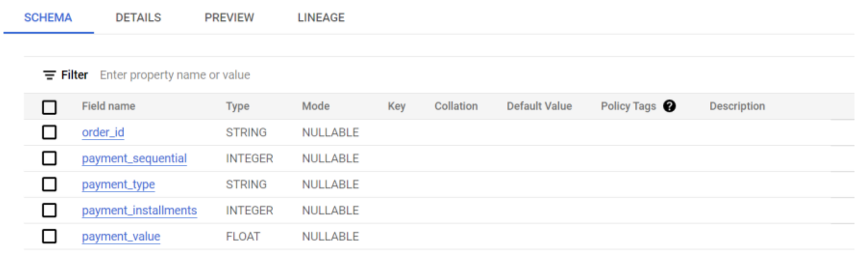
1. **Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset**

1. ***Data type of columns in a table.***


SCHEMA DETAILS PREVIEW LINEAGE

Filter Enter property name or value

Field name	Type	Mode	Key	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					

2. ***Time period for which the data is given.***

****

```
SELECT
order_id,
order_purchase_timestamp,
EXTRACT (YEAR FROM order_purchase_timestamp) as YEAR,
EXTRACT (QUARTER FROM order_purchase_timestamp) as QUARTER,
EXTRACT(MONTH FROM order_purchase_timestamp) as MONTH
FROM `Assignment.orders_dataset`
ORDER BY order_purchase_timestamp
LIMIT 10
```

****

Insights : -

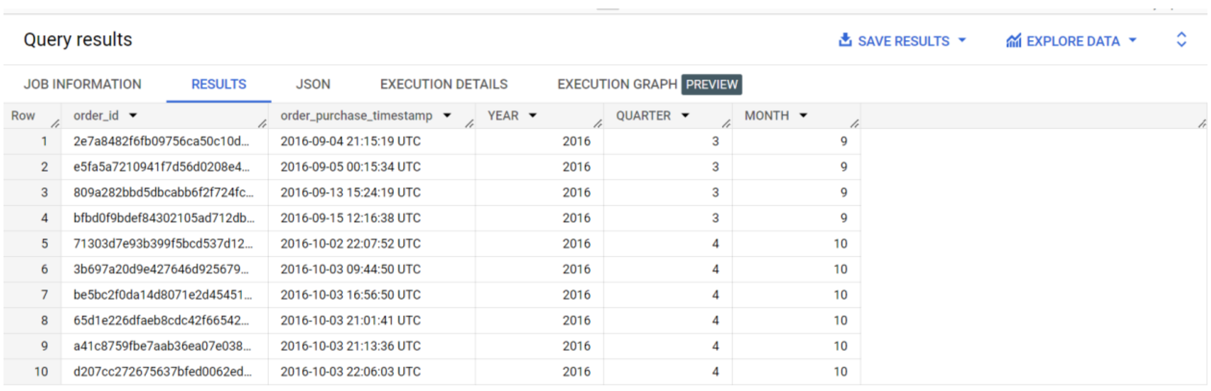
YEAR WISE data from 2016 -2018.

QUARTER WISE – 3rd Quarter in 2016 – 4th Quarter in 2018

Month Wise – Sep 2016 to Oct 2018

****

Query result screen shot : -


Query results

Row	order_id	order_purchase_timestamp	YEAR	QUARTER	MONTH
1	2e7a8482f6fb09756ca50c10d...	2016-09-04 21:15:19 UTC	2016	3	9
2	e5fa5a7210941f7d56d0208e4...	2016-09-05 00:15:34 UTC	2016	3	9
3	809a282bbd5dbcabb6f2f724fc...	2016-09-13 15:24:19 UTC	2016	3	9
4	bfbd0f9bdef84302105ad712db...	2016-09-15 12:16:38 UTC	2016	3	9
5	71303d7e93b399f5bcd537d12...	2016-10-02 22:07:52 UTC	2016	4	10
6	3b697a20d9e427646d925679...	2016-10-03 09:44:50 UTC	2016	4	10
7	be5bc2f0da14d8071e2d45451...	2016-10-03 16:56:50 UTC	2016	4	10
8	65d1e226dfaeb8cdc42f66542...	2016-10-03 21:01:41 UTC	2016	4	10
9	a41c8759fbe7aab36ea07e038...	2016-10-03 21:13:36 UTC	2016	4	10
10	d207cc272675637bfed0062ed...	2016-10-03 22:06:03 UTC	2016	4	10

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

****

```
SELECT
 c.customer_id,
 order_id,
 c.customer_city,
 c.customer_state

FROM `Assignment.customers_dataset` c
JOIN `Assignment.orders_dataset` as o
ON c.customer_id = o.customer_id
WHERE EXTRACT(YEAR FROM order_purchase_timestamp) BETWEEN 2016 and 2018
ORDER BY c.customer_id
LIMIT 10
```

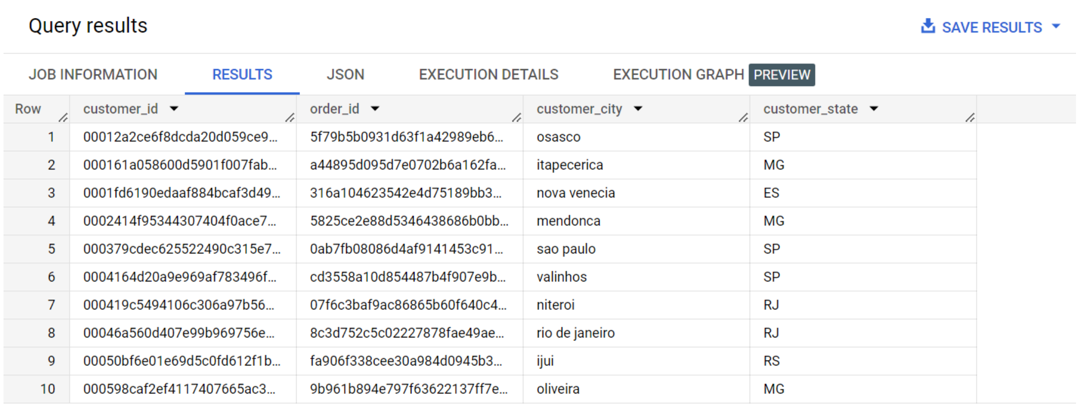
****

Insights :-

Incorrect city names have resulted in different rows if data is grouped by city.

****

Query result screenshot : -



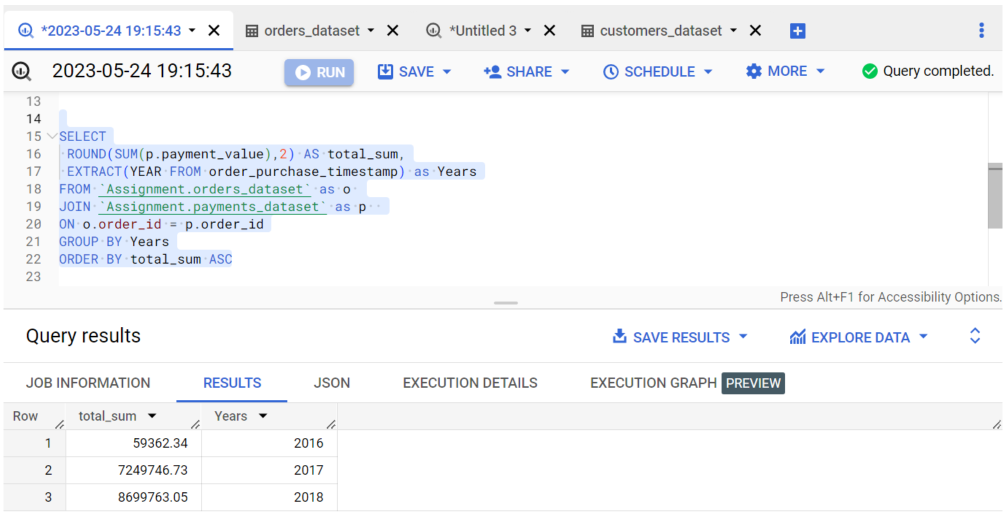
Row	customer_id	order_id	customer_city	customer_state
1	00012a2ce6f8dcda20d059ce9...	5f79b5b0931d63f1a42989eb6...	osasco	SP
2	000161a058600d5901f007fab...	a44895d095d7e0702b6a162fa...	itapecerica	MG
3	0001fd6190edaaf884bcaf3d49...	316a104623542e4d75189bb3...	nova venecia	ES
4	0002414f95344307404f0ace7...	5825ce2e88d5346438686b0bb...	mendonca	MG
5	000379cdec625522490c315e7...	0ab7fb08086d4af9141453c91...	sao paulo	SP
6	0004164d20a9e969af783496f...	cd3558a10d854487b4f907e9b...	valinhos	SP
7	000419c5494106c306a97b56...	07f6c3baf9ac86865b60f640c4...	niteroi	RJ
8	00046a560d407e99b969756e...	8c3d752c5c02227878fae49ae...	rio de janeiro	RJ
9	00050bf6e01e69d5c0fd612f1b...	fa906f338cee30a984d0945b3...	ijui	RS
10	000598caf2ef4117407665ac3...	9b961b894e797f63622137ff7e...	oliveira	MG

2. 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
 ROUND(SUM(p.payment_value),2) AS total_sum,
 EXTRACT(YEAR FROM order_purchase_timestamp) as Years
FROM `Assignment.orders_dataset` as o
JOIN `Assignment.payments_dataset` as p
ON o.order_id = p.order_id
GROUP BY Years
ORDER BY total_sum ASC
```

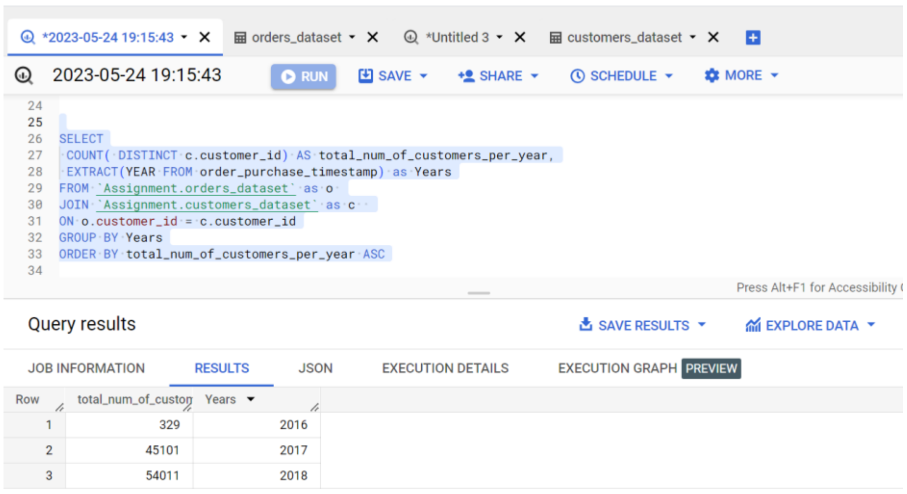


```
SELECT
 ROUND(SUM(p.payment_value),2) AS total_sum,
 EXTRACT(YEAR FROM order_purchase_timestamp) as Years
FROM `Assignment.orders_dataset` as o
JOIN `Assignment.payments_dataset` as p
ON o.order_id = p.order_id
GROUP BY Years
ORDER BY total_sum ASC
```

Row	total_sum	Years
1	59362.34	2016
2	7249746.73	2017
3	8699763.05	2018

■■

```
SELECT
 COUNT( DISTINCT c.customer_id) AS total_num_of_customers_per_year,
 EXTRACT(YEAR FROM order_purchase_timestamp) as Years
FROM `Assignment.orders_dataset` as o
JOIN `Assignment.customers_dataset` as c
ON o.customer_id = c.customer_id
GROUP BY Years
ORDER BY total_num_of_customers_per_year ASC
```



```sql

SELECT
 COUNT( DISTINCT c.customer_id) AS total_num_of_customers_per_year,
 EXTRACT(YEAR FROM order_purchase_timestamp) as Years
FROM `Assignment.orders_dataset` as o
JOIN `Assignment.customers_dataset` as c
ON o.customer_id = c.customer_id
GROUP BY Years
ORDER BY total_num_of_customers_per_year ASC

```

Query results

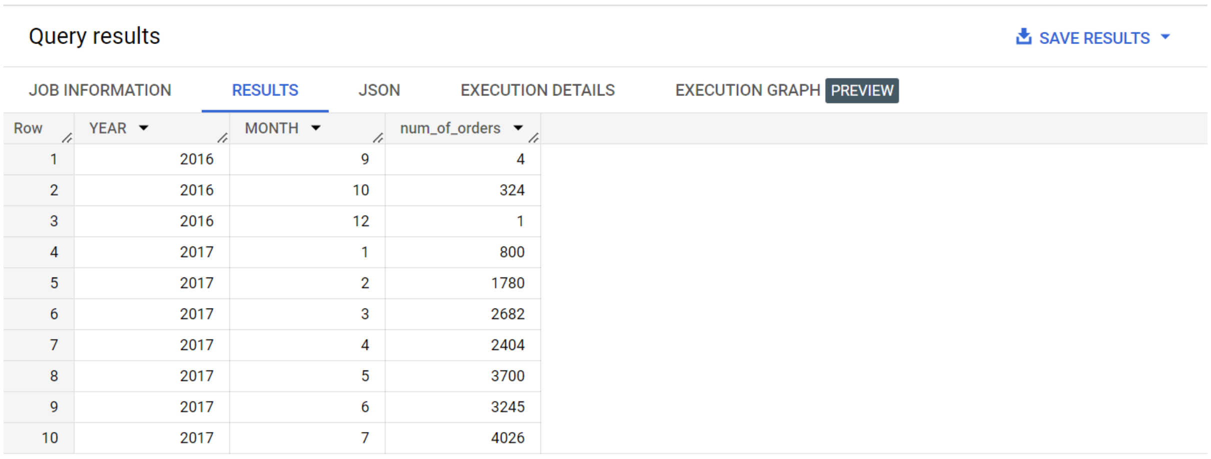
Row	total_num_of_custon	Years
1	329	2016
2	45101	2017
3	54011	2018

Insights :-

There is a definite increase in e commerce over the duration on which data is processed. We can see an increase in total number of customers per year as well as increase in total sum of payments per year.

*** ***

```sql
SELECT
 EXTRACT(YEAR from order_purchase_timestamp) as YEAR,
 EXTRACT(MONTH from order_purchase_timestamp) as MONTH,

 COUNT(order_id) as num_of_orders
FROM `Assignment.orders_dataset` as o
GROUP BY YEAR, MONTH
ORDER BY YEAR, MONTH
LIMIT 10
```



Query results

Row	YEAR	MONTH	num_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

Insights :-

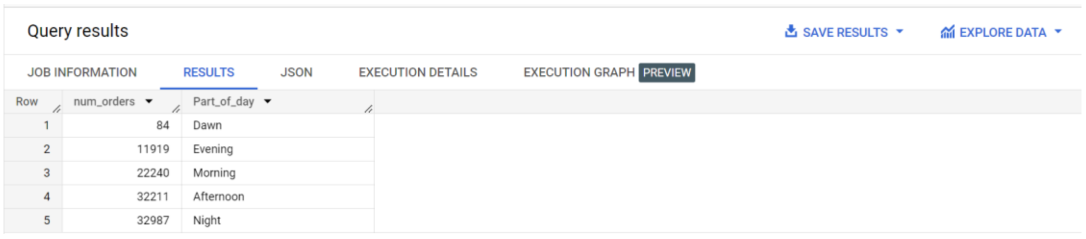
With the growth of e commerce in Brazil from 2016 to 2018 more users start to place orders. Hence we see an increase in number of orders each year with the month of Nov 2017 hitting the peak in terms of total orders placed.

Max orders each year are placed during the 3rd and 4th Quarter i.e Between the Jul- Dec

****

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

```sql
SELECT
 COUNT(d.order_id) as num_orders,
 d.Part_of_day

FROM

(
SELECT
 order_id,
 order_purchase_timestamp,
 CASE
    WHEN EXTRACT(TIME FROM order_purchase_timestamp) BETWEEN "05:00:00" AND "05:30:00"
    THEN "Dawn"
    WHEN EXTRACT(TIME FROM order_purchase_timestamp) BETWEEN "06:00:00" AND "11:59:59"
    THEN "Morning"
    WHEN EXTRACT(TIME FROM order_purchase_timestamp) BETWEEN "1:00:00" AND "16:59:59"
    THEN "Afternoon"
    WHEN EXTRACT(TIME FROM order_purchase_timestamp) BETWEEN "1:00:00" AND "18:59:59"
    THEN "Evening"
    ELSE "Night"
 END AS Part_of_day

FROM `Assignment.orders_dataset`
ORDER BY order_purchase_timestamp
) as d

GROUP BY d.Part_of_day
ORDER BY num_orders
```


Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW

Row	num_orders	Part_of_day
1	84	Dawn
2	11919	Evening
3	22240	Morning
4	32211	Afternoon
5	32987	Night

Insights : -

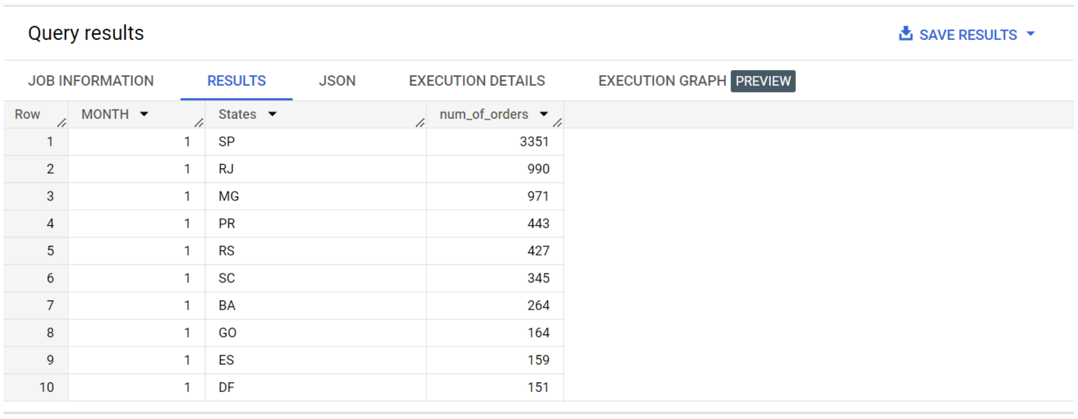
Max customers tend to buy during afternoon and night.

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

1. Get month on month orders by states.

Month on month data comparison among all states.

```
SELECT

EXTRACT(MONTH from order_purchase_timestamp) as MONTH,
c.customer_state as States,
COUNT(order_id) as num_of_orders
FROM `Assignment.orders_dataset` as o
JOIN `Assignment.customers_dataset` as c
ON c.customer_id = o.customer_id
GROUP BY States, MONTH
ORDER BY MONTH,num_of_orders DESC
LIMIT 10
```


Query results

SAVE RESULTS

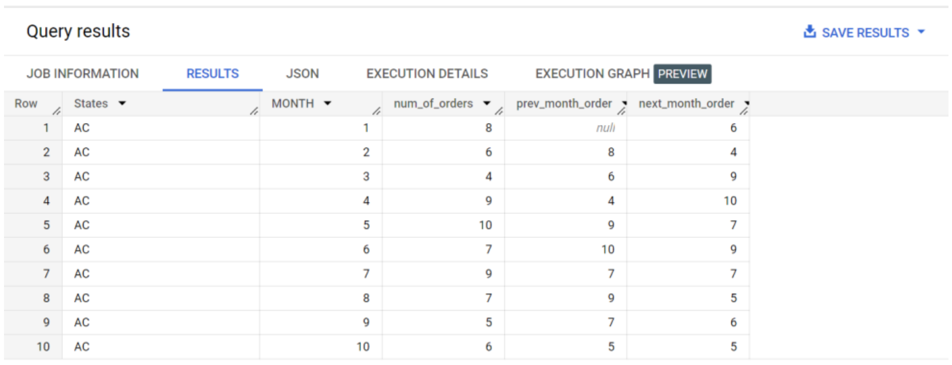
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	MONTH	States	num_of_orders
1	1	SP	3351
2	1	RJ	990
3	1	MG	971
4	1	PR	443
5	1	RS	427
6	1	SC	345
7	1	BA	264
8	1	GO	164
9	1	ES	159
10	1	DF	151

States : SP, RJ and MG lead in total number of orders per month for each month when data checked over total duration available.

Month on month data comparison for each state separately.

```
SELECT*,
 LAG(d.num_of_orders,1) OVER (PARTITION BY d.States ORDER BY d.MONTH) as prev_month_order,
 LEAD(d.num_of_orders,1) OVER (PARTITION BY d.States ORDER BY d.MONTH) as next_month_order
FROM
(
SELECT DISTINCT
c.customer_state as States,
EXTRACT(MONTH from order_purchase_timestamp) as MONTH,
COUNT(order_id) as num_of_orders,

FROM `Assignment.orders_dataset` as o
JOIN `Assignment.customers_dataset` as c
ON c.customer_id = o.customer_id
GROUP BY States, MONTH
) as d
ORDER BY d.States, d.MONTH
LIMIT 10
```


Query results

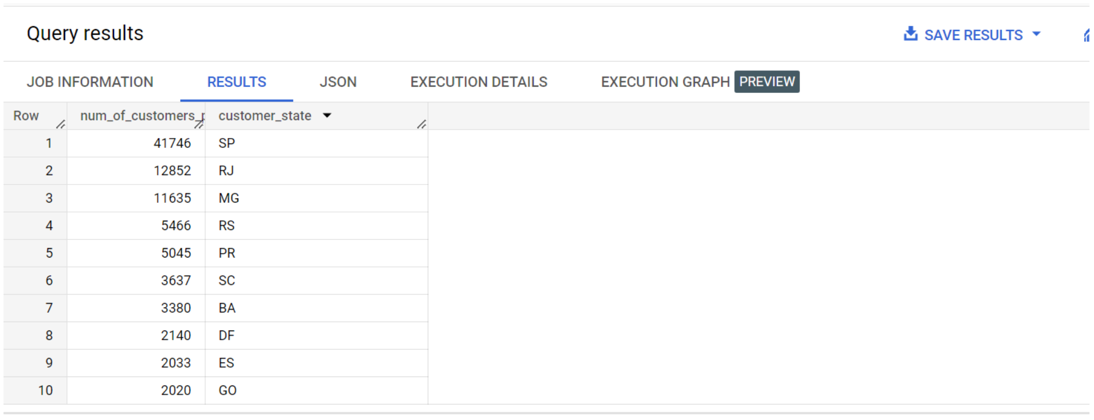
SAVE RESULTS

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW

Row	States	MONTH	num_of_orders	prev_month_order	next_month_order
1	AC	1	8	null	6
2	AC	2	6	8	4
3	AC	3	4	6	9
4	AC	4	9	4	10
5	AC	5	10	9	7
6	AC	6	7	10	9
7	AC	7	9	7	7
8	AC	8	7	9	5
9	AC	9	5	7	6
10	AC	10	6	5	5

2. Distribution of customers across the states in Brazil.

```
SELECT
 COUNT(DISTINCT customer_id) as num_of_customers_per_state,
 customer_state

FROM `Assignment.customers_dataset`
GROUP BY customer_state
ORDER BY num_of_customers_per_state DESC
LIMIT 10
```

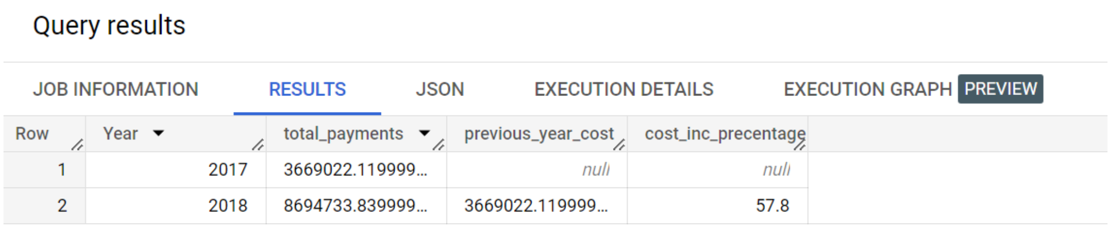

Query results

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

4. ***Impact on Economy: Analyze 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**

```
SELECT *,
 ROUND(((e.total_payments - e.previous_year_cost)/e.total_payments) * 100 ,2) as
cost_inc_precentage
FROM
(
SELECT *,
 LAG(d.total_payments,1) OVER (ORDER BY d.Year) as previous_year_cost
FROM
(
SELECT
 EXTRACT(YEAR from order_purchase_timestamp) as Year,
 SUM(payment_value) as total_payments
FROM `Assignment.orders_dataset` as o
JOIN `Assignment.payments_dataset` 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
GROUP BY Year
ORDER BY Year
) as d
) as e
ORDER BY e.Year
```


Query results

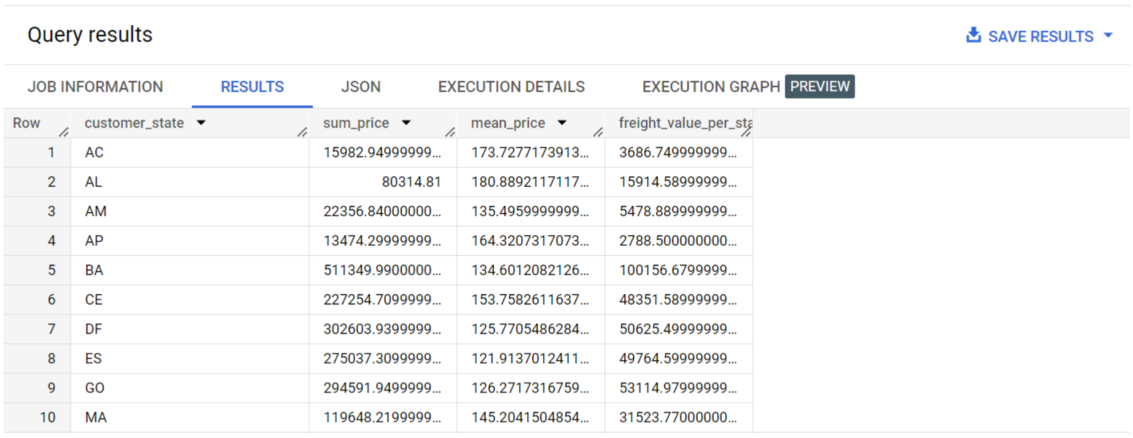
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	Year	total_payments	previous_year_cost	cost_inc_precentage
1	2017	3669022.119999...	null	null
2	2018	8694733.839999...	3669022.119999...	57.8

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

```
SELECT
  c.customer_state,
  SUM(oi.price) as sum_price,
  AVG(oi.price) as mean_price,
  SUM(oi.freight_value) as freight_value_per_state


FROM `Assignment.order_items_dataset` as oi
JOIN `Assignment.orders_dataset` as o
ON oi.order_id = o.order_id
```

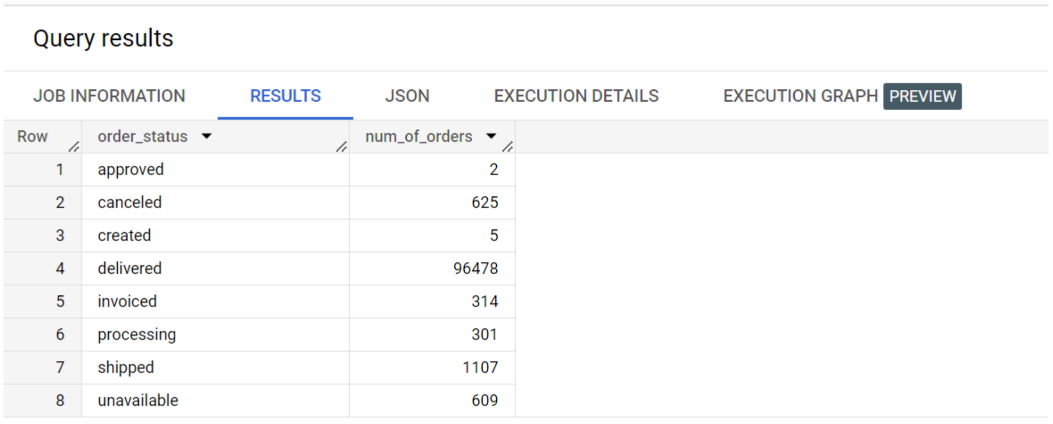
```
JOIN `Assignment.customers_dataset` as c
ON o.customer_id = c.customer_id

GROUP BY c.customer_state
ORDER BY c.customer_state
LIMIT 10
```



Row	customer_state	sum_price	mean_price	freight_value_per_sta
1	AC	15982.94999999...	173.7277173913...	3686.749999999...
2	AL	80314.81	180.8892117117...	15914.58999999...
3	AM	22356.84000000...	135.4959999999...	5478.889999999...
4	AP	13474.29999999...	164.3207317073...	2788.500000000...
5	BA	511349.9900000...	134.6012082126...	100156.6799999...
6	CE	227254.7099999...	153.7582611637...	48351.58999999...
7	DF	302603.9399999...	125.7705486284...	50625.49999999...
8	ES	275037.3099999...	121.9137012411...	49764.59999999...
9	GO	294591.9499999...	126.2717316759...	53114.97999999...
10	MA	119648.2199999...	145.2041504854...	31523.77000000...

5. Analysis on sales, freight and delivery time

```
SELECT
 order_status,
 COUNT(order_id) as num_of_orders
FROM `Assignment.orders_dataset`
GROUP BY order_status
ORDER BY order_status
```


Query results

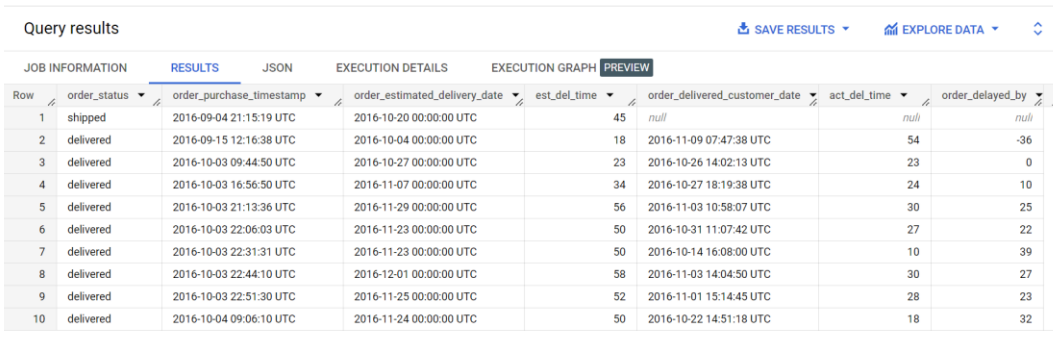
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	order_status	num_of_orders
1	approved	2
2	canceled	625
3	created	5
4	delivered	96478
5	invoiced	314
6	processing	301
7	shipped	1107
8	unavailable	609

We see that we have different number of orders based on their order status. It is found when the date in order_dataset is ordered by order status. Only orders with status as delivered, shipped and some orders which were cancelled after the product was shipped have an actual date value ***order_deliver_carrier_date*** *populated for them in the table. Rest all orders have NULL. This means only the above mentioned orders have freight value populated for them. Hence in the below query while calculating freight value we have filtered the status condition to meet this requirement.*

1. Calculate days between purchasing, delivering and estimated delivery

```
SELECT

 order_status,

 order_purchase_timestamp,

 order_estimated_delivery_date,

 DATE_DIFF(order_estimated_delivery_date,order_purchase_timestamp, DAY) as
est_del_time,

 order_delivered_customer_date,

 DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp, DAY) as
act_del_time,

 DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY) as
order_delayed_by

FROM `Assignment.orders_dataset`

WHERE order_status IN ("delivered", "shipped", "cancelled") AND
order_delivered_carrier_date IS NOT NULL

ORDER BY order_purchase_timestamp

LIMIT 10
```



Row	order_status	order_purchase_timestamp	order_estimated_delivery_date	est_del_time	order_delivered_customer_date	act_del_time	order_delayed_by
1	shipped	2016-09-04 21:15:19 UTC	2016-10-20 00:00:00 UTC	45	null	null	null
2	delivered	2016-09-15 12:16:38 UTC	2016-10-04 00:00:00 UTC	18	2016-11-09 07:47:38 UTC	54	-36
3	delivered	2016-10-03 09:44:50 UTC	2016-10-27 00:00:00 UTC	23	2016-10-26 14:02:13 UTC	23	0
4	delivered	2016-10-03 16:56:50 UTC	2016-11-07 00:00:00 UTC	34	2016-10-27 18:19:38 UTC	24	10
5	delivered	2016-10-03 21:13:36 UTC	2016-11-29 00:00:00 UTC	56	2016-11-03 10:58:07 UTC	30	25
6	delivered	2016-10-03 22:06:03 UTC	2016-11-23 00:00:00 UTC	50	2016-10-31 11:07:42 UTC	27	22
7	delivered	2016-10-03 22:31:31 UTC	2016-11-23 00:00:00 UTC	50	2016-10-14 16:08:00 UTC	10	39
8	delivered	2016-10-03 22:44:10 UTC	2016-12-01 00:00:00 UTC	58	2016-11-03 14:04:50 UTC	30	27
9	delivered	2016-10-03 22:51:30 UTC	2016-11-25 00:00:00 UTC	52	2016-11-01 15:14:45 UTC	28	23
10	delivered	2016-10-04 09:06:10 UTC	2016-11-24 00:00:00 UTC	50	2016-10-22 14:51:18 UTC	18	32

2. Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:
   - time_to_delivery = order_delivered_customer_date-order_purchase_timestamp
   - diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

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

   SELECT

   c.customer_state,

ROUND(AVG(DATE_DIFF(order_estimated_delivery_date,order_purchase_timestamp, DAY)),2) as time_to_delivery,

ROUND(AVG(DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date, DAY)),2) as diff_estimated_delivery,

ROUND(AVG(oi.freight_value),2) as avg_freight_value

FROM `Assignment.orders_dataset` as o

INNER JOIN `Assignment.order_items_dataset` as oi

ON o.order_id = oi.order_id

INNER JOIN `Assignment.customers_dataset` as c

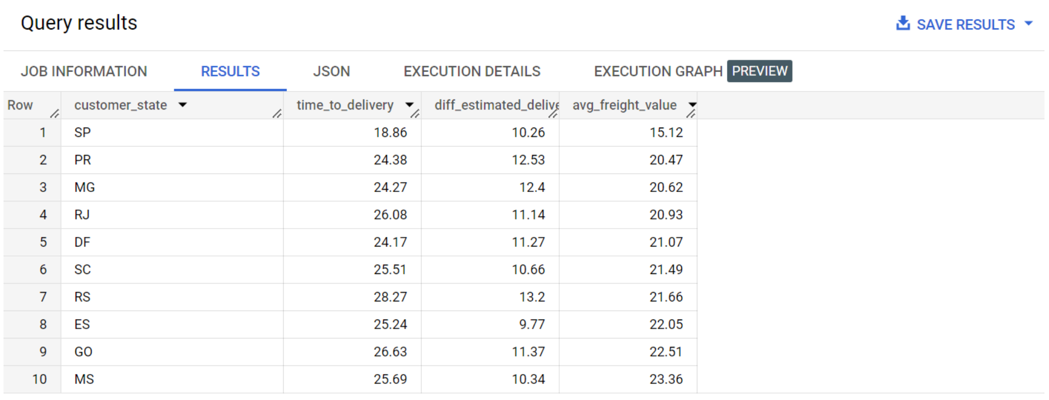
ON o.customer_id = c.customer_id

WHERE order_status IN ("delivered", "shipped", "cancelled") AND order_delivered_carrier_date IS NOT NULL

GROUP BY c.customer_state

ORDER BY avg_freight_value

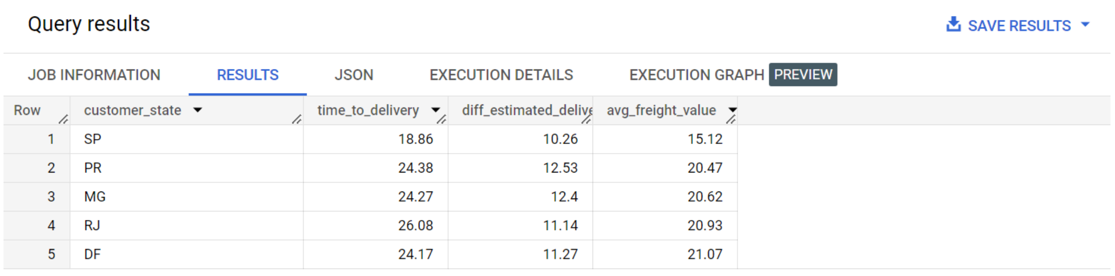
LIMIT 10


Query results

Row	customer_state	time_to_delivery	diff_estimated_delive	avg_freight_value
1	SP	18.86	10.26	15.12
2	PR	24.38	12.53	20.47
3	MG	24.27	12.4	20.62
4	RJ	26.08	11.14	20.93
5	DF	24.17	11.27	21.07
6	SC	25.51	10.66	21.49
7	RS	28.27	13.2	21.66
8	ES	25.24	9.77	22.05
9	GO	26.63	11.37	22.51
10	MS	25.69	10.34	23.36

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

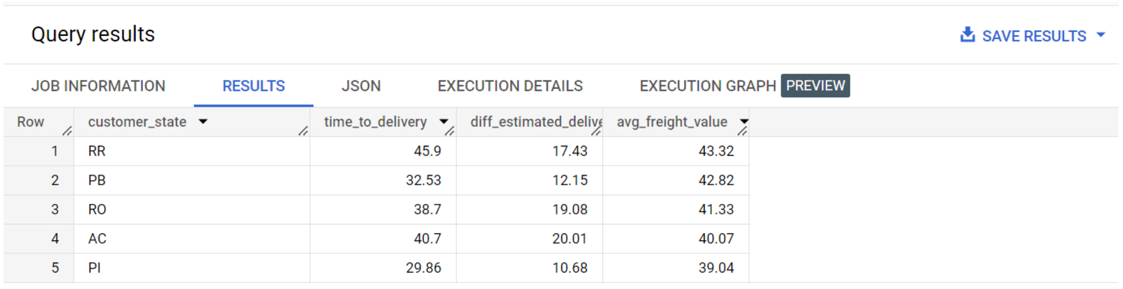
Same query as above except ORDER BY avg_freight_value DESC in last second line and LIMIT 5 after that.


Query results

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

Row	customer_state	time_to_delivery	diff_estimated_delive	avg_freight_value
1	SP	18.86	10.26	15.12
2	PR	24.38	12.53	20.47
3	MG	24.27	12.4	20.62
4	RJ	26.08	11.14	20.93
5	DF	24.17	11.27	21.07

Same query as above except ORDER BY avg_freight_value DESC in last second line and LIMIT 5 after that.


Query results

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

Row	customer_state	time_to_delivery	diff_estimated_delive	avg_freight_value
1	RR	45.9	17.43	43.32
2	PB	32.53	12.15	42.82
3	RO	38.7	19.08	41.33
4	AC	40.7	20.01	40.07
5	PI	29.86	10.68	39.04

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

SELECT

c.customer_state,

ROUND(AVG(DATE_DIFF(order_estimated_delivery_date,order_purchase_timestamp, DAY)),2) as avg_time_to_delivery,

ROUND(AVG(DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date, DAY)),2) as avg_diff_estimated_delivery,

ROUND(AVG(oi.freight_value),2) as avg_freight_value

FROM `Assignment.orders_dataset` as o

INNER JOIN `Assignment.order_items_dataset` as oi

ON o.order_id = oi.order_id

INNER JOIN `Assignment.customers_dataset` as c

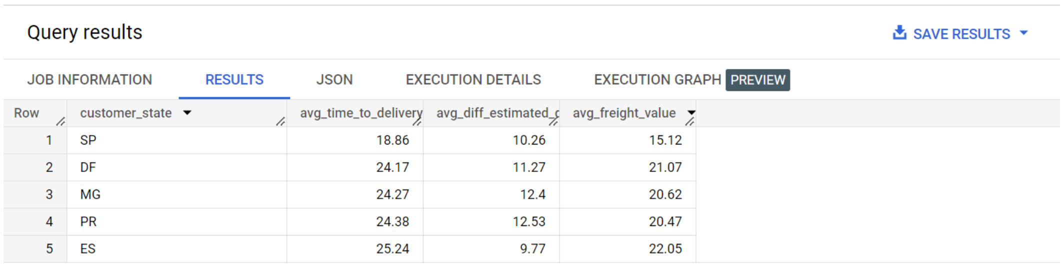
ON o.customer_id = c.customer_id

WHERE order_status IN ("delivered", "shipped", "cancelled") AND order_delivered_carrier_date IS NOT NULL

GROUP BY c.customer_state

ORDER BY avg_time_to_delivery

LIMIT 5

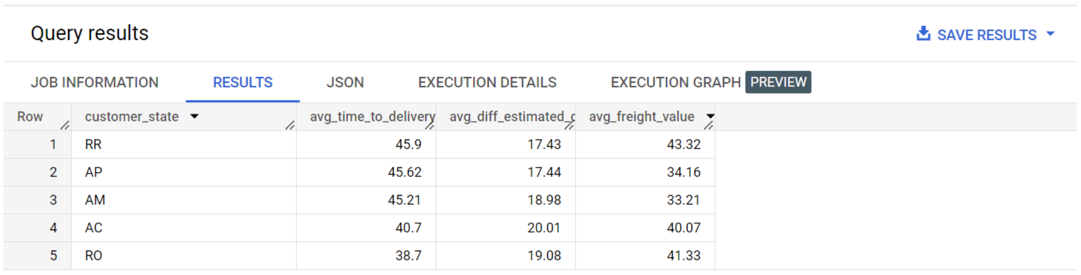


Query results SAVE RESULTS

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

Row	customer_state	avg_time_to_delivery	avg_diff_estimated_c	avg_freight_value
1	SP	18.86	10.26	15.12
2	DF	24.17	11.27	21.07
3	MG	24.27	12.4	20.62
4	PR	24.38	12.53	20.47
5	ES	25.24	9.77	22.05

Highest time to delivery. Same query as above, except ORDER BY avg_time_to_delivery DESC in last second line.



Query results SAVE RESULTS

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

Row	customer_state	avg_time_to_delivery	avg_diff_estimated_c	avg_freight_value
1	RR	45.9	17.43	43.32
2	AP	45.62	17.44	34.16
3	AM	45.21	18.98	33.21
4	AC	40.7	20.01	40.07
5	RO	38.7	19.08	41.33

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

SELECT

c.customer_state,

ROUND(AVG(DATE_DIFF(order_estimated_delivery_date,order_purchase_timestamp, DAY)),2) as avg_time_to_delivery,

ROUND(AVG(DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date, DAY)),2) as avg_diff_estimated_delivery,

ROUND(AVG(oi.freight_value),2) as avg_freight_value

FROM `Assignment.orders_dataset` as o

INNER JOIN `Assignment.order_items_dataset` as oi

ON o.order_id = oi.order_id

INNER JOIN `Assignment.customers_dataset` as c

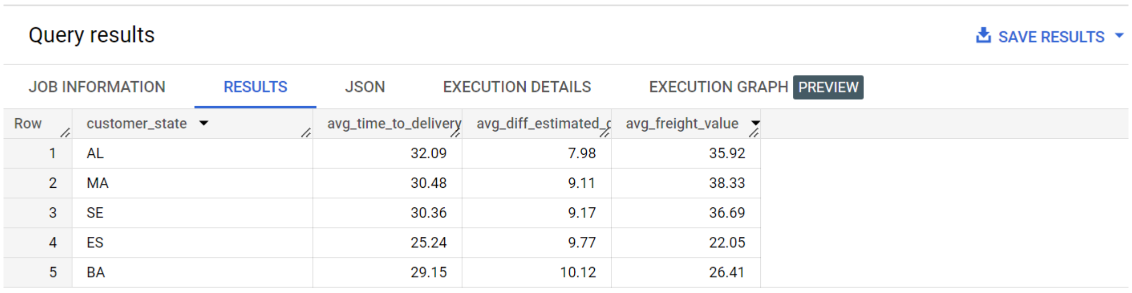
ON o.customer_id = c.customer_id

WHERE order_status IN ("delivered", "shipped", "cancelled") AND order_delivered_carrier_date IS NOT NULL

GROUP BY c.customer_state

ORDER BY avg_diff_estimated_delivery

LIMIT 5



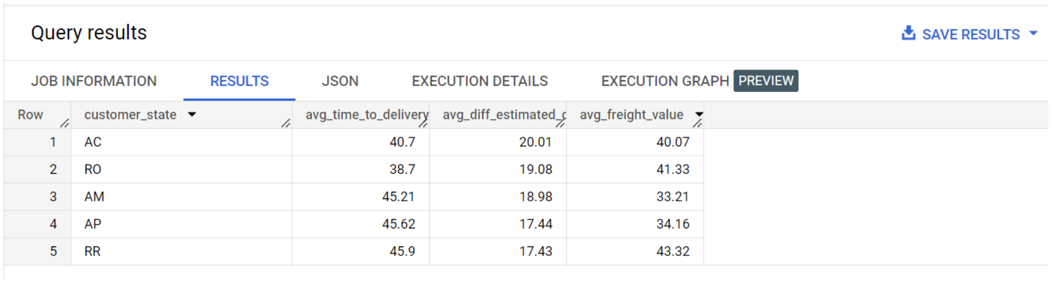
Query results

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

Row	customer_state	avg_time_to_delivery	avg_diff_estimated_c	avg_freight_value
1	AL	32.09	7.98	35.92
2	MA	30.48	9.11	38.33
3	SE	30.36	9.17	36.69
4	ES	25.24	9.77	22.05
5	BA	29.15	10.12	26.41

Delivery not so fast : -

Same query except ORDER BY avg_diff_estimated_deliveryDESC in last second line.



Query results

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

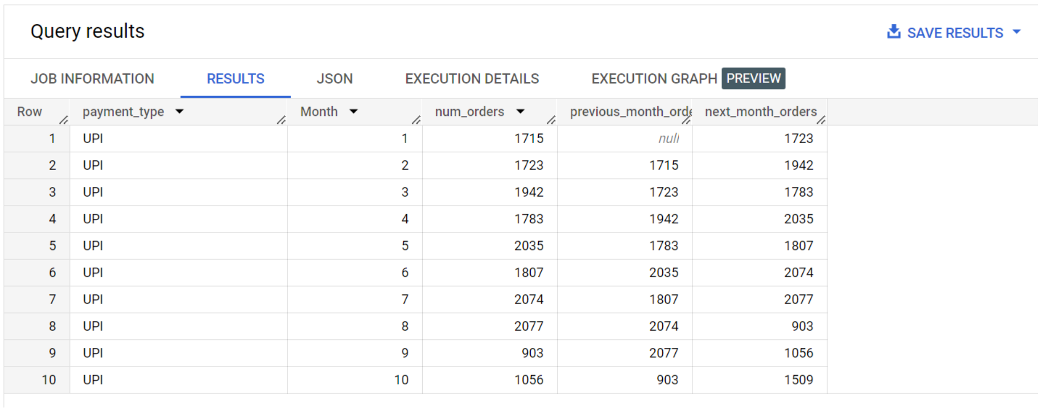
Row	customer_state	avg_time_to_delivery	avg_diff_estimated_c	avg_freight_value
1	AC	40.7	20.01	40.07
2	RO	38.7	19.08	41.33
3	AM	45.21	18.98	33.21
4	AP	45.62	17.44	34.16
5	RR	45.9	17.43	43.32

6. Payment type analysis:

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

SELECT,*

```
 LAG(d.num_orders,1) OVER (PARTITION BY d.payment_type ORDER BY
d.payment_type,d.Month) as previous_month_orders,

 LEAD(d.num_orders,1) OVER (PARTITION BY d.payment_type ORDER BY
d.payment_type,d.Month) as next_month_orders



FROM

(

SELECT

 payment_type,

 EXTRACT(MONTH FROM o.order_purchase_timestamp) as Month,

 COUNT(o.order_id) as num_orders

FROM `Assignment.payments_dataset` as p

JOIN `Assignment.orders_dataset` as o

ON o.order_id = p.order_id

GROUP BY payment_type, Month

ORDER BY payment_type, Month

) as d

ORDER by d.payment_type, d.Month
```

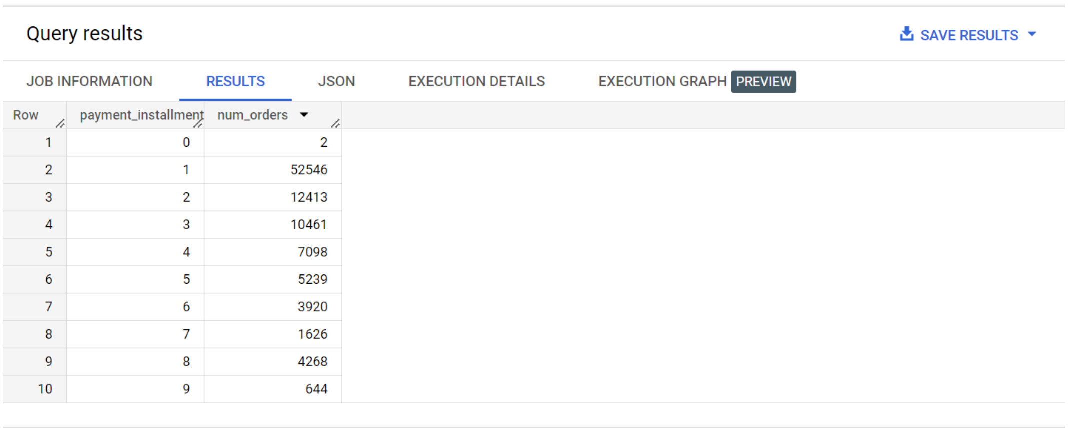


Row	payment_type	Month	num_orders	previous_month_orde	next_month_orders
1	UPI	1	1715	null	1723
2	UPI	2	1723	1715	1942
3	UPI	3	1942	1723	1783
4	UPI	4	1783	1942	2035
5	UPI	5	2035	1783	1807
6	UPI	6	1807	2035	2074
7	UPI	7	2074	1807	2077
8	UPI	8	2077	2074	903
9	UPI	9	903	2077	1056
10	UPI	10	1056	903	1509

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

```
SELECT
```

```
 payment_installments,

 COUNT(o.order_id) as num_orders


FROM `Assignment.payments_dataset` as p

JOIN `Assignment.orders_dataset` as o

ON o.order_id = p.order_id

GROUP BY payment_installments

ORDER BY payment_installments

LIMIT 10
```



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