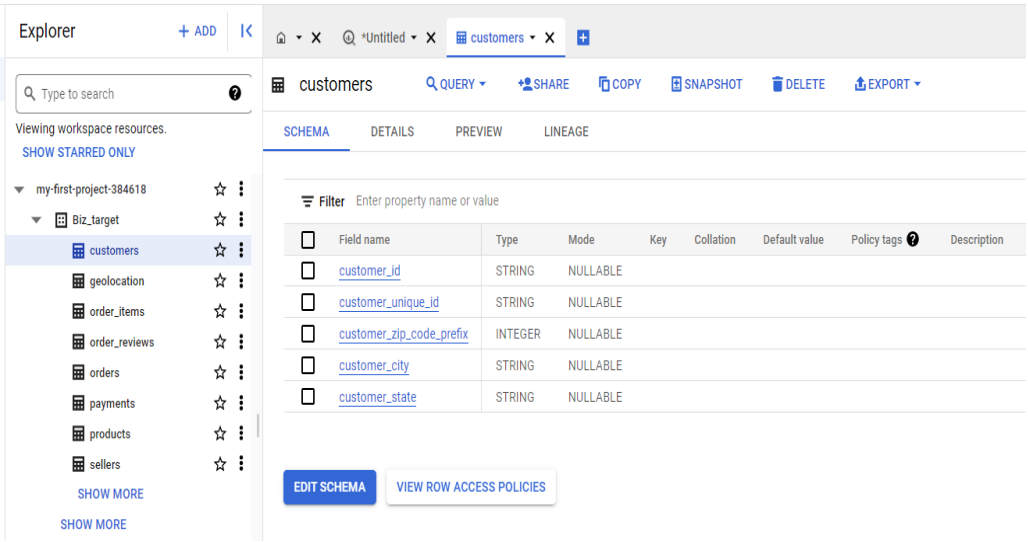
1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset

1.1 . Data type of columns in a table

We can check the datatypes by clicking on the respective table and go to schema, under **TYPE** column. As no particular table has been mentioned. For all the given table Data only the following Datatypes of the columns were included:

String, Integer, Float, Timestamp

Schema Snapshot for Customers Table (below).

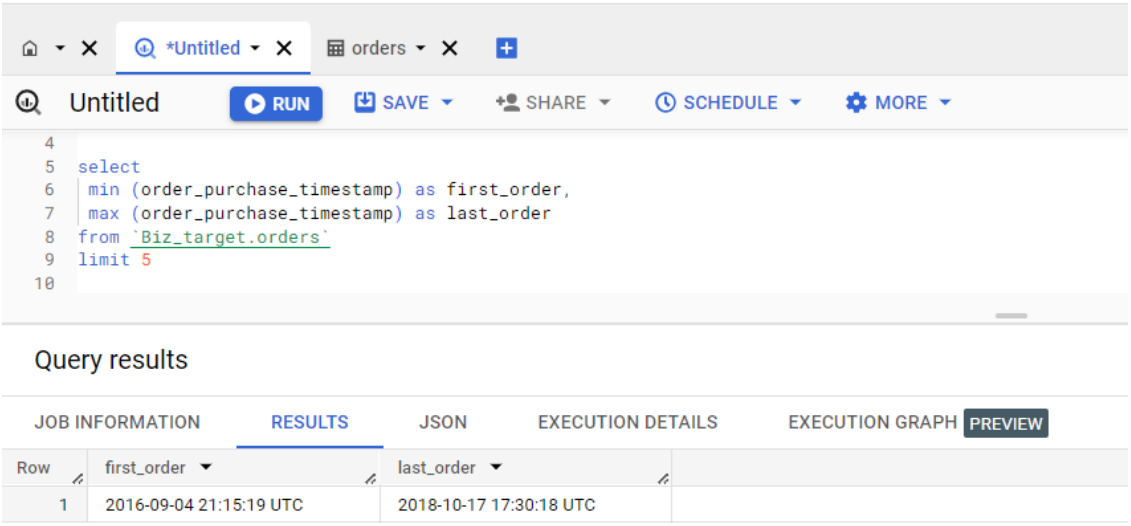


1.2 . Time period for which the data is given

As explained in the Context, the data contains information of 100k orders from 2016 to 2018. We can confirm this time period by sending the below query:

Query:

```
select

 min (order_purchase_timestamp) as first_order,

 max (order_purchase_timestamp) as last_order

from `Biz_target.orders`

limit 5
```



Insight:

The Orders table dataset contains the purchase timestamp of each order. We can find the time period by extracting the first order and last order from the orders table data by using aggregate functions MIN, MAX. MIN gives the first or least value and MAX gives the last or highest value from the dataset

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

Here we need to include the cities and states of customers for the above query which is:

Query:

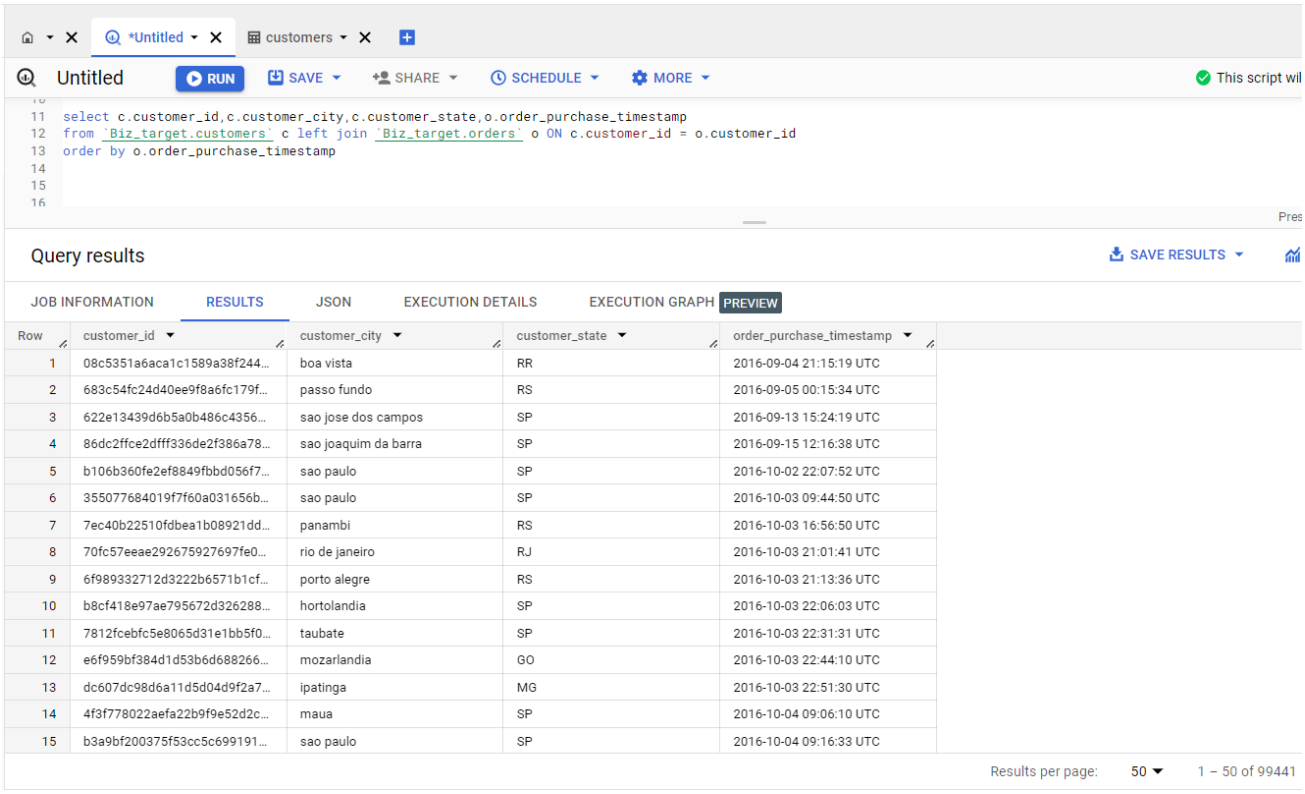
```
select
c.customer_id,c.customer_city,c.customer_state,o.order_purchase_timestamp

from `Biz_target.customers` c left join `Biz_target.orders` o

ON c.customer_id = o.customer_id

order by o.order_purchase_timestamp
```

Insight:

For this query we need ***customers and orders*** tables because the required data can be shown only by intercepting these two tables which in our case, we have used LEFT JOIN. The reason to use LEFT JOIN is because we only need to show the data from customers table only which holds the city and state columns as these are the required data.

From the above query you can see that we have joined these two tables with customer_id as it is the common column from the two tables and selected the customer_id, city, state and purchase timestamp which results with customers city and state at the particular time period.

Snap of Query result:



2. In-depth Exploration:

2.1. 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?

If strictly considering the data in that time period, then NO.

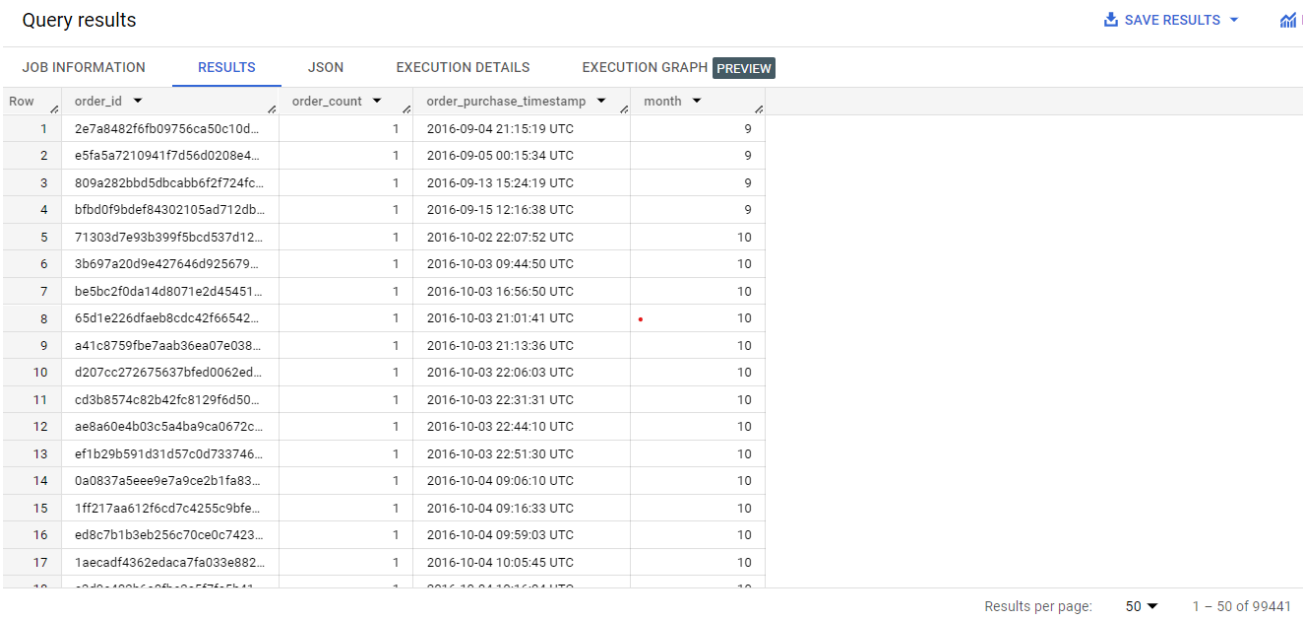
I can say that the e-commerce trend in Brazil has made a good track for a short period. But from the given data in the span of three years from 2016 -2018 the order purchase has grown drastically for quite some time and has faced instant breakdown at the last two months. Which makes us think that there may be a possibility of low purchase rate in the future.

If we consider the previous purchase rate there might be a chance to bring back the growing trend, if there are new products available which makes the people in Brazil happy. There should be particular products available only in the region of Brazil which are suitable for the people by making the lives better or release multiple discounts for products which have been purchased from the previous years.

Retrieving the seasonality with monthly peak purchases data

Through Big Query:

```
select order_id,count(order_id) as order_count,order_purchase_timestamp,
extract (month from order_purchase_timestamp) as month,
from `Biz_target.orders`
group by order_id,order_purchase_timestamp
order by order_purchase_timestamp;
```


Query results

SAVE RESULTS

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

Row	order_id	order_count	order_purchase_timestamp	month
1	2e7a8482f6fb09756ca50c10d...	1	2016-09-04 21:15:19 UTC	9
2	e5fa5a7210941f7d56d0208e4...	1	2016-09-05 00:15:34 UTC	9
3	809a282bbd5dbcabb6f2f724fc...	1	2016-09-13 15:24:19 UTC	9
4	bfbd0f9bdef84302105ad712db...	1	2016-09-15 12:16:38 UTC	9
5	71303d7e93b399f5bcd537d12...	1	2016-10-02 22:07:52 UTC	10
6	3b697a20d9e427646d925679...	1	2016-10-03 09:44:50 UTC	10
7	be5bc2f0da14d8071e2d45451...	1	2016-10-03 16:56:50 UTC	10
8	65d1e226dfaeb8cdc42f66542...	1	2016-10-03 21:01:41 UTC	10
9	a41c8759fbe7aab36ea07e038...	1	2016-10-03 21:13:36 UTC	10
10	d207cc272675637bfed0062ed...	1	2016-10-03 22:06:03 UTC	10
11	cd3b8574c82b42fc8129f6d50...	1	2016-10-03 22:31:31 UTC	10
12	ae8a60e4b03c5a4ba9ca0672c...	1	2016-10-03 22:44:10 UTC	10
13	ef1b29b591d31d57c0d733746...	1	2016-10-03 22:51:30 UTC	10
14	0a0837a5eee9e7a9ce2b1fa83...	1	2016-10-04 09:06:10 UTC	10
15	1ff217aa612f6cd7c4255c9bfe...	1	2016-10-04 09:16:33 UTC	10
16	ed8c7b1b3eb256c70ce0c7423...	1	2016-10-04 09:59:03 UTC	10
17	1aecadf4362edaca7fa033e882...	1	2016-10-04 10:05:45 UTC	10

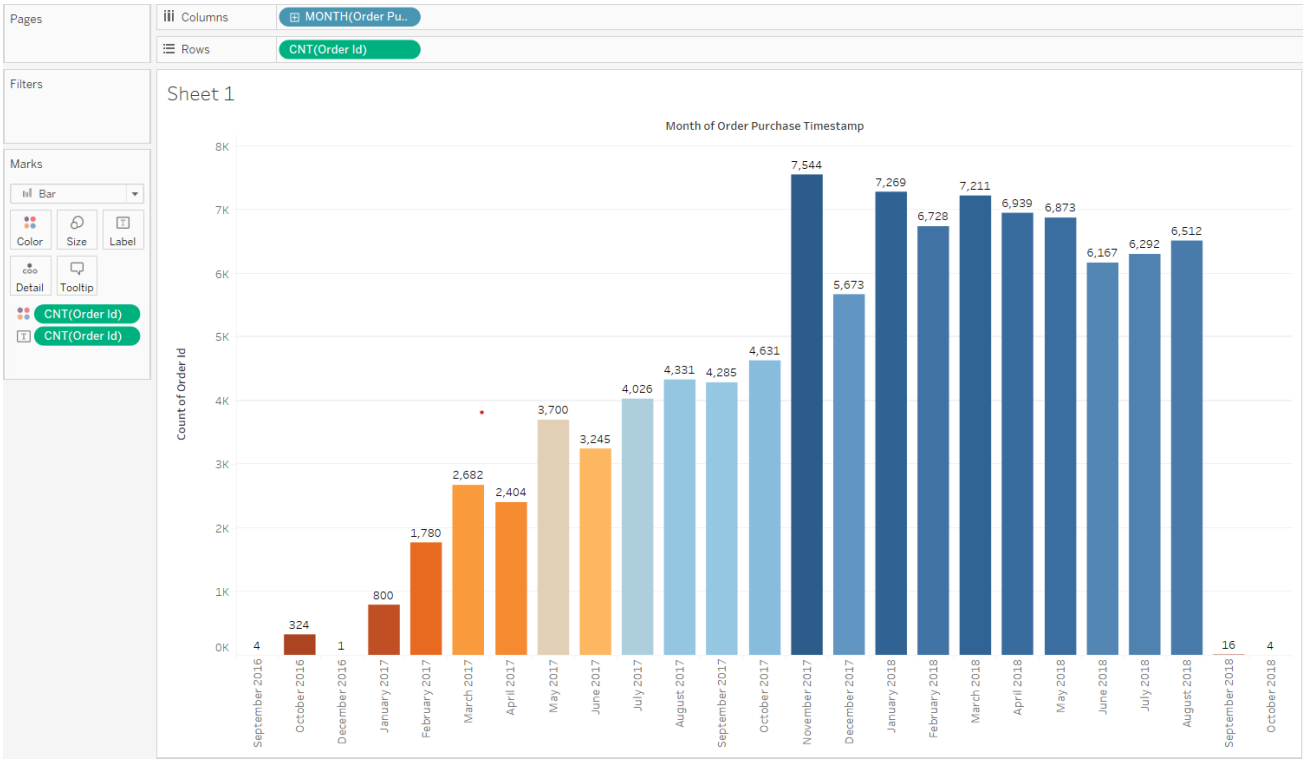
Results per page: 50 1 – 50 of 99441

Insight:

Used the count of order_id with purchase timestamp, extracted monthly count with respect to order_count column. In my query under Month column the same number of months is equal to number orders purchased. I realised that my query is correct after using tableau visualization.

For example, 4 orders were purchased in the 9th month (September). We can find in the below tableau view the first month has 4 orders.

Through Tableau:



After visualizing in tableau, we can see that the peak monthly purchase in the given period is from Nov 2017 to Aug 2018 with November, 2017 as highest with 7544 orders.

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

At **Afternoon time** Brazilian customers tend to make more purchases.

Here I categorized the time as below:

Dawn: 5 – 6

Morning: 6 – 12

Afternoon: 12 – 17

Evening: 17 – 21

Night: 21 - 5

The SQL Query for customers tend to buy at the given time period is

QUERY:

```
select tbl.hour, count(hour) as hour_count
from
(select order_id,order_purchase_timestamp,
extract(hour from datetime(order_purchase_timestamp)) as hour
from `Biz_target.orders`
group by order_id, order_purchase_timestamp) tbl
where hour between 0 and 23
group by tbl.hour
order by hour_count desc;
```

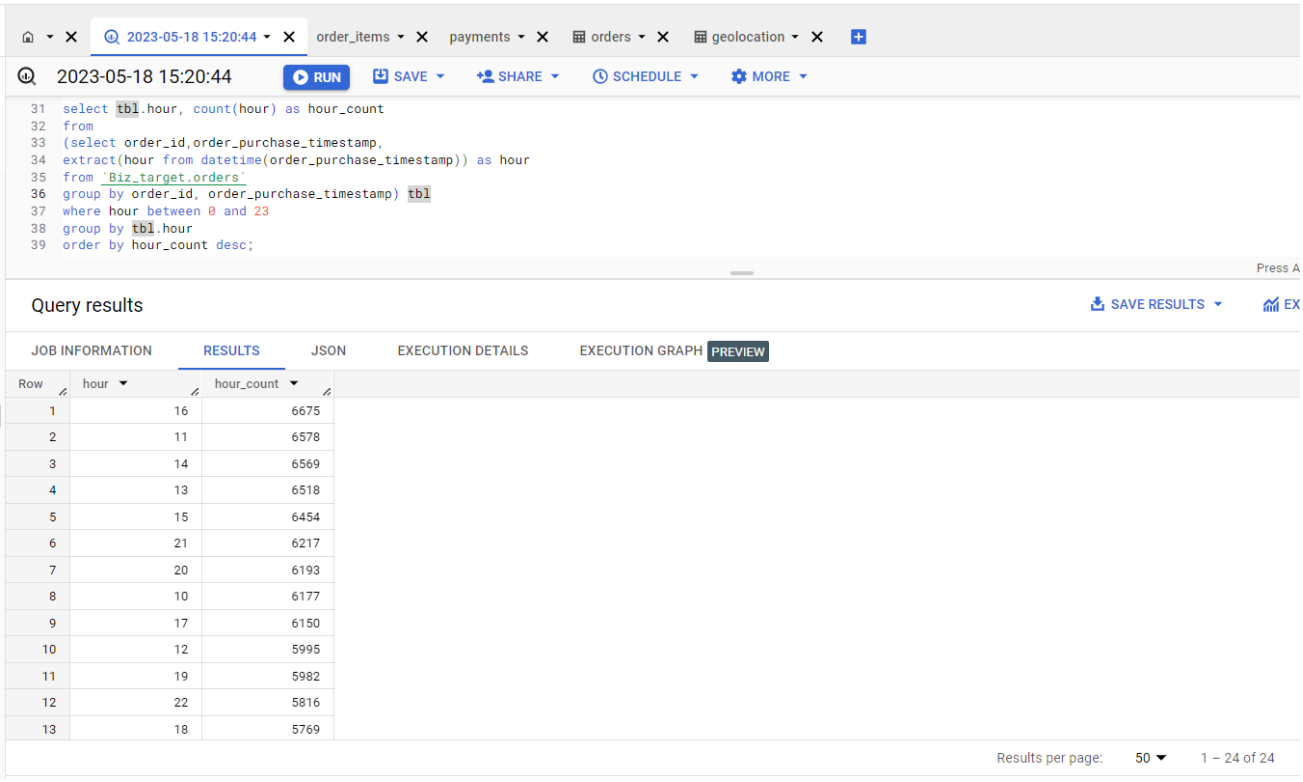
Insights:

From the above query we can see that I have extracted **hour** from the given timestamp and grouped it by order_id and timestamp and completely made the query as a sub query (TBL). And from the sub query I have used the hour column to count the number of hours for each order_id as hour_count. Distinguished the hour from 0 to 23 and grouped it by itself and order by with count from high to low as we need the maximum purchases happened in particular hour.

So, for the maximum purchases in a particular hour, we can finalise the hour with our categorized timeline of dawn to night, which is **Afternoon** at **16:00 hour** with purchase count of **6675. Through Tableau I have cross checked with timestamp(hour) and order (count) and resulted as Afternoon (16:00) as highest which is same.**

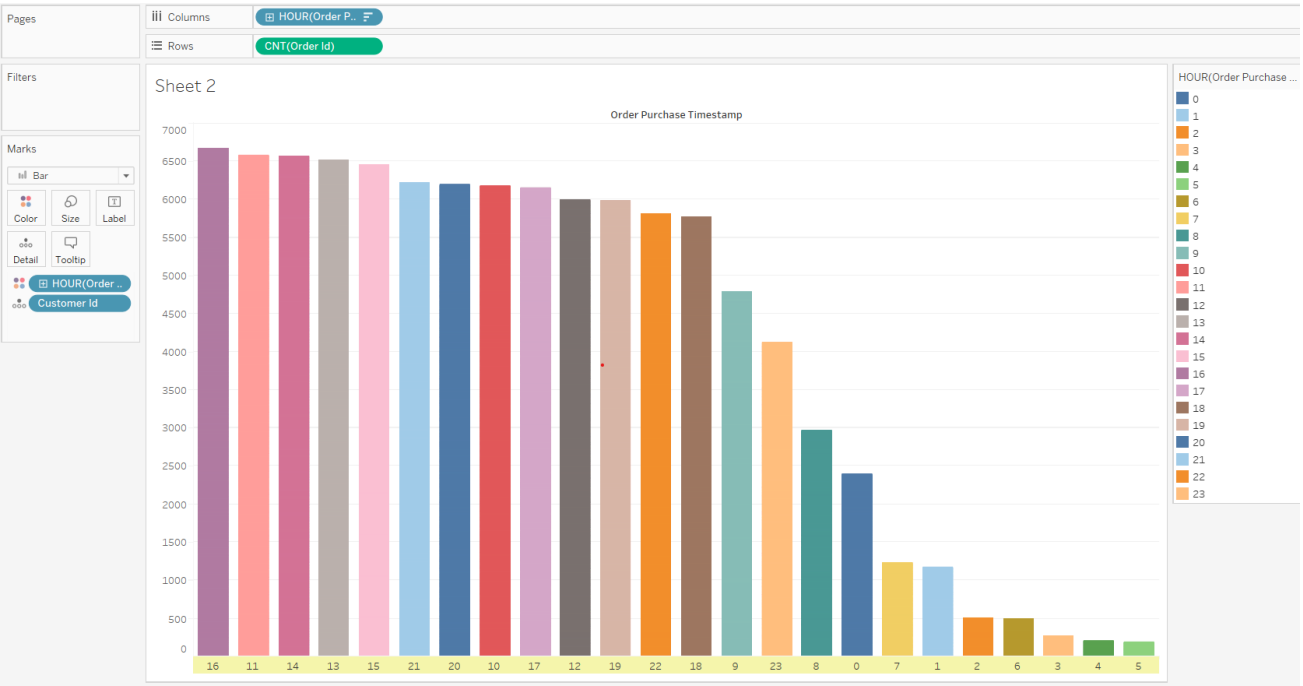
QUERY RESULT:

From Big Query:



Row	hour	hour_count
1	16	6675
2	11	6578
3	14	6569
4	13	6518
5	15	6454
6	21	6217
7	20	6193
8	10	6177
9	17	6150
10	12	5995
11	19	5982
12	22	5816
13	18	5769

From Tableau:



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

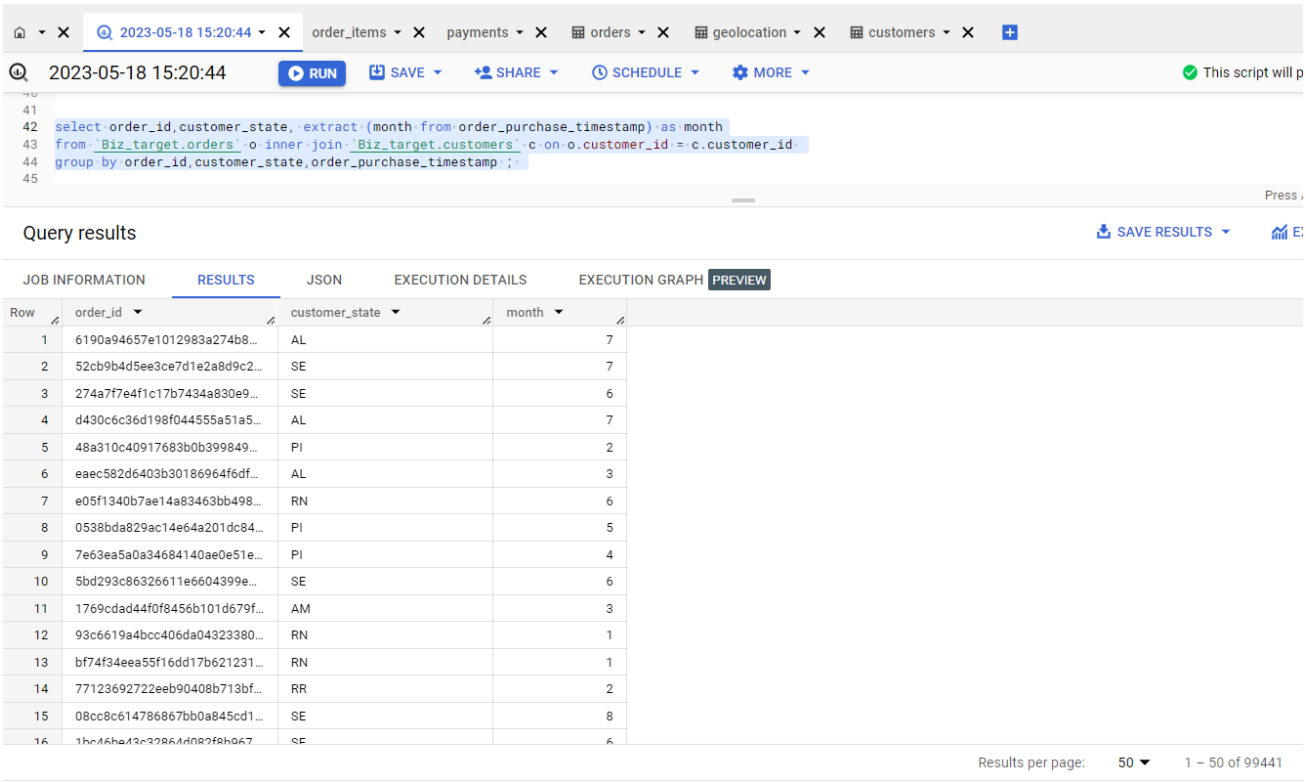
3.1. Get month on month orders by states

Got the data by using inner join for orders table and customers table with customer_id as common column.

Query:

```
select order_id,customer_state, extract (month from
order_purchase_timestamp) as month
from `Biz_target.orders` o inner join `Biz_target.customers` c on
o.customer_id = c.customer_id
group by order_id,customer_state,order_purchase_timestamp ;
```

Query Result:



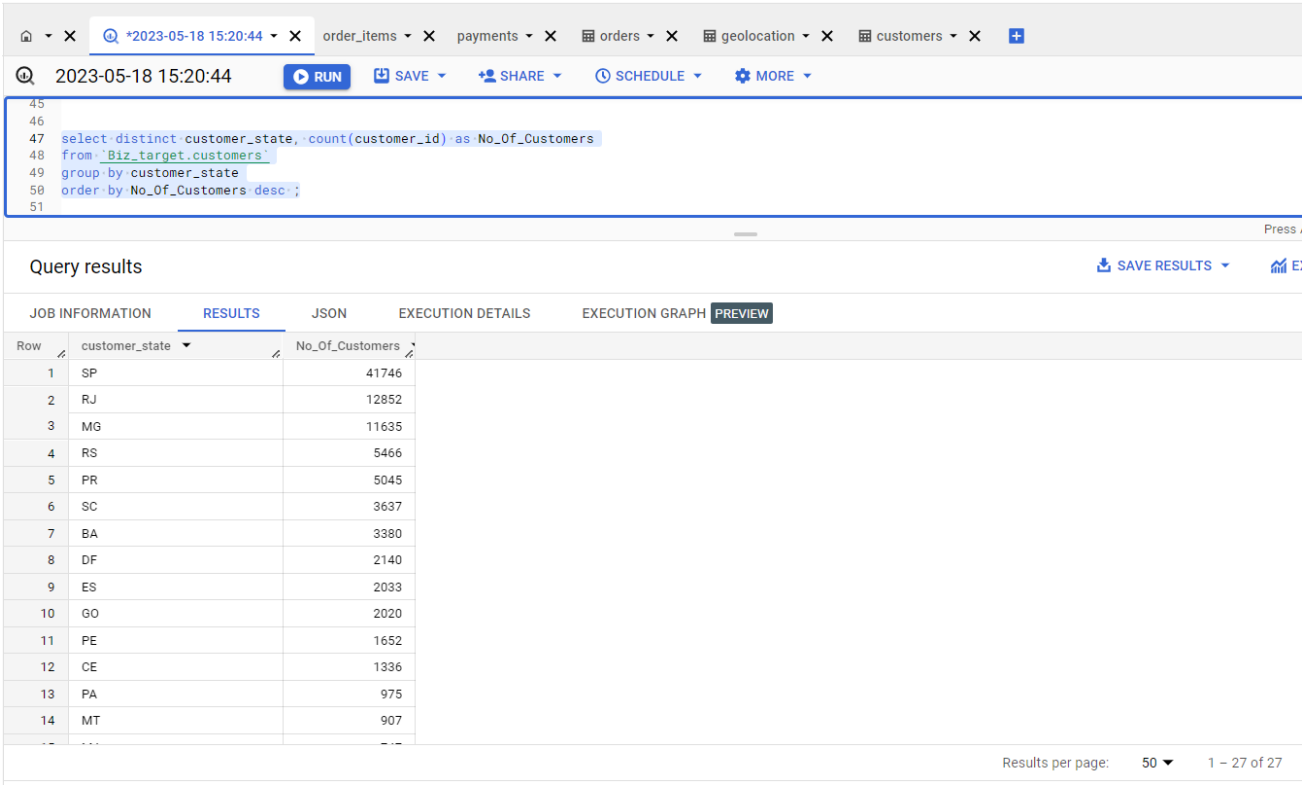
3.2. Distribution of customers across the states in Brazil

Retrieved the data from customers table using count of customer_id for total number of customers and distinct customer_state. Finally used group by for the aligned data of state wise customer count.

SQL Query:

```sql
select distinct customer_state, count(customer_id) as
No_Of_Customers
from `Biz_target.customers`
group by customer_state
order by No_Of_Customers desc ;
```

Query Result:



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

4.1 . 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

SQL Query:

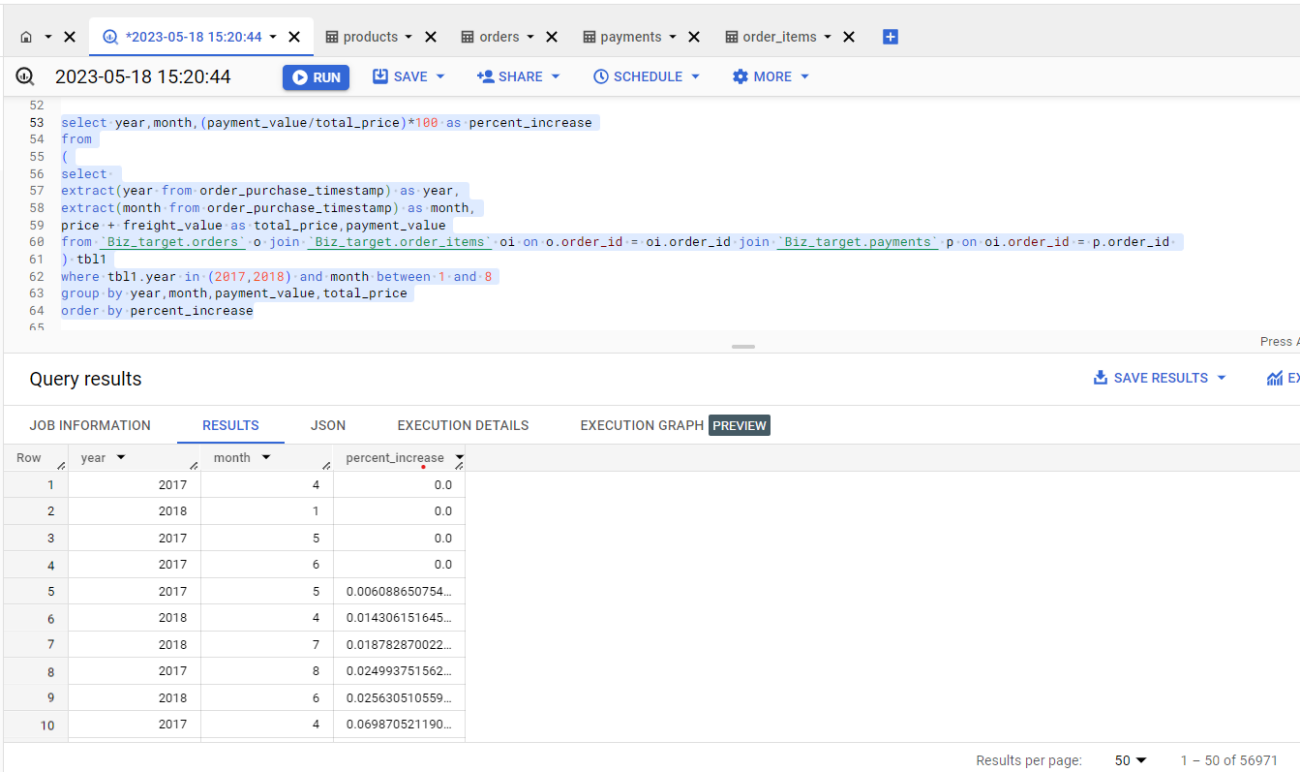
```
select year,month,(payment_value/total_price)*100 as percent_increase
from
(
select
extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month,
price + freight_value as total_price,payment_value
from `Biz_target.orders` o join `Biz_target.order_items` oi on o.order_id
= oi.order_id join `Biz_target.payments` p on oi.order_id = p.order_id
) tbl1
where tbl1.year in (2017,2018) and month between 1 and 8
group by year,month,payment_value,total_price
order by percent_increase
```

Insights:

For this question, I have used 3 tables which are orders table for extracting year and month and order_items table for using price and freight_value as total_price and payments table for using payment_value.

So, with all the data I have for providing the percentage in increase cost using payment_value divided bytotal_price multiplied by 100. And finally filtered using where for year and month as asked.

Query Results:



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

SQL Query:

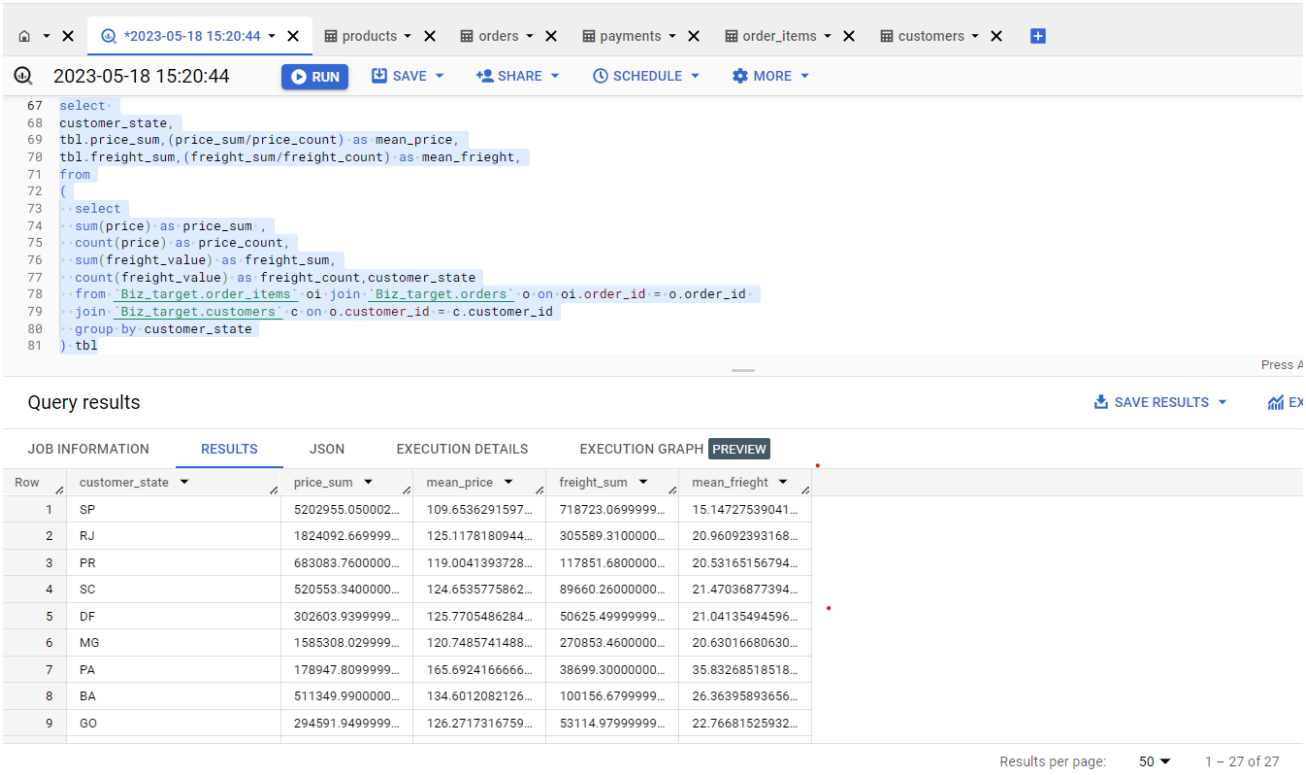
```
select
customer_state,
tbl.price_sum,(price_sum/price_count) as mean_price,
tbl.freight_sum,(freight_sum/freight_count) as mean_frieght,
from
(
  select
  sum(price) as price_sum ,
  count(price) as price_count,
  sum(freight_value) as freight_sum,
  count(freight_value) as freight_count,customer_state
  from `Biz_target.order_items` oi join `Biz_target.orders` o on
oi.order_id = o.order_id
  join `Biz_target.customers` c on o.customer_id = c.customer_id
  group by customer_state
) tbl
```

Insights:

For the mean and sum for price and freight value columns, I have taken 3 tables which are order_ items table for calculating the mean and sum for price and freight value columns and customers table for grouping by customer_ state column and orders table for intersecting the above two tables.

Here, I have calculated the mean by using sum of respective column with count of the rows for the same column.

Query Result:



5. Analysis on sales, freight and delivery time

5.1 . Calculate days between purchasing, delivering and estimated delivery

SQL Query:

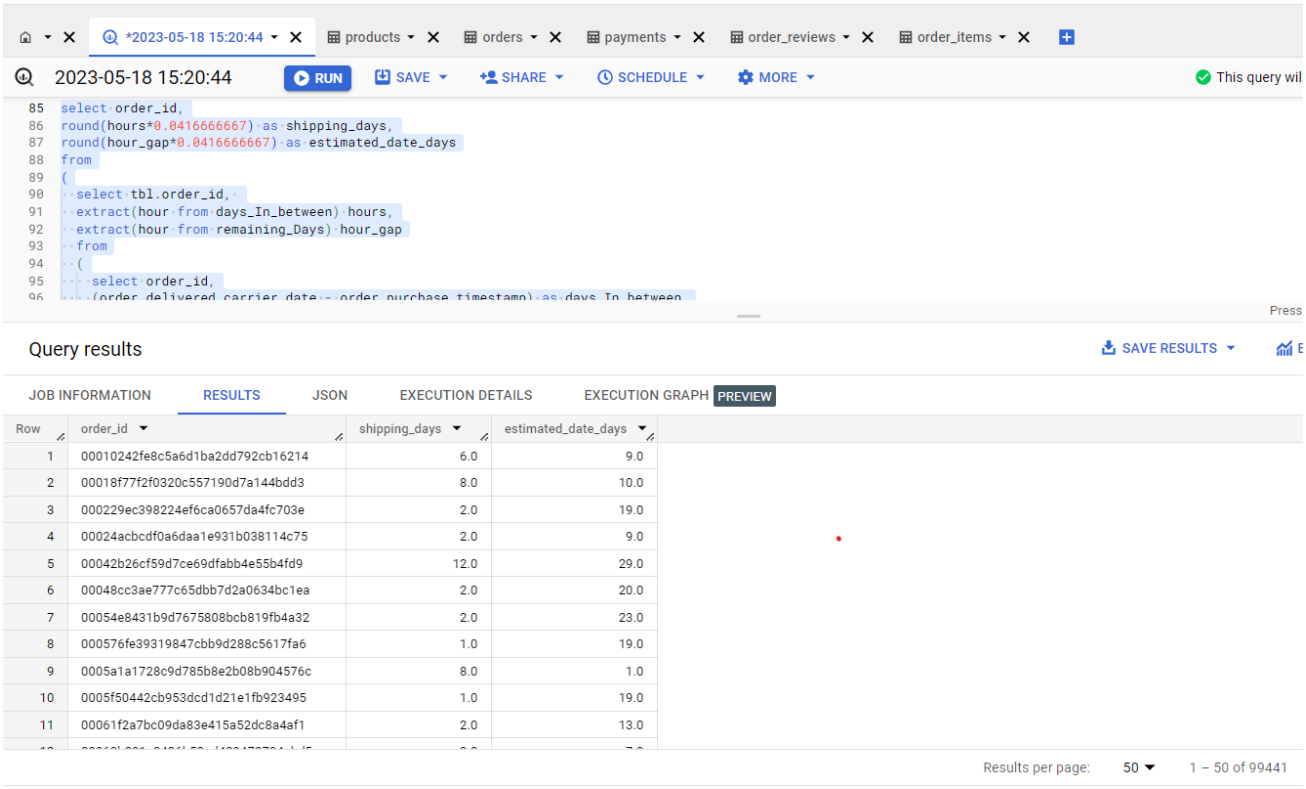
```
select order_id,
round(hours*0.0416666667) as shipping_days,
round(hour_gap*0.0416666667) as estimated_date_days
from
(
  select tbl.order_id,
  extract(hour from days_In_between) hours,
  extract(hour from remaining_Days) hour_gap
  from
  (
    select order_id,
    (order_delivered_carrier_date - order_purchase_timestamp) as days_In_between,
    (order_estimated_delivery_date - order_delivered_carrier_date)as
remaining_Days
    from `Biz_target.orders`
  )tbl
)tbl2
order by order_id
```

Insights:

For this problem I have used 3 columns which are carrier_ date, purchase_ timestamp and delivered_ carrier_ date. Simply subtracted the columns for getting the remaining days.

But got the data in hours (E.g.: 0-0-0 126:13:12), so extracted only hours from the first sub query(tbl) and converted those hours into days (1 hour = `0.0416666667` days, multiply hours with the value) from the second sub query which gave us the required data in days.

Query Results:



Row	order_id	shipping_days	estimated_date_days
1	00010242fe8c5a6d1ba2dd792cb16214	6.0	9.0
2	00018f77f2f0320c557190d7a144bdd3	8.0	10.0
3	000229ec398224ef6ca0657da4fc703e	2.0	19.0
4	00024acbcdf0a6daa1e931b038114c75	2.0	9.0
5	00042b26cf59d7ce69dfabb4e55b4fd9	12.0	29.0
6	00048cc3ae777c65dbb7d2a0634bc1ea	2.0	20.0
7	00054e8431b9d7675808bcb819fb4a32	2.0	23.0
8	000576fe39319847cbb9d288c5617fa6	1.0	19.0
9	0005a1a1728c9d785b8e2b08b904576c	8.0	1.0
10	0005f50442cb953dcd1d21e1fb923495	1.0	19.0
11	00061f2a7bc09da83e415a52dc8a4af1	2.0	13.0

5.2. Find time_to_delivery & diff_estimated_delivery.

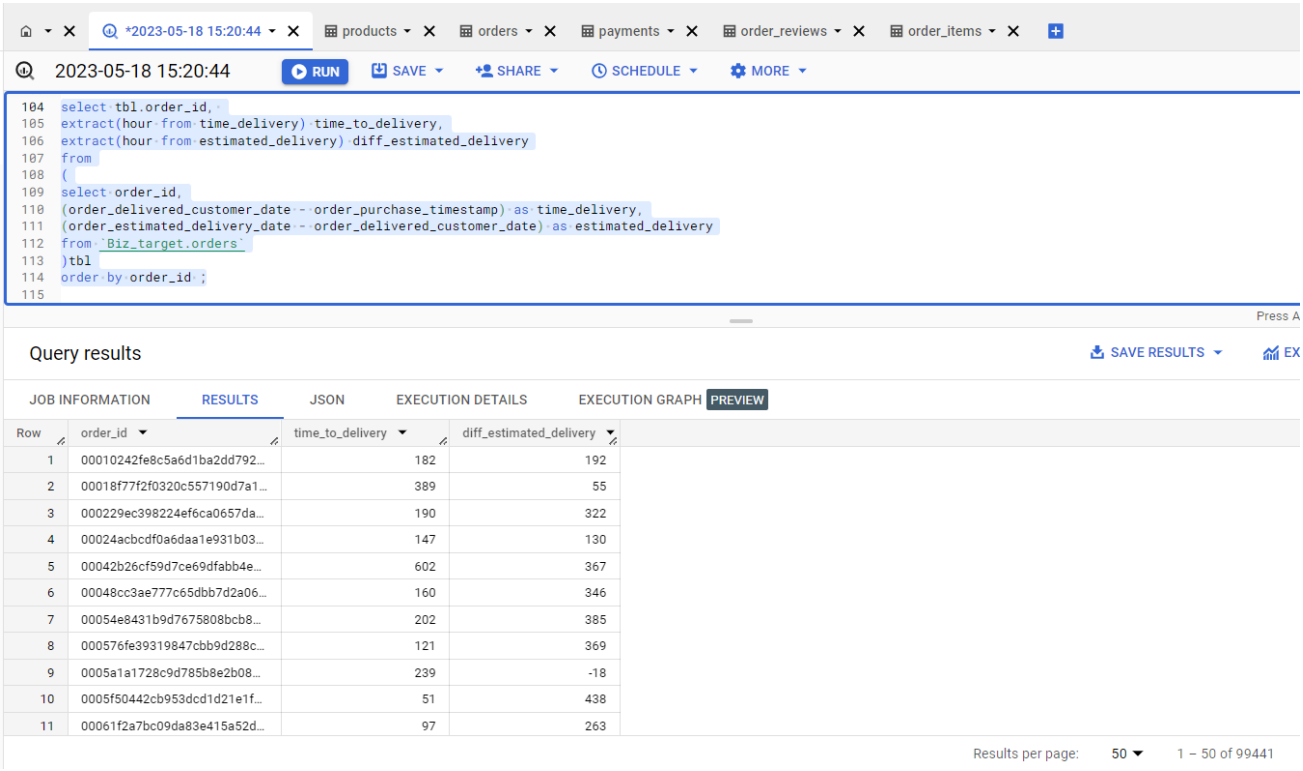
SQL Query:

```
select tbl.order_id,
extract(hour from time_delivery) time_to_delivery,
extract(hour from estimated_delivery) diff_estimated_delivery
from
(
select order_id,
(order_delivered_customer_date - order_purchase_timestamp) as
time_delivery,
(order_estimated_delivery_date - order_delivered_customer_date) as
estimated_delivery
from `Biz_target.orders`
)tbl
order by order_id ;
```

Insights:

From the given formula, used orders table and retrieved data was giving the time in hours format from the first query. So created it as subquery and extracted only hours from the data. Finally ordered it by Order_id.

Query Results:



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

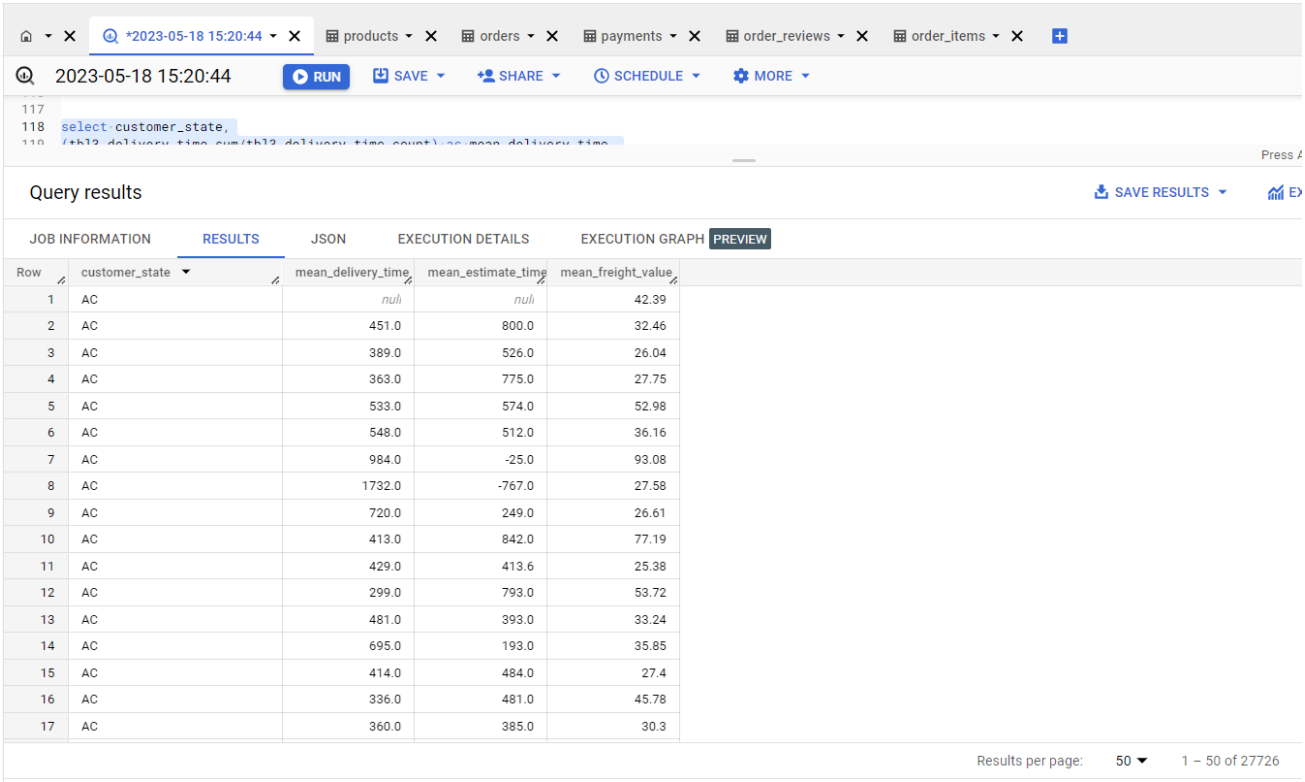
SQL Query:

```
select customer_state,
(tbl3.delivery_time_sum/tbl3.delivery_time_count) as mean_delivery_time,
(tbl3.delivery_estimate_sum/tbl3.delivery_estimate_count) as mean_estimate_time,
(tbl3.freight_sum/tbl3.freight_count) as mean_freight_value
from
  (
  select customer_state,tbl2.freight_count,tbl2.freight_sum,
  sum(time_to_delivery) as delivery_time_sum, count(time_to_delivery) as
delivery_time_count,
  sum(diff_estimated_delivery) as delivery_estimate_sum,
count(diff_estimated_delivery) as delivery_estimate_count
  from
    (
    select tbl1.customer_state,tbl1.freight_count,tbl1.freight_sum,
    extract(hour from time_delivery) time_to_delivery,
    extract(hour from estimated_delivery) diff_estimated_delivery
    from
      (
      select customer_state,
      sum(freight_value) as freight_sum,
      count(freight_value) as freight_count,
      (order_delivered_customer_date - order_purchase_timestamp) as
time_delivery,
      (order_estimated_delivery_date - order_delivered_customer_date) as
estimated_delivery
      from `Biz_target.order_items` oi join `Biz_target.orders` o on oi.order_id
= o.order_id
      join `Biz_target.customers` c on o.customer_id = c.customer_id
      group by customer_state, order_delivered_customer_date,
order_purchase_timestamp, order_estimated_delivery_date
      ) tbl1
    ) tbl2
  group by customer_state,tbl2.freight_count,tbl2.freight_sum
  ) tbl3
order by customer_state ;
```

Insights:

For this problem I have considered previous queries from problems 4.2 & 5.2 and created 3rd sub query for calculating the mean for freight value, delivery time, estimated time. As we cannot calculate the newly created columns on the same query itself, we got 3 subqueries for these calculations.

Query Results:



5.4. Sort the data to get the following:

1. 5.5. Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5

SQL Query:

Highest average

```
select
customer_state,
(freight_sum/freight_count) as mean_freight,
from
(
  select
  sum(freight_value) as freight_sum,
  count(freight_value) as freight_count,customer_state
  from `Biz_target.order_items` oi join `Biz_target.orders` o on
oi.order_id = o.order_id
  join `Biz_target.customers` c on o.customer_id = c.customer_id
  group by customer_state
) tbl
order by mean_freight desc
limit 5 ;
```

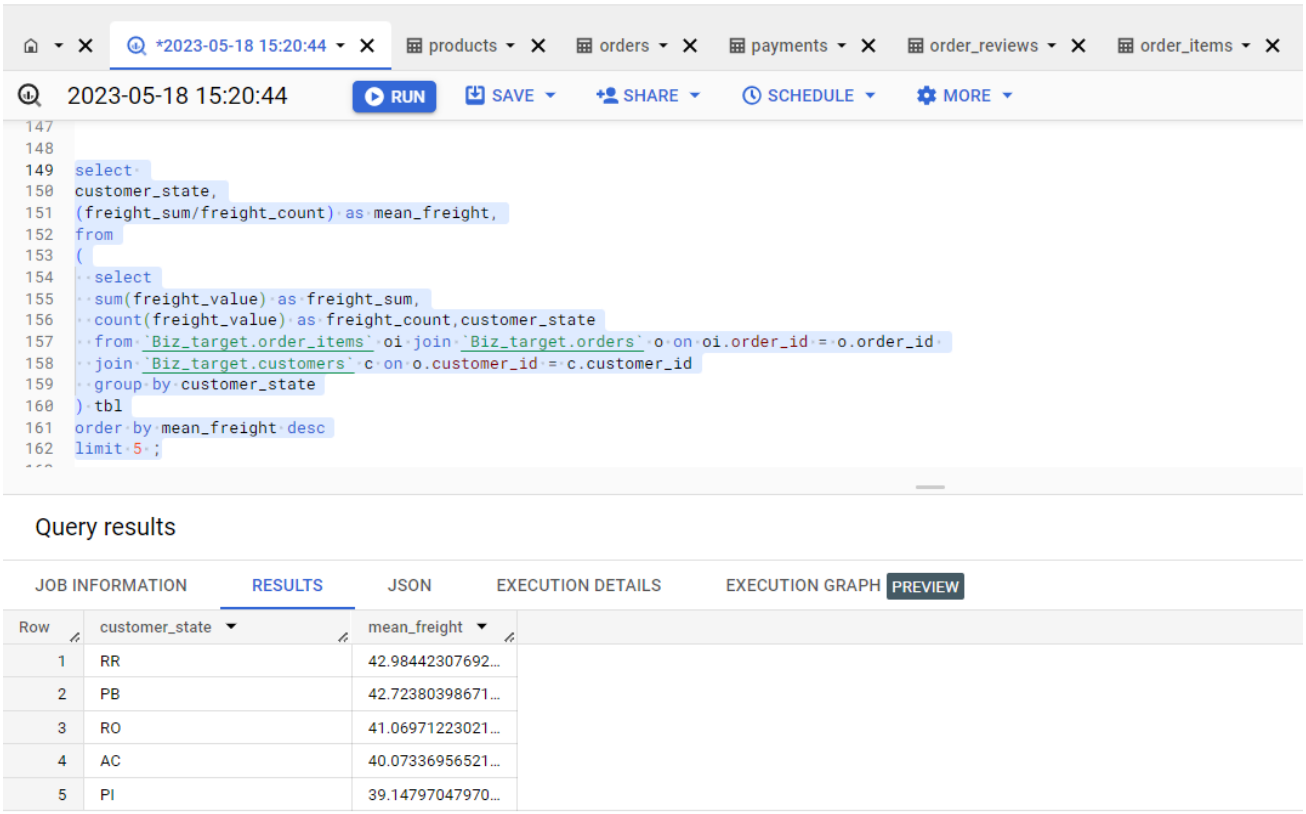
SQL Query:

Lowest average

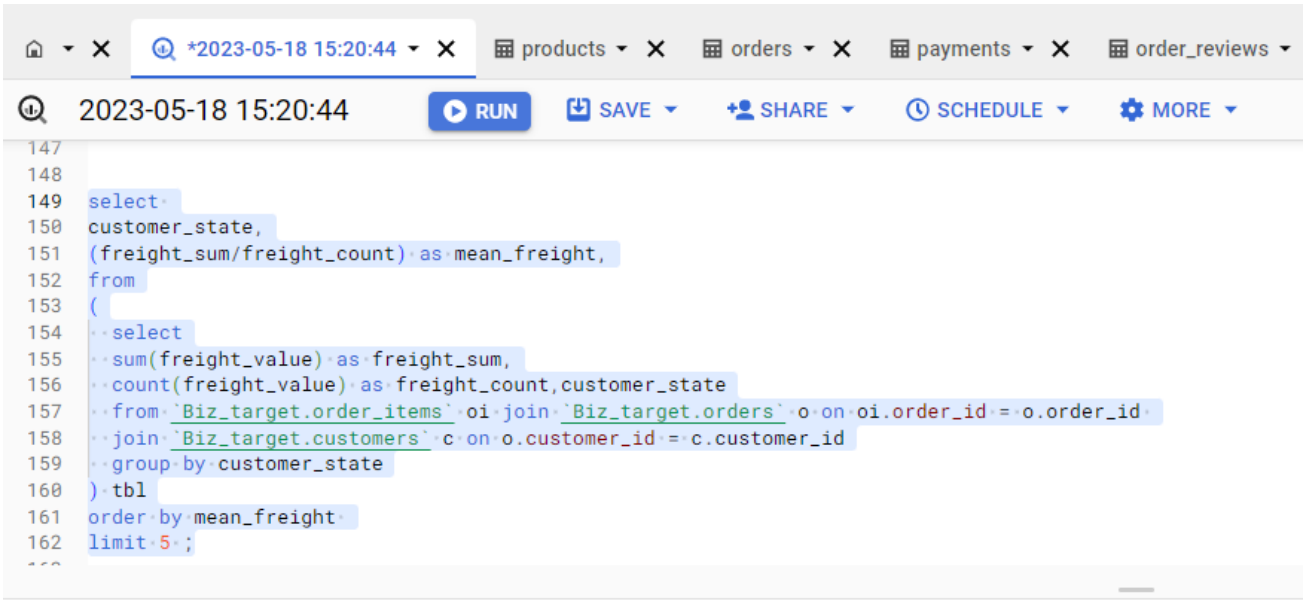
```
select
customer_state,
(freight_sum/freight_count) as mean_freight,
from
(
  select
  sum(freight_value) as freight_sum,
  count(freight_value) as freight_count,customer_state
  from `Biz_target.order_items` oi join `Biz_target.orders` o on
oi.order_id = o.order_id
  join `Biz_target.customers` c on o.customer_id = c.customer_id
  group by customer_state
) tbl
order by mean_freight
limit 5 ;
```

Query Results:

Highest



Lowest



Query results

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

Row	customer_state	mean_freight
1	SP	15.14727539041...
2	PR	20.53165156794...
3	MG	20.63016680630...
4	RJ	20.96092393168...
5	DF	21.04135494596...

2. 5.6. Top 5 states with highest/lowest average time to delivery

SQL Query: highest

```
select customer_state,
(tbl3.delivery_time_sum/tbl3.delivery_time_count) as Average_delivery_time,
from
  (
  select customer_state,
  sum(time_to_delivery) as delivery_time_sum, count(time_to_delivery) as
delivery_time_count
  from
    (
    select tbl1.customer_state,
    extract(hour from time_delivery) time_to_delivery
    from
      (
      select customer_state,
      (order_delivered_customer_date - order_purchase_timestamp) as
time_delivery,
      from `Biz_target.orders` o join `Biz_target.customers` c on o.customer_id =
c.customer_id
      group by customer_state, order_delivered_customer_date,
order_purchase_timestamp
      ) tbl1
    ) tbl2
  group by customer_state
  ) tbl3
order by Average_delivery_time desc
limit 5 ;
```

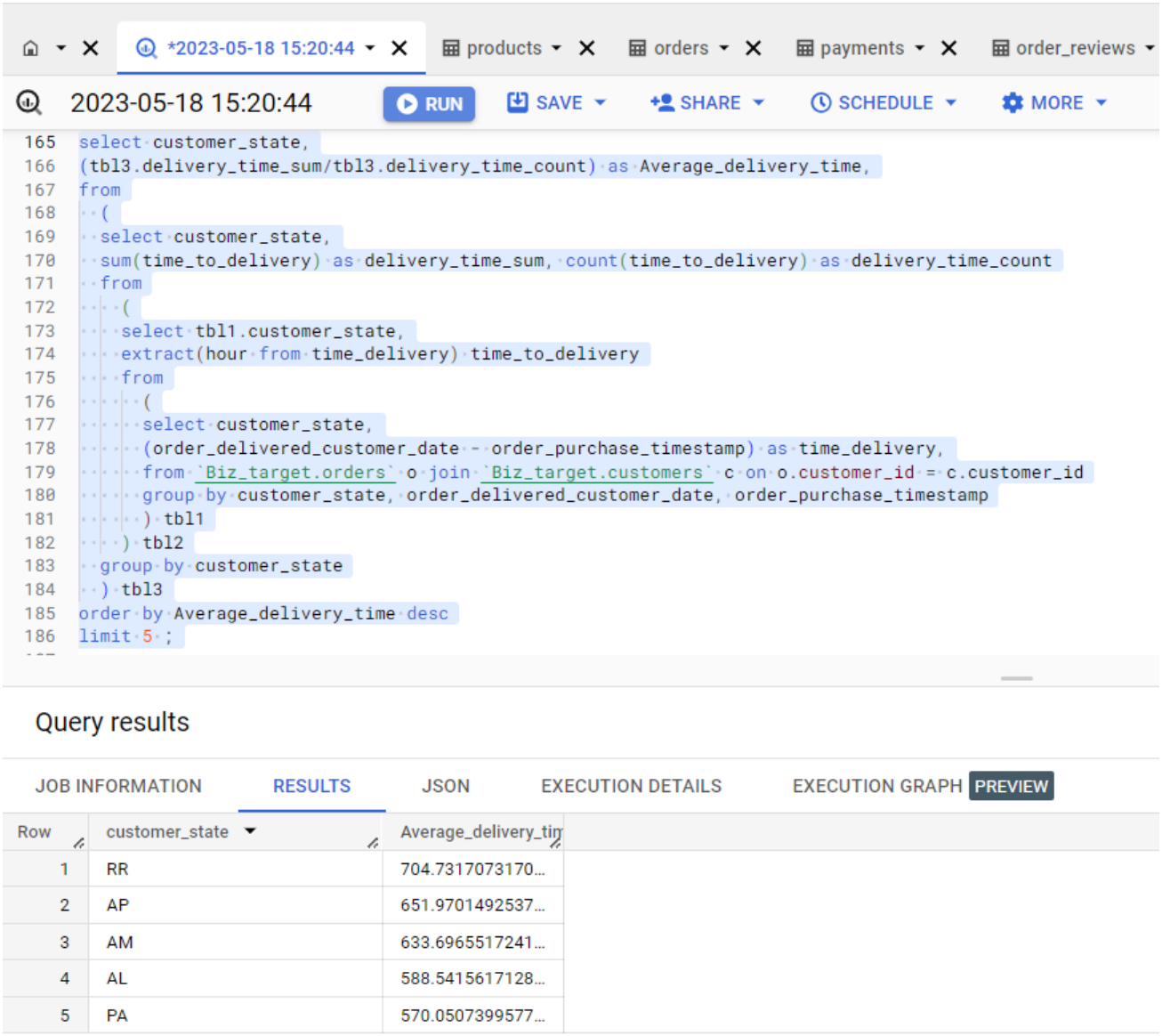
SQL Query: lowest

```
select customer_state,
(tbl3.delivery_time_sum/tbl3.delivery_time_count) as Average_delivery_time,
from
  (
  select customer_state,
  sum(time_to_delivery) as delivery_time_sum, count(time_to_delivery) as
delivery_time_count
  from
    (
    select tbl1.customer_state,
    extract(hour from time_delivery) time_to_delivery
    from
      (
      select customer_state,
      (order_delivered_customer_date - order_purchase_timestamp) as
time_delivery,
      from `Biz_target.orders` o join `Biz_target.customers` c on o.customer_id =
c.customer_id
```

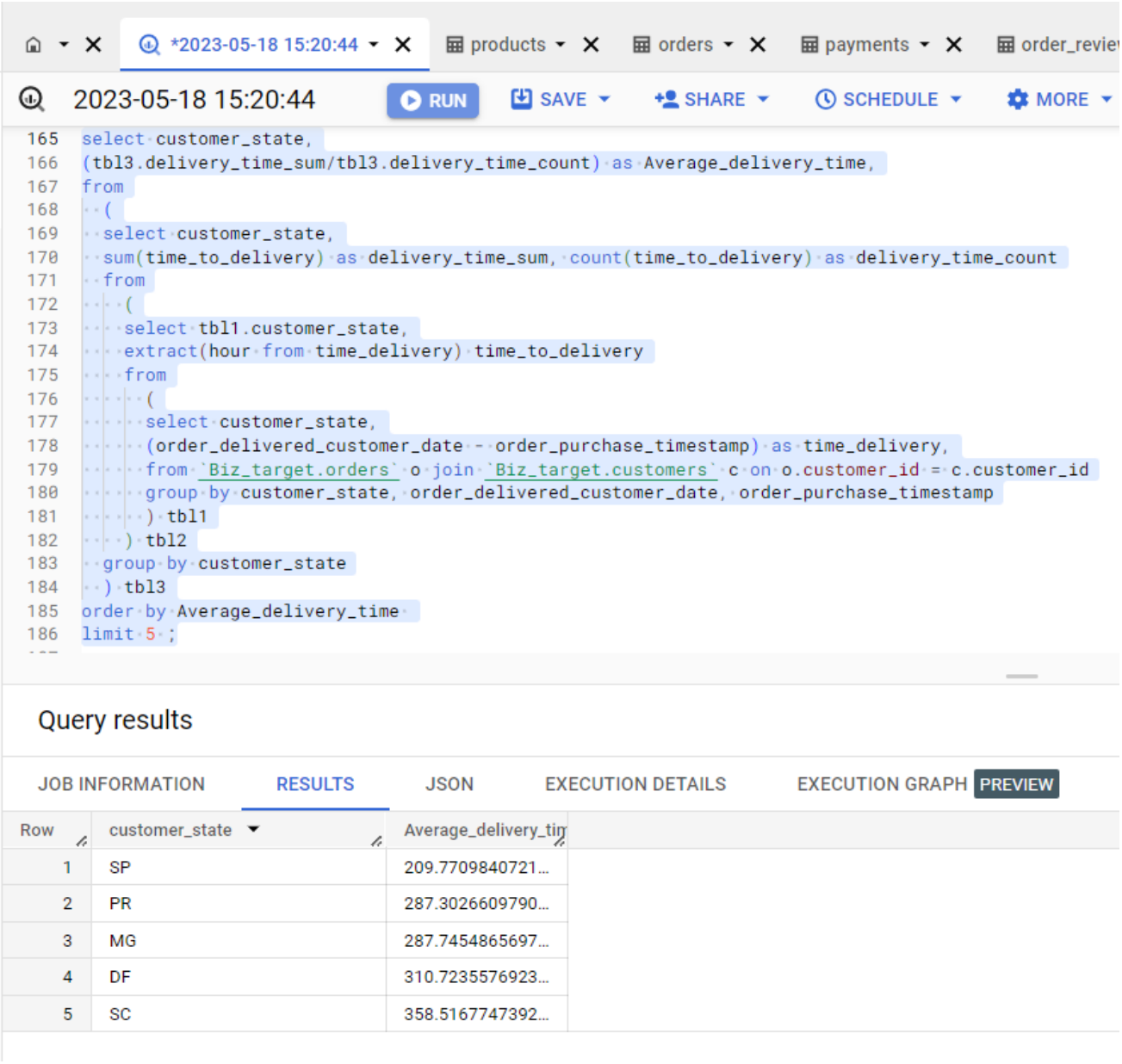
```
      group by customer_state, order_delivered_customer_date,
order_purchase_timestamp
      ) tbl1
    ) tbl2
  group by customer_state
  ) tbl3
order by Average_delivery_time
limit 5 ;
```

Query Results:

Highest



Lowest

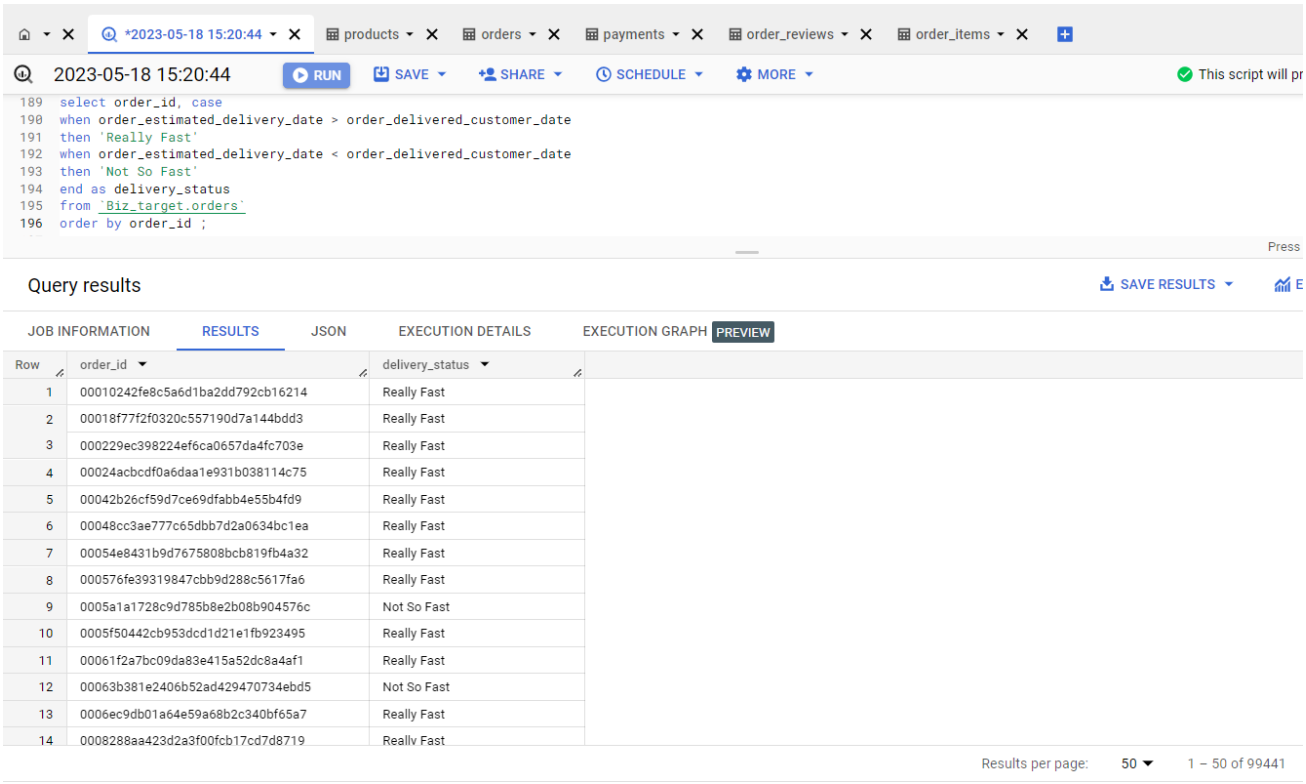


3. 5.7. Top 5 states where delivery is really fast/ not so fast compared to estimated date

SQL Query:

```
select order_id, case
when order_estimated_delivery_date > order_delivered_customer_date
then 'Really Fast'
when order_estimated_delivery_date < order_delivered_customer_date
then 'Not So Fast'
end as delivery_status
from `Biz_target.orders`
order by order_id
```

Query Results:



6. Payment type analysis:

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

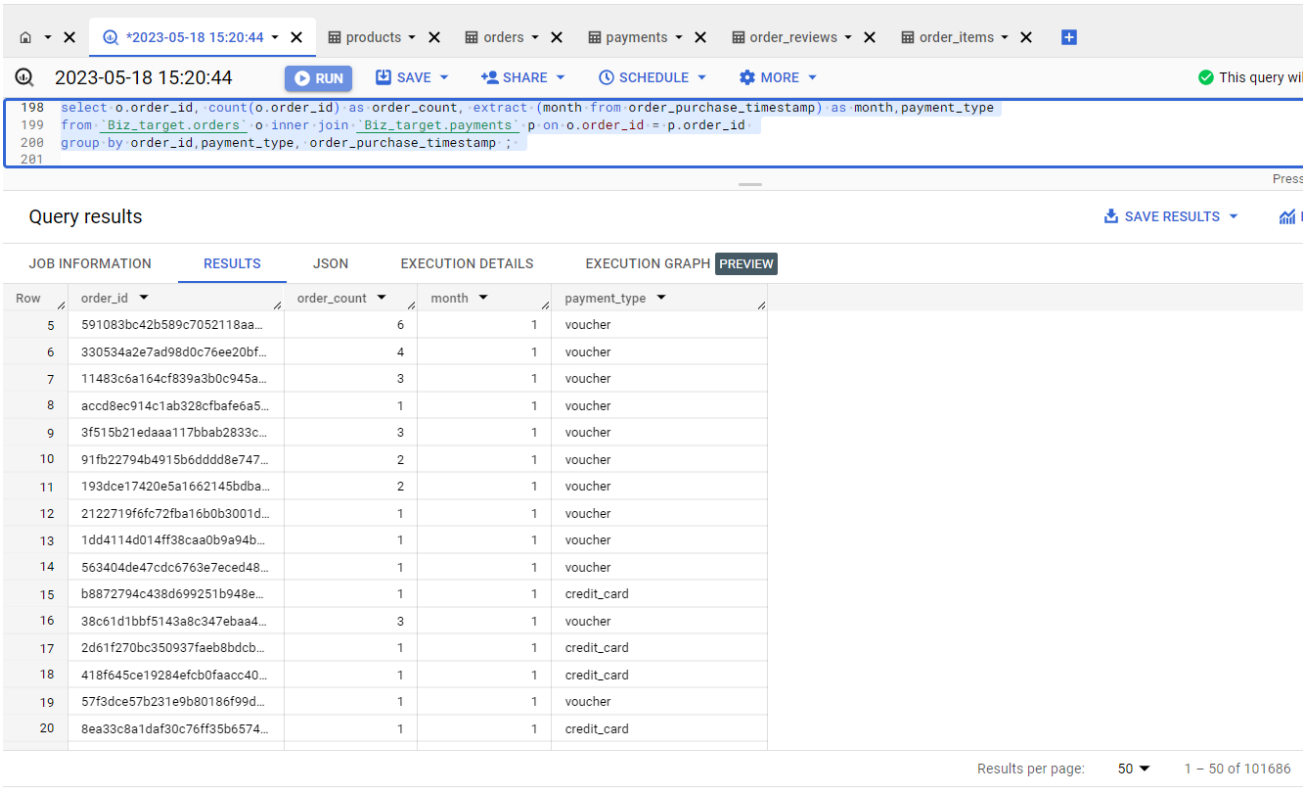
SQL Query:

```
select o.order_id, count(o.order_id) as order_count, extract (month from
order_purchase_timestamp) as month,payment_type
from `Biz_target.orders` o inner join `Biz_target.payments` p on o.order_id =
p.order_id
group by order_id,payment_type, order_purchase_timestamp ;
```

Insights:

Here I have extracted month and count of orders from orders table and payment_ type column from payments table. Using join for orders and payments tables and grouping the columns respectively

Query Results:



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

SQL Query:

```
select o.order_id, count(o.order_id) as order_count,payment_installments
from `Biz_target.orders` o inner join `Biz_target.payments` p on
o.order_id = p.order_id
group by order_id,payment_installments, order_purchase_timestamp ;
```

Query Results: