Target SQL Business Case

Topic: SQL

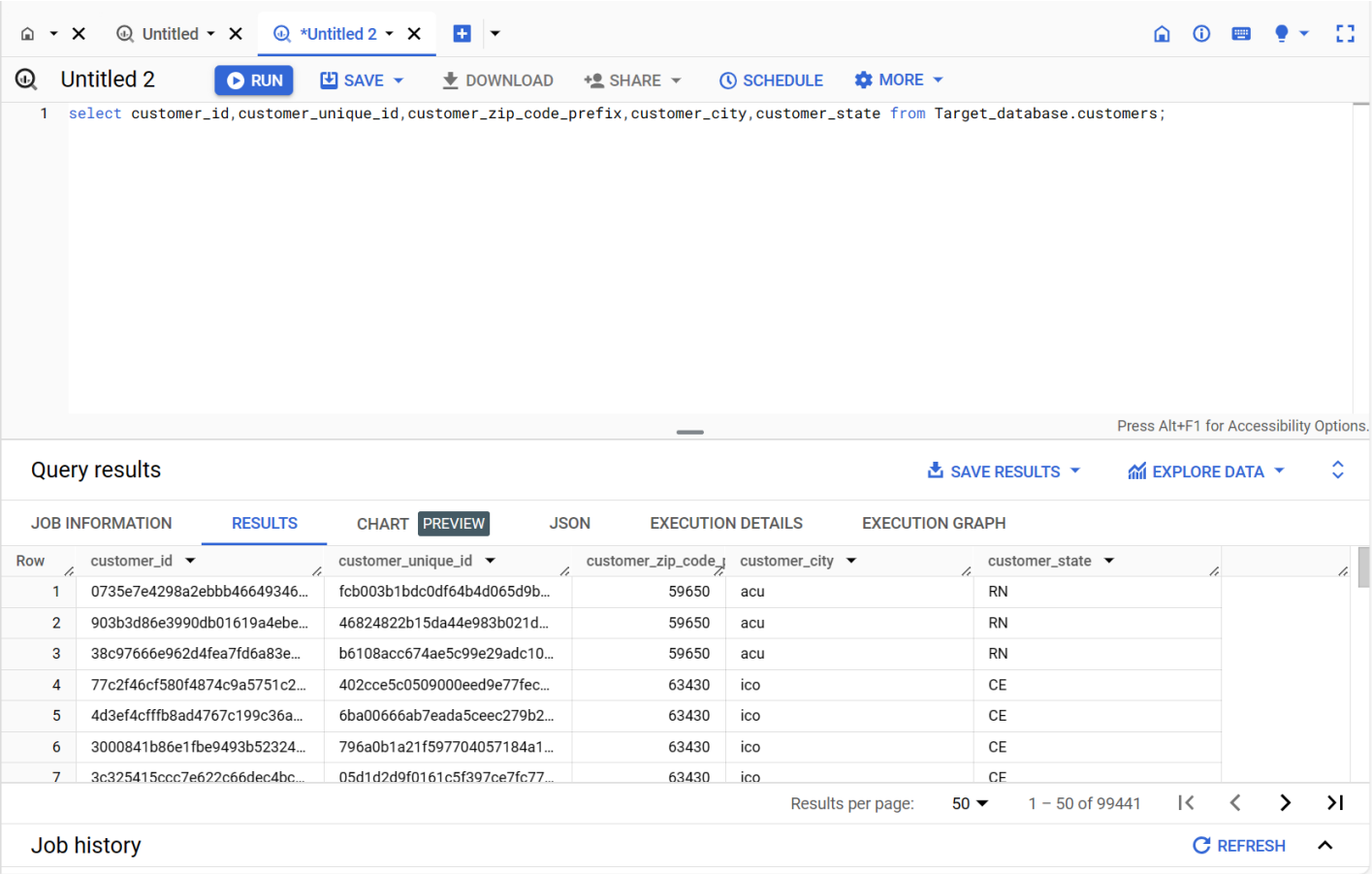
PROJECT: Target SQL

QUERIES:

QUESTION 1: Data type of all columns in the “customers” table.

```
select customer_id,customer_unique_id,customer_zip_code_prefix,customer_city,customer_state from Target_database.customers;
```

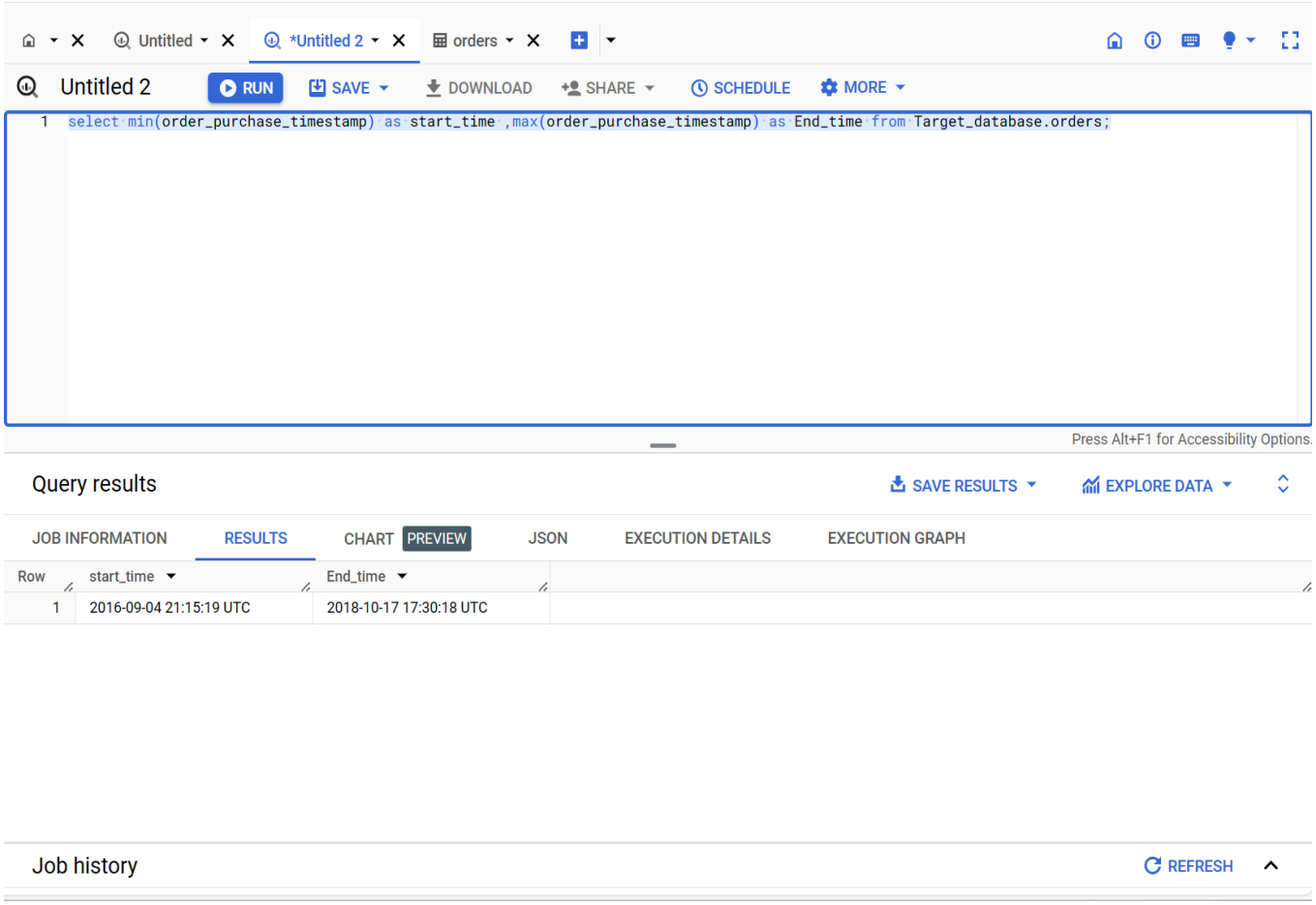
output:



QUESTION2: Get the time range between which the orders were placed

```
select min(order_purchase_timestamp) as start_time
,max(order_purchase_timestamp) as End_time from
Target_database.orders;
```

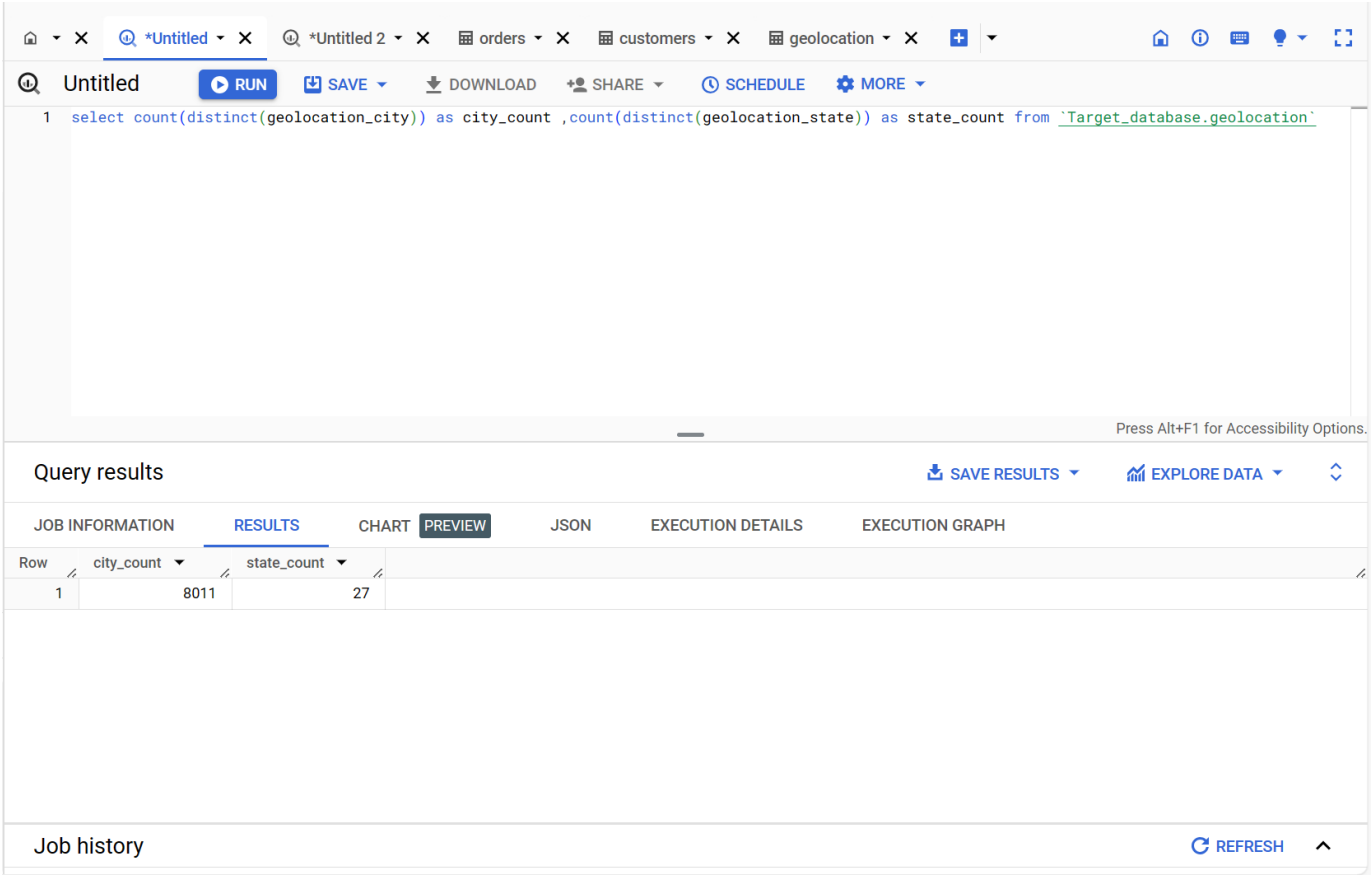
output:



QUESTION3: Count the number of Cities and States in our dataset.

```
select count(distinct(geolocation_city)) as city_count
,count(distinct(geolocation_state)) as state_count from
`Target_database.geolocation`
```

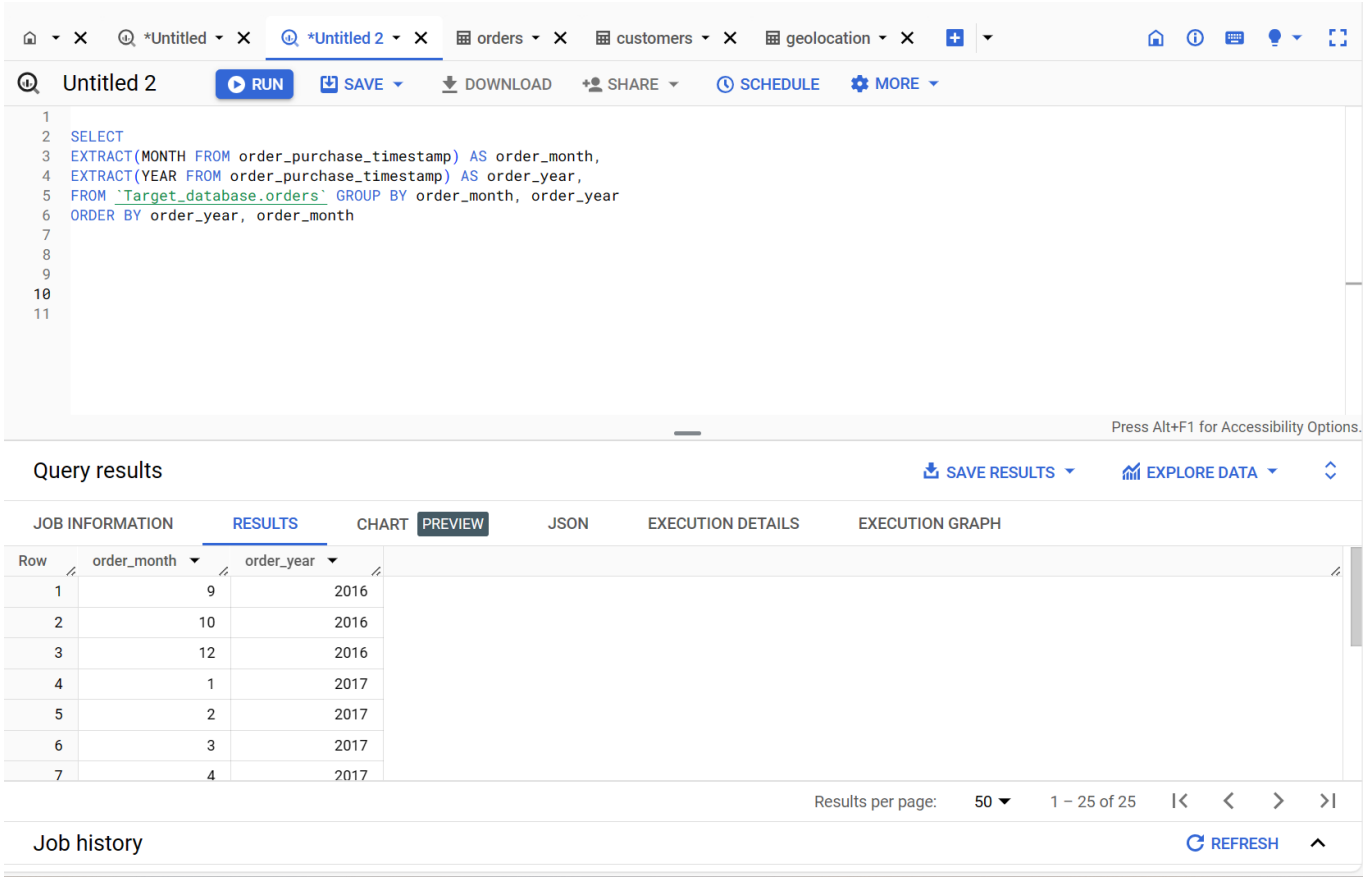
output:



QUESTION4: Is there a growing trend in the no. of orders placed over the past years?

```
SELECT
EXTRACT(MONTH FROM order_purchase_timestamp) AS order_month,
EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
FROM `Target_database.orders` GROUP BY order_month, order_year
ORDER BY order_year, order_month
```

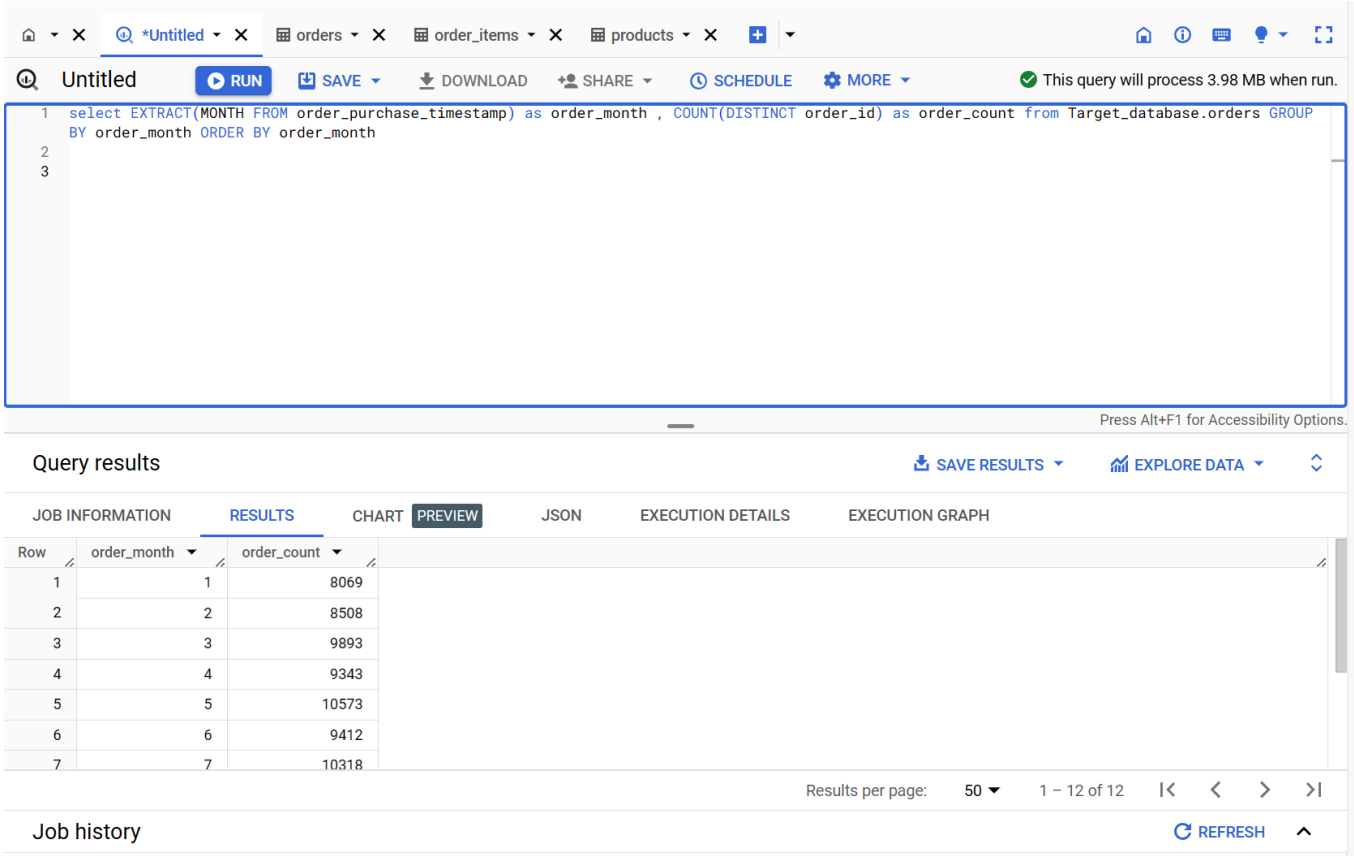
Output:



QUESTION 5: Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

```
Select
 EXTRACT(MONTH FROM
order_purchase_timestamp) as order_month ,
COUNT(DISTINCT order_id) as order_count from
Target_database.orders GROUP BY order_month
ORDER BY order_month
```

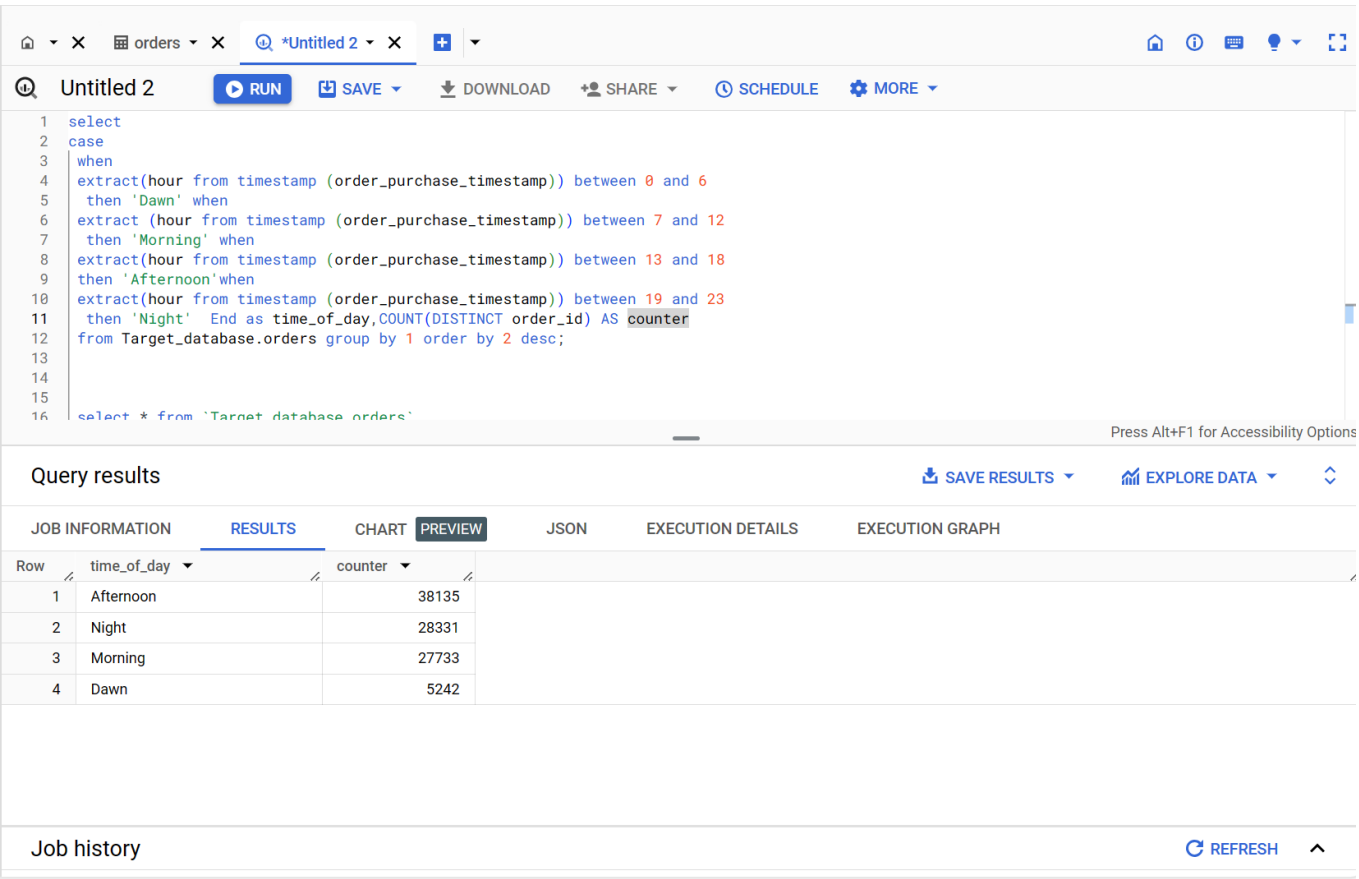
Output:



QUESTION6: During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night) ● 0-6 hrs : Dawn ● 7-12 hrs : Mornings ● 13-18 hrs : Afternoon ● 19-23 hrs : Night

```
select
case
 when
 extract(hour from timestamp (order_purchase_timestamp)) between 0 and 6
  then 'Dawn' when
 extract (hour from timestamp (order_purchase_timestamp)) between 7 and 12
  then 'Morning' when
 extract(hour from timestamp (order_purchase_timestamp)) between 13 and 18
 then 'Afternoon'when
 extract(hour from timestamp (order_purchase_timestamp)) between 19 and 23
  then 'Night'  End as time_of_day,COUNT(DISTINCT order_id) AS counter
 from Target_database.orders group by 1 order by 2 desc;
```

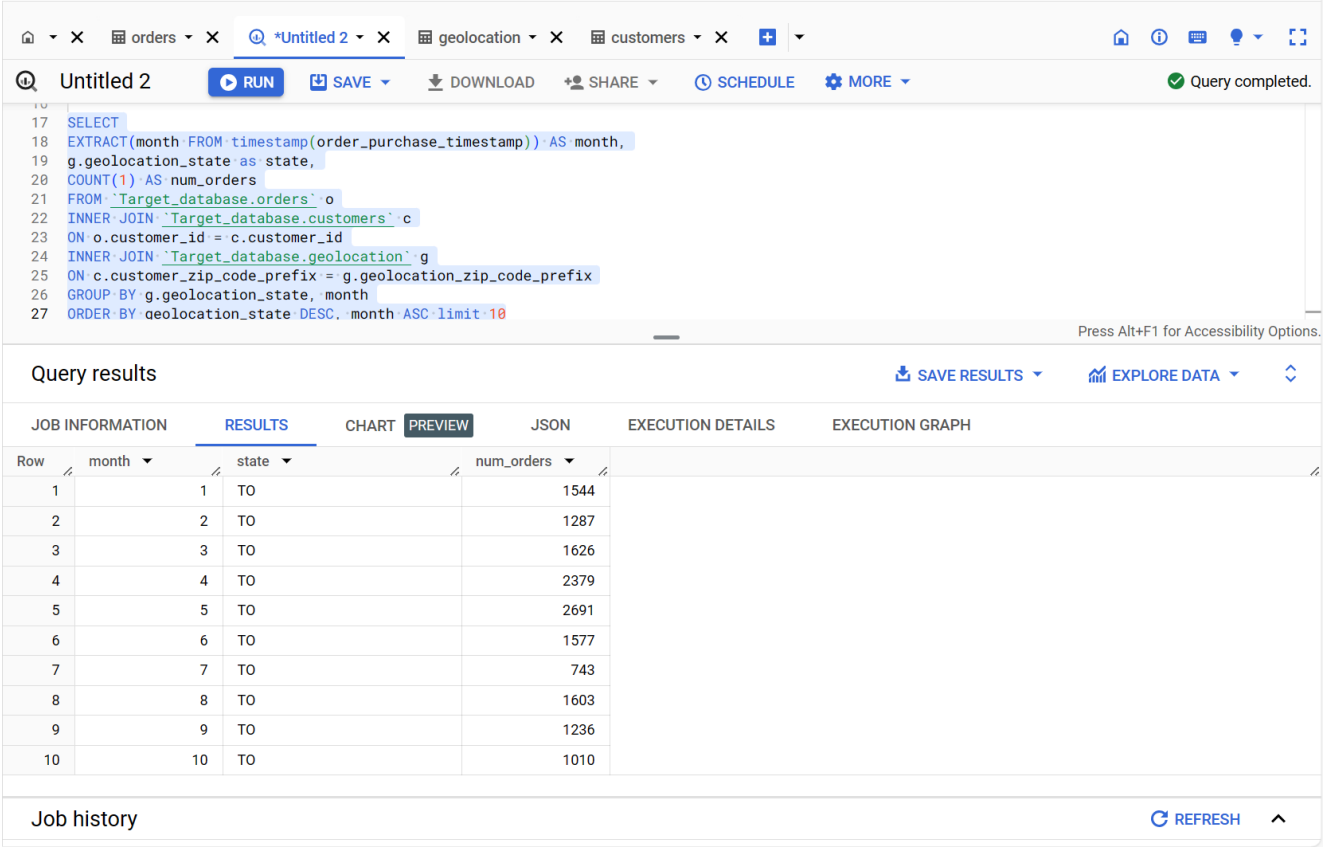
output:



QUESTION7: Get the month on month no. of orders placed in each state.

```
SELECT
EXTRACT(month FROM timestamp(order_purchase_timestamp)) AS month,
g.geolocation_state as state,
COUNT(1) AS num_orders
FROM `Target_database.orders` o
INNER JOIN `Target_database.customers` c
ON o.customer_id = c.customer_id
INNER JOIN `Target_database.geolocation` g
ON c.customer_zip_code_prefix = g.geolocation_zip_code_prefix
GROUP BY g.geolocation_state, month
ORDER BY geolocation_state DESC, month ASC limit 10
```

Output:

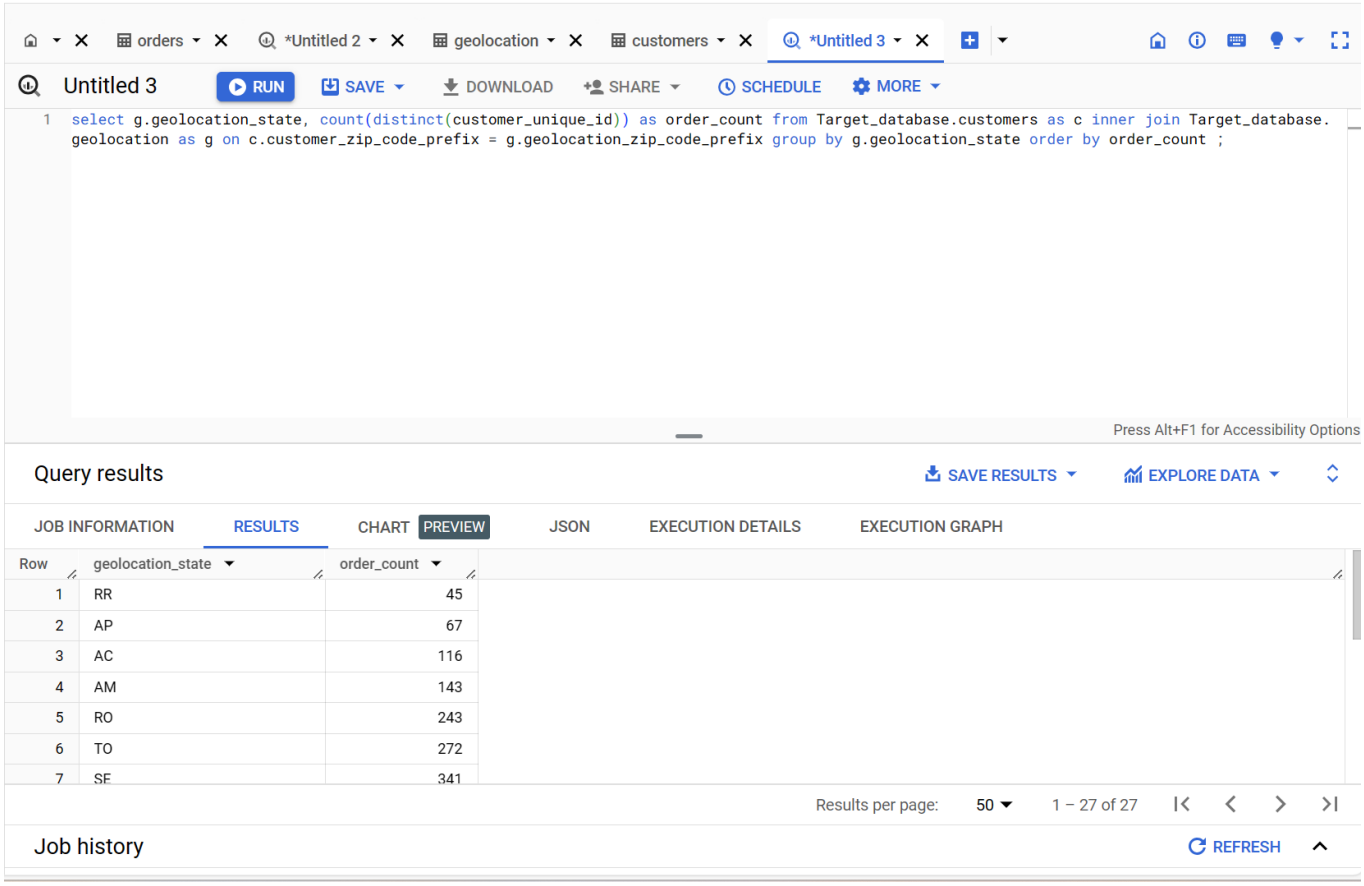


Row	month	state	num_orders
1	1	TO	1544
2	2	TO	1287
3	3	TO	1626
4	4	TO	2379
5	5	TO	2691
6	6	TO	1577
7	7	TO	743
8	8	TO	1603
9	9	TO	1236
10	10	TO	1010

QUESTION8: How are the customers distributed across all the states?

```
Select
 g.geolocation_state, count(distinct(customer_unique_id)) as
order_count from Target_database.customers as c inner join
Target_database.geolocation as g on c.customer_zip_code_prefix =
g.geolocation_zip_code_prefix group by g.geolocation_state order by
order_count ;
```

output:



QUESTION9: Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

```
with demo as(
select extract(year from timestamp(o.order_purchase_timestamp)) as
year,extract(month from timestamp(o.order_purchase_timestamp))as
month, sum(price)/count(distinct(o.order_id)) as
order_count,(sum(freight_value) / COUNT(distinct o.order_id)) AS
freight_value from Target_database.orders as o inner join
Target_database.order_items as i on o.order_id=i.order_id GROUP
BY year, month)
SELECT
month,
price_2017,
price_2018,
round((price_2018 - price_2017) / price_2017 * 100, 2) AS Avgg
FROM
(
SELECT
month,
sum(CASE WHEN year = 2017 THEN order_count ELSE 0 END)
AS price_2017,
sum(CASE WHEN year = 2018 THEN order_count ELSE 0 END)
AS price_2018
FROM demo
WHERE (year = 2017 OR year = 2018) AND month BETWEEN 1
AND 8
GROUP BY month
order by month
);
```

Output:

*Untitled | orders | order_items

Untitled

RUN | SAVE | DOWNLOAD | SHARE | SCHEDULE | MORE

```
with demo as(
select extract(year from timestamp(o.order_purchase_timestamp)) as year,extract(month from timestamp(o.order_purchase_timestamp))as month, sum
(price)/count(distinct(o.order_id)) as order_count,(sum(freight_value) / COUNT(distinct o.order_id)) AS freight_value from Target_database.
orders as o inner join Target_database.order_items as i on o.order_id=i.order_id GROUP BY year, month)
SELECT
month,
price_2017,
price_2018,
```

Press Alt+F1 for Accessibility Options

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	month	price_2017	price_2018	Avgg
1	1	152.4877946768...	131.5831523545...	-13.71
2	2	142.7022619734...	126.1097564983...	-11.63
3	3	141.7433926542...	136.7853978853...	-3.5
4	4	150.5341823504...	143.7334511104...	-4.52
5	5	138.2708032786...	145.4133488982...	5.17
6	6	134.6094497979...	140.4422581168...	4.33
7	7	125.4803426555...	142.7558138052...	13.77
8	8	133.6994362916...	132.4684330440...	-0.92

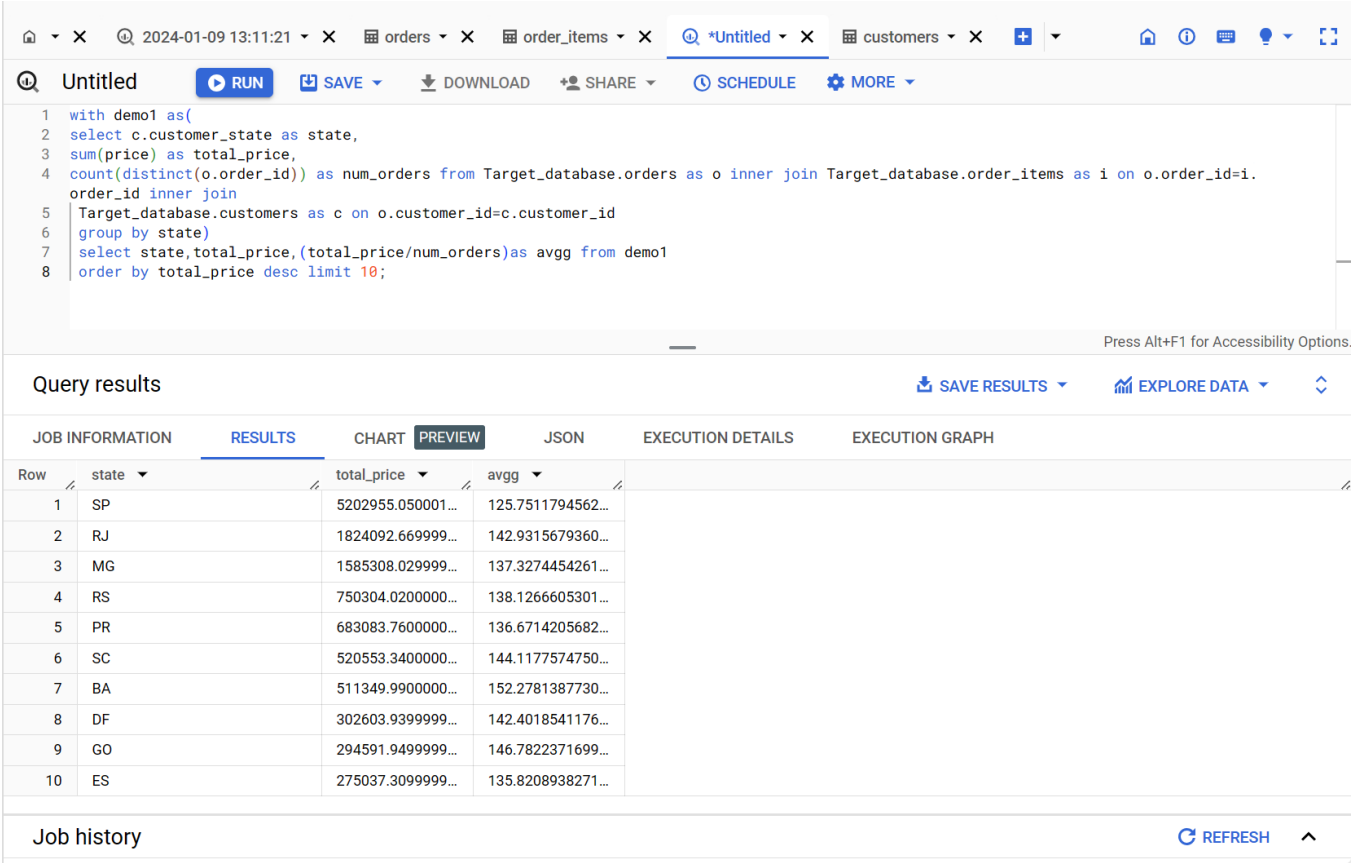
Job history

REFRESH

QUESTION10: Calculate the Total & Average value of order price for each state.

```
with demo1 as(
select c.customer_state as state,
sum(price) as total_price,
count(distinct(o.order_id)) as num_orders from
Target_database.orders as o inner join Target_database.order_items as
i on o.order_id=i.order_id inner join
 Target_database.customers as c on o.customer_id=c.customer_id
 group by state)
 select state,total_price,(total_price/num_orders)as avgg from demo1
 order by total_price desc limit 10;
```

output:



Row	state	total_price	avgg
1	SP	5202955.050001...	125.7511794562...
2	RJ	1824092.669999...	142.9315679360...
3	MG	1585308.029999...	137.3274454261...
4	RS	750304.0200000...	138.1266605301...
5	PR	683083.7600000...	136.6714205682...
6	SC	520553.3400000...	144.1177574750...
7	BA	511349.9900000...	152.2781387730...
8	DF	302603.9399999...	142.4018541176...
9	GO	294591.9499999...	146.7822371699...
10	ES	275037.3099999...	135.8208938271...

QUESTION11: Calculate the Total & Average value of order freight for each state.

```
with demo3 as
 (  select c.customer_state as state , extract(year from
timestamp(o.order_purchase_timestamp)) as year,extract(month from
timestamp(o.order_purchase_timestamp))as
month,(sum(freight_value) / COUNT(distinct o.order_id)) AS
freight_value  from  Target_database.orders as o inner join
Target_database.order_items
 as i
on o.order_id = i.order_id inner join Target_database.customers as c
on o.customer_id=c.customer_id    GROUP BY year, month,state )
 SELECT
month,state,
price_2017,
price_2018,
price_2016,
round((price_2016 + price_2017 + price_2016 / 3) )AS freight_Avgg
FROM
(
SELECT
month,demo3.state,
sum(CASE WHEN year = 2016 THEN freight_value ELSE 0 END)
AS price_2016,
sum(CASE WHEN year = 2017 THEN freight_value ELSE 0 END)
AS price_2017,
sum(CASE WHEN year = 2018 THEN freight_value ELSE 0 END)
AS price_2018

FROM demo3
WHERE (year = 2017 OR year = 2018 or year= 2016) AND month
BETWEEN 1 AND 12
GROUP BY month,state
order by month
);
```

Output:

Untitled | 2024-01-10 00:28:13 | *2024-01-10 00:00:31 | 2024-01-09 13:11:21 | orders | cus

2024-01-10 00:00:31 RUN SAVE QUERY SHARE SCHEDULE MORE Query completed.

```
with demo3 as
( select c.customer_state as state , extract(year from timestamp(o.order_purchase_timestamp)) as year,extract(month from timestamp(o.
order_purchase_timestamp))as month,(sum(freight_value) / COUNT(distinct o.order_id)) AS freight_value  from  Target_database.orders as o inner
join Target_database.order_items
as i
on o.order_id = i.order_id inner join Target_database.customers as c on o.customer_id=c.customer_id   GROUP BY year, month,state )
SELECT
month,state,
price_2017,
```

Press Alt+F1 for Accessibility Options.

Query results SAVE RESULTS EXPLORE DATA

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

Row	month	state	price_2017	price_2018	price_2016	freight_Avgg
1	1	RJ	21.50260416666...	23.68042841037...	0.0	22.0
2	1	SP	18.33176271186...	16.73730731061...	0.0	18.0
3	1	DF	21.39384615384...	21.67637681159...	0.0	21.0
4	1	RS	26.90415094339...	23.55986595174...	0.0	27.0
5	1	CE	24.07111111111...	34.26244444444...	0.0	24.0
6	1	PE	29.71666666666...	34.06913461538...	0.0	30.0
7	1	PR	15.33400000000...	20.74618666666...	0.0	15.0
8	1	BA	22.21399999999...	30.72485106382...	0.0	22.0
9	1	MG	22.39877358490...	22.48529069767...	0.0	22.0
10	1	RN	28.65199999999...	31.14622222222...	0.0	29.0
11	1	PA	45.95833333333...	39.79728571428...	0.0	46.0

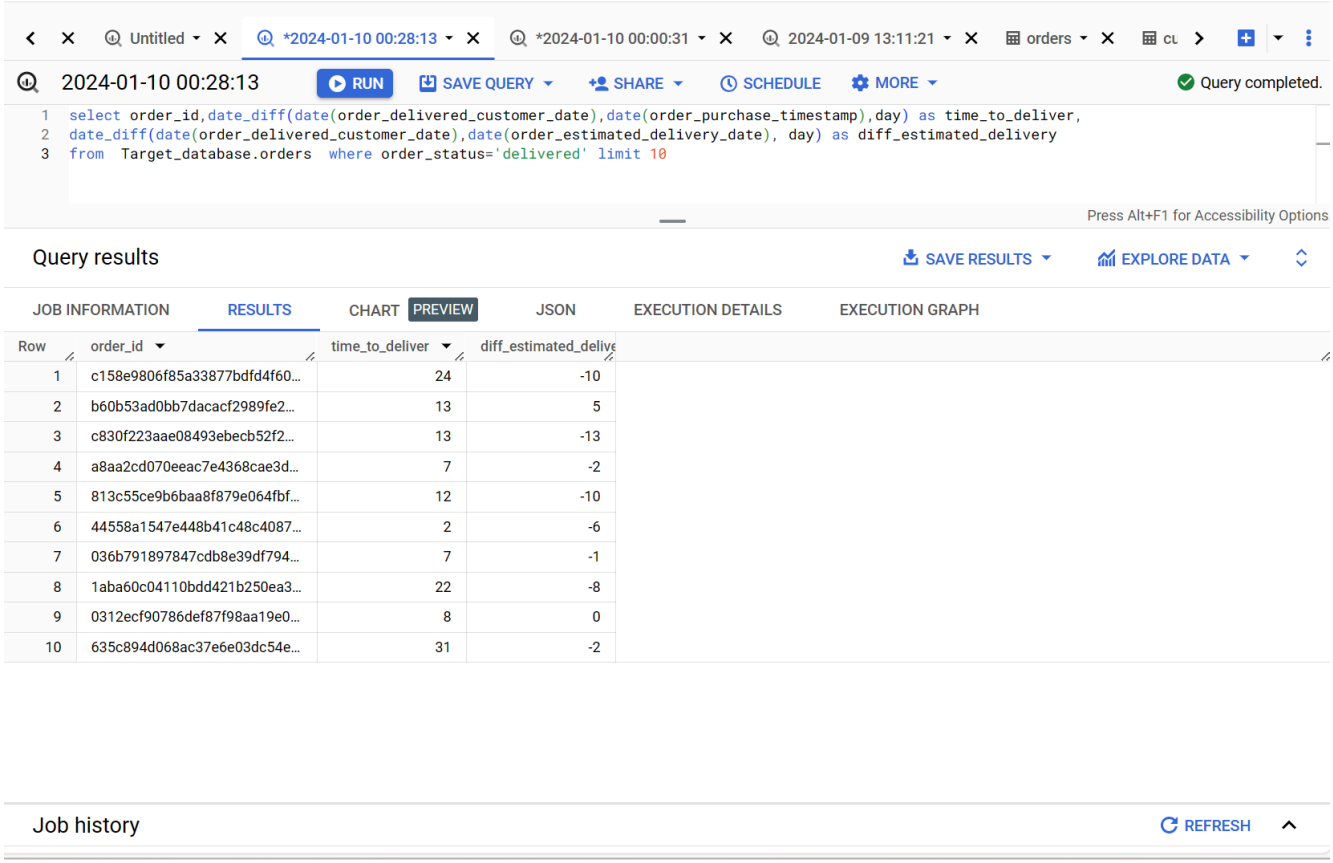
Results per page: 50 1 – 50 of 322

Job history REFRESH

QUESTION12: . Find the no. of days taken to deliver each order from the order's purchase date as delivery time. Also, calculate the difference (in days) between the estimated & actual delivery date of an order. Do this in a single query.

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

output:

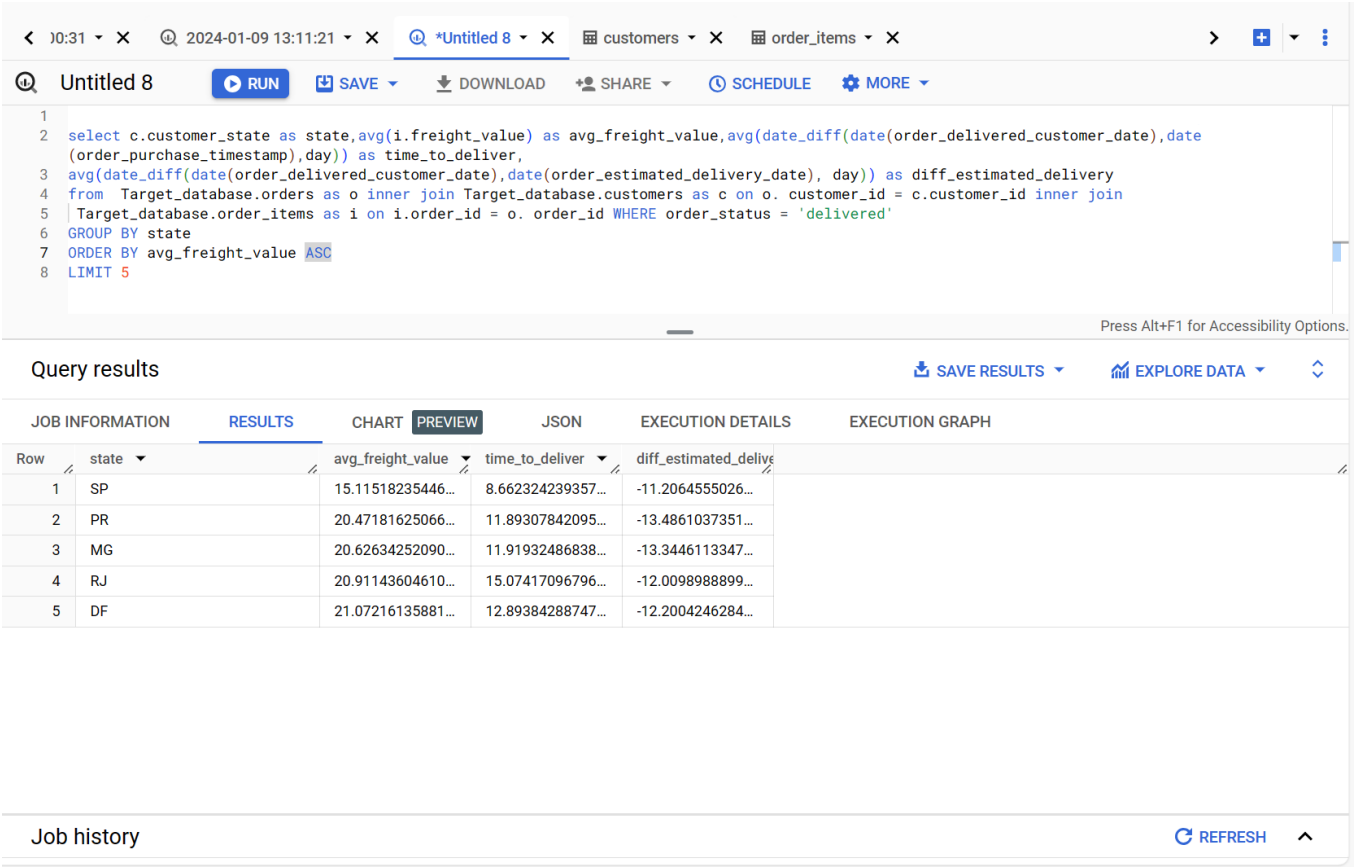


Row	order_id	time_to_deliver	diff_estimated_delive
1	c158e9806f85a33877bdfd4f60...	24	-10
2	b60b53ad0bb7dacacf2989fe2...	13	5
3	c830f223aae08493ebecb52f2...	13	-13
4	a8aa2cd070eeac7e4368cae3d...	7	-2
5	813c55ce9b6baa8f879e064fbf...	12	-10
6	44558a1547e448b41c48c4087...	2	-6
7	036b791897847cdb8e39df794...	7	-1
8	1aba60c04110bdd421b250ea3...	22	-8
9	0312ecf90786def87f98aa19e0...	8	0
10	635c894d068ac37e6e03dc54e...	31	-2

QUESTION13 :Find out the top 5 states with the highest & lowest average freight value

```
select c.customer_state as state,avg(i.freight_value) as avg_freight_value,avg(date_diff(date(order_delivered_customer_date),date(order_purchase_timestamp),day)) as time_to_deliver,
avg(date_diff(date(order_delivered_customer_date),date(order_estimated_delivery_date), day)) as diff_estimated_delivery
from  Target_database.orders as o inner join Target_database.customers as c on o. customer_id = c.customer_id inner join
 Target_database.order_items as i on i.order_id = o. order_id WHERE order_status = 'delivered'
GROUP BY state
ORDER BY avg_freight_value ASC
LIMIT 5
```

Output:



Row	state	avg_freight_value	time_to_deliver	diff_estimated_delive
1	SP	15.11518235446...	8.662324239357...	-11.2064555026...
2	PR	20.47181625066...	11.89307842095...	-13.4861037351...
3	MG	20.62634252090...	11.91932486838...	-13.3446113347...
4	RJ	20.91143604610...	15.07417096796...	-12.0098988899...
5	DF	21.07216135881...	12.89384288747...	-12.2004246284...

QUESTION14: Find out the top 5 states with the highest & lowest average delivery time.

```
(with demo as (
select g.geolocation_state as
state,SUM(TIMESTAMP_DIFF(TIMESTAMP(order_estimat
ed_delivery_date),
TIMESTAMP(order_purchase_timestamp),
DAY)) / COUNT(ORDER_ID) AS avg_del_time from
Target_database.orders as o inner join
Target_database.customers as c on o.customer_id =
c.customer_id
 inner join Target_database.geolocation as g on
g.geolocation_zip_code_prefix = c.customer_zip_code_prefix
WHERE order_status = 'delivered'group by state order by
avg_del_time desc limit 5)
 select demo.state,demo.avg_del_time from demo)

 UNION ALL

 (with demo1 as (
select g.geolocation_state as
state,SUM(TIMESTAMP_DIFF(TIMESTAMP(order_estimat
ed_delivery_date),
TIMESTAMP(order_purchase_timestamp),
DAY)) / COUNT(ORDER_ID) AS avg_del_time from
Target_database.orders as o inner join
Target_database.customers as c on o.customer_id =
c.customer_id
 inner join Target_database.geolocation as g on
g.geolocation_zip_code_prefix = c.customer_zip_code_prefix
WHERE order_status = 'delivered'group by state order by
avg_del_time ASC limit 5)
select demo1.state,demo1.avg_del_time from demo1)
```

output:

Untitled | geolocation | top 5 states with the high... ime | 2024-01-10 00:00:31

top 5 states with the highe...ime

RUN | SAVE QUERY | SHARE | SCHEDULE | MORE | This query will process 24.44 MB

```
(with demo as (
select g.geolocation_state as state,SUM(TIMESTAMP_DIFF(TIMESTAMP(order_estimated_delivery_date), TIMESTAMP(order_purchase_timestamp),
DAY)) / COUNT(ORDER_ID) AS avg_del_time from Target_database.orders as o inner join Target_database.customers as c on o.customer_id = c.
customer_id
 inner join Target_database.geolocation as g on g.geolocation_zip_code_prefix = c.customer_zip_code_prefix WHERE order_status =
```

Press Alt+F1 for Accessibility Options.

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	state	avg_del_time
1	AP	46.56841445581...
2	RR	45.25946547884...
3	AM	45.13338205737...
4	AC	39.21026049973...
5	RO	37.63690709525...
6	SP	19.04831245981...
7	PR	23.96693288404...
8	DF	24.08380053436...
9	MG	24.11110763087...
10	ES	25.00719754878...

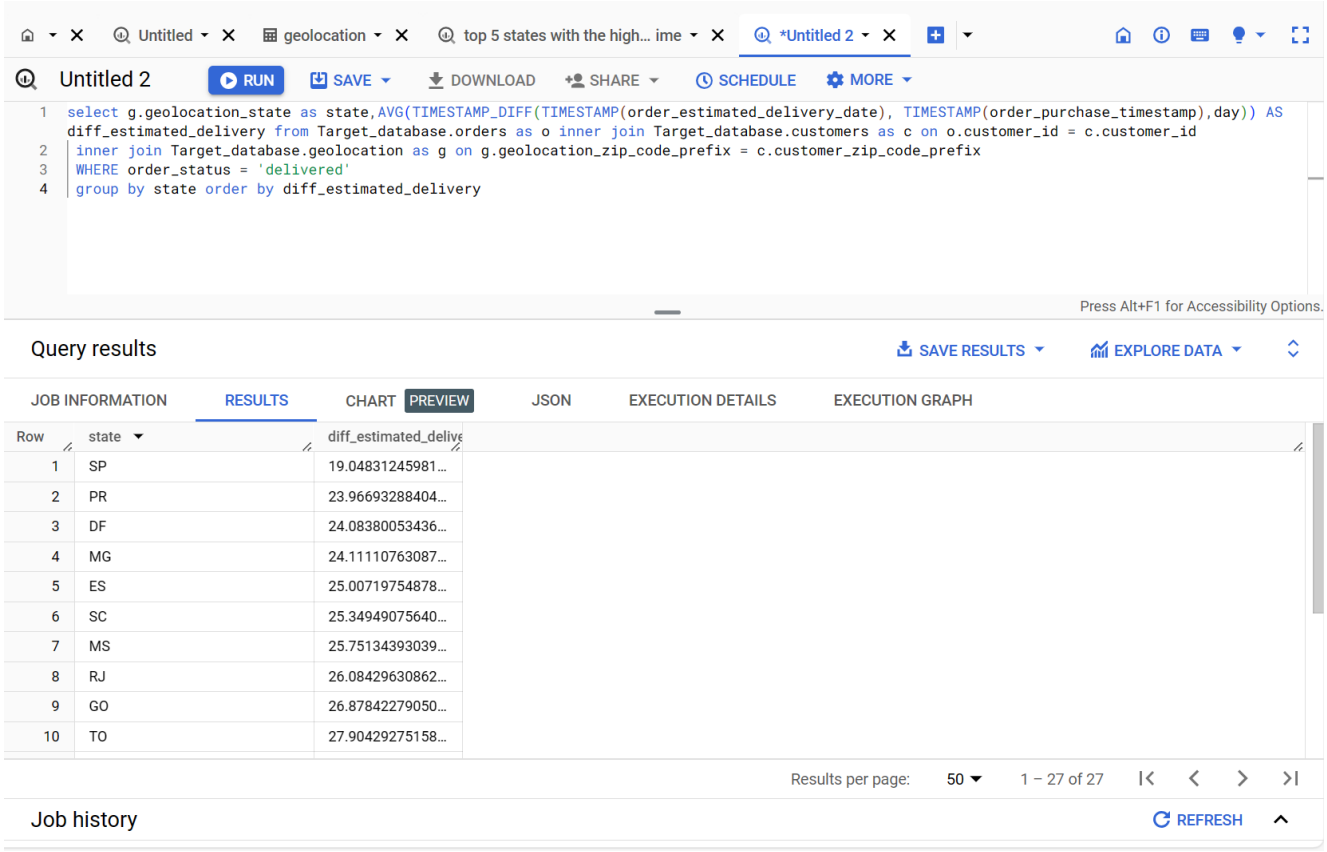
Job history

REFRESH

QUESTION15: Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery

```
select g.geolocation_state as
state,AVG(TIMESTAMP_DIFF(TIMESTAMP(order_estimated_deli
very_date), TIMESTAMP(order_purchase_timestamp),day)) AS
diff_estimated_delivery from Target_database.orders as o inner join
Target_database.customers as c on o.customer_id = c.customer_id
 inner join Target_database.geolocation as g on
g.geolocation_zip_code_prefix = c.customer_zip_code_prefix
 WHERE order_status = 'delivered'
 group by state order by diff_estimated_delivery
```

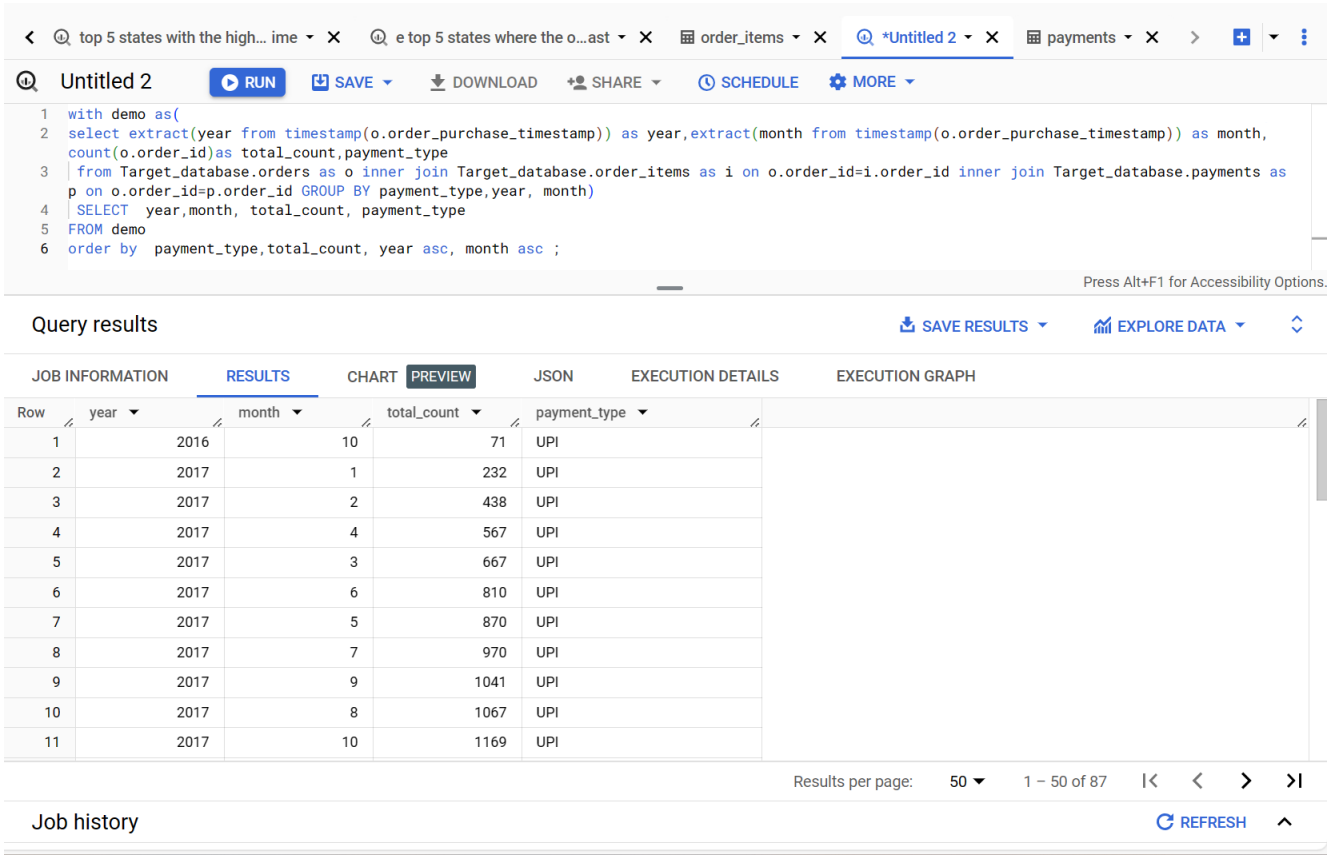
output:



QUESTION16: Find the month on month no. of orders placed using different payment types

```
with demo as(
select extract(year from timestamp(o.order_purchase_timestamp)) as year,extract(month from timestamp(o.order_purchase_timestamp)) as month,count(o.order_id)as total_count,payment_type
 from Target_database.orders as o inner join Target_database.order_items as i on o.order_id=i.order_id inner join Target_database.payments as p on o.order_id=p.order_id GROUP BY payment_type,year, month)
 SELECT  year,month, total_count, payment_type
FROM demo
order by  payment_type,total_count, year asc, month asc ;
```

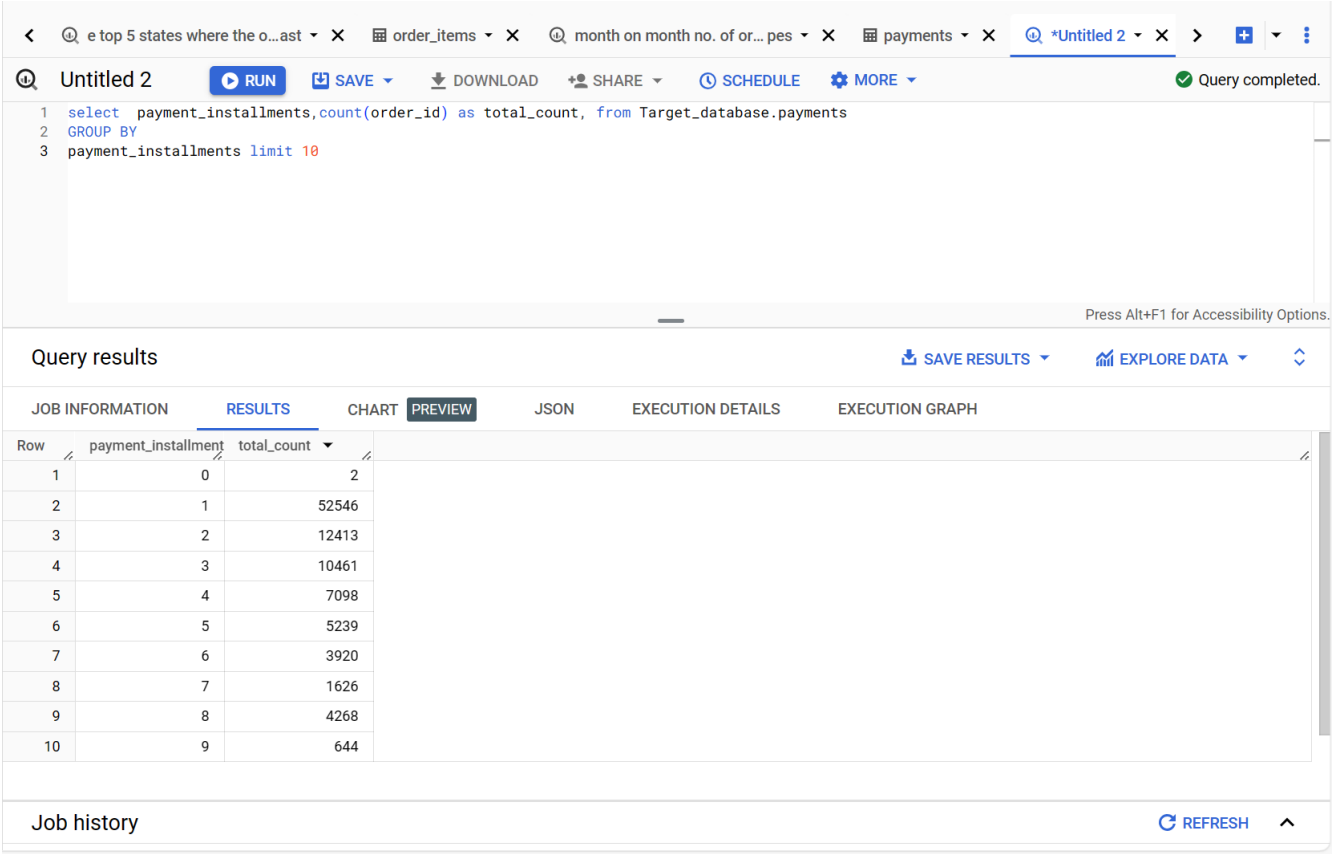
Output:



QUESTION17: Find the no. of orders placed on the basis of the payment installments that have been paid.

```
select  payment_installments,count(order_id) as total_count,
from Target_database.payments
GROUP BY
payment_installments
```

output:



Row	payment_installment	total_count
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