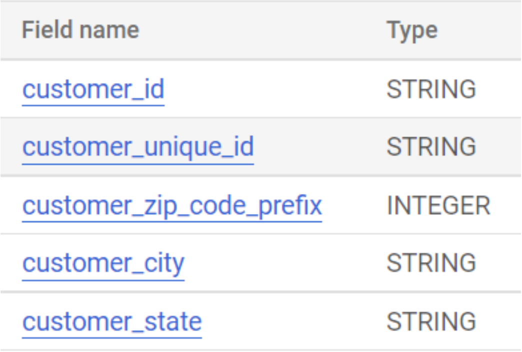
Q1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset

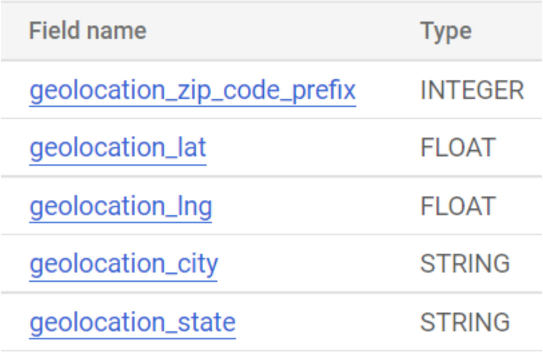
1. Data type of columns in a table

- CUSTOMERS TABLE



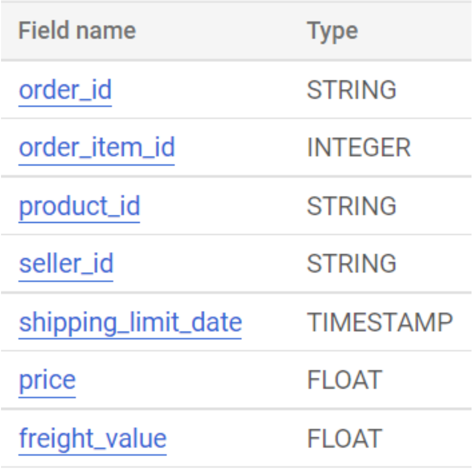
Field name	Type
customer_id	STRING
customer_unique_id	STRING
customer_zip_code_prefix	INTEGER
customer_city	STRING
customer_state	STRING

- GEOLOCATION TABLE



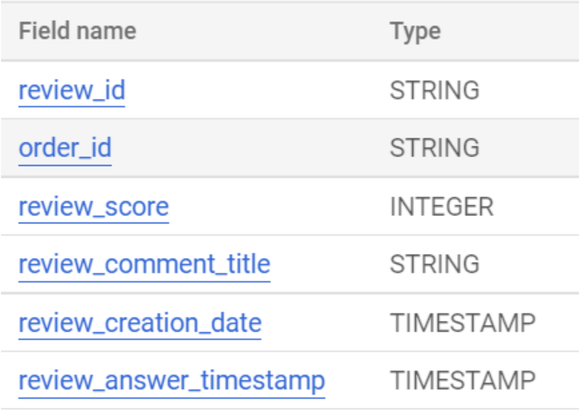
Field name	Type
geolocation_zip_code_prefix	INTEGER
geolocation_lat	FLOAT
geolocation_lng	FLOAT
geolocation_city	STRING
geolocation_state	STRING

- ORDER_ITEMS TABLE



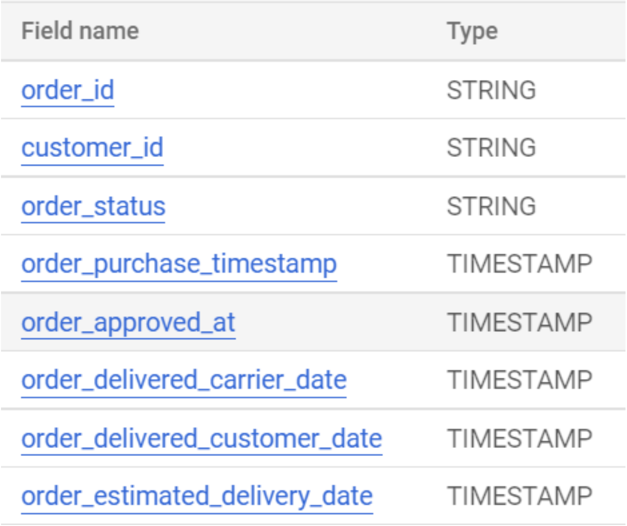
Field name	Type
order_id	STRING
order_item_id	INTEGER
product_id	STRING
seller_id	STRING
shipping_limit_date	TIMESTAMP
price	FLOAT
freight_value	FLOAT

- ORDER_REVIEWS TABLE



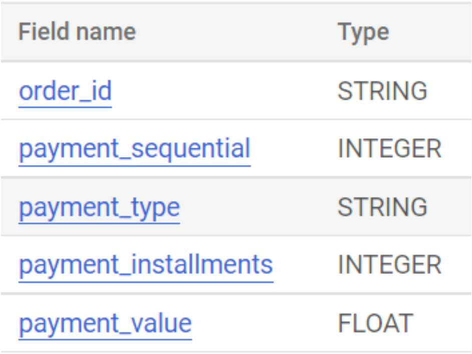
Field name	Type
review_id	STRING
order_id	STRING
review_score	INTEGER
review_comment_title	STRING
review_creation_date	TIMESTAMP
review_answer_timestamp	TIMESTAMP

- ORDERS TABLE



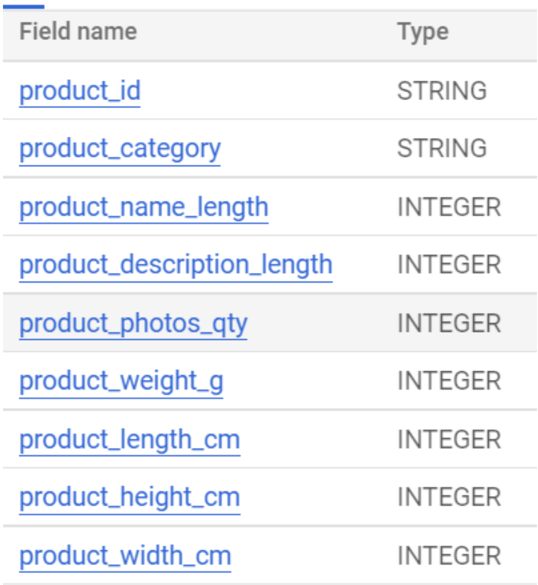
Field name	Type
order_id	STRING
customer_id	STRING
order_status	STRING
order_purchase_timestamp	TIMESTAMP
order_approved_at	TIMESTAMP
order_delivered_carrier_date	TIMESTAMP
order_delivered_customer_date	TIMESTAMP
order_estimated_delivery_date	TIMESTAMP

- PAYMENTS TABLE



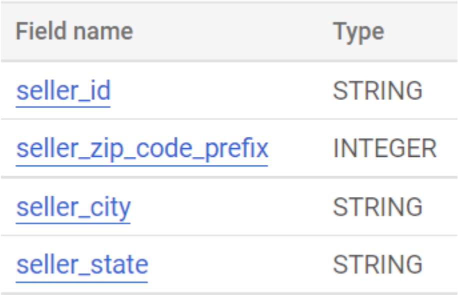
Field name	Type
order_id	STRING
payment_sequential	INTEGER
payment_type	STRING
payment_installments	INTEGER
payment_value	FLOAT

- PRODUCTS TABLE



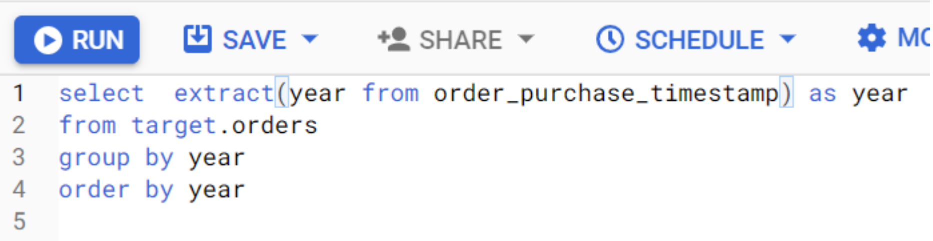
Field name	Type
product_id	STRING
product_category	STRING
product_name_length	INTEGER
product_description_length	INTEGER
product_photos_qty	INTEGER
product_weight_g	INTEGER
product_length_cm	INTEGER
product_height_cm	INTEGER
product_width_cm	INTEGER

- SELLERS TABLE

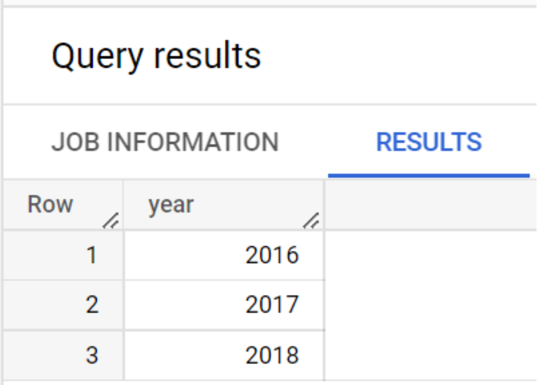


Field name	Type
seller_id	STRING
seller_zip_code_prefix	INTEGER
seller_city	STRING
seller_state	STRING

2. Time period for which the data is given



```
select  extract(year from order_purchase_timestamp) as year
from target.orders
group by year
order by year
```

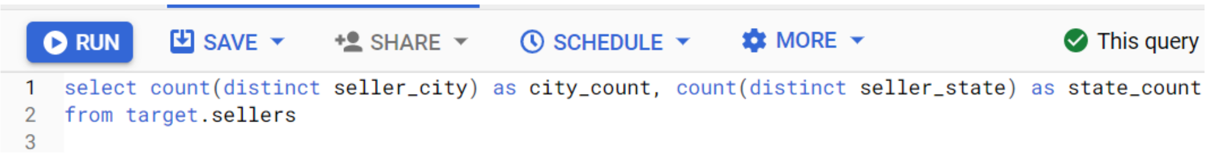


Row	year
1	2016
2	2017
3	2018

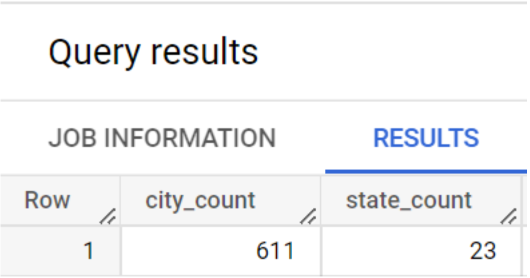
- **The time period of data in the dataset is from 2016 to 2018.**

3. Cities and States covered in the dataset

The count of cities and states of seller in the dataset.



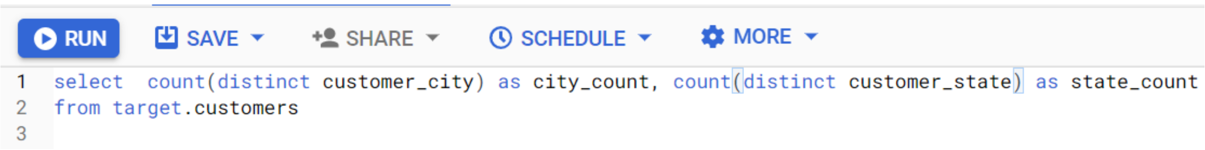
```sql
select count(distinct seller_city) as city_count, count(distinct seller_state) as state_count
from target.sellers
```



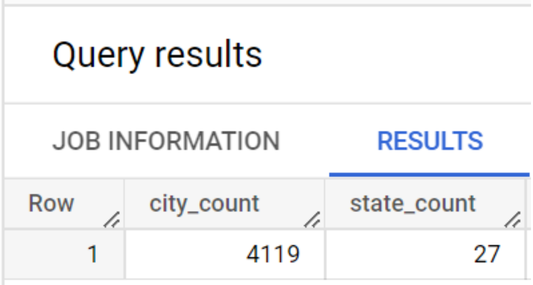
Row	city_count	state_count
1	611	23

- **There are sellers from 611 different cities and 23 different states.**

The count of cities and states of customers in the dataset.



```sql
select  count(distinct customer_city) as city_count, count(distinct customer_state) as state_count
from target.customers
```

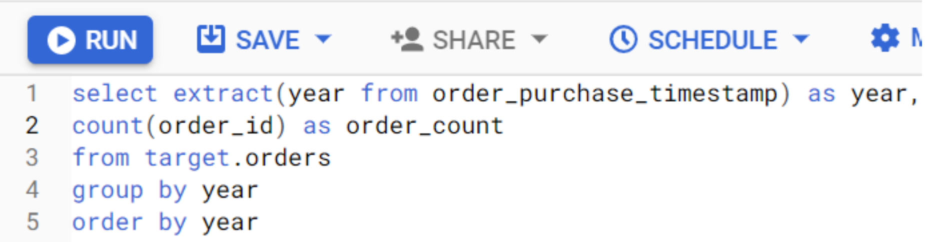


Row	city_count	state_count
1	4119	27

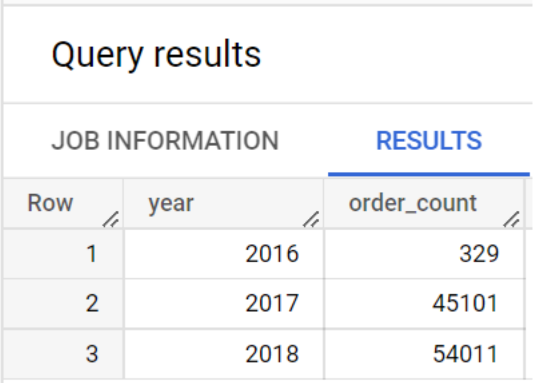
- **There are customers from 4119 different cities and 27 different states.**

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?

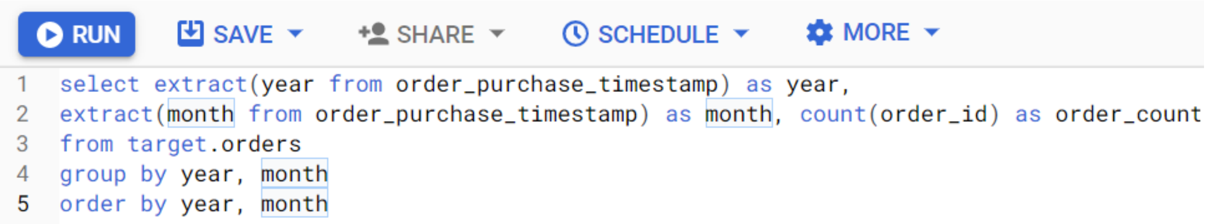


```sql
select extract(year from order_purchase_timestamp) as year,
count(order_id) as order_count
from target.orders
group by year
order by year
```

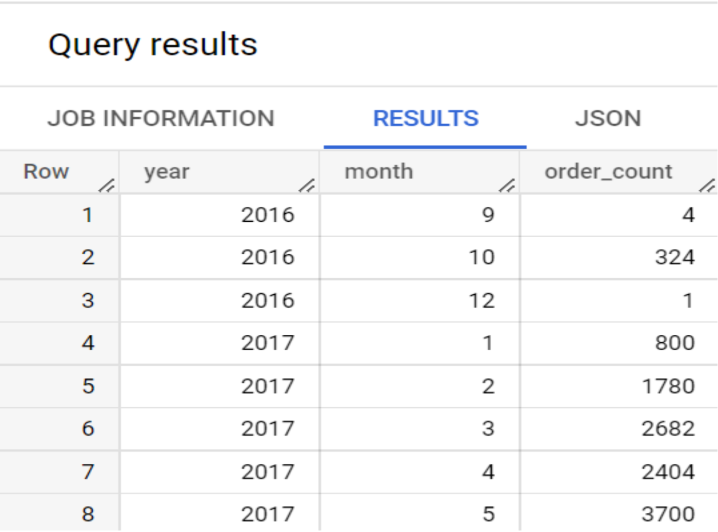

Query results

Row	year	order_count
1	2016	329
2	2017	45101
3	2018	54011

- **There is a growing trend in e-commerce in Brazil as there is increase in total number of orders per year.**

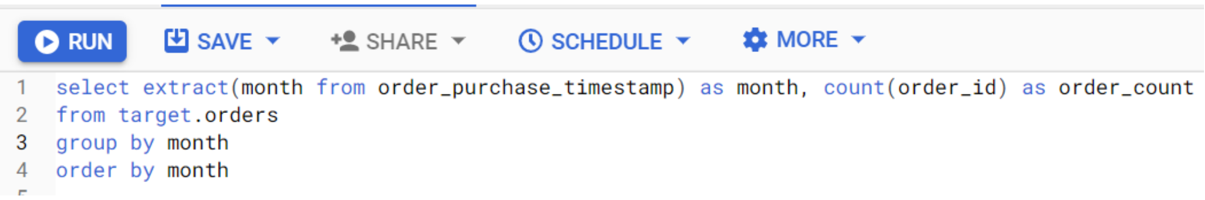


```sql
select extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month, count(order_id) as order_count
from target.orders
group by year, month
order by year, month
```

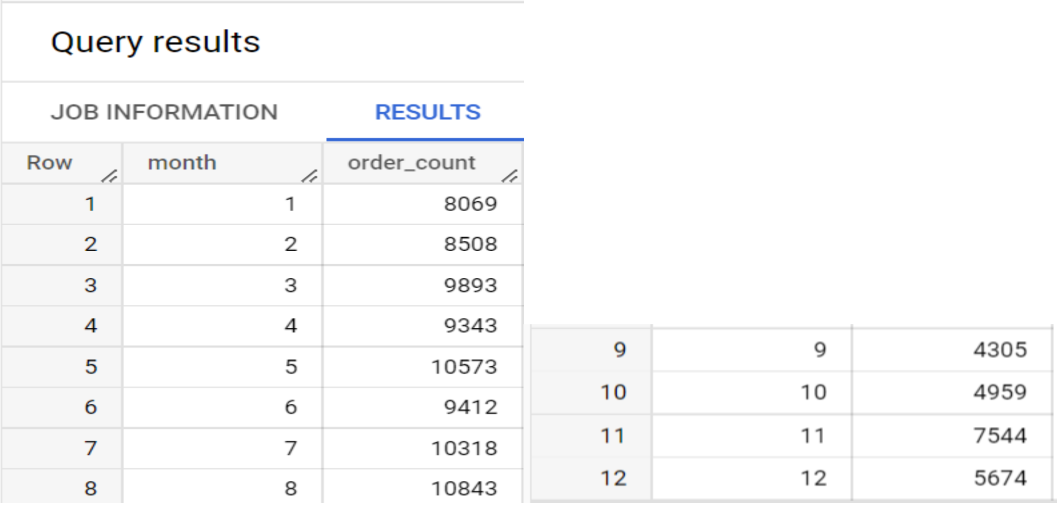


Row	year	month	order_count
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700

- **We can see that the number of purchases since 2017 has been increasing.**



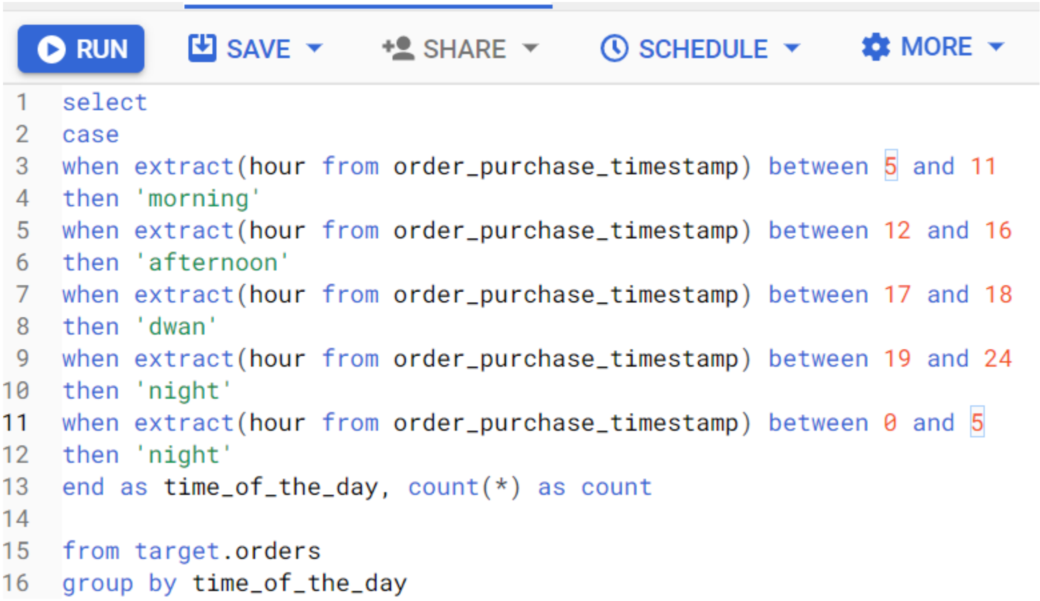
```
select extract(month from order_purchase_timestamp) as month, count(order_id) as order_count
from target.orders
group by month
order by month
```



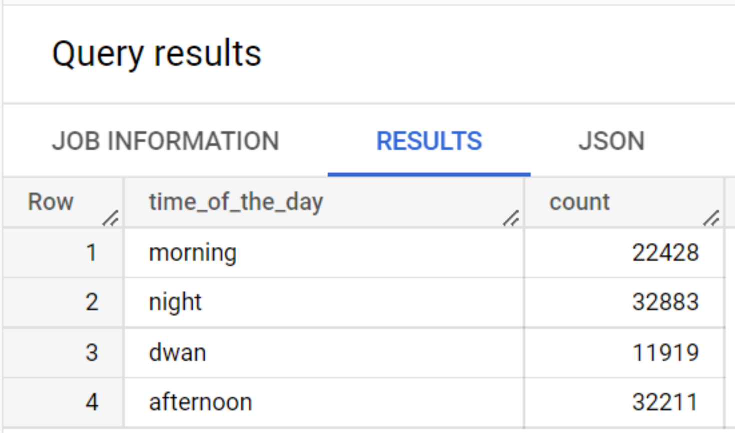
Row	month	order_count
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318
8	8	10843
9	9	4305
10	10	4959
11	11	7544
12	12	5674

- **We can see that months may, July, august has highest number of orders than any other months.**

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



```sql
select
case
when extract(hour from order_purchase_timestamp) between 5 and 11
then 'morning'
when extract(hour from order_purchase_timestamp) between 12 and 16
then 'afternoon'
when extract(hour from order_purchase_timestamp) between 17 and 18
then 'dwan'
when extract(hour from order_purchase_timestamp) between 19 and 24
then 'night'
when extract(hour from order_purchase_timestamp) between 0 and 5
then 'night'
end as time_of_the_day, count(*) as count

from target.orders
group by time_of_the_day
```

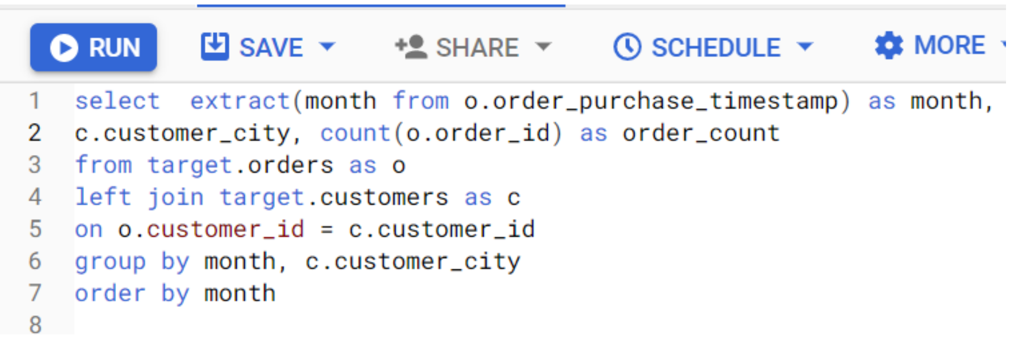


Row	time_of_the_day	count
1	morning	22428
2	night	32883
3	dwan	11919
4	afternoon	32211

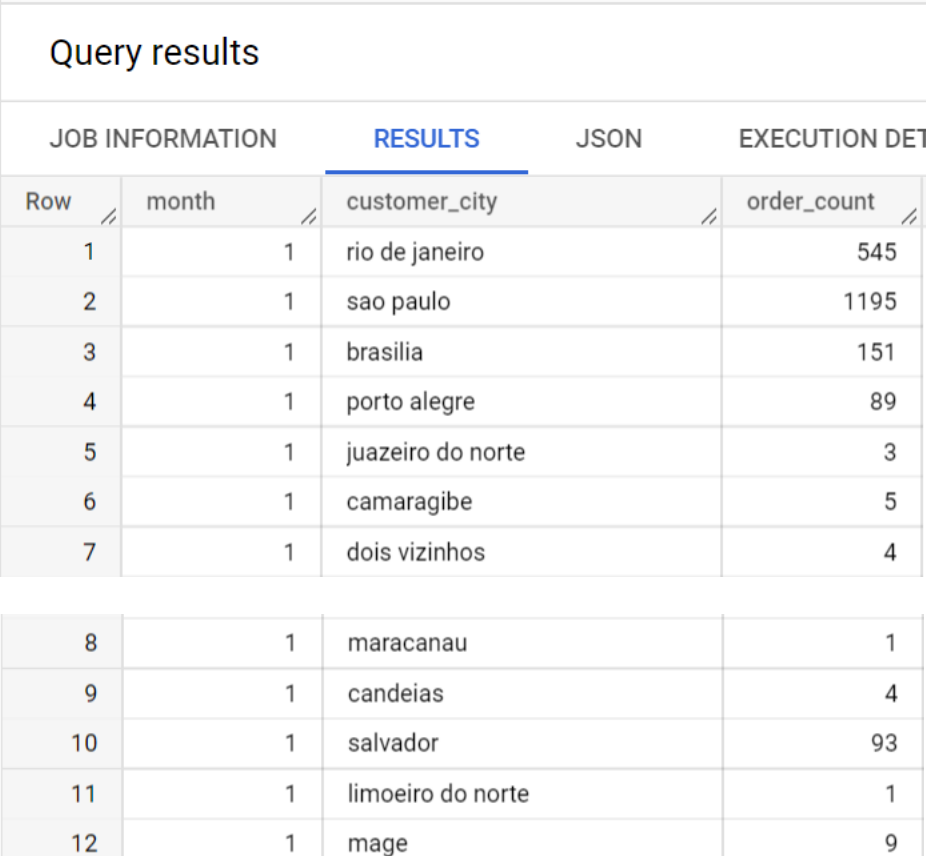
- **Brazilian customers trend to place orders more at night and afternoon.**

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

1. Get month on month orders by region



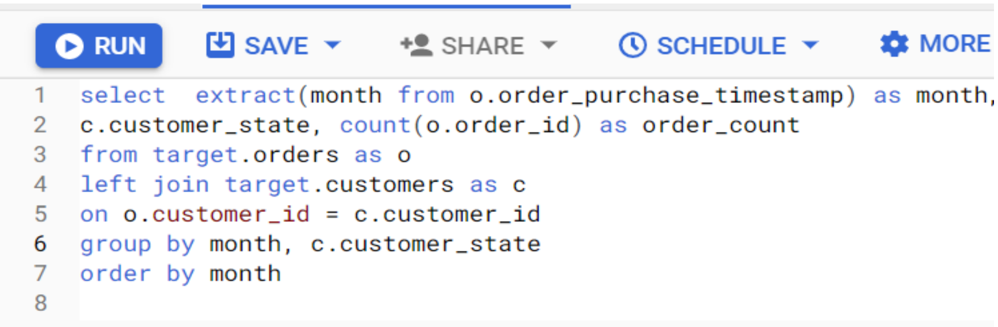
```
select  extract(month from o.order_purchase_timestamp) as month,
c.customer_city, count(o.order_id) as order_count
from target.orders as o
left join target.customers as c
on o.customer_id = c.customer_id
group by month, c.customer_city
order by month
```



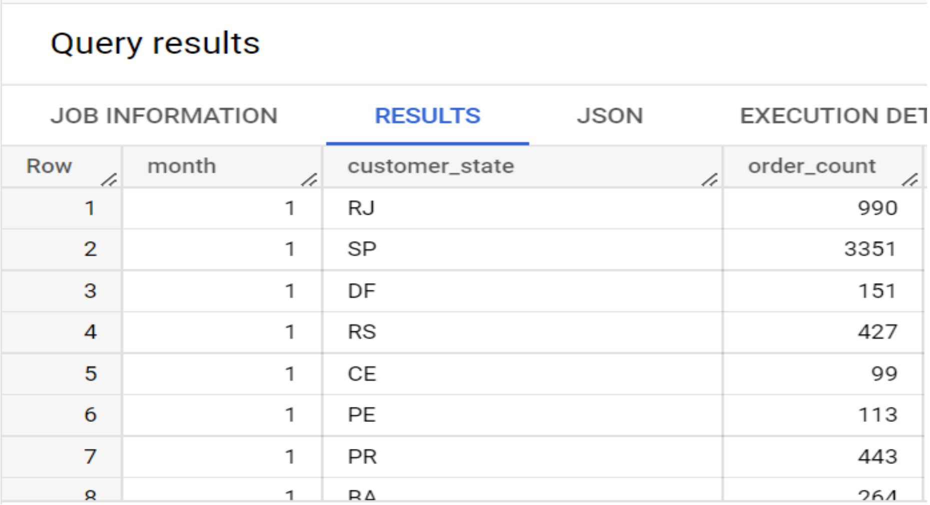
Query results

Row	month	customer_city	order_count
1	1	rio de janeiro	545
2	1	sao paulo	1195
3	1	brasilia	151
4	1	porto alegre	89
5	1	juazeiro do norte	3
6	1	camaragibe	5
7	1	dois vizinhos	4
8	1	maracanau	1
9	1	candeias	4
10	1	salvador	93
11	1	limoeiro do norte	1
12	1	mage	9

2. Get month on month orders by states

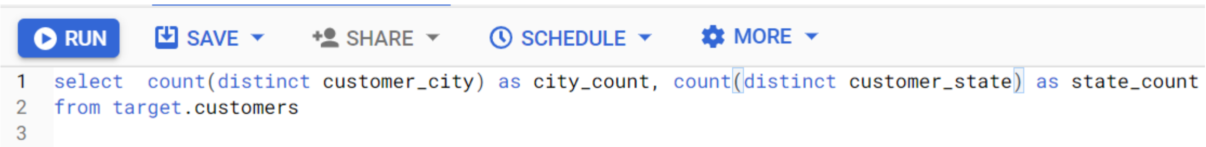


```
select  extract(month from o.order_purchase_timestamp) as month,
c.customer_state, count(o.order_id) as order_count
from target.orders as o
left join target.customers as c
on o.customer_id = c.customer_id
group by month, c.customer_state
order by month
```

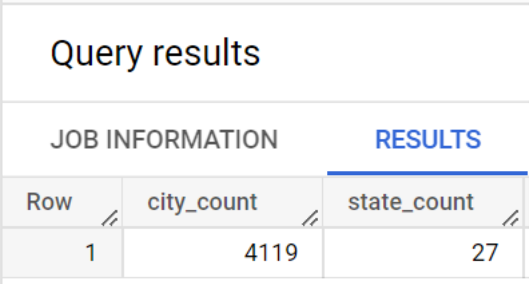


Row	month	customer_state	order_count
1	1	RJ	990
2	1	SP	3351
3	1	DF	151
4	1	RS	427
5	1	CE	99
6	1	PE	113
7	1	PR	443
8	1	BA	264

2. How are customers distributed in Brazil



```sql
select  count(distinct customer_city) as city_count, count(distinct customer_state) as state_count
from target.customers
```

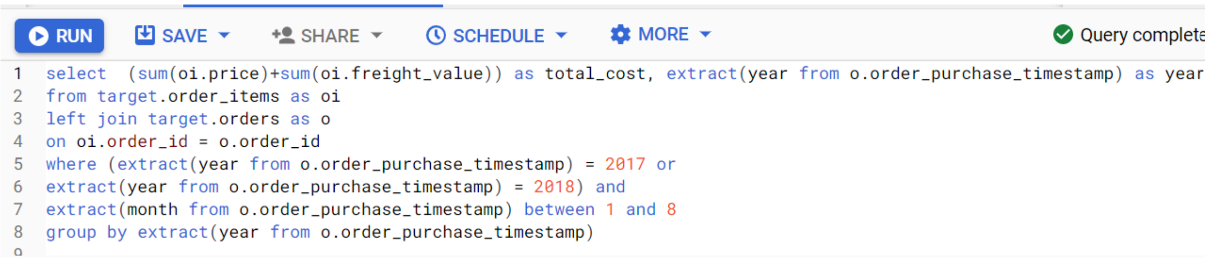


Row	city_count	state_count
1	4119	27

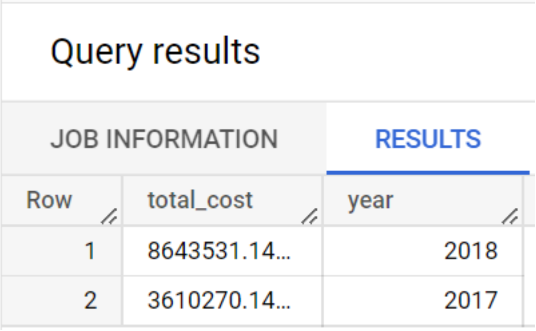
- **There are customers from 4119 different cities and 27 different states.**

Q4. Impact on Economy: Analyze the money movemented 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)



```
select  (sum(oi.price)+sum(oi.freight_value)) as total_cost, extract(year from o.order_purchase_timestamp) as year
from target.order_items as oi
left join target.orders as o
on oi.order_id = o.order_id
where (extract(year from o.order_purchase_timestamp) = 2017 or
extract(year from o.order_purchase_timestamp) = 2018) and
extract(month from o.order_purchase_timestamp) between 1 and 8
group by extract(year from o.order_purchase_timestamp)
```

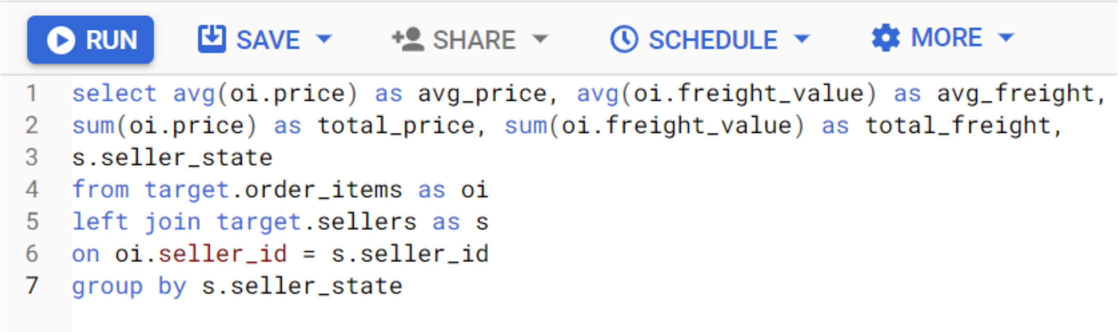


Query results

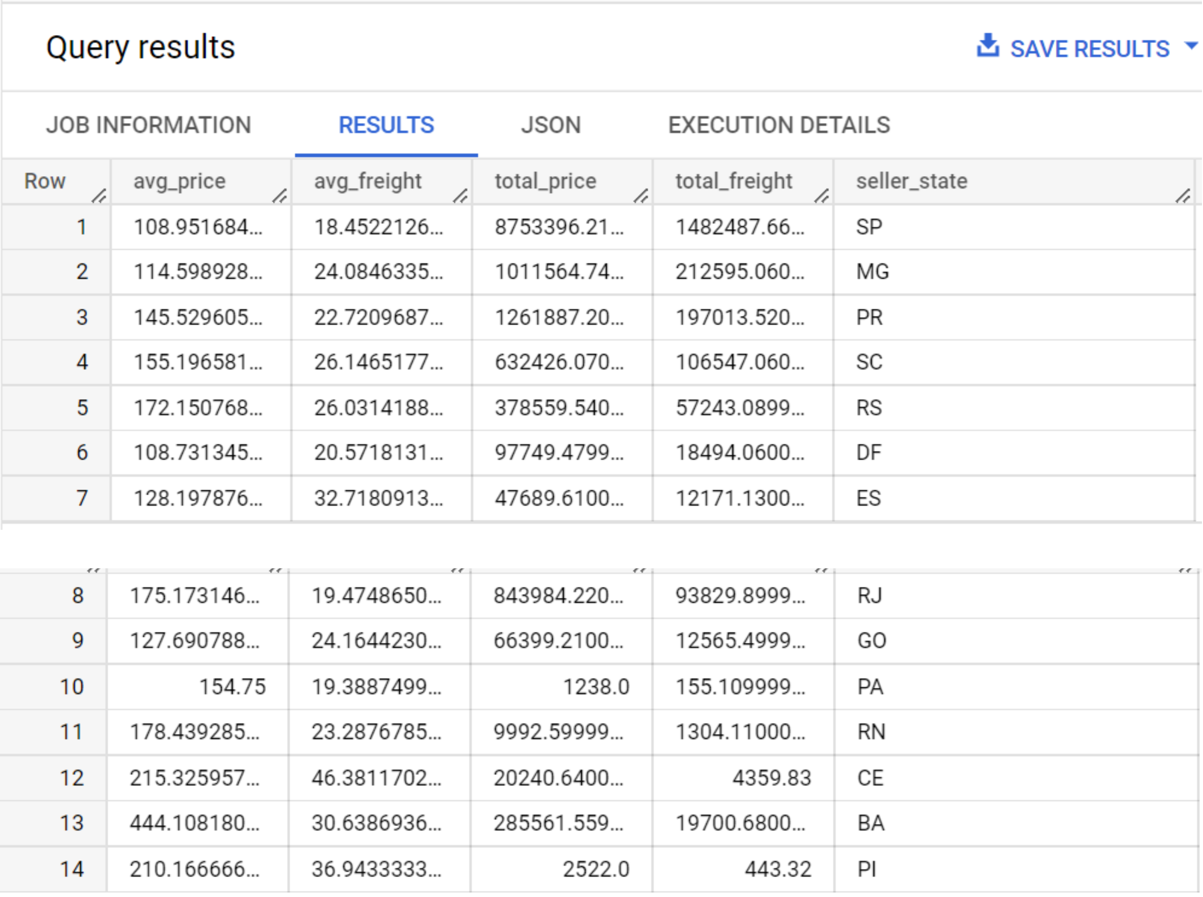
Row	total_cost	year
1	8643531.14...	2018
2	3610270.14...	2017

- **The total cost of orders on 2017 from Jan to Aug was found out to be 8643531.14 and the total cost of orders on 2018 from Jan to Aug was found out to be 3610270.14.**
- **The % of increase in cost from 2017 to 2018** (include months between Jan   to Aug only) **is 23.94%.**

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



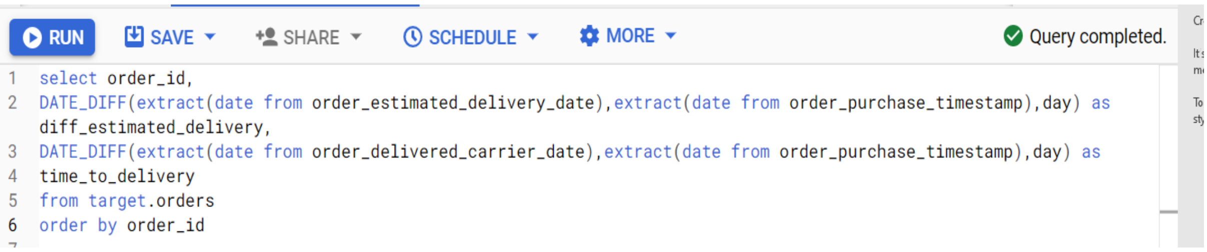
```sql
select avg(oi.price) as avg_price, avg(oi.freight_value) as avg_freight,
sum(oi.price) as total_price, sum(oi.freight_value) as total_freight,
s.seller_state
from target.order_items as oi
left join target.sellers as s
on oi.seller_id = s.seller_id
group by s.seller_state
```



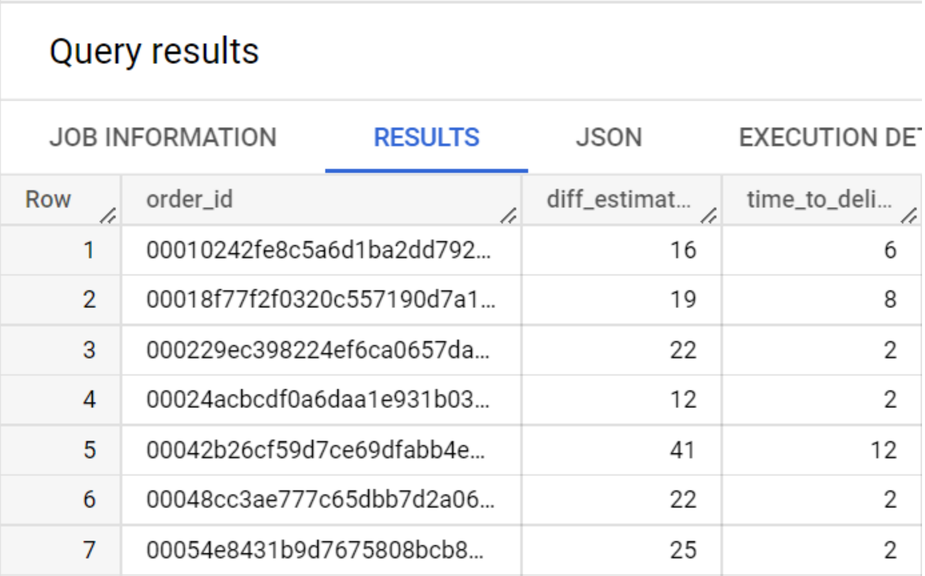
Row	avg_price	avg_freight	total_price	total_freight	seller_state
1	108.951684...	18.4522126...	8753396.21...	1482487.66...	SP
2	114.598928...	24.0846335...	1011564.74...	212595.060...	MG
3	145.529605...	22.7209687...	1261887.20...	197013.520...	PR
4	155.196581...	26.1465177...	632426.070...	106547.060...	SC
5	172.150768...	26.0314188...	378559.540...	57243.0899...	RS
6	108.731345...	20.5718131...	97749.4799...	18494.0600...	DF
7	128.197876...	32.7180913...	47689.6100...	12171.1300...	ES
8	175.173146...	19.4748650...	843984.220...	93829.8999...	RJ
9	127.690788...	24.1644230...	66399.2100...	12565.4999...	GO
10	154.75	19.3887499...	1238.0	155.109999...	PA
11	178.439285...	23.2876785...	9992.59999...	1304.11000...	RN
12	215.325957...	46.3811702...	20240.6400...	4359.83	CE
13	444.108180...	30.6386936...	285561.559...	19700.6800...	BA
14	210.166666...	36.9433333...	2522.0	443.32	PI

Q5. Analysis on sales, freight and delivery time

1. Calculate days between purchasing, delivering and estimated delivery
2. Create columns:
    - time_to_delivery = order_purchase_timestamp-order_delivered_customer_date
    - diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

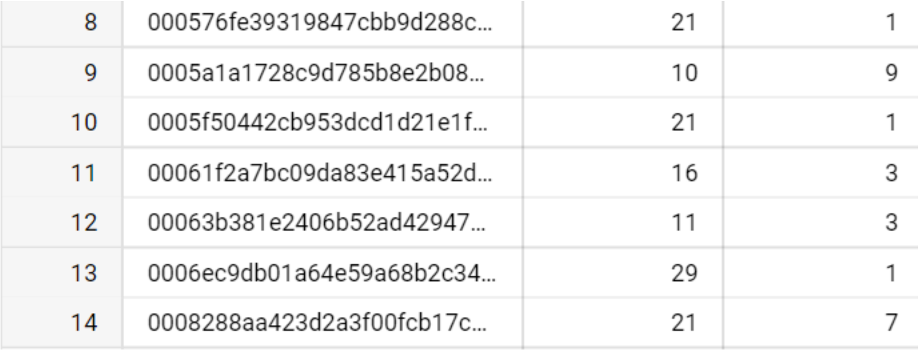


```
select order_id,
DATE_DIFF(extract(date from order_estimated_delivery_date),extract(date from order_purchase_timestamp),day) as
diff_estimated_delivery,
DATE_DIFF(extract(date from order_delivered_carrier_date),extract(date from order_purchase_timestamp),day) as
time_to_delivery
from target.orders
order by order_id
```



Query results

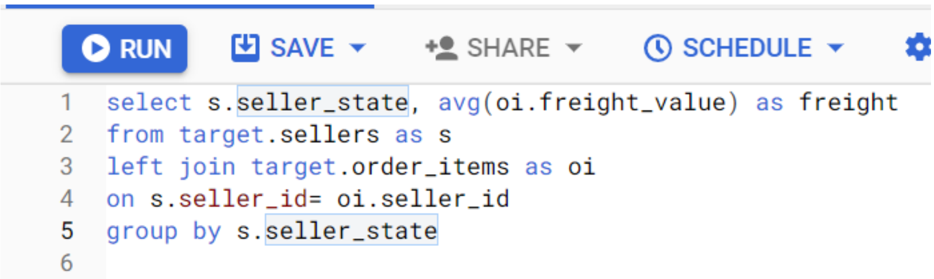
JOB INFORMATION | RESULTS | JSON | EXECUTION DE



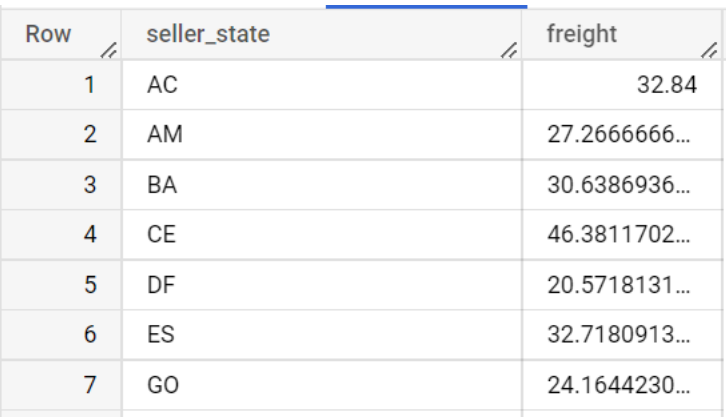
Row	order_id	diff_estimat...	time_to_deli...
1	00010242fe8c5a6d1ba2dd792...	16	6
2	00018f77f2f0320c557190d7a1...	19	8
3	000229ec398224ef6ca0657da...	22	2
4	00024acbcdf0a6daa1e931b03...	12	2
5	00042b26cf59d7ce69dfabb4e...	41	12
6	00048cc3ae777c65dbb7d2a06...	22	2
7	00054e8431b9d7675808bcb8...	25	2
8	000576fe39319847cbb9d288c...	21	1
9	0005a1a1728c9d785b8e2b08...	10	9
10	0005f50442cb953dcd1d21e1f...	21	1
11	00061f2a7bc09da83e415a52d...	16	3
12	00063b381e2406b52ad42947...	11	3
13	0006ec9db01a64e59a68b2c34...	29	1
14	0008288aa423d2a3f00fcb17c...	21	7

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

The data of mean fright value grouped state wise.

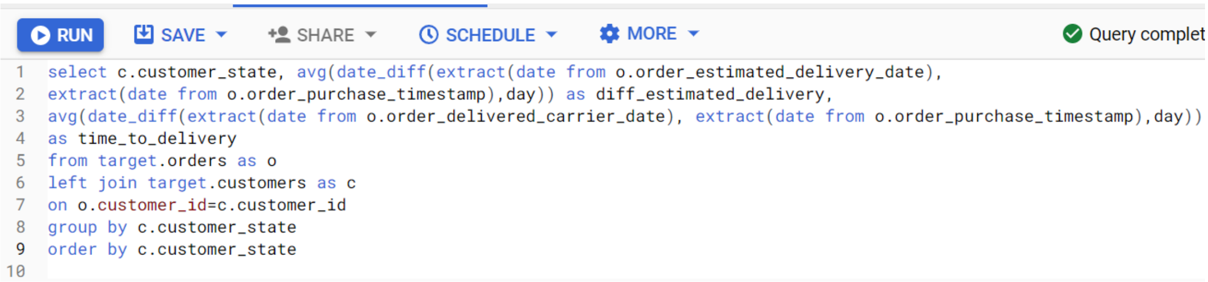


```
select s.seller_state, avg(oi.freight_value) as freight
from target.sellers as s
left join target.order_items as oi
on s.seller_id= oi.seller_id
group by s.seller_state
```

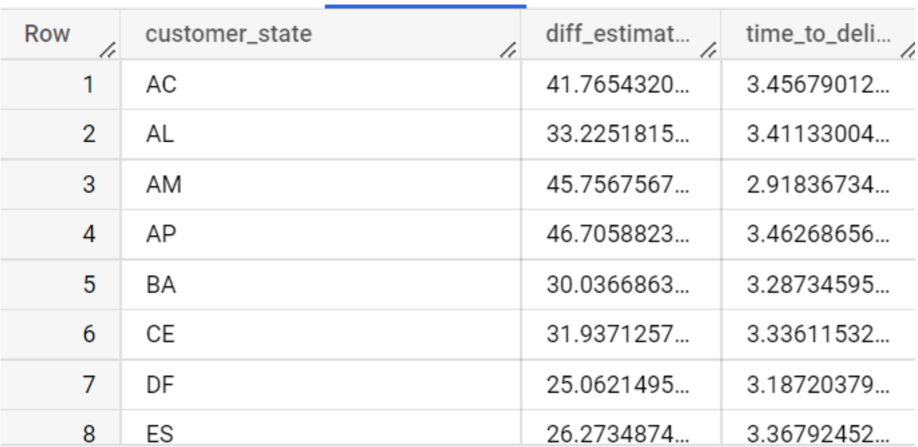


Row	seller_state	freight
1	AC	32.84
2	AM	27.2666666...
3	BA	30.6386936...
4	CE	46.3811702...
5	DF	20.5718131...
6	ES	32.7180913...
7	GO	24.1644230...

The data of mean time to delivery and diff estimated delivery



```
select c.customer_state, avg(date_diff(extract(date from o.order_estimated_delivery_date),
extract(date from o.order_purchase_timestamp),day)) as diff_estimated_delivery,
avg(date_diff(extract(date from o.order_delivered_carrier_date), extract(date from o.order_purchase_timestamp),day))
as time_to_delivery
from target.orders as o
left join target.customers as c
on o.customer_id=c.customer_id
group by c.customer_state
order by c.customer_state
```

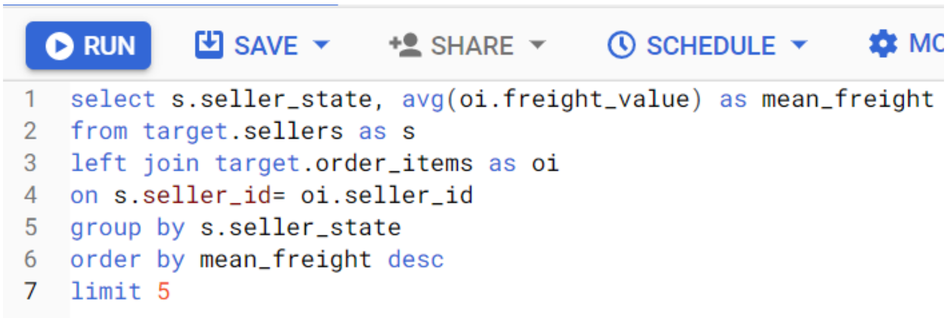


Row	customer_state	diff_estimat...	time_to_deli...
1	AC	41.7654320...	3.45679012...
2	AL	33.2251815...	3.41133004...
3	AM	45.7567567...	2.91836734...
4	AP	46.7058823...	3.46268656...
5	BA	30.0366863...	3.28734595...
6	CE	31.9371257...	3.33611532...
7	DF	25.0621495...	3.18720379...
8	ES	26.2734874...	3.36792452...

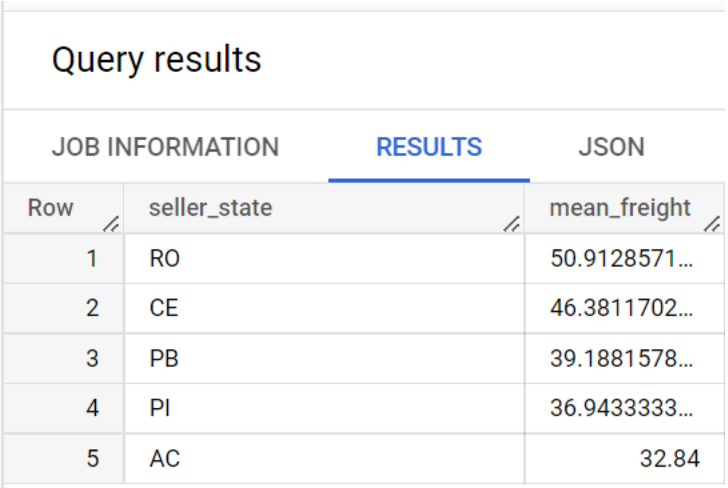
4.Sort the data to get the following:

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

Top 5 states with highest mean freight value.

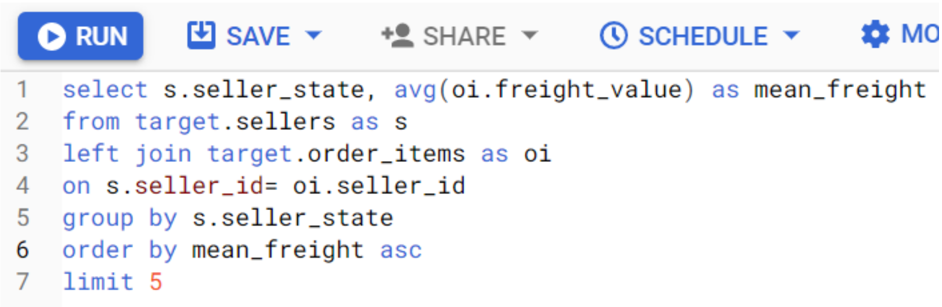


```
select s.seller_state, avg(oi.freight_value) as mean_freight
from target.sellers as s
left join target.order_items as oi
on s.seller_id= oi.seller_id
group by s.seller_state
order by mean_freight desc
limit 5
```

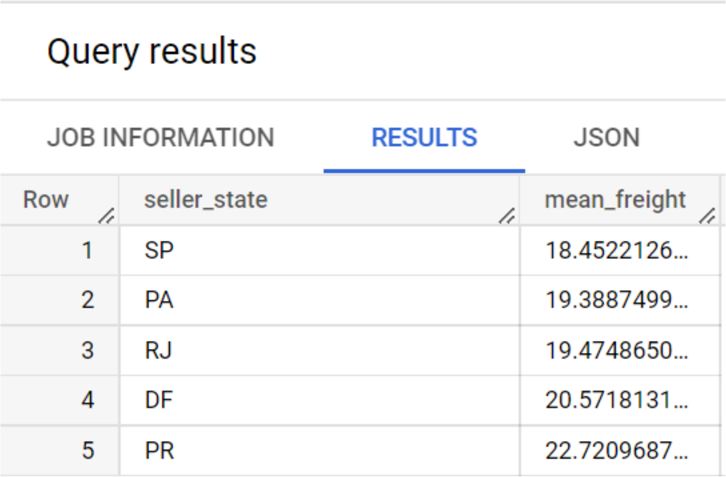


Row	seller_state	mean_freight
1	RO	50.9128571...
2	CE	46.3811702...
3	PB	39.1881578...
4	PI	36.9433333...
5	AC	32.84

Top 5 states with lowest mean freight value.



```
select s.seller_state, avg(oi.freight_value) as mean_freight
from target.sellers as s
left join target.order_items as oi
on s.seller_id= oi.seller_id
group by s.seller_state
order by mean_freight asc
limit 5
```

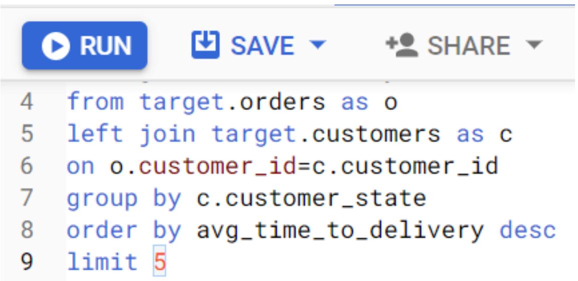


Query results

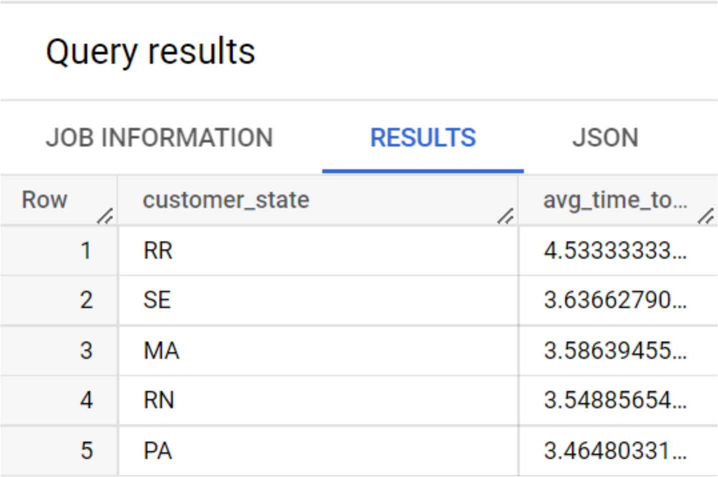
Row	seller_state	mean_freight
1	SP	18.4522126...
2	PA	19.3887499...
3	RJ	19.4748650...
4	DF	20.5718131...
5	PR	22.7209687...

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

Top 5 states with highest average delivery time.



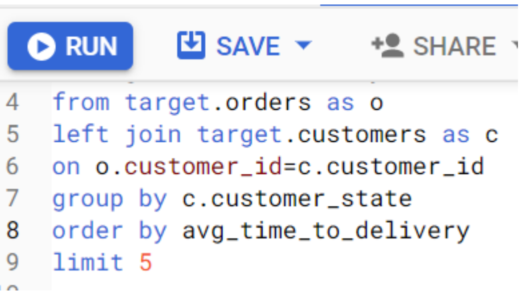
```
from target.orders as o
left join target.customers as c
on o.customer_id=c.customer_id
group by c.customer_state
order by avg_time_to_delivery desc
limit 5
```



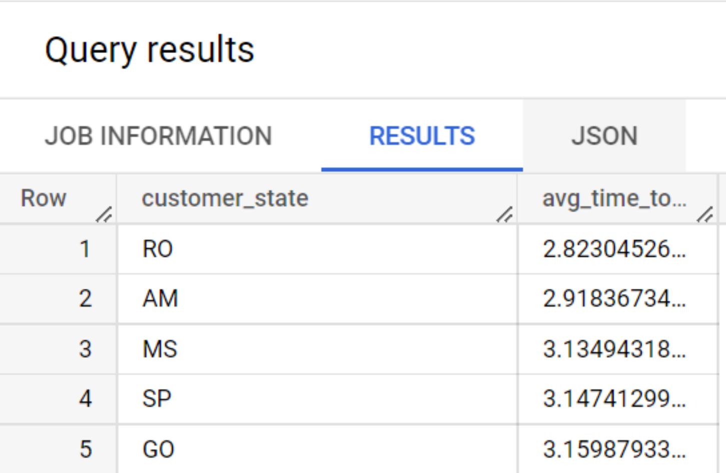
Query results

Row	customer_state	avg_time_to...
1	RR	4.53333333...
2	SE	3.63662790...
3	MA	3.58639455...
4	RN	3.54885654...
5	PA	3.46480331...

Top 5 states with lowest average delivery time.



```
from target.orders as o
left join target.customers as c
on o.customer_id=c.customer_id
group by c.customer_state
order by avg_time_to_delivery
limit 5
```

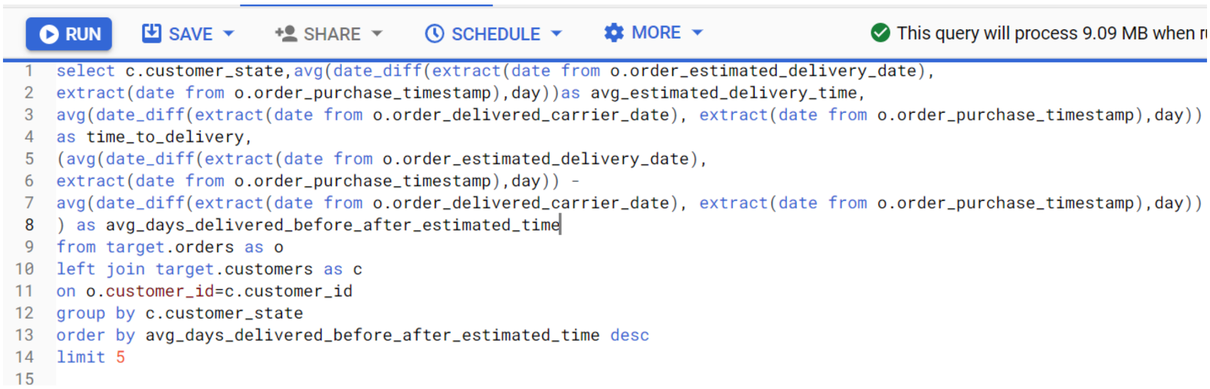


Query results

Row	customer_state	avg_time_to...
1	RO	2.82304526...
2	AM	2.91836734...
3	MS	3.13494318...
4	SP	3.14741299...
5	GO	3.15987933...

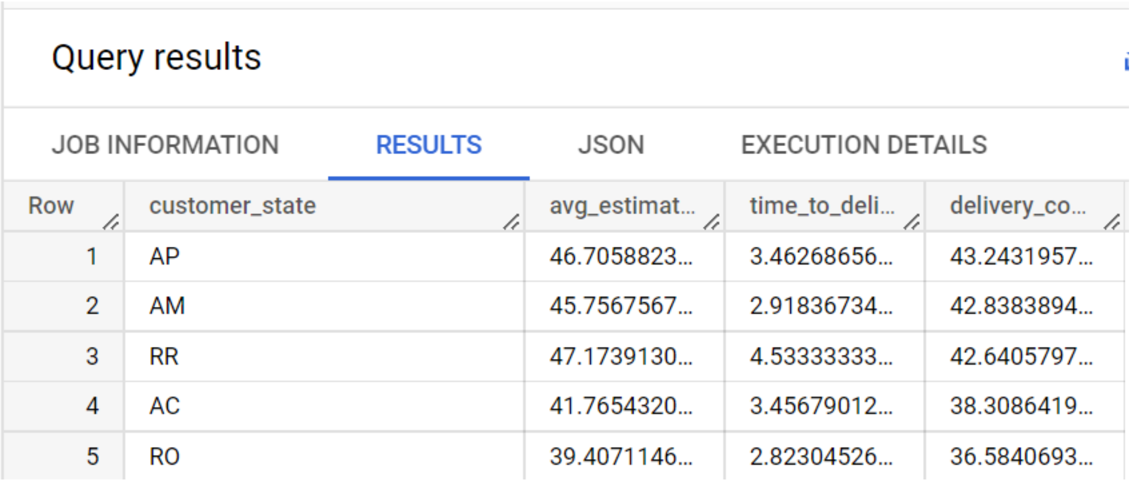
3. Top 5 states where delivery is really fast/ not so fast compared to estimated date

Top 5 states where delivery is faster compared to estimated date.



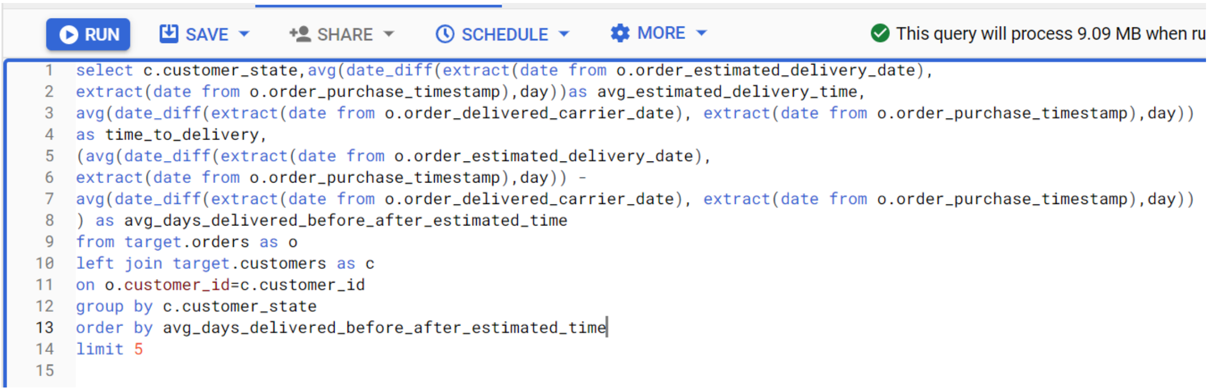
This query will process 9.09 MB when r

```
select c.customer_state,avg(date_diff(extract(date from o.order_estimated_delivery_date),
extract(date from o.order_purchase_timestamp),day))as avg_estimated_delivery_time,
avg(date_diff(extract(date from o.order_delivered_carrier_date), extract(date from o.order_purchase_timestamp),day))
as time_to_delivery,
(avg(date_diff(extract(date from o.order_estimated_delivery_date),
extract(date from o.order_purchase_timestamp),day)) -
avg(date_diff(extract(date from o.order_delivered_carrier_date), extract(date from o.order_purchase_timestamp),day))
) as avg_days_delivered_before_after_estimated_time
from target.orders as o
left join target.customers as c
on o.customer_id=c.customer_id
group by c.customer_state
order by avg_days_delivered_before_after_estimated_time desc
limit 5

```



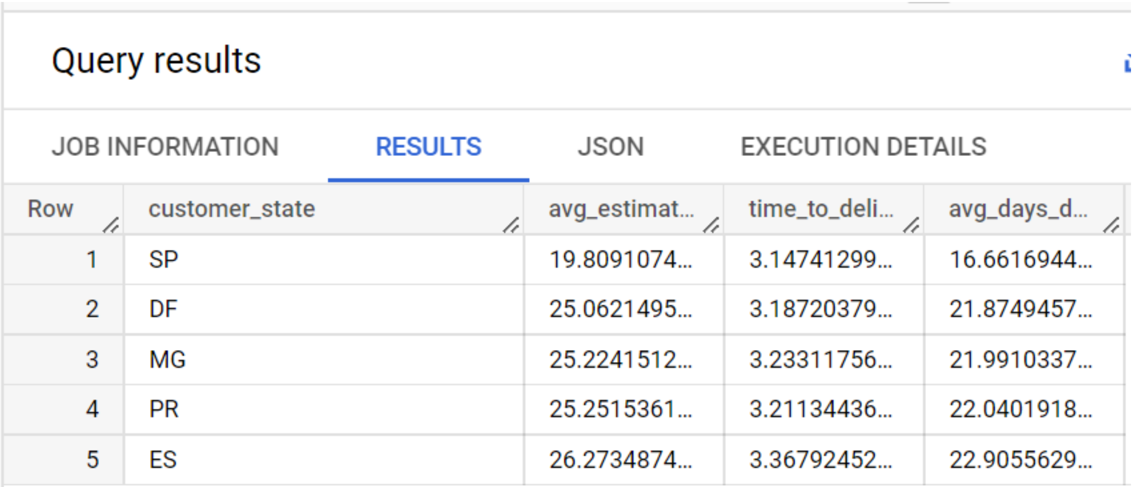
Row	customer_state	avg_estimat...	time_to_deli...	delivery_co...
1	AP	46.7058823...	3.46268656...	43.2431957...
2	AM	45.7567567...	2.91836734...	42.8383894...
3	RR	47.1739130...	4.53333333...	42.6405797...
4	AC	41.7654320...	3.45679012...	38.3086419...
5	RO	39.4071146...	2.82304526...	36.5840693...

Top 5 states where delivery is not so faster compared to estimated date.



This query will process 9.09 MB when ru

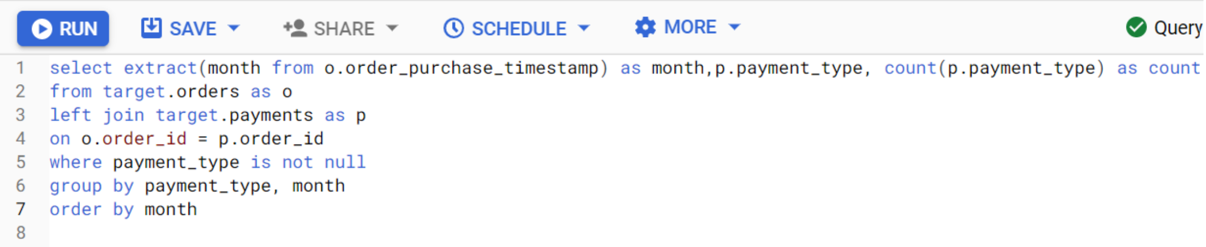
```
select c.customer_state,avg(date_diff(extract(date from o.order_estimated_delivery_date),
extract(date from o.order_purchase_timestamp),day))as avg_estimated_delivery_time,
avg(date_diff(extract(date from o.order_delivered_carrier_date), extract(date from o.order_purchase_timestamp),day))
as time_to_delivery,
(avg(date_diff(extract(date from o.order_estimated_delivery_date),
extract(date from o.order_purchase_timestamp),day)) -
avg(date_diff(extract(date from o.order_delivered_carrier_date), extract(date from o.order_purchase_timestamp),day))
) as avg_days_delivered_before_after_estimated_time
from target.orders as o
left join target.customers as c
on o.customer_id=c.customer_id
group by c.customer_state
order by avg_days_delivered_before_after_estimated_time
limit 5

```



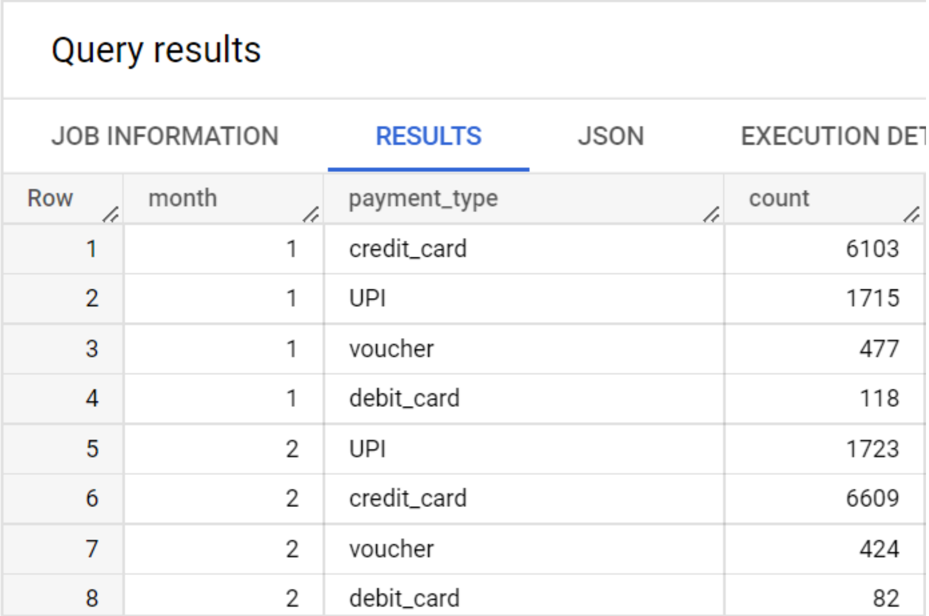
Row	customer_state	avg_estimat...	time_to_deli...	avg_days_d...
1	SP	19.8091074...	3.14741299...	16.6616944...
2	DF	25.0621495...	3.18720379...	21.8749457...
3	MG	25.2241512...	3.23311756...	21.9910337...
4	PR	25.2515361...	3.21134436...	22.0401918...
5	ES	26.2734874...	3.36792452...	22.9055629...

Q6. Payment type analysis:

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

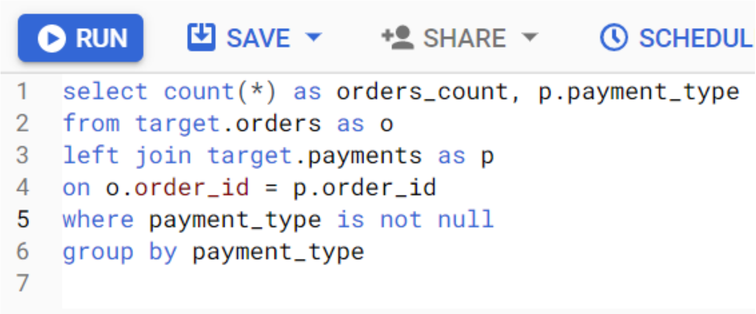


```sql
select extract(month from o.order_purchase_timestamp) as month,p.payment_type, count(p.payment_type) as count
from target.orders as o
left join target.payments as p
on o.order_id = p.order_id
where payment_type is not null
group by payment_type, month
order by month
```

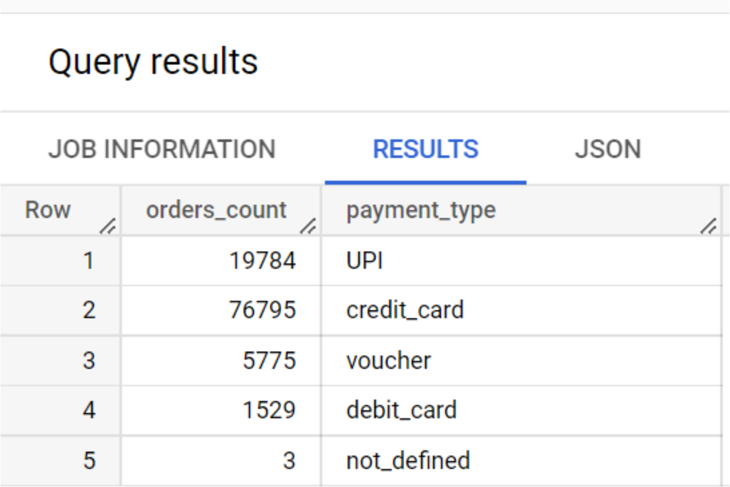

Query results

Row	month	payment_type	count
1	1	credit_card	6103
2	1	UPI	1715
3	1	voucher	477
4	1	debit_card	118
5	2	UPI	1723
6	2	credit_card	6609
7	2	voucher	424
8	2	debit_card	82

2. Distribution of payment installments and count of orders



```
select count(*) as orders_count, p.payment_type
from target.orders as o
left join target.payments as p
on o.order_id = p.order_id
where payment_type is not null
group by payment_type
```


Query results

Row	orders_count	payment_type
1	19784	UPI
2	76795	credit_card
3	5775	voucher
4	1529	debit_card
5	3	not_defined

The UPI is the most used type of payment followed by credit_card.