers` c
on o.customer_id = c.customer_id
group by 1
order by 2) t
where t.avg_aesc< 6 or t.avg_desc<6
```


Query results

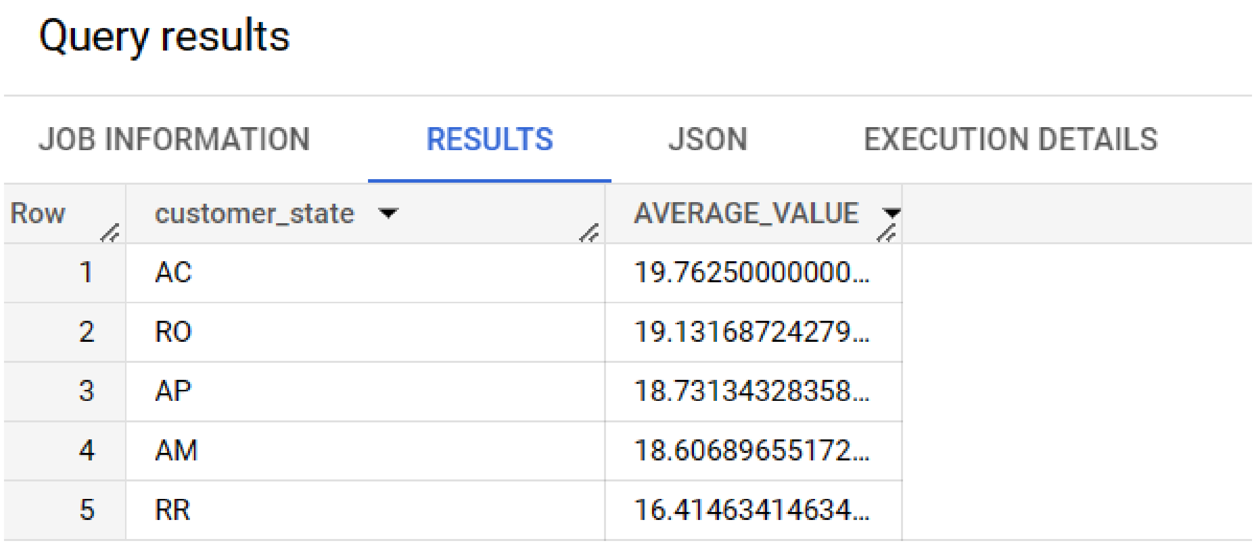
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART PREVIEW

Row	customer_state	AVERAGE_VALUE
1	SP	8.298093544722...
2	PR	11.52671135486...
3	MG	11.54218777523...
4	DF	12.50913461538...
5	SC	14.47518330513...
6	PA	23.31606765327...
7	AL	24.04030226700...
8	AM	25.98620689655...
9	AP	26.73134328358...
10	RR	28.97560975609...

5.4 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
c.customer_state,
AVG(o.delivery_time) as AVERAGE_VALUE
from
(select
*,
date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as
delivery_time,
from `Target_SQL.orders`
where order_status = 'delivered') o
join `Target_SQL.customers` c
on o.customer_id = c.customer_id
group by 1
order by 2 desc
limit 5;
```



Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

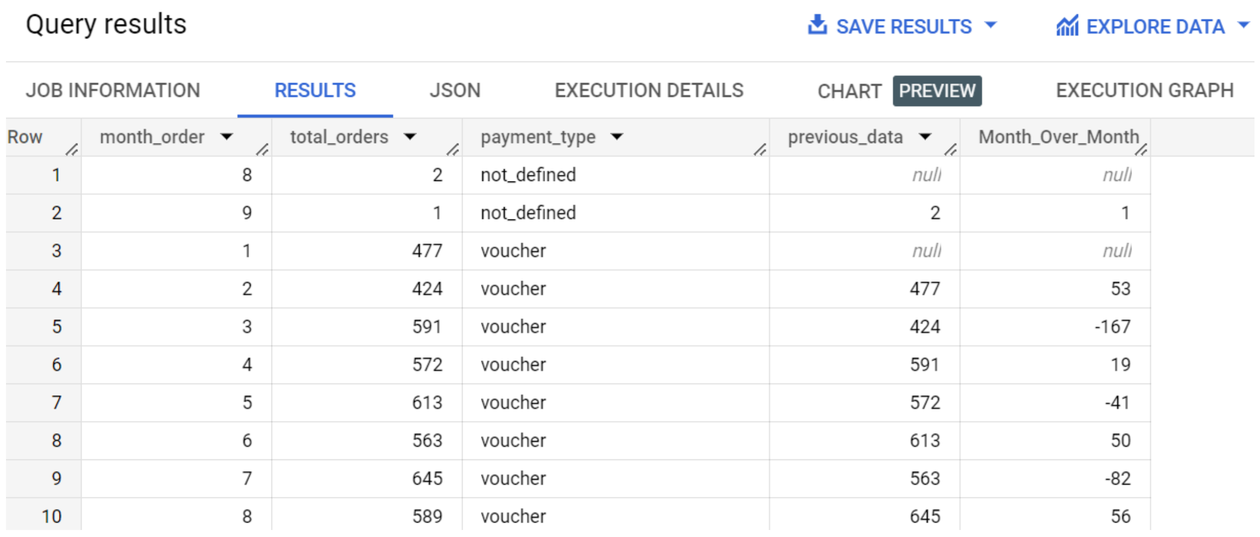
Row	customer_state	AVERAGE_VALUE
1	AC	19.76250000000...
2	RO	19.13168724279...
3	AP	18.73134328358...
4	AM	18.60689655172...
5	RR	16.41463414634...

QUESTION 6: Analysis based on the payments:

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

QUERY:

```
select
*, (FL.previous_data-FL.total_orders) as Month_Over_Month
from
(select
*,
lag(t.total_orders) over(partition by t.payment_type order by t.month_order) as
previous_data
from
(select
extract(month from o.order_purchase_timestamp) as month_order,
count(o.order_id) as total_orders,
p.payment_type
from `Target_SQL.orders` o join `Target_SQL.payments` p
on o.order_id = p.order_id
group by 1, 3) t )FL
```



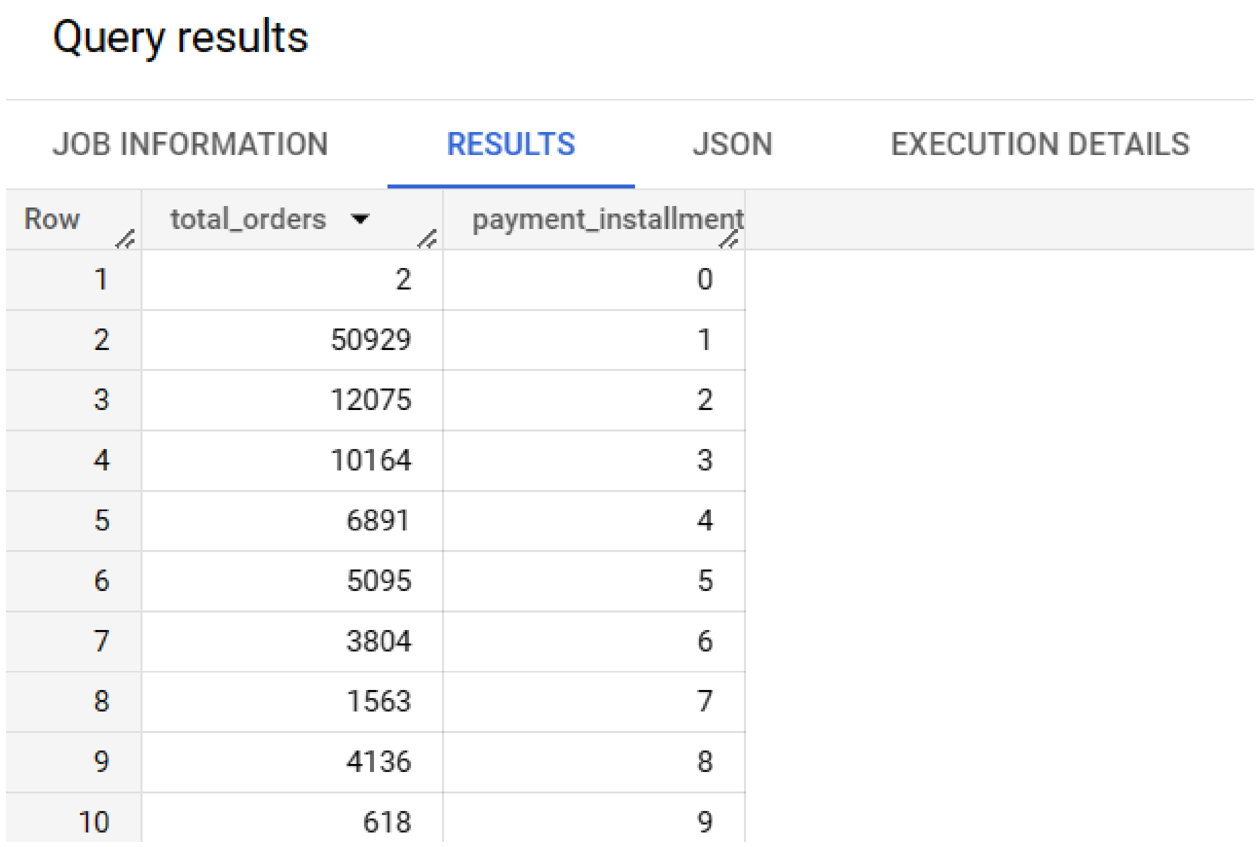
Row	month_order	total_orders	payment_type	previous_data	Month_Over_Month
1	8	2	not_defined	null	null
2	9	1	not_defined	2	1
3	1	477	voucher	null	null
4	2	424	voucher	477	53
5	3	591	voucher	424	-167
6	4	572	voucher	591	19
7	5	613	voucher	572	-41
8	6	563	voucher	613	50
9	7	645	voucher	563	-82
10	8	589	voucher	645	56

Actionable Insights: Maximum orders were made through online mode.

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

QUERY:

```
select
count(o.order_id) as total_orders,
p.payment_installments
from `Target_SQL.orders` o join `Target_SQL.payments` p
on o.order_id = p.order_id
where o.order_status = 'delivered'
group by 2
order by 2
```



Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	total_orders	payment_installment
1	2	0
2	50929	1
3	12075	2
4	10164	3
5	6891	4
6	5095	5
7	3804	6
8	1563	7
9	4136	8
10	618	9

Actionable Insights: Maximum orders were placed in 1 installments