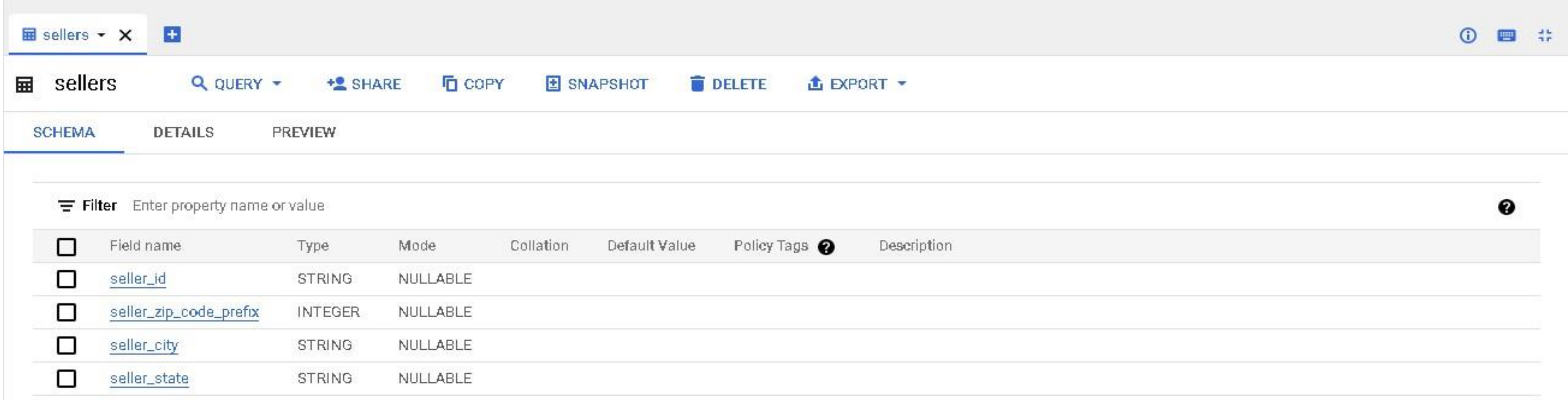
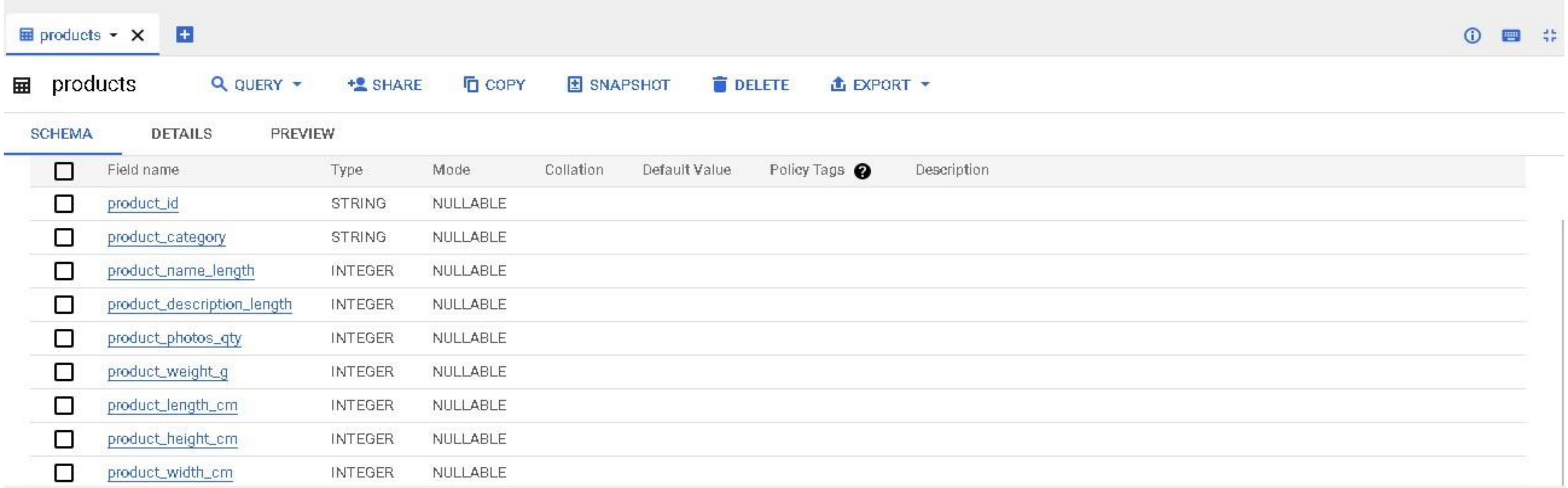
TARGET SQL PROJECT

Q1. Import the dataset and do usual exploratory analysis steps like checking the structure and characteristics of the dataset:

1. Data type of columns in a table

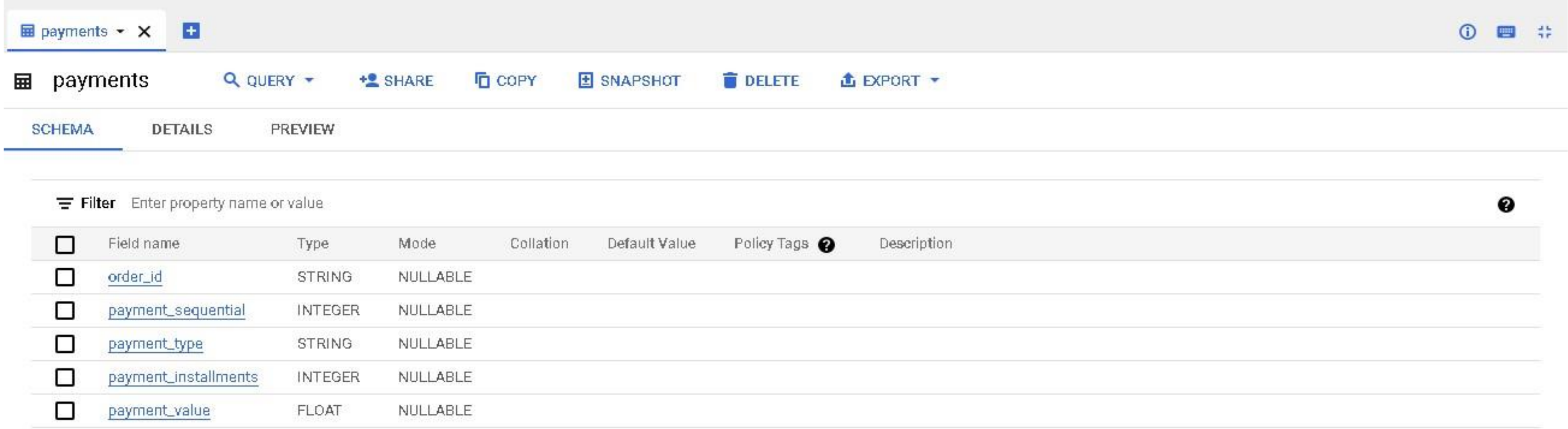


□	Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
□	seller_id	STRING	NULLABLE				
□	seller_zip_code_prefix	INTEGER	NULLABLE				
□	seller_city	STRING	NULLABLE				
□	seller_state	STRING	NULLABLE				



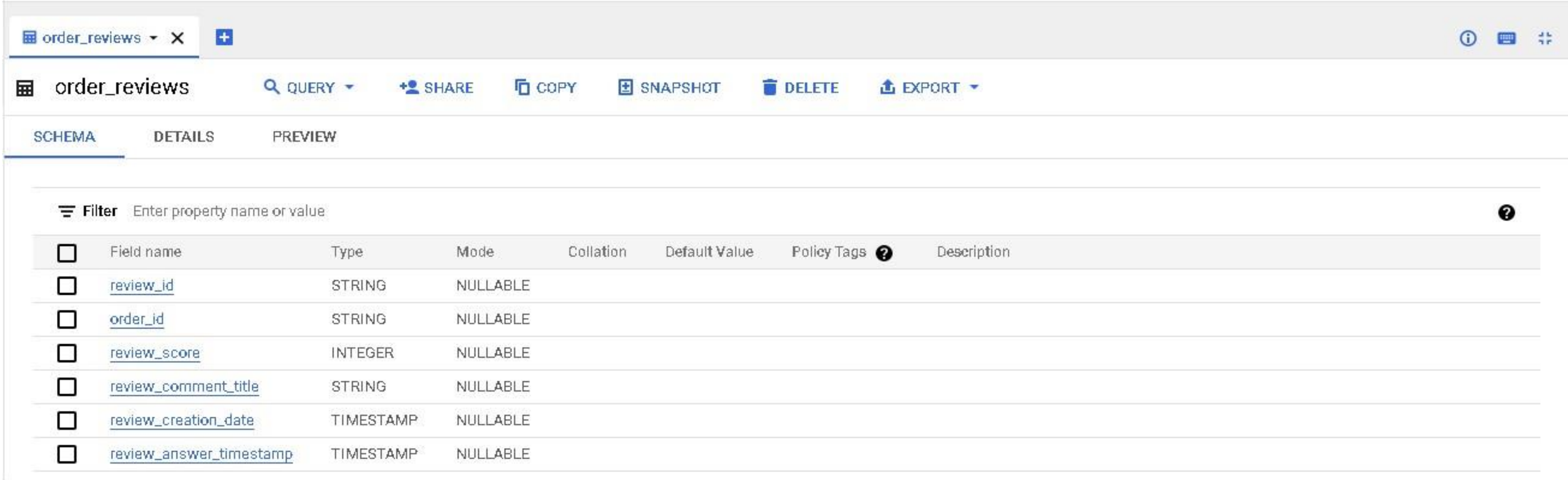
products

□	Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
□	product_id	STRING	NULLABLE				
□	product_category	STRING	NULLABLE				
□	product_name_length	INTEGER	NULLABLE				
□	product_description_length	INTEGER	NULLABLE				
□	product_photos_qty	INTEGER	NULLABLE				
□	product_weight_g	INTEGER	NULLABLE				
□	product_length_cm	INTEGER	NULLABLE				
□	product_height_cm	INTEGER	NULLABLE				
□	product_width_cm	INTEGER	NULLABLE				



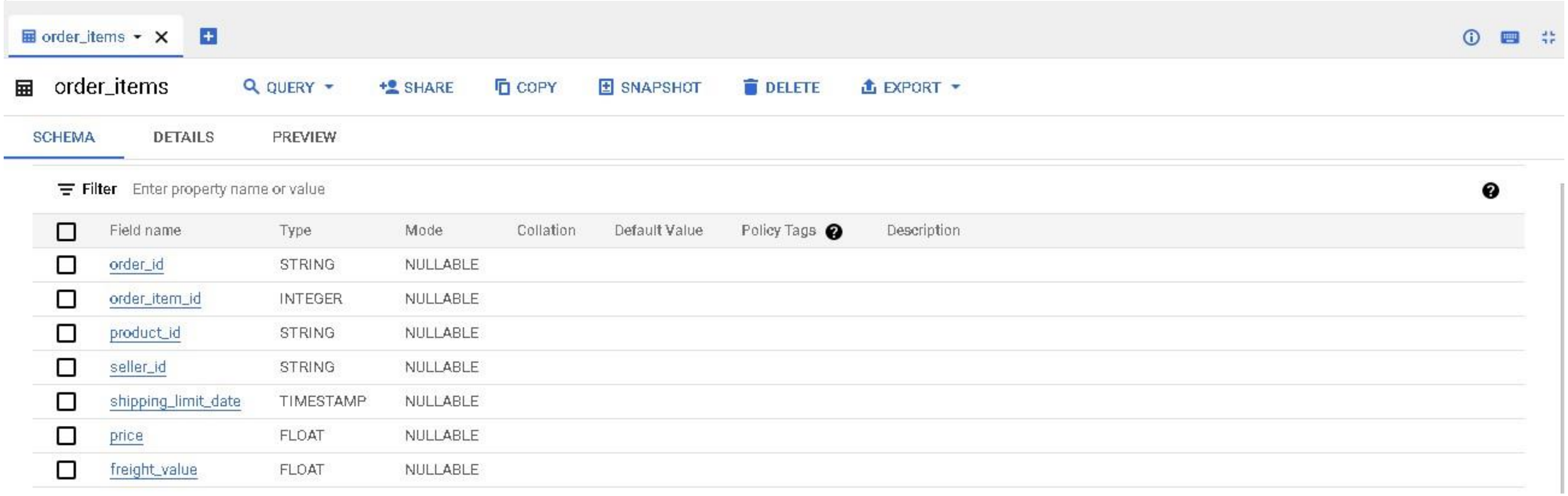
payments

□	Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
□	order_id	STRING	NULLABLE				
□	payment_sequential	INTEGER	NULLABLE				
□	payment_type	STRING	NULLABLE				
□	payment_installments	INTEGER	NULLABLE				
□	payment_value	FLOAT	NULLABLE				



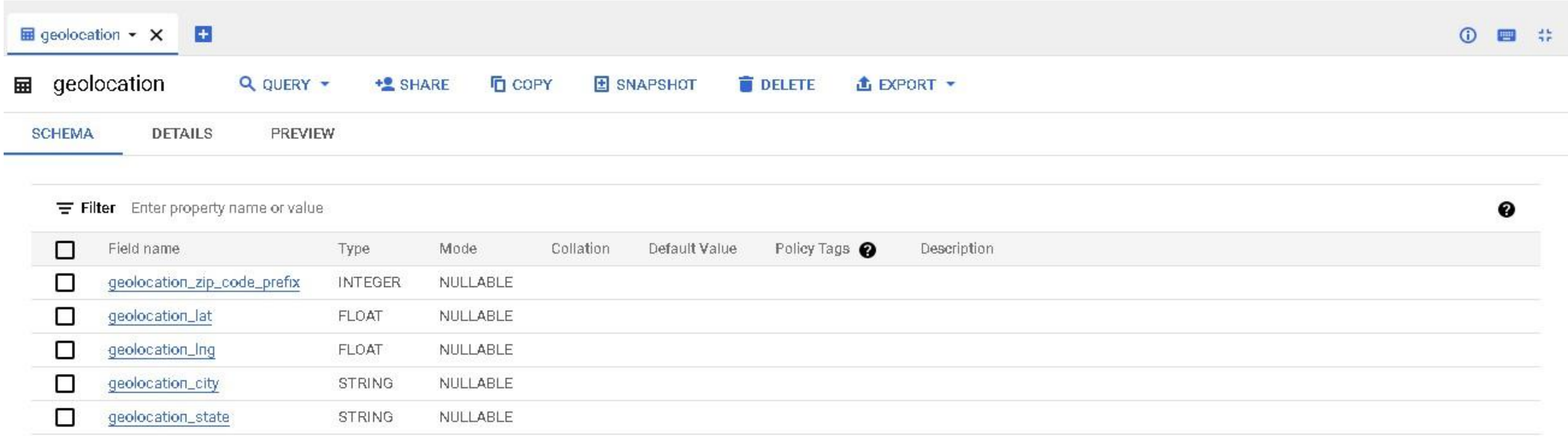
order_reviews

□	Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
□	review_id	STRING	NULLABLE				
□	order_id	STRING	NULLABLE				
□	review_score	INTEGER	NULLABLE				
□	review_comment_title	STRING	NULLABLE				
□	review_creation_date	TIMESTAMP	NULLABLE				
□	review_answer_timestamp	TIMESTAMP	NULLABLE				



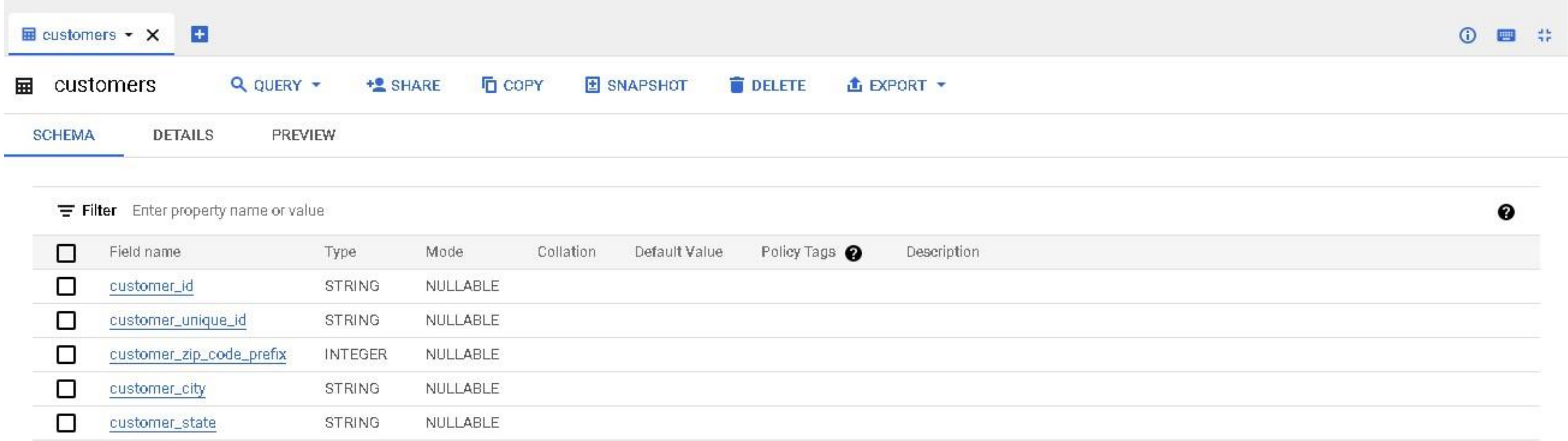
order_items

□	Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
□	order_id	STRING	NULLABLE				
□	order_item_id	INTEGER	NULLABLE				
□	product_id	STRING	NULLABLE				
□	seller_id	STRING	NULLABLE				
□	shipping_limit_date	TIMESTAMP	NULLABLE				
□	price	FLOAT	NULLABLE				
□	freight_value	FLOAT	NULLABLE				



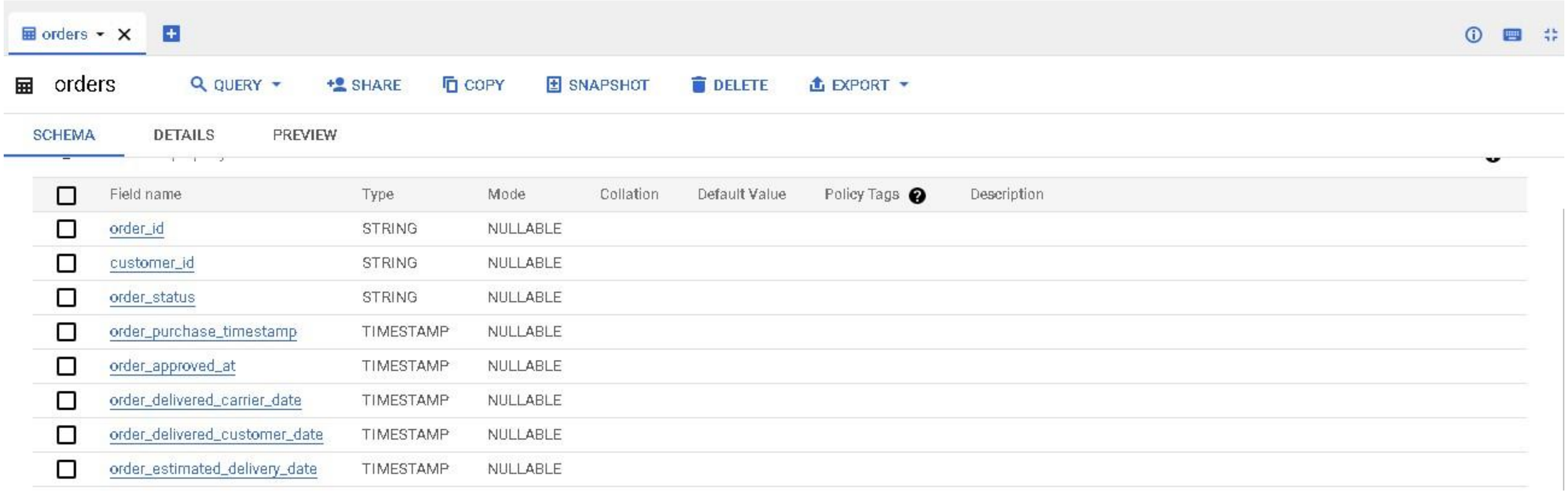
geolocation

☐	Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
☐	geolocation_zip_code_prefix	INTEGER	NULLABLE				
☐	geolocation_lat	FLOAT	NULLABLE				
☐	geolocation_lng	FLOAT	NULLABLE				
☐	geolocation_city	STRING	NULLABLE				
☐	geolocation_state	STRING	NULLABLE				



customers

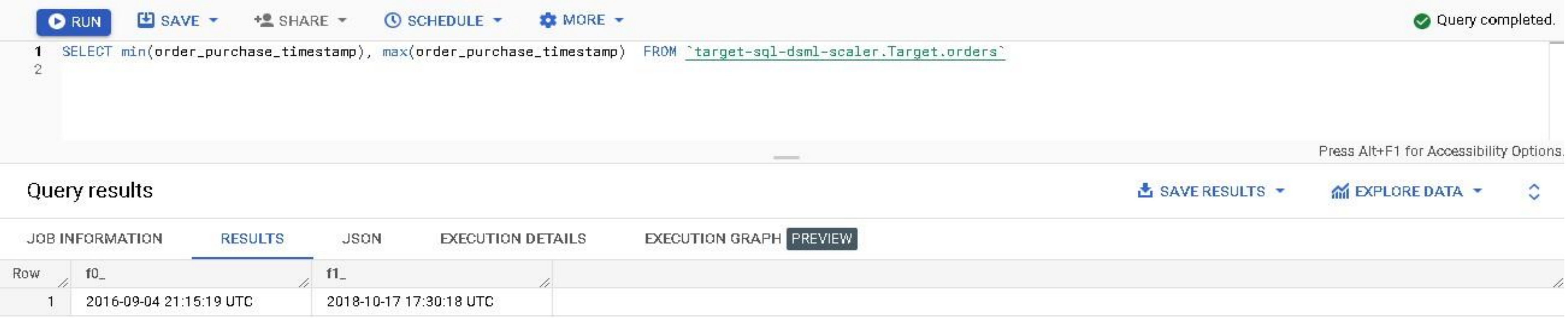
☐	Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
☐	customer_id	STRING	NULLABLE				
☐	customer_unique_id	STRING	NULLABLE				
☐	customer_zip_code_prefix	INTEGER	NULLABLE				
☐	customer_city	STRING	NULLABLE				
☐	customer_state	STRING	NULLABLE				



□	Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
□	order_id	STRING	NULLABLE				
□	customer_id	STRING	NULLABLE				
□	order_status	STRING	NULLABLE				
□	order_purchase_timestamp	TIMESTAMP	NULLABLE				
□	order_approved_at	TIMESTAMP	NULLABLE				
□	order_delivered_carrier_date	TIMESTAMP	NULLABLE				
□	order_delivered_customer_date	TIMESTAMP	NULLABLE				
□	order_estimated_delivery_date	TIMESTAMP	NULLABLE				

2. Time period for which the data is given

```
SELECT min(order_purchase_timestamp), max(order_purchase_timestamp)
FROM `target-sql-dsml-scaler.Target.orders`
```



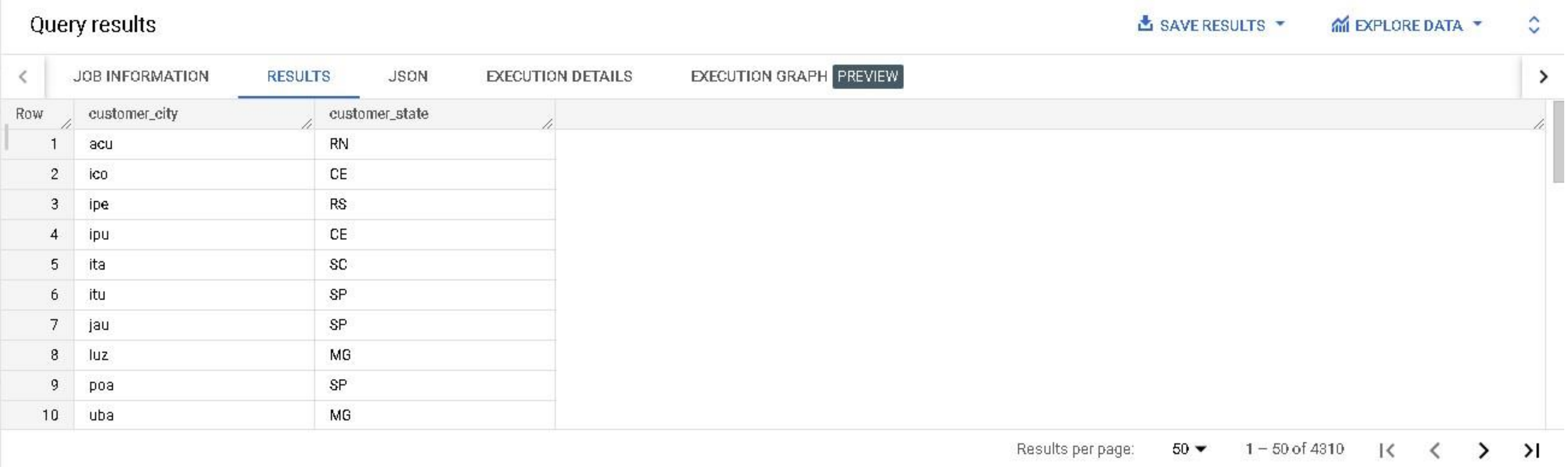
```
SELECT min(order_purchase_timestamp), max(order_purchase_timestamp)  FROM `target-sql-dsml-scaler.Target.orders`
```

Row	f0_	f1_
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

The time period of the data is from 4th September 2016 to 17th October 2018.

3. Cities and States of customers ordered during the given period.

```
SELECT distinct customer_city, customer_state
FROM `target-sql-dsml-scaler.Target.customers`
```



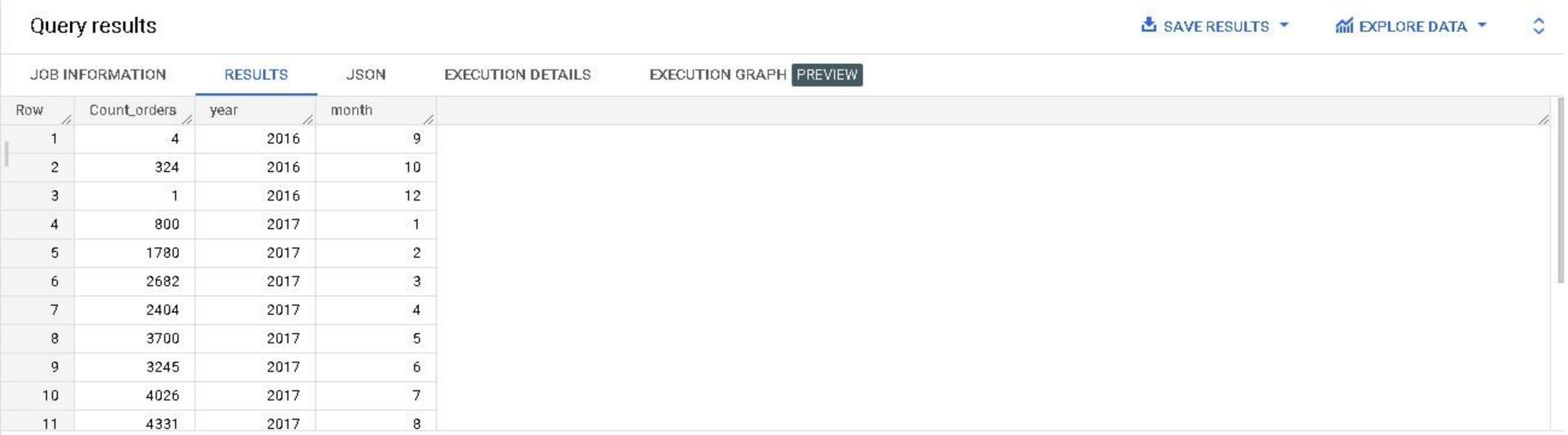
Row	customer_city	customer_state
1	acu	RN
2	ico	CE
3	ipe	RS
4	ipu	CE
5	ita	SC
6	itu	SP
7	jau	SP
8	luz	MG
9	poa	SP
10	uba	MG

Results per page: 50 1 – 50 of 4310

Q2. In-depth Exploration:

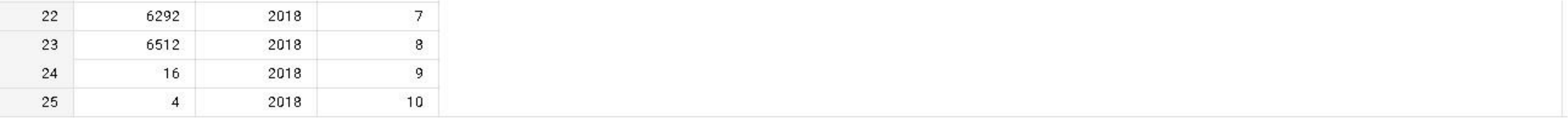
1. Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?

```
SELECT count(order_id) as Count_orders, extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month  from `target-sql-dsml-scaler.Target.orders`
GROUP BY month, year
ORDER BY year, month asc
```



Row	Count_orders	year	month
1	4	2016	9
2	324	2016	10
3	1	2016	12
4	800	2017	1
5	1780	2017	2
6	2682	2017	3
7	2404	2017	4
8	3700	2017	5
9	3245	2017	6
10	4026	2017	7
11	4331	2017	8

12	4285	2017	9
13	4631	2017	10
14	7544	2017	11
15	5673	2017	12
16	7269	2018	1
17	6728	2018	2
18	7211	2018	3
19	6939	2018	4
20	6873	2018	5
21	6167	2018	6



22	6292	2018	7
23	6512	2018	8
24	16	2018	9
25	4	2018	10

OBSERVATION:

After analysing the data, it can be conferred that there was a constant increase in customer orders throughout the time period of the data before there was a huge downfall in the number of orders in the month of September and October in 2018. Hence, we can conclude that there is a growing trend of e-commerce in Brazil.

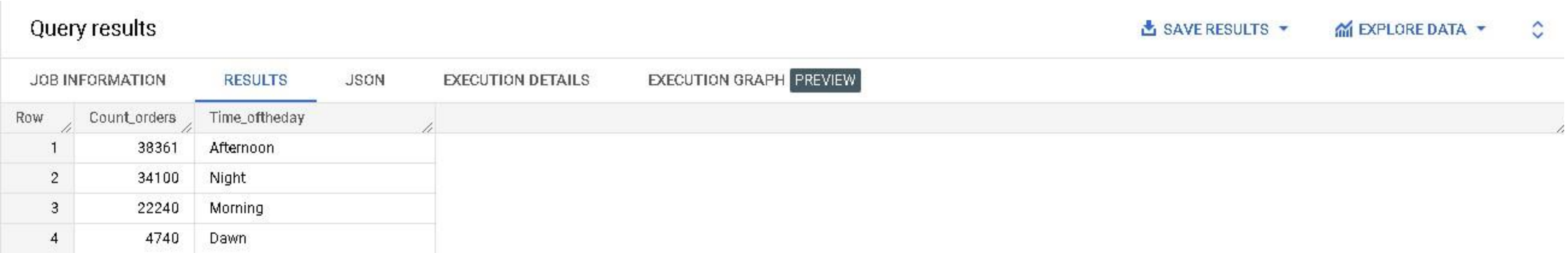
There was a peak in the number of orders in November 2017 with 7544 orders. The number of orders again came close to the peak in January 2018 with 7269 orders. This could be due to the festive season and discounts offered during that time.

To summarize, we can say that even though the number of orders were low during the initial 2-3 months, there was a gradual rise in the number of orders in the months that followed with the highest number of orders in November 2017 and the lowest number of orders in December 2016.

2. What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?

```
With A as
(SELECT order_id, extract (time from order_purchase_timestamp) as time_day
FROM `Target.orders`)

SELECT count(order_id) as Count_orders,
CASE
    WHEN time_day between '00:00:00' and '05:59:59' THEN 'Dawn'
    WHEN time_day between '06:00:00' and '11:59:59' THEN 'Morning'
    WHEN time_day between '12:00:00' and '17:59:59' THEN 'Afternoon'
    WHEN time_day between '18:00:00' and '23:59:59' THEN 'Night'
END AS Time_oftheday
FROM A
GROUP BY Time_oftheday
ORDER BY Count_orders DESC
```



Row	Count_orders	Time_oftheday
1	38361	Afternoon
2	34100	Night
3	22240	Morning
4	4740	Dawn

OBSERVATION:

The day has been divided into four equal intervals for convenience and categorised as follows:

Dawn: 00:00:00 to 05:59:59 hrs

Morning: 06:00:00 to 11:59:59 hrs

Afternoon: 12:00:00 to 17:59:59 hrs

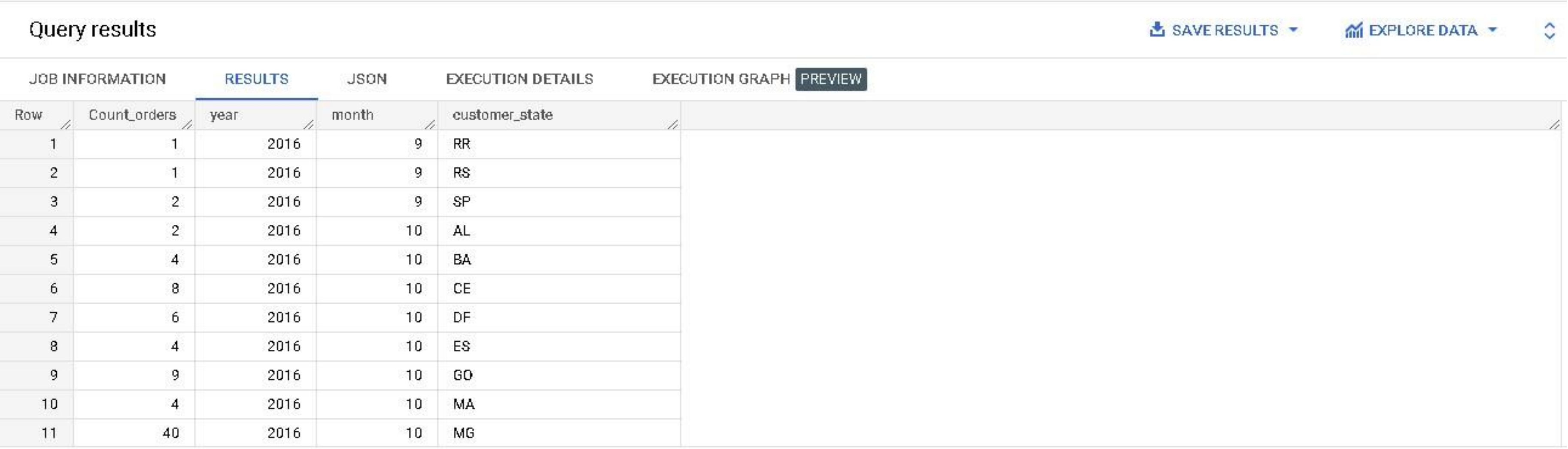
Night: 18:00:00 to 23:59:59 hrs

It can be observed from the query results that the highest number of orders are made during afternoon. Hence, the Brazilian customers tend to buy more during afternoon.

Q3. Evolution of E-commerce orders in the Brazil region:

1. Get month on month orders by states

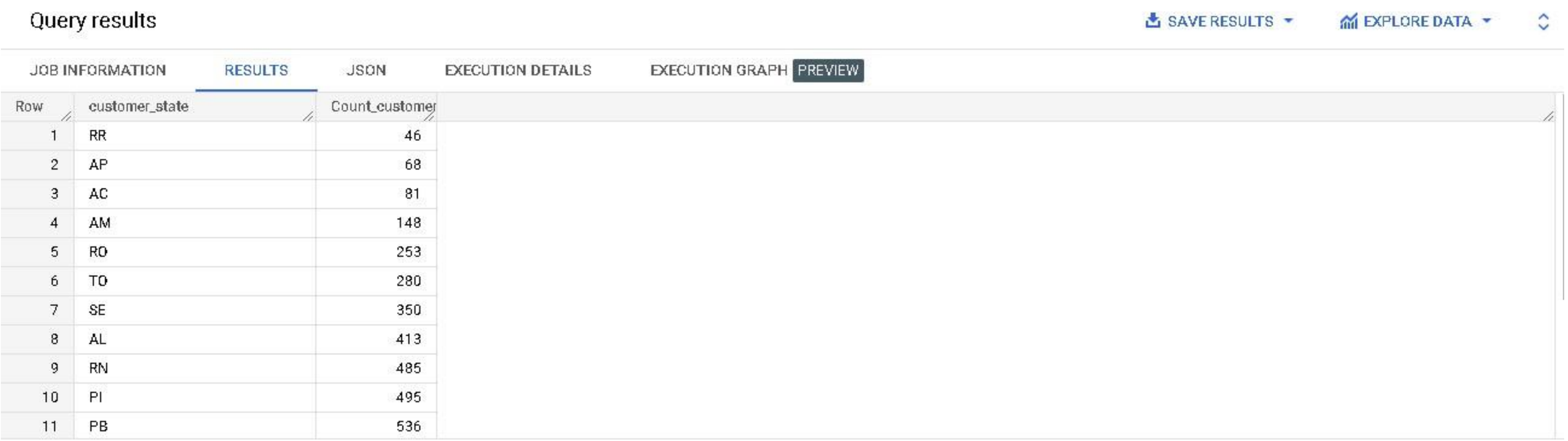
```
SELECT count(o.order_id) as Count_orders, extract(year from o.order_purchase_timestamp) as year,
extract(month from o.order_purchase_timestamp) as month, c.customer_state
from `target-sql-dsml-scaler.Target.orders` as o
JOIN `target-sql-dsml-scaler.Target.customers` as c
ON o.customer_id = c.customer_id
GROUP BY month, year, c.customer_state
ORDER BY year, month, c.customer_state asc
```



Row	Count_orders	year	month	customer_state
1	1	2016	9	RR
2	1	2016	9	RS
3	2	2016	9	SP
4	2	2016	10	AL
5	4	2016	10	BA
6	8	2016	10	CE
7	6	2016	10	DF
8	4	2016	10	ES
9	9	2016	10	GO
10	4	2016	10	MA
11	40	2016	10	MG

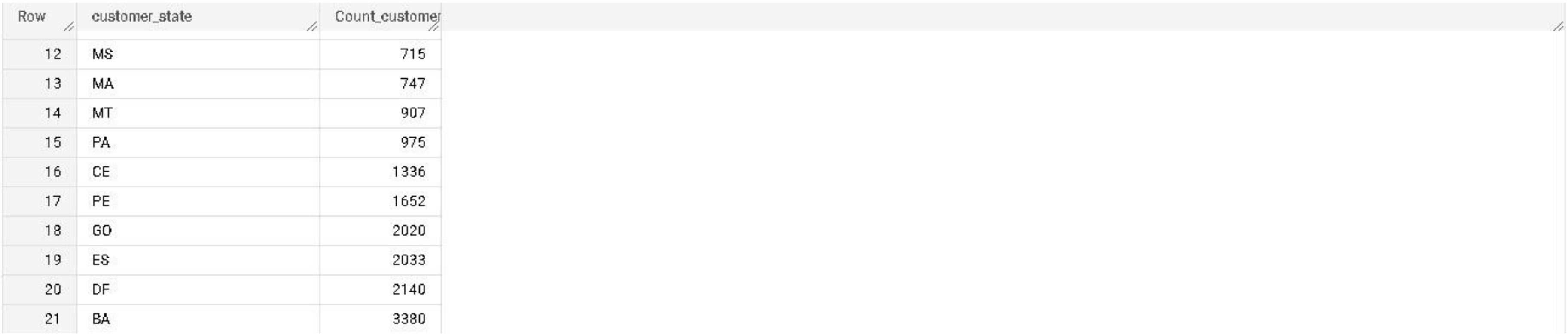
2. Distribution of customers across the states in Brazil

```
SELECT customer_state, count(customer_id) as Count_customers,
FROM `Target.customers`
GROUP BY customer_state
ORDER BY Count_customers asc
```

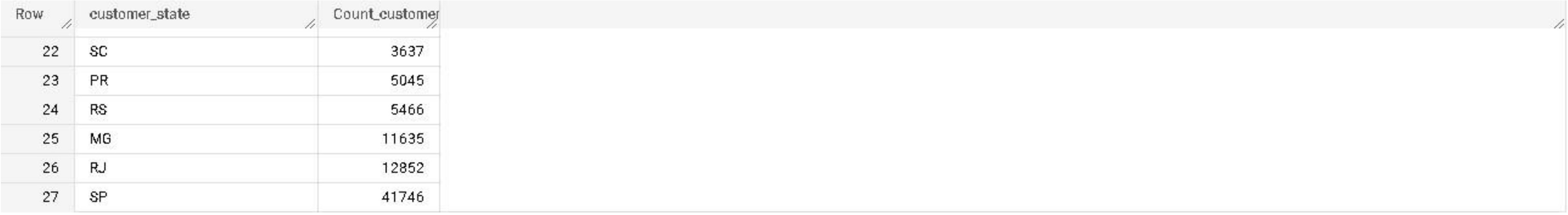


Query results

Row	customer_state	Count_customer
1	RR	46
2	AP	68
3	AC	81
4	AM	148
5	RO	253
6	TO	280
7	SE	350
8	AL	413
9	RN	485
10	PI	495
11	PB	536



Row	customer_state	Count_customer
12	MS	715
13	MA	747
14	MT	907
15	PA	975
16	CE	1336
17	PE	1652
18	GO	2020
19	ES	2033
20	DF	2140
21	BA	3380



Row	customer_state	Count_customer
22	SC	3637
23	PR	5045
24	RS	5466
25	MG	11635
26	RJ	12852
27	SP	41746

OBSERVATION:

It can be observed from query results that the minimum number of customers are from the State of Roraima (RR) i.e. 46 and the maximum number of customers are from the state of State of Sao Paulo (SP) i.e. 41746. The reason for variance in the number of customers could be the population of that particular state of Brazil.

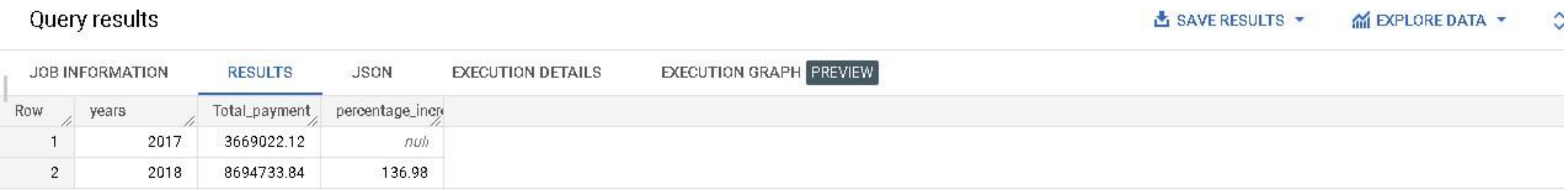
Q4. Impact on Economy: Analyse the money movement by e-commerce by looking at order prices, freight and others.

1. Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use “payment_value” column in payments table

```
With A as
(select round(sum(total_payment),2) as Total_payment,years
from
(select round(sum(p.payment_value),2) as total_payment,
extract( month from o.order_purchase_timestamp) as months,
extract( year from o.order_purchase_timestamp) as years,
from `Target.payments` as p
inner join `Target.orders` as o
on p.order_id = o.order_id
group by months, years, o.order_purchase_timestamp
having years>2016 and months <9
order by years, months)
group by years),

B as
(SELECT Total_payment, years,
lag(Total_payment) over (order by years) as previous_year_payments
FROM A)

SELECT years, Total_payment, round(((Total_payment - previous_year_payments)/previous_year_payments * 100), 2) as percentage_increase
FROM B
```



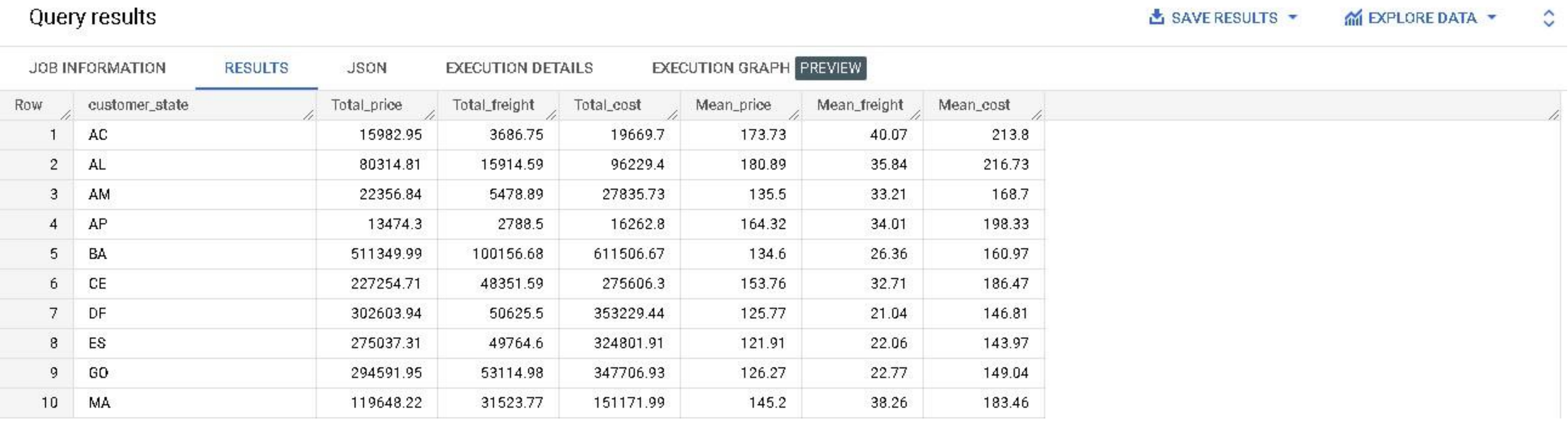
Row	years	Total_payment	percentage_incre
1	2017	3669022.12	null
2	2018	8694733.84	136.98

OBSERVATION:

The percentage increase in orders from 2017 to 2018 (including months from January to August) is 136.98%.

2. Mean and Sum of price and freight value by customer state

```
SELECT c.customer_state, round(sum(oi.price),2) as Total_price, round(sum(oi.freight_value),2) as Total_freight, round(sum(oi.price)+
sum(oi.freight_value),2) as Total_cost,round(avg(oi.price),2) as Mean_price, round(avg(oi.freight_value),2) as Mean_freight,
round(avg(oi.price)+avg(oi.freight_value),2) as Mean_cost
FROM `Target.customers` c
JOIN `Target.orders` o
ON c.customer_id = o.customer_id
JOIN `Target.order_items` oi
ON o.order_id = oi.order_id
GROUP BY c.customer_state
ORDER BY c.customer_state
```

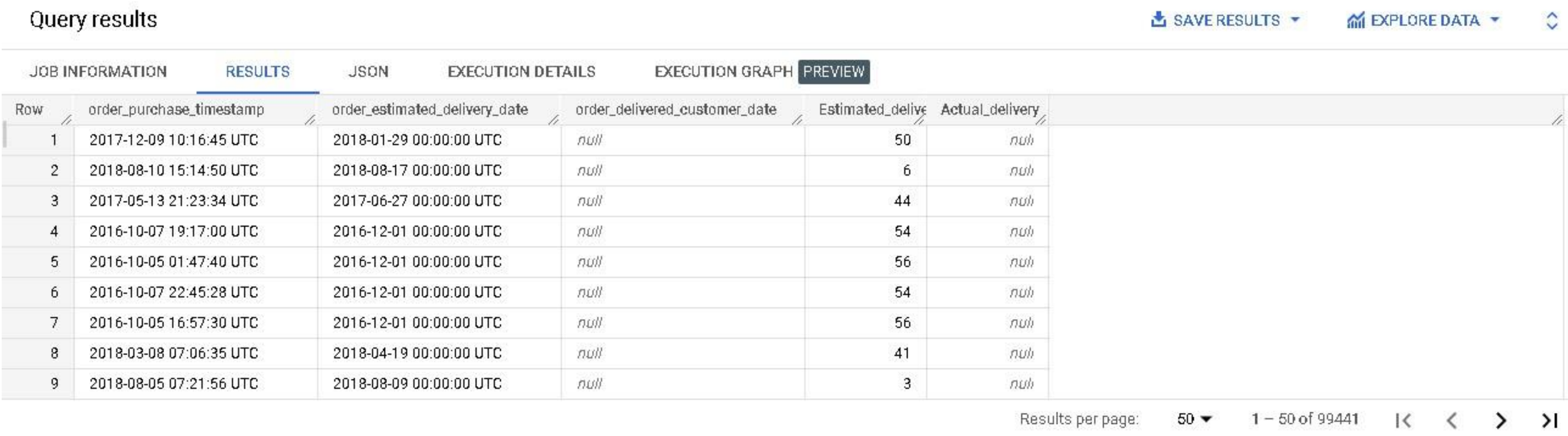


Row	customer_state	Total_price	Total_freight	Total_cost	Mean_price	Mean_freight	Mean_cost
1	AC	15982.95	3686.75	19669.7	173.73	40.07	213.8
2	AL	80314.81	15914.59	96229.4	180.89	35.84	216.73
3	AM	22356.84	5478.89	27835.73	135.5	33.21	168.7
4	AP	13474.3	2788.5	16262.8	164.32	34.01	198.33
5	BA	511349.99	100156.68	611506.67	134.6	26.36	160.97
6	CE	227254.71	48351.59	275606.3	153.76	32.71	186.47
7	DF	302603.94	50625.5	353229.44	125.77	21.04	146.81
8	ES	275037.31	49764.6	324801.91	121.91	22.06	143.97
9	GO	294591.95	53114.98	347706.93	126.27	22.77	149.04
10	MA	119648.22	31523.77	151171.99	145.2	38.26	183.46

Q5. Analysis on sales, freight and delivery time

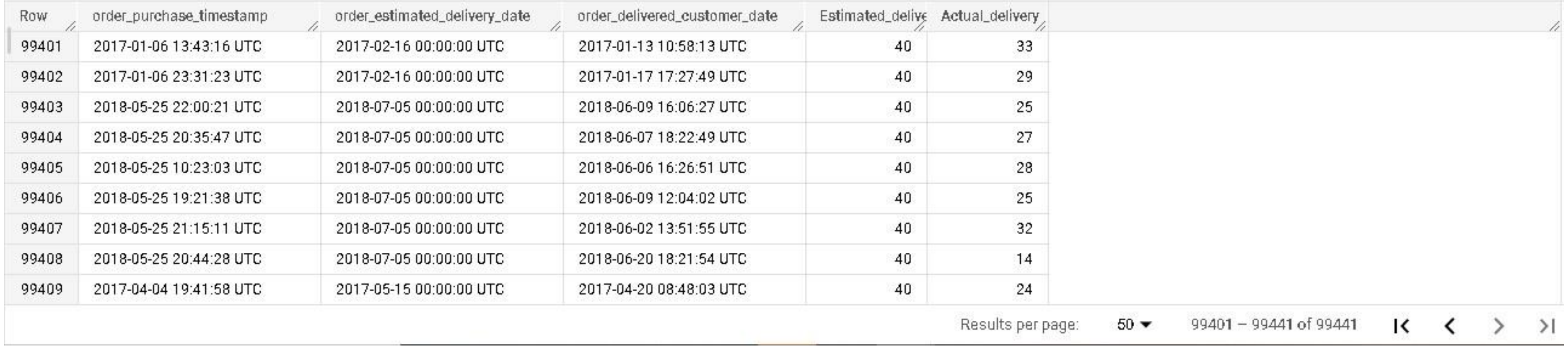
1. Calculate days between purchasing, delivering and estimated delivery

```
SELECT order_purchase_timestamp, order_estimated_delivery_date, order_delivered_customer_date,
date_diff(order_estimated_delivery_date, order_purchase_timestamp, day) as Estimated_delivery,
date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as Actual_delivery
FROM `target-sql-dsml-scaler.Target.orders`
```



Row	order_purchase_timestamp	order_estimated_delivery_date	order_delivered_customer_date	Estimated_delive	Actual_delivery
1	2017-12-09 10:16:45 UTC	2018-01-29 00:00:00 UTC	null	50	null
2	2018-08-10 15:14:50 UTC	2018-08-17 00:00:00 UTC	null	6	null
3	2017-05-13 21:23:34 UTC	2017-06-27 00:00:00 UTC	null	44	null
4	2016-10-07 19:17:00 UTC	2016-12-01 00:00:00 UTC	null	54	null
5	2016-10-05 01:47:40 UTC	2016-12-01 00:00:00 UTC	null	56	null
6	2016-10-07 22:45:28 UTC	2016-12-01 00:00:00 UTC	null	54	null
7	2016-10-05 16:57:30 UTC	2016-12-01 00:00:00 UTC	null	56	null
8	2018-03-08 07:06:35 UTC	2018-04-19 00:00:00 UTC	null	41	null
9	2018-08-05 07:21:56 UTC	2018-08-09 00:00:00 UTC	null	3	null

Results per page: 50 ▾ 1 – 50 of 99441



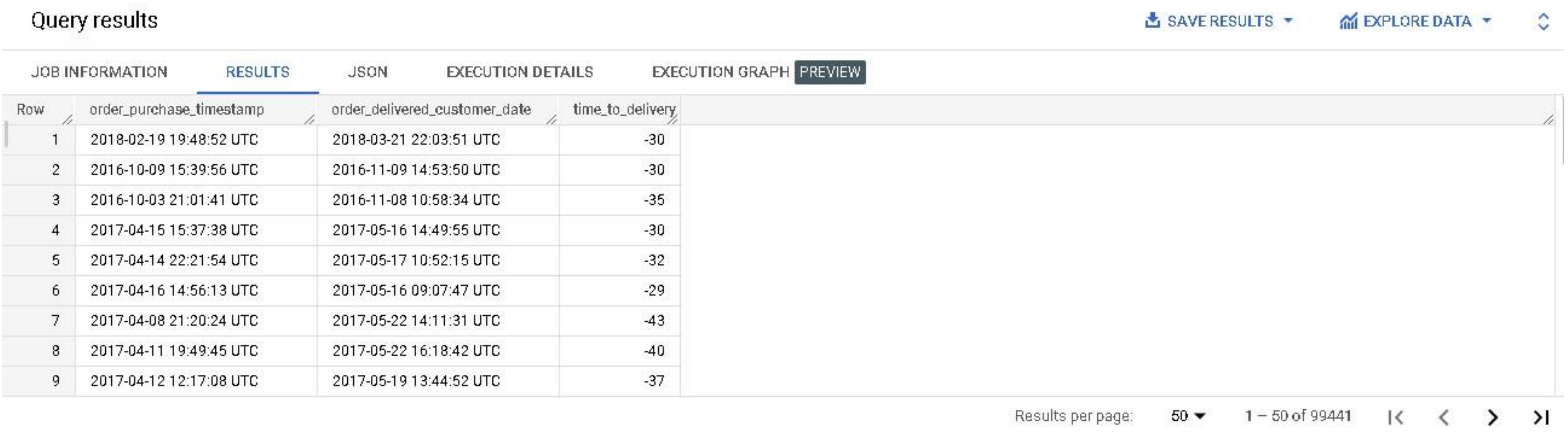
Row	order_purchase_timestamp	order_estimated_delivery_date	order_delivered_customer_date	Estimated_delive	Actual_delivery
99401	2017-01-06 13:43:16 UTC	2017-02-16 00:00:00 UTC	2017-01-13 10:58:13 UTC	40	33
99402	2017-01-06 23:31:23 UTC	2017-02-16 00:00:00 UTC	2017-01-17 17:27:49 UTC	40	29
99403	2018-05-25 22:00:21 UTC	2018-07-05 00:00:00 UTC	2018-06-09 16:06:27 UTC	40	25
99404	2018-05-25 20:35:47 UTC	2018-07-05 00:00:00 UTC	2018-06-07 18:22:49 UTC	40	27
99405	2018-05-25 10:23:03 UTC	2018-07-05 00:00:00 UTC	2018-06-06 16:26:51 UTC	40	28
99406	2018-05-25 19:21:38 UTC	2018-07-05 00:00:00 UTC	2018-06-09 12:04:02 UTC	40	25
99407	2018-05-25 21:15:11 UTC	2018-07-05 00:00:00 UTC	2018-06-02 13:51:55 UTC	40	32
99408	2018-05-25 20:44:28 UTC	2018-07-05 00:00:00 UTC	2018-06-20 18:21:54 UTC	40	14
99409	2017-04-04 19:41:58 UTC	2017-05-15 00:00:00 UTC	2017-04-20 08:48:03 UTC	40	24

Results per page: 50 ▾ 99401 – 99441 of 99441

2. Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:

   a. time_to_delivery = order_purchase_timestamp-order_delivered_customer_date

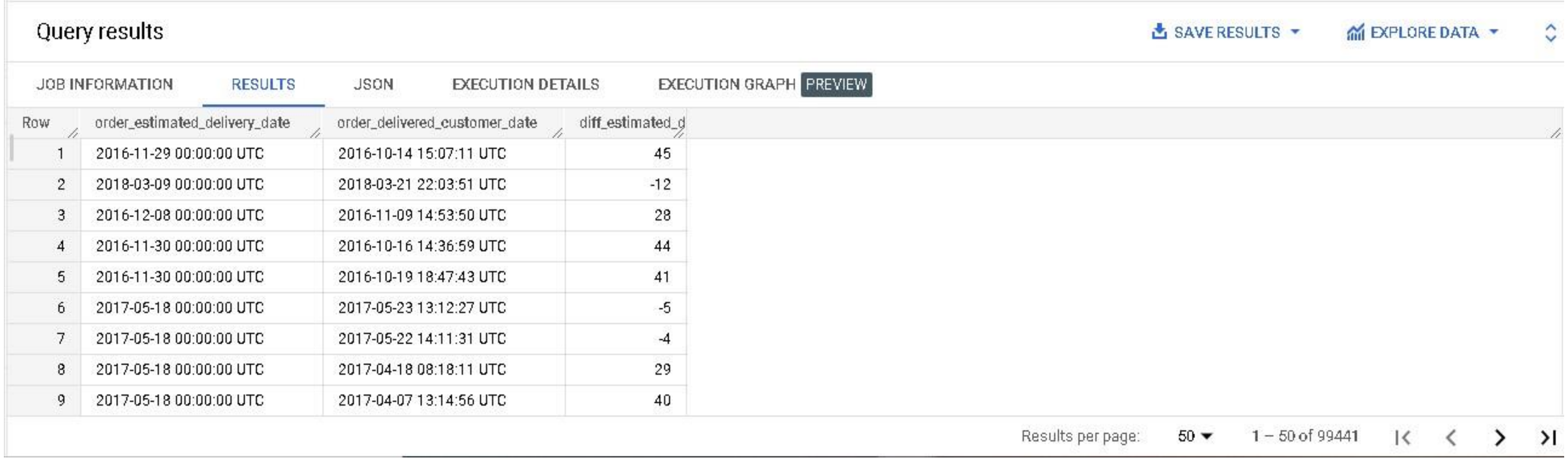
```
SELECT order_purchase_timestamp, order_delivered_customer_date,
date_diff(order_purchase_timestamp, order_delivered_customer_date, day) as time_to_delivery
FROM `target-sql-dsml-scaler.Target.orders`
```



Row	order_purchase_timestamp	order_delivered_customer_date	time_to_delivery
1	2018-02-19 19:48:52 UTC	2018-03-21 22:03:51 UTC	-30
2	2016-10-09 15:39:56 UTC	2016-11-09 14:53:50 UTC	-30
3	2016-10-03 21:01:41 UTC	2016-11-08 10:58:34 UTC	-35
4	2017-04-15 15:37:38 UTC	2017-05-16 14:49:55 UTC	-30
5	2017-04-14 22:21:54 UTC	2017-05-17 10:52:15 UTC	-32
6	2017-04-16 14:56:13 UTC	2017-05-16 09:07:47 UTC	-29
7	2017-04-08 21:20:24 UTC	2017-05-22 14:11:31 UTC	-43
8	2017-04-11 19:49:45 UTC	2017-05-22 16:18:42 UTC	-40
9	2017-04-12 12:17:08 UTC	2017-05-19 13:44:52 UTC	-37

b. diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

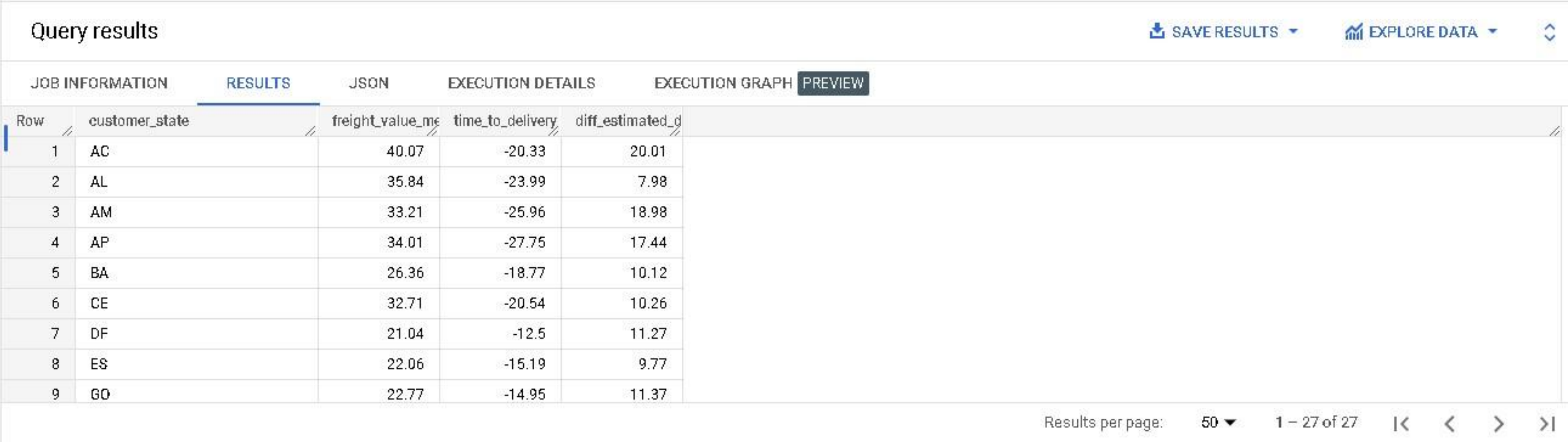
```
SELECT order_estimated_delivery_date, order_delivered_customer_date, date_diff(order_estimated_delivery_date, order_delivered_cus
tomer_date, day) as diff_estimated_delivery
FROM `Target.orders`
```



Row	order_estimated_delivery_date	order_delivered_customer_date	diff_estimated_d
1	2016-11-29 00:00:00 UTC	2016-10-14 15:07:11 UTC	45
2	2018-03-09 00:00:00 UTC	2018-03-21 22:03:51 UTC	-12
3	2016-12-08 00:00:00 UTC	2016-11-09 14:53:50 UTC	28
4	2016-11-30 00:00:00 UTC	2016-10-16 14:36:59 UTC	44
5	2016-11-30 00:00:00 UTC	2016-10-19 18:47:43 UTC	41
6	2017-05-18 00:00:00 UTC	2017-05-23 13:12:27 UTC	-5
7	2017-05-18 00:00:00 UTC	2017-05-22 14:11:31 UTC	-4
8	2017-05-18 00:00:00 UTC	2017-04-18 08:18:11 UTC	29
9	2017-05-18 00:00:00 UTC	2017-04-07 13:14:56 UTC	40

3. Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

```
SELECT c.customer_state, round(avg(oi.freight_value),2) as freight_value_mean, round(avg(date_diff(o.order_purchase_timestamp, o.order_delivered_customer_date, day)),2) as time_to_delivery , round(avg(date_diff(o.order_estimated_delivery_date, o.order_delivered_customer_date, day)),2) as diff_estimated_delivery
FROM `Target.orders` o
JOIN `Target.order_items` oi
ON o.order_id = oi.order_id
JOIN `Target.customers` c
ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY c.customer_state
```

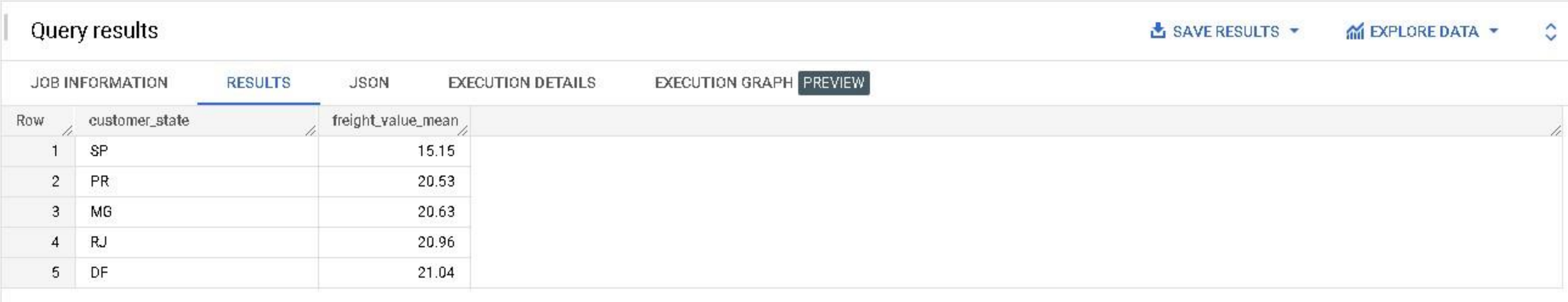


Row	customer_state	freight_value_me	time_to_delivery	diff_estimated_d
1	AC	40.07	-20.33	20.01
2	AL	35.84	-23.99	7.98
3	AM	33.21	-25.96	18.98
4	AP	34.01	-27.75	17.44
5	BA	26.36	-18.77	10.12
6	CE	32.71	-20.54	10.26
7	DF	21.04	-12.5	11.27
8	ES	22.06	-15.19	9.77
9	GO	22.77	-14.95	11.37

4. Sort the data to get the following:
5. Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5

Lowest Average Freight Value

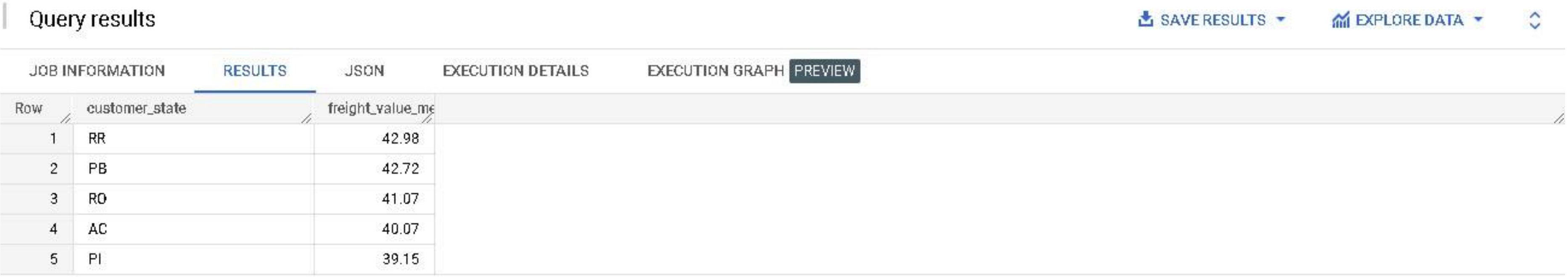
```
SELECT c.customer_state, round(avg(freight_value),2) as freight_value_mean
FROM `Target.orders` o
JOIN `Target.order_items` oi
ON o.order_id = oi.order_id
JOIN `Target.customers` c
ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY freight_value_mean
LIMIT 5
```



Row	customer_state	freight_value_mean
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04

Highest Average Freight Value

```
SELECT c.customer_state, round(avg(freight_value),2) as freight_value_mean
FROM `Target.orders` o
JOIN `Target.order_items` oi
ON o.order_id = oi.order_id
JOIN `Target.customers` c
ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY freight_value_mean DESC
LIMIT 5
```

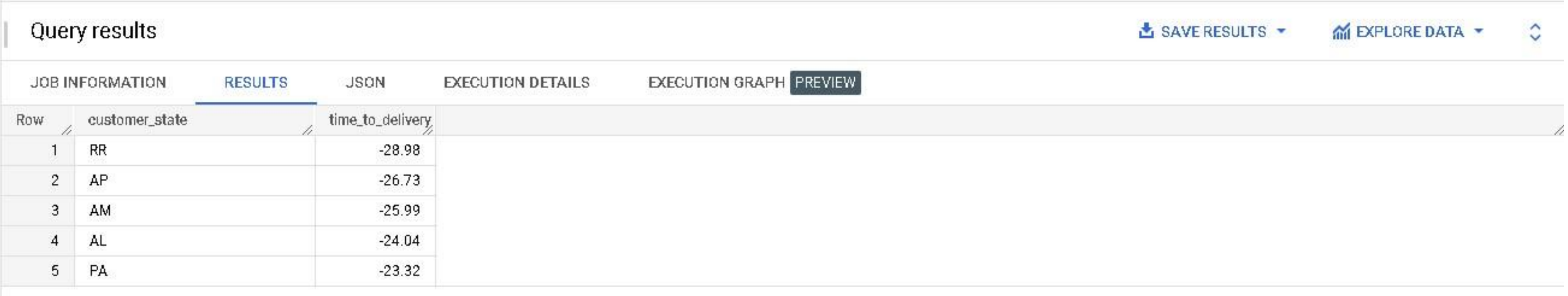


Row	customer_state	freight_value_me
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15

6. Top 5 states with highest/lowest average time to delivery

Lowest Time To Delivery

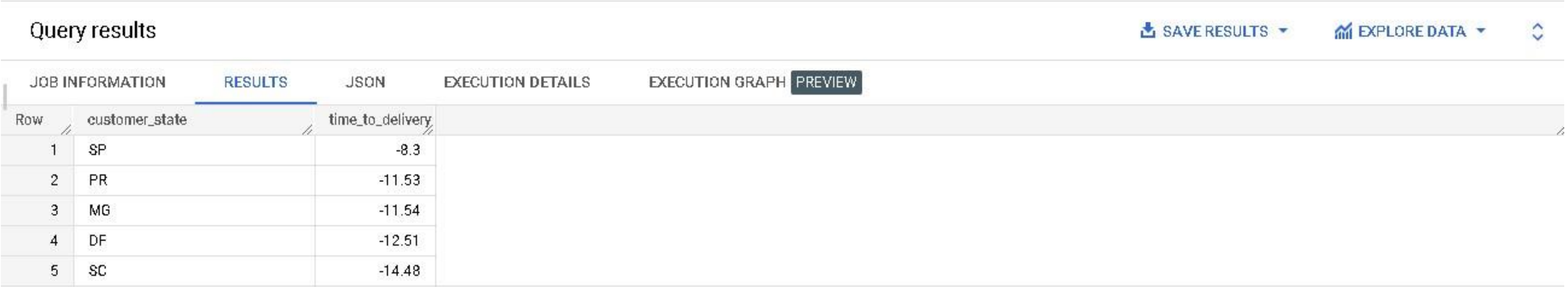
```
SELECT c.customer_state, round(avg(date_diff(o.order_purchase_timestamp, o.order_delivered_customer_date, day)),2)
as time_to_delivery
FROM `Target.orders` o
JOIN `Target.customers` c
ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY time_to_delivery
LIMIT 5
```



Row	customer_state	time_to_delivery
1	RR	-28.98
2	AP	-26.73
3	AM	-25.99
4	AL	-24.04
5	PA	-23.32

Highest Time To Delivery

```
SELECT c.customer_state, round(avg(date_diff(o.order_purchase_timestamp, o.order_delivered_customer_date, day)),2)
as time_to_delivery
FROM `Target.orders` o
JOIN `Target.customers` c
ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY time_to_delivery DESC
LIMIT 5
```

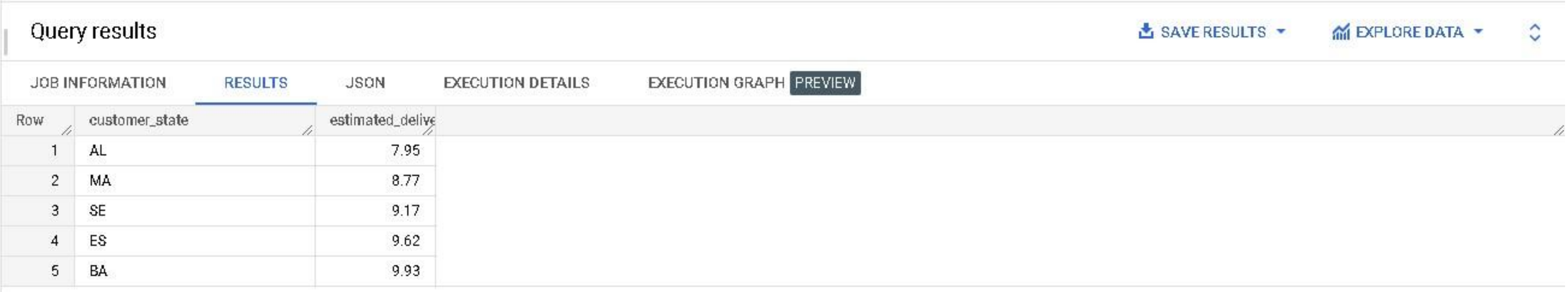


Row	customer_state	time_to_delivery
1	SP	-8.3
2	PR	-11.53
3	MG	-11.54
4	DF	-12.51
5	SC	-14.48

7. Top 5 states where delivery is really fast/ not so fast compared to estimated date

Really Fast

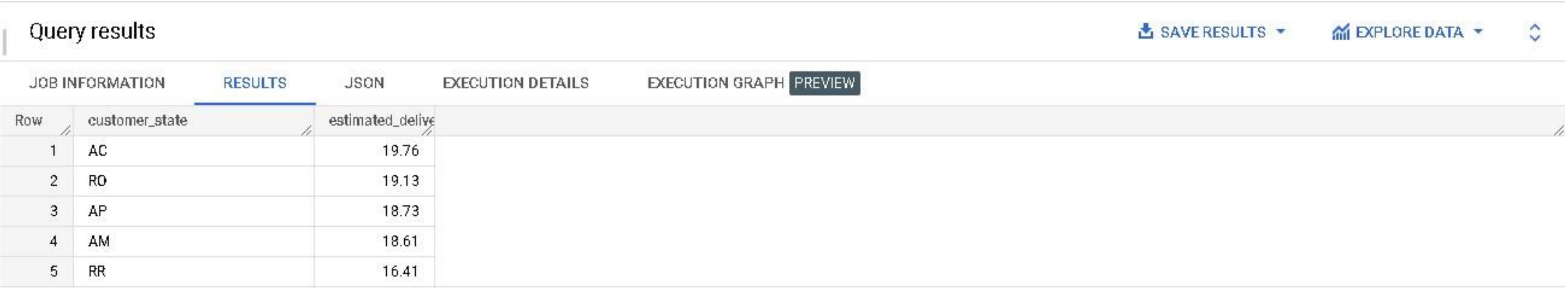
```
SELECT c.customer_state, round(avg(date_diff(o.order_estimated_delivery_date, o.order_delivered_customer_date, day)),2)
as estimated_delivery
FROM `Target.orders` o
JOIN `Target.customers` c
ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY estimated_delivery
LIMIT 5
```



Row	customer_state	estimated_delive
1	AL	7.95
2	MA	8.77
3	SE	9.17
4	ES	9.62
5	BA	9.93

Not So Fast

```
SELECT c.customer_state, round(avg(date_diff(o.order_estimated_delivery_date, o.order_delivered_customer_date, day)),2)
as estimated_delivery
FROM `Target.orders` o
JOIN `Target.customers` c
ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY estimated_delivery DESC
LIMIT 5
```

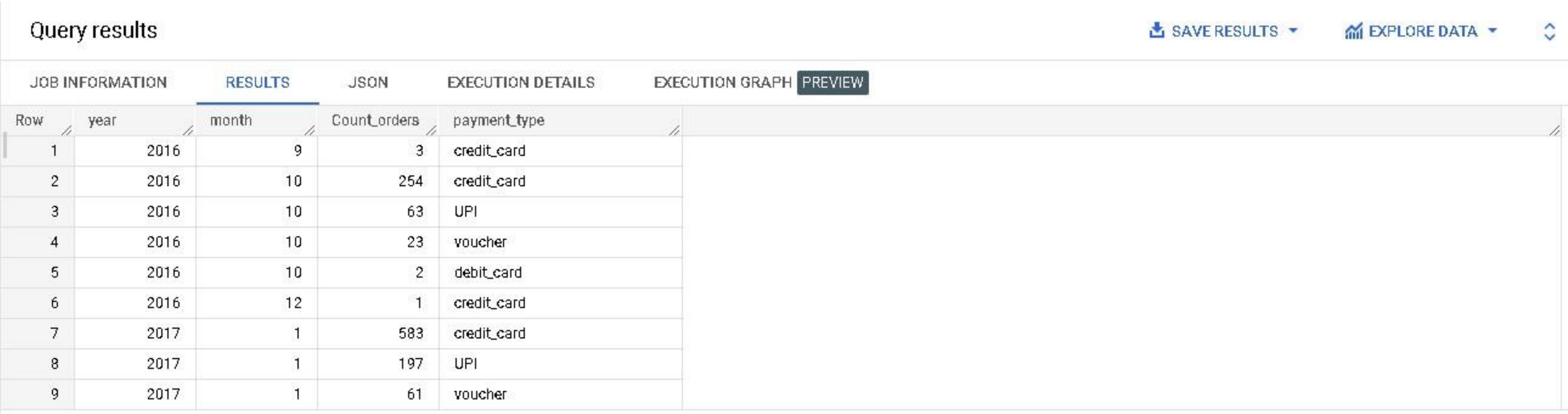


Row	customer_state	estimated_delive
1	AC	19.76
2	RO	19.13
3	AP	18.73
4	AM	18.61
5	RR	16.41

Q6. Payment type analysis:

1. Month over Month count of orders for different payment types

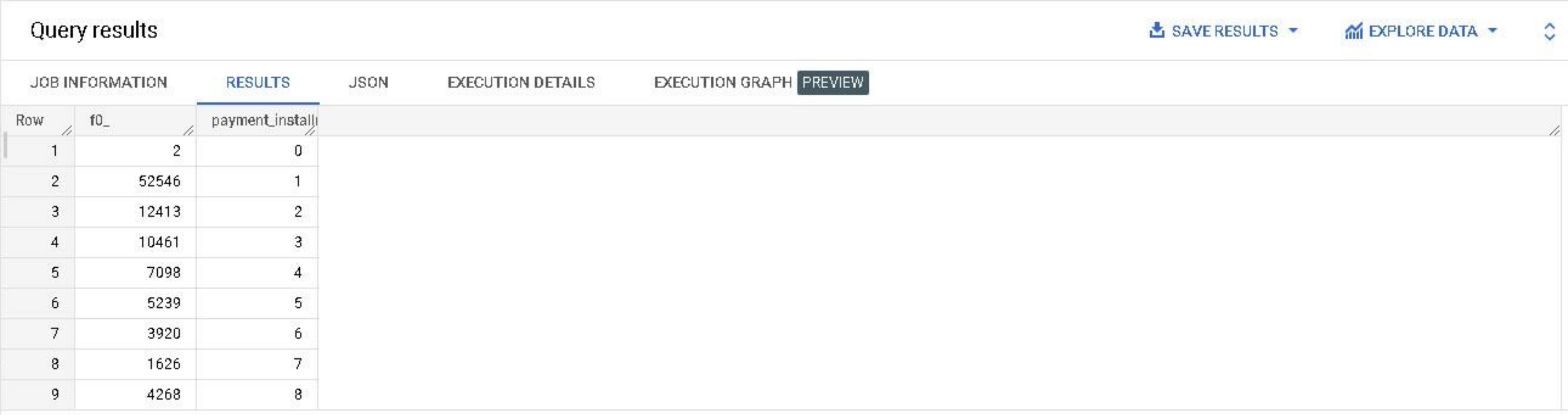
```
SELECT  extract(year from order_purchase_timestamp) as year, extract(month from o.order_purchase_timestamp) as month, count(o.order_i
d) as Count_orders, p.payment_type
FROM `target-sql-dsml-scaler.Target.payments` p
JOIN `Target.orders` o
ON p.order_id = o.order_id
GROUP BY year, month, p.payment_type
ORDER BY year, month
```



Row	year	month	Count_orders	payment_type
1	2016	9	3	credit_card
2	2016	10	254	credit_card
3	2016	10	63	UPI
4	2016	10	23	voucher
5	2016	10	2	debit_card
6	2016	12	1	credit_card
7	2017	1	583	credit_card
8	2017	1	197	UPI
9	2017	1	61	voucher

2. Count of orders based on the no. of payment installments

```
SELECT count(order_id), payment_installments
FROM `target-sql-dsml-scaler.Target.payments`
GROUP BY payment_installments
```



Row	f0_	payment_install
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

ACTIONABLE INSIGHTS AND RECOMMENDATIONS:

E-commerce provides smooth door-to-door service to people and save a lot of time as well.

After analysing the data, we can see that there is an increase in the online orders over the years. It was also observed that the number of orders went up during the festive season. It is advised that the companies should stock up well before the peak months in order to be prepared for the same.

It was also observed that many customers preferred to pay in installments. In order to make it more convenient for the customers, the same should be taken care of and the manner of making payment for the orders in installments should be made easy for the customers. This would encourage the customers to buy more via e-commerce.

Online payment methods are being used for the customers for making payment for the orders. The method for online payment should be made easy and convenient for the customers as many people are not comfortable with online transactions.

Online payment often comes with the threat of online frauds and thefts. The companies should make sure that their payment gateways are secure and the customers feel safe doing transactions on their website.

It is also observed that the delivery of orders is not so fast in some of the states and this might discourage the customers from buying online. The companies should take measures to improve in this aspect and make timely delivery of the orders.