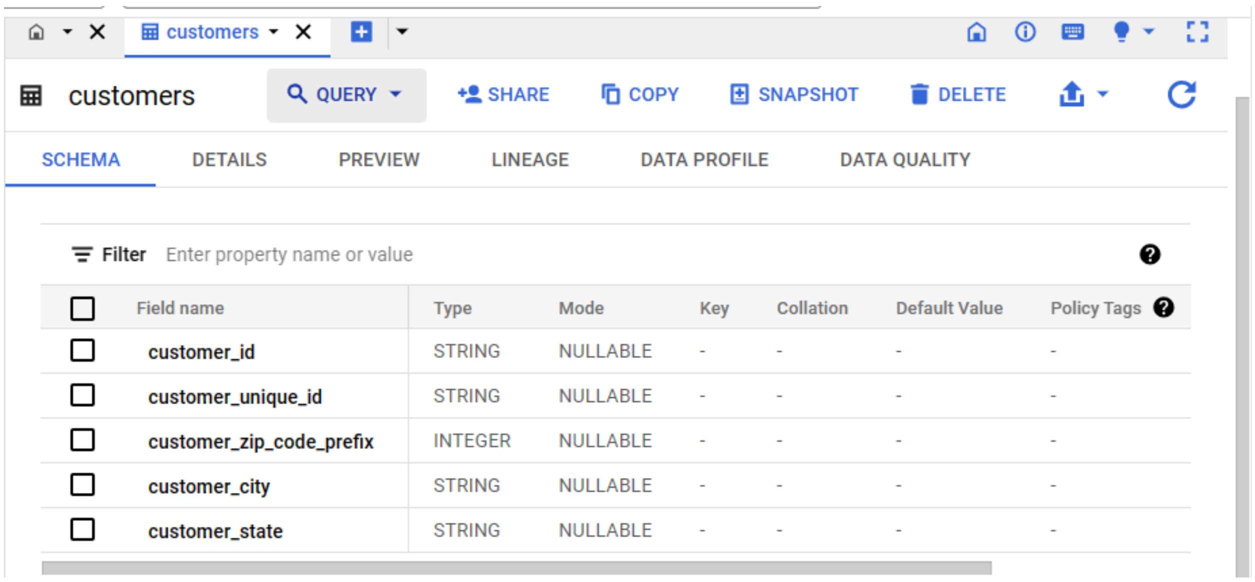
TARGET BUSINESS CASE

1. **Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset.**
   a. Data type of all columns in the “customers” table.



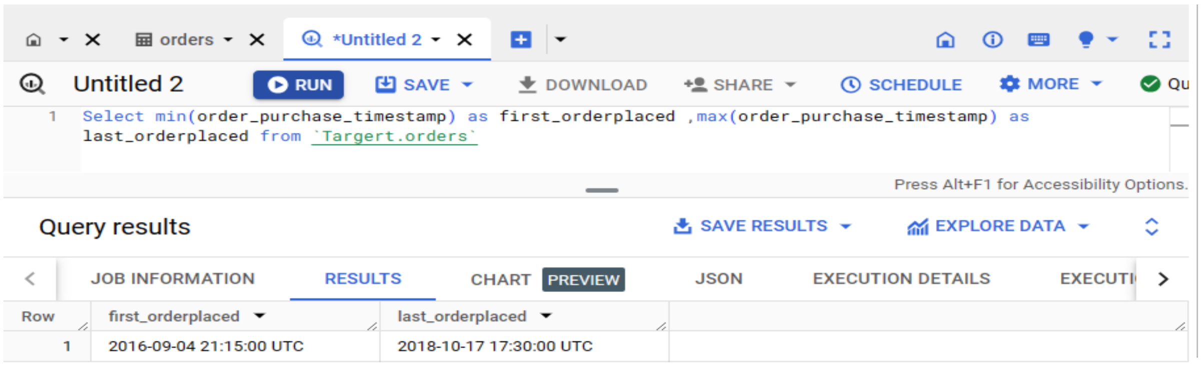
customers

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

Observation: It was observed that various fields have different data types

- customer_id is string
- Customer_unique_id is string
- Customer_zip_code_prefix is Integer
- Customer_city is string, Customer_state is string

b. Get the time range between which the orders were placed.



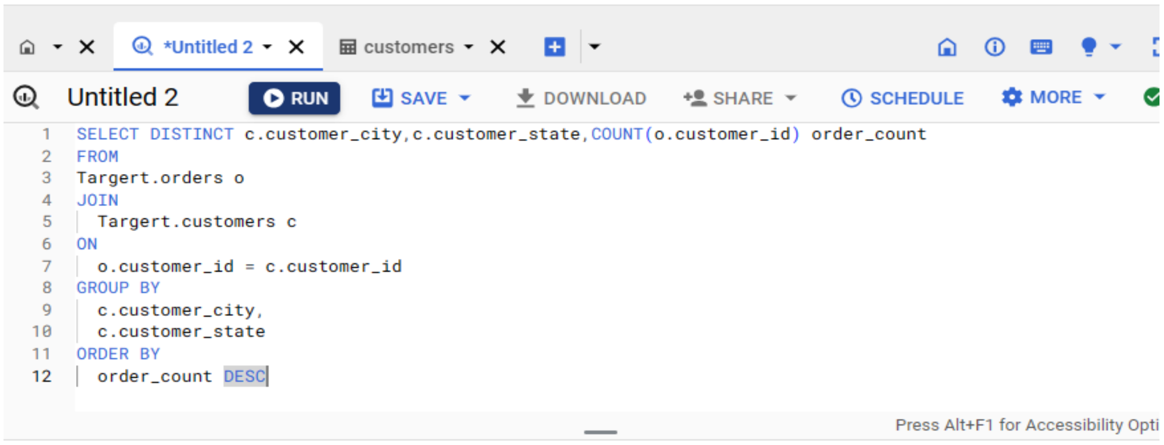
```
Select min(order_purchase_timestamp) as first_orderplaced ,max(order_purchase_timestamp) as
last_orderplaced from `Targert.orders`
```

Row	first_orderplaced	last_orderplaced
1	2016-09-04 21:15:00 UTC	2018-10-17 17:30:00 UTC

Observation: It was observed that first order and placed in 04th of September 2016 and last order was placed on 17th of October 2018

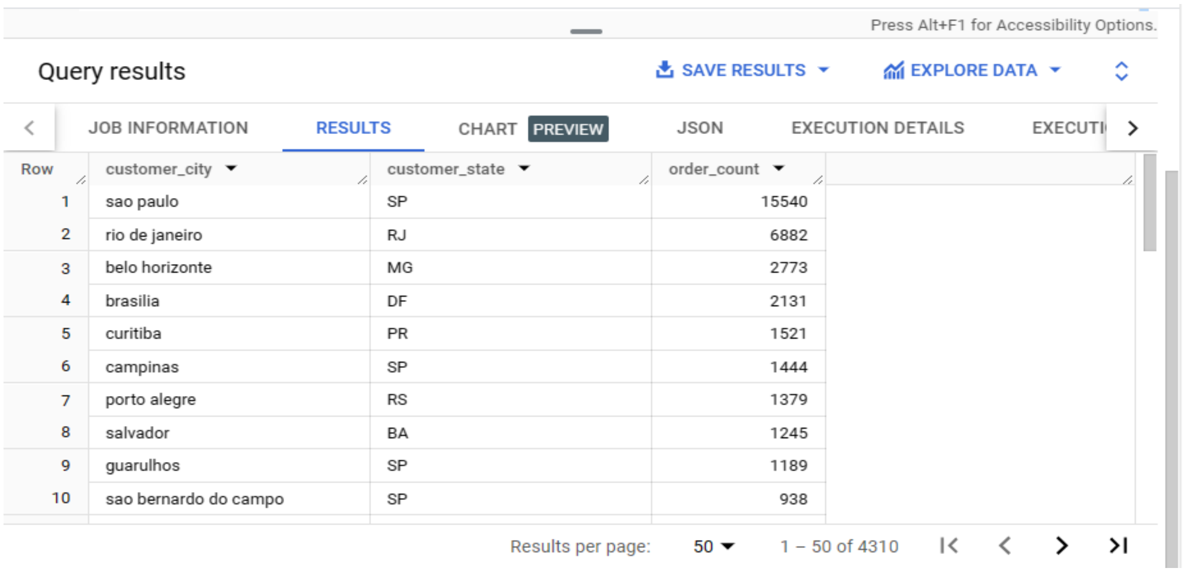
1. **Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset.**

   c. Count the Cities & States of customers who ordered during the given period.



```sql
SELECT DISTINCT c.customer_city,c.customer_state,COUNT(o.customer_id) order_count
FROM
Targert.orders o
JOIN
  Targert.customers c
ON
  o.customer_id = c.customer_id
GROUP BY
  c.customer_city,
  c.customer_state
ORDER BY
  order_count DESC
```

Press Alt+F1 for Accessibility Opti



Press Alt+F1 for Accessibility Options.

Query results

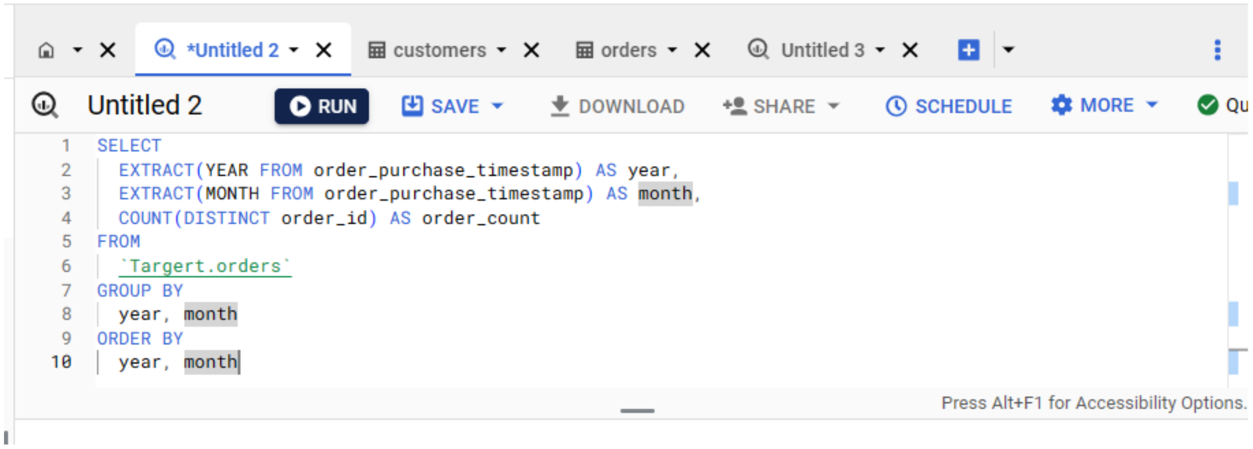
Row	customer_city	customer_state	order_count
1	sao paulo	SP	15540
2	rio de janeiro	RJ	6882
3	belo horizonte	MG	2773
4	brasilia	DF	2131
5	curitiba	PR	1521
6	campinas	SP	1444
7	porto alegre	RS	1379
8	salvador	BA	1245
9	guarulhos	SP	1189
10	sao bernardo do campo	SP	938

Results per page: 50 1 – 50 of 4310

Observation: It was observed maximum orders were placed from Sap paulo from SP state 15540 orders

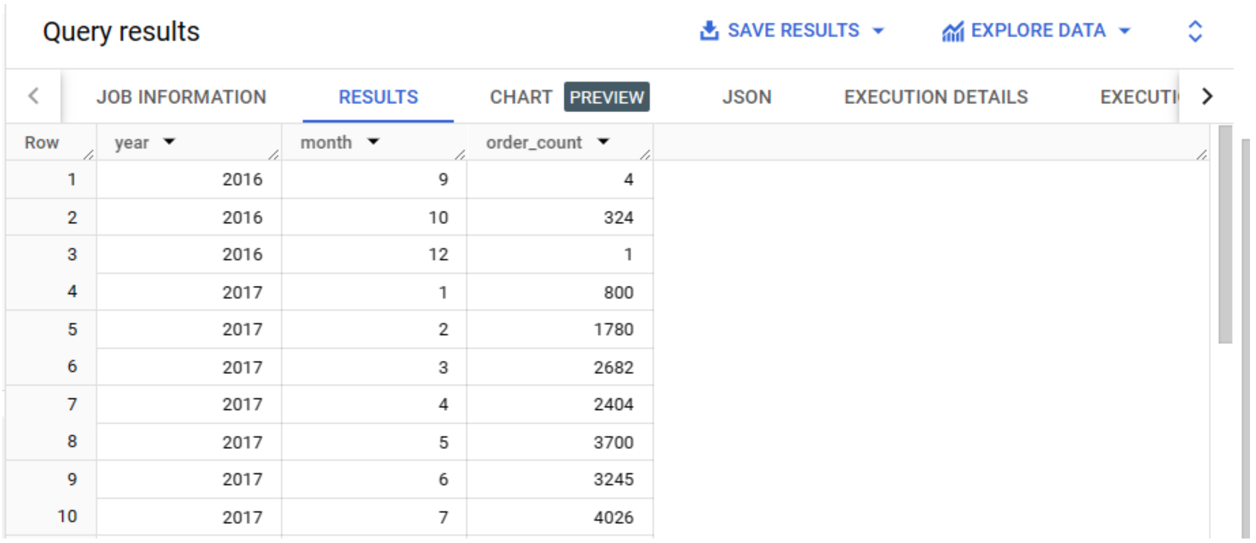
2. **In-depth Exploration:**
    a. Is there a growing trend in the no. of orders placed over the past years?

Query



```
SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
  COUNT(DISTINCT order_id) AS order_count
FROM
  `Targert.orders`
GROUP BY
  year, month
ORDER BY
  year, month
```

Result



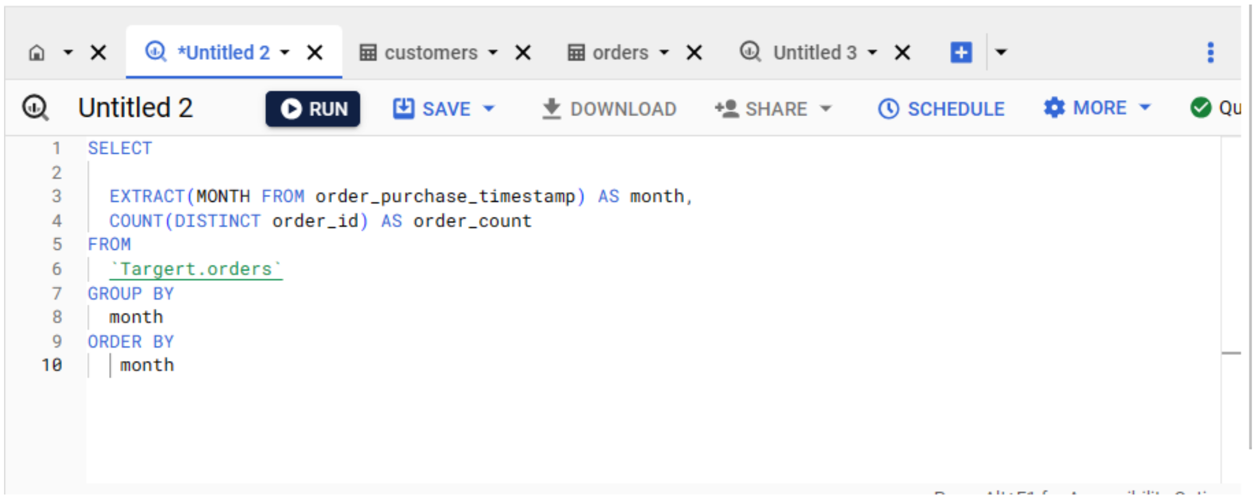
Row	year	month	order_count
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026

Observation: It was oberserved that first order were placed on November 2016 and in December of 2016 was the worst time only 4 orders were placed with the time order were increasing from 800 in Jan 2017 to 4026 in July 2017

2. **In-depth Exploration:**

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

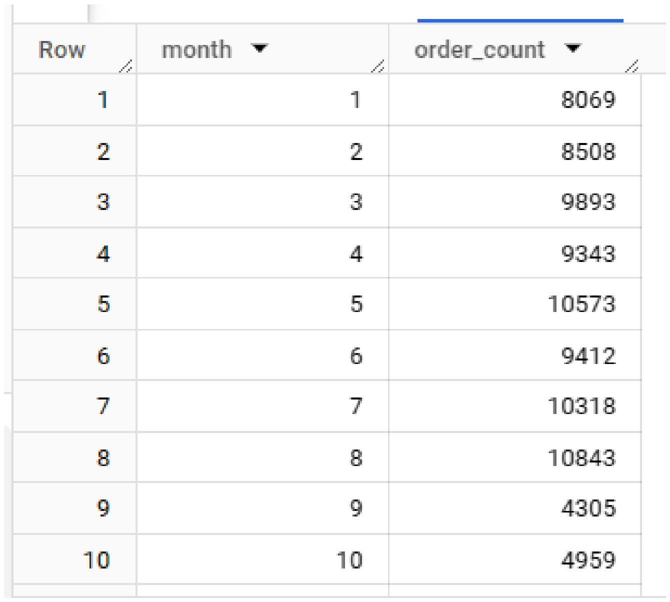
   Query



```
SELECT

  EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
  COUNT(DISTINCT order_id) AS order_count
FROM
  `Targert.orders`
GROUP BY
  month
ORDER BY
  month
```

   Result

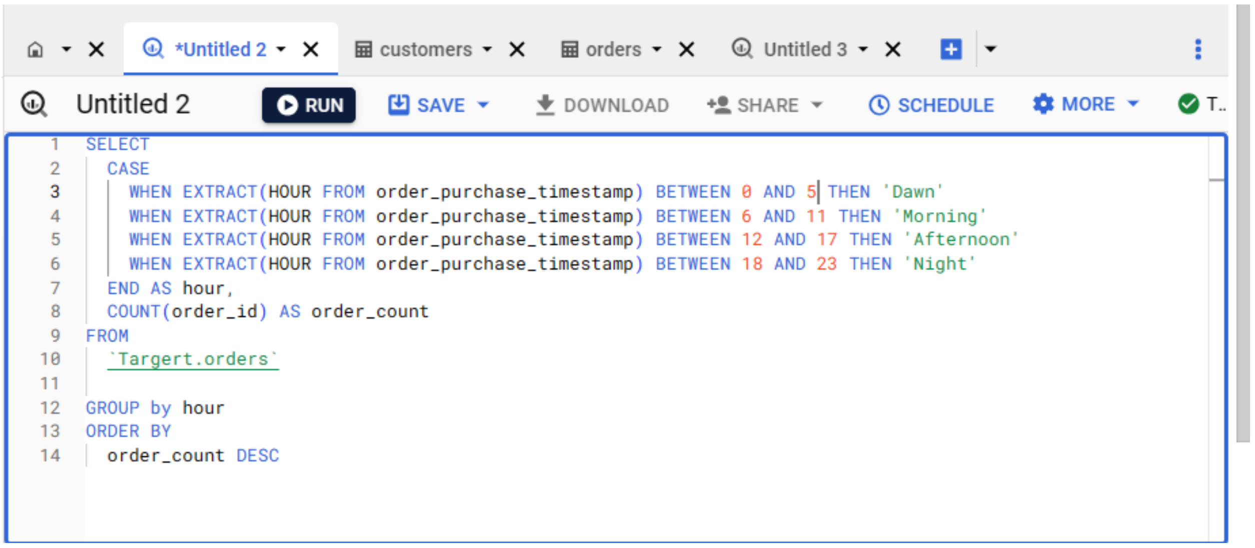


Row	month	order_count
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318
8	8	10843
9	9	4305
10	10	4959

Observation: it was Seen there was gradual increase in market from jan to oct sales were orders were increasing every month from 8069 to 10843. In August sales were maximum .but there was a decrease in orders in November month from 10843 to 4305

2. **In-depth Exploration:**
    c. During what time of the day, do the Brazilian customers mostly place theirorders? (Dawn, Morning, Afternoon or Night)

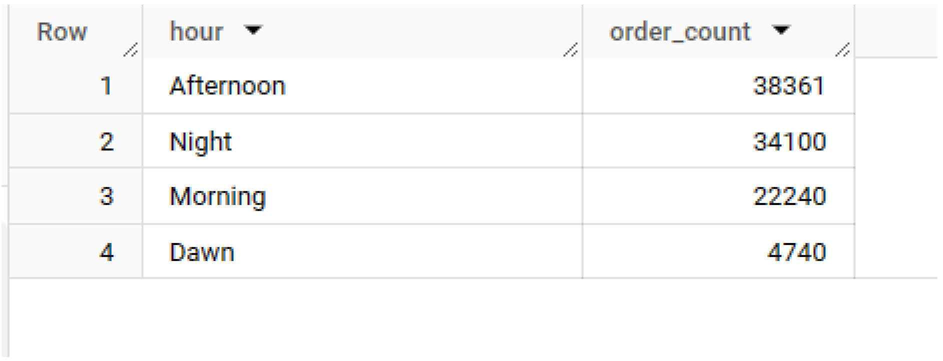
Query



```
SELECT
  CASE
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 5 THEN 'Dawn'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 6 AND 11 THEN 'Morning'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 12 AND 17 THEN 'Afternoon'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 18 AND 23 THEN 'Night'
  END AS hour,
  COUNT(order_id) AS order_count
FROM
  `Targert.orders`

GROUP by hour
ORDER BY
  order_count DESC
```

Result



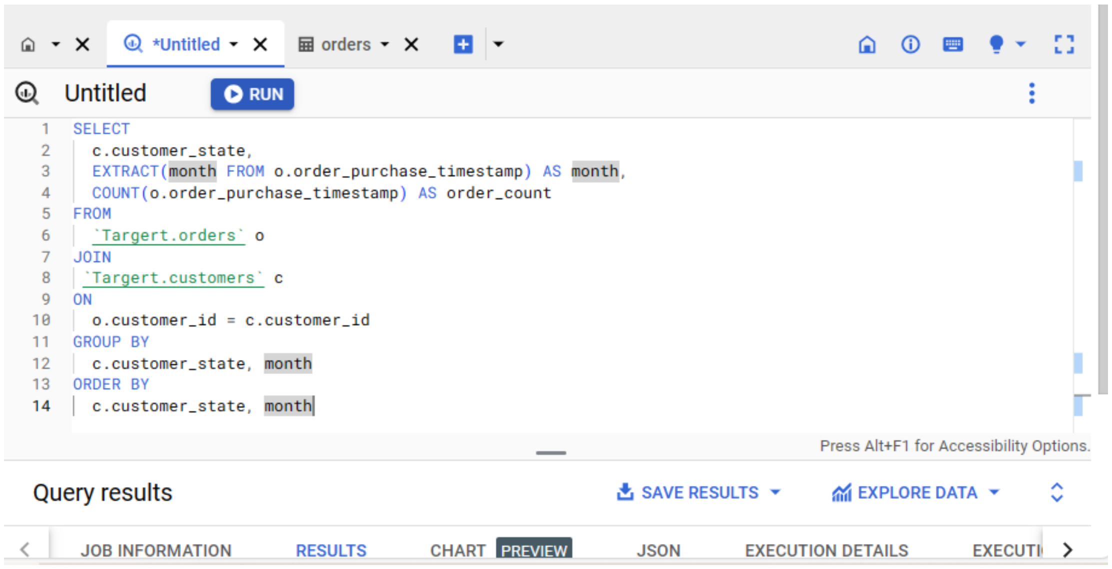
Row	hour	order_count
1	Afternoon	38361
2	Night	34100
3	Morning	22240
4	Dawn	4740

Observation: It was observed that During afternoon maximum people were used to placed orders

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

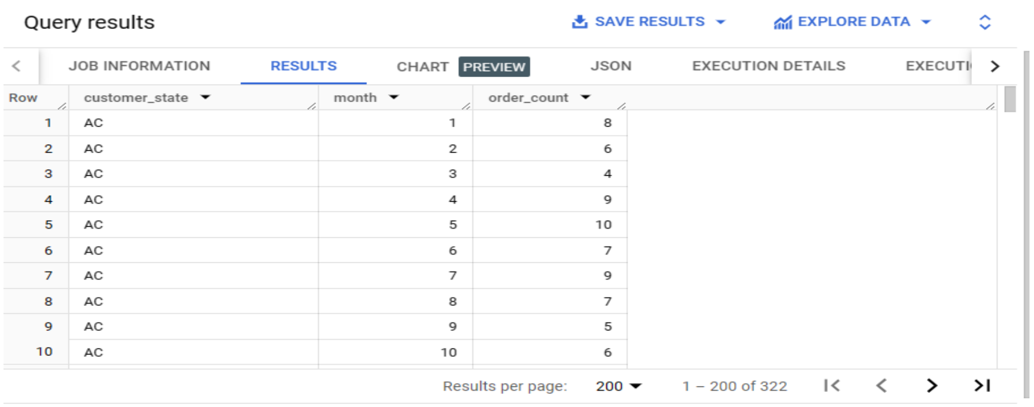
a. Get the month on month no. of orders placed in each state.

QUERY



```
SELECT
  c.customer_state,
  EXTRACT(month FROM o.order_purchase_timestamp) AS month,
  COUNT(o.order_purchase_timestamp) AS order_count
FROM
  `Targert.orders` o
JOIN
 `Targert.customers` c
ON
  o.customer_id = c.customer_id
GROUP BY
  c.customer_state, month
ORDER BY
  c.customer_state, month
```

Result



Row	customer_state	month	order_count
1	AC	1	8
2	AC	2	6
3	AC	3	4
4	AC	4	9
5	AC	5	10
6	AC	6	7
7	AC	7	9
8	AC	8	7
9	AC	9	5
10	AC	10	6

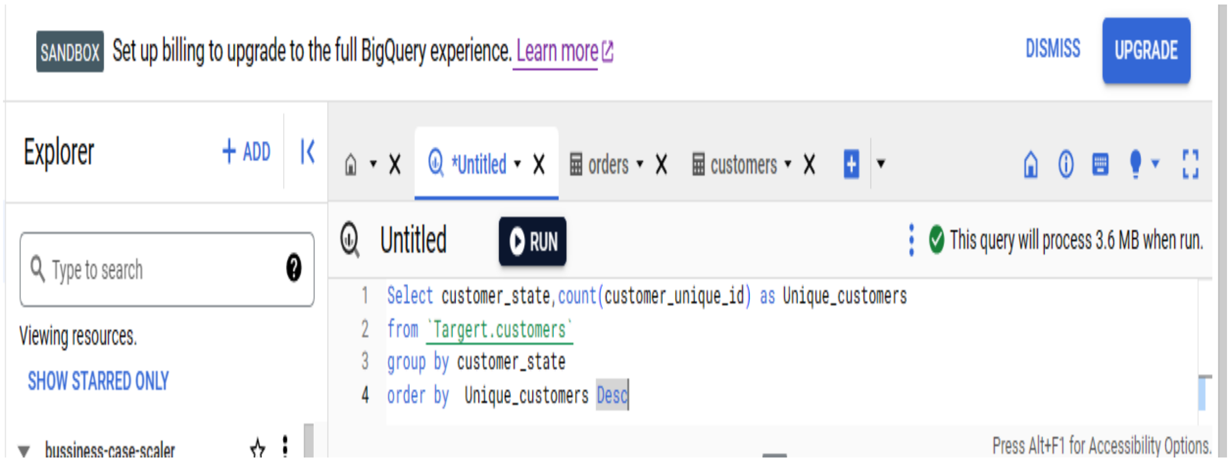
Results per page: 200 1 – 200 of 322

Observation:- It is observed that SP consistently has the highest number of orders in any given month, followed by Rio de Janeiro (RJ) and Minas Gerais (MG).

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

b. How are the customers distributed across all the states?

QUERY



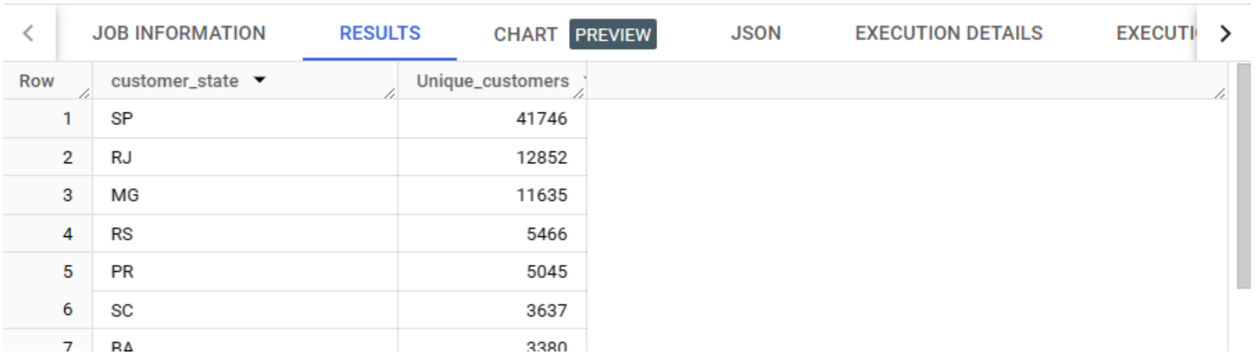
SANDBOX Set up billing to upgrade to the full BigQuery experience. Learn more

Untitled

This query will process 3.6 MB when run.

```
Select customer_state,count(customer_unique_id) as Unique_customers
from `Targert.customers`
group by customer_state
order by  Unique_customers Desc
```

Result



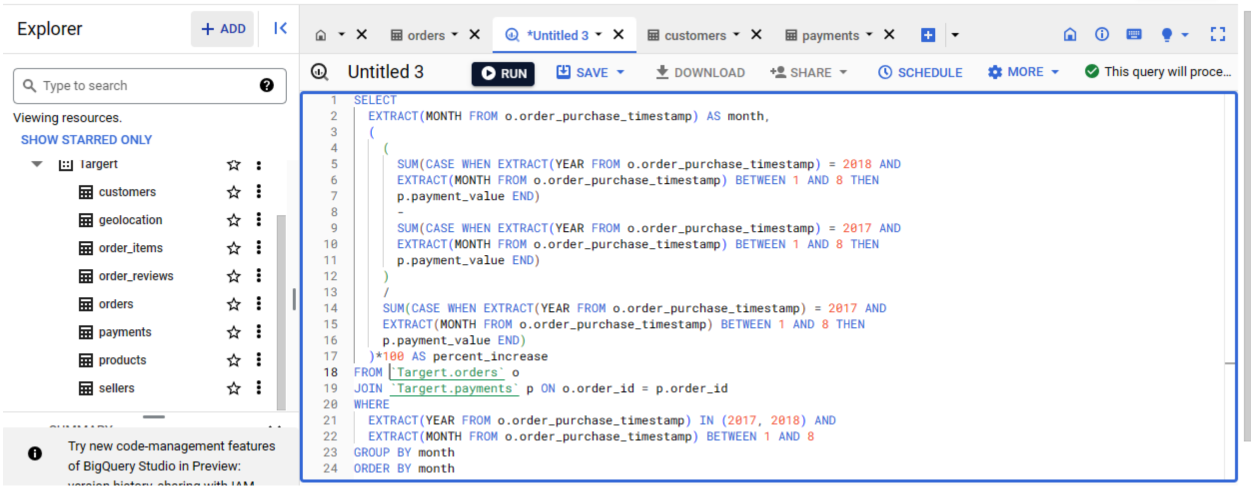
Row	customer_state	Unique_customers
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380

Observation:- It was observed that SP state has maximum customers in Brazilian market

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

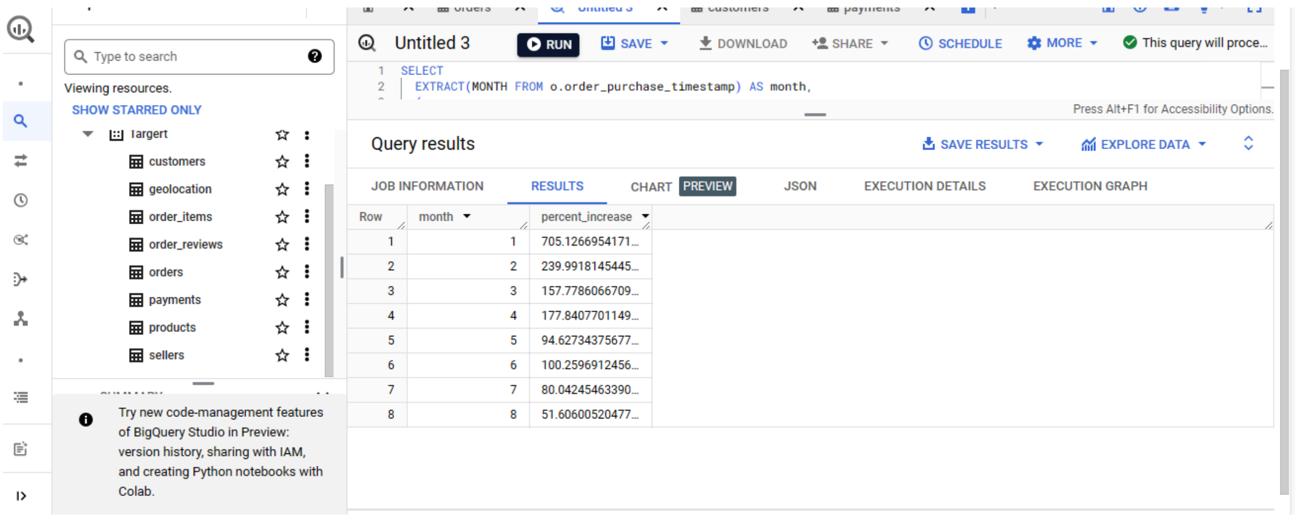
    a. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

QUERY



```
SELECT
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
  (
    (
      SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2018 AND
      EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8 THEN
      p.payment_value END)
      -
      SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017 AND
      EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8 THEN
      p.payment_value END)
    )
    /
    SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017 AND
    EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8 THEN
    p.payment_value END)
  )*100 AS percent_increase
FROM `Targert.orders` o
JOIN `Targert.payments` p ON o.order_id = p.order_id
WHERE
  EXTRACT(YEAR FROM o.order_purchase_timestamp) IN (2017, 2018) AND
  EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
GROUP BY month
ORDER BY month
```

RESULT

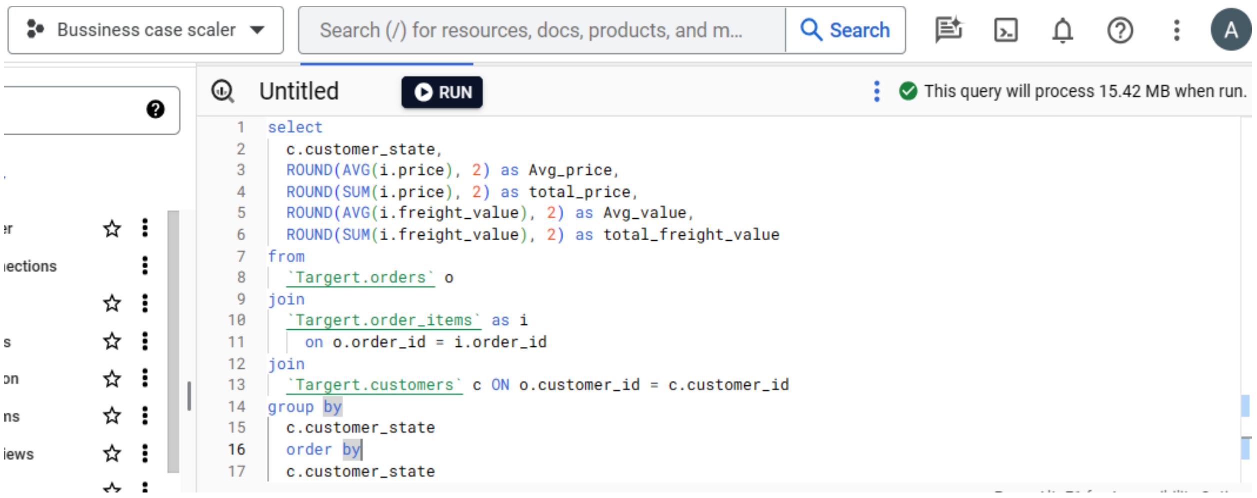


Row	month	percent_increase
1	1	705.1266954171...
2	2	239.9918145445...
3	3	157.7786066709...
4	4	177.8407701149...
5	5	94.62734375677...
6	6	100.2596912456...
7	7	80.04245463390...
8	8	51.60600520477...

OBSERVATION: It is observed that maximum increase is in the month January 705 %

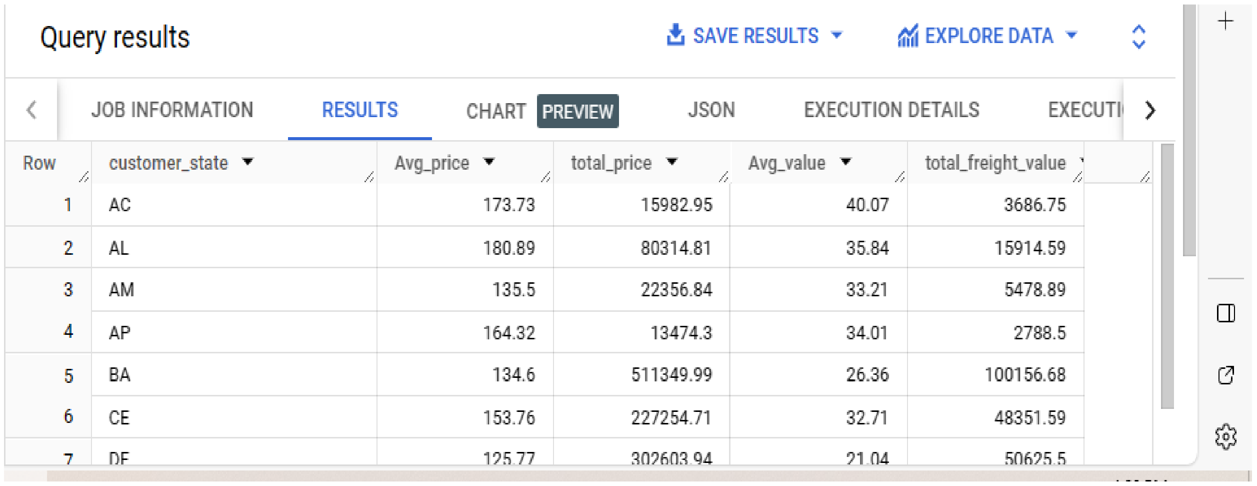
4. **Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.**
   b. Calculate the Total & Average value of order price for each state.
   c. Calculate the Total & Average value of order freight for each state.

QUERY



```
select
  c.customer_state,
  ROUND(AVG(i.price), 2) as Avg_price,
  ROUND(SUM(i.price), 2) as total_price,
  ROUND(AVG(i.freight_value), 2) as Avg_value,
  ROUND(SUM(i.freight_value), 2) as total_freight_value
from
  `Targert.orders` o
join
  `Targert.order_items` as i
   on o.order_id = i.order_id
join
  `Targert.customers` c ON o.customer_id = c.customer_id
group by
  c.customer_state
  order by
  c.customer_state
```

Result



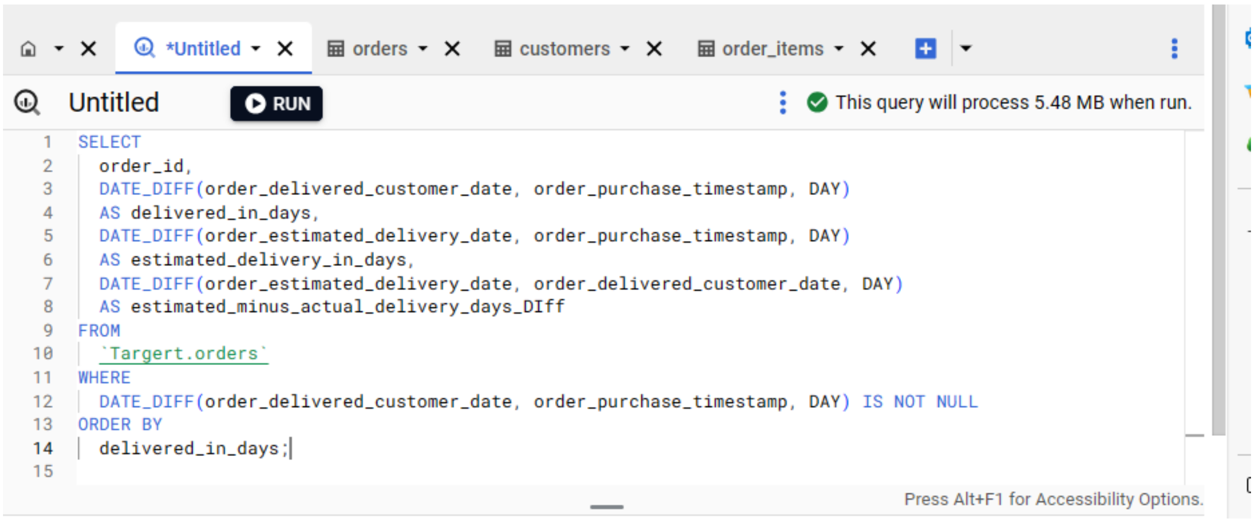
Row	customer_state	Avg_price	total_price	Avg_value	total_freight_value
1	AC	173.73	15982.95	40.07	3686.75
2	AL	180.89	80314.81	35.84	15914.59
3	AM	135.5	22356.84	33.21	5478.89
4	AP	164.32	13474.3	34.01	2788.5
5	BA	134.6	511349.99	26.36	100156.68
6	CE	153.76	227254.71	32.71	48351.59
7	DF	125.77	302603.94	21.04	50625.5

Observation:- It was observed SP state has highest Total price, Average price, Total freight value, Average freight value

5. **Analysis based on sales, freight and delivery time.**

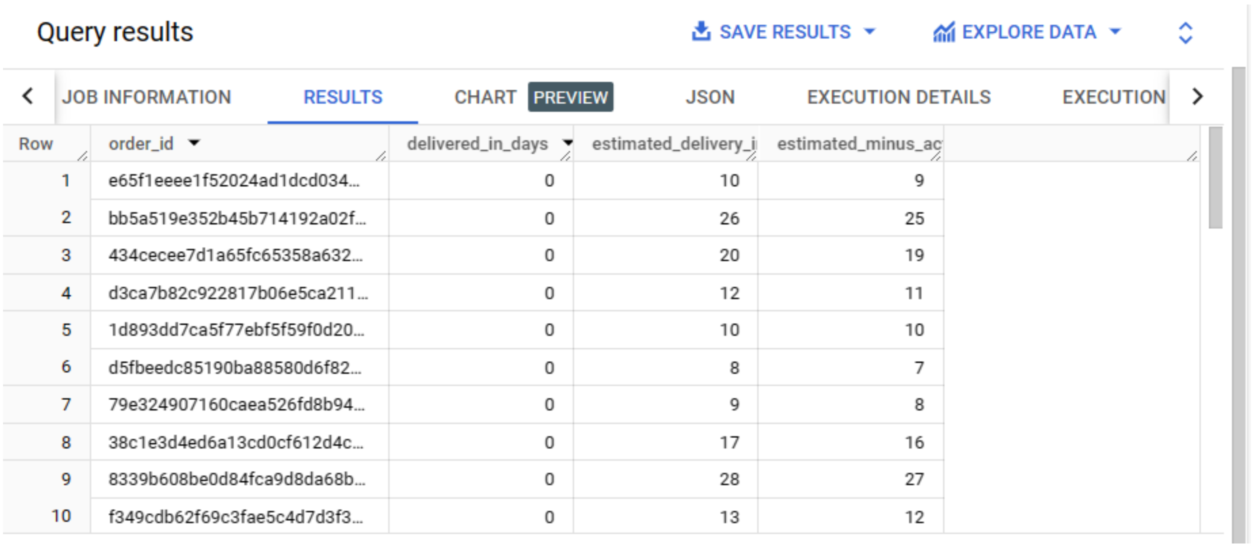
   a. Find the no. of days taken to deliver each order from the order's purchase date as delivery time.Also, calculate the difference (in days) between the estimated & actual delivery date of an order.

QUERY


*Untitled
orders
customers
order_items
Untitled
RUN
This query will process 5.48 MB when run.
Press Alt+F1 for Accessibility Options.

```sql
SELECT
  order_id,
  DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)
  AS delivered_in_days,
  DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, DAY)
  AS estimated_delivery_in_days,
  DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY)
  AS estimated_minus_actual_delivery_days_DIff
FROM
  `Targert.orders`
WHERE
  DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) IS NOT NULL
ORDER BY
  delivered_in_days;
```

Result

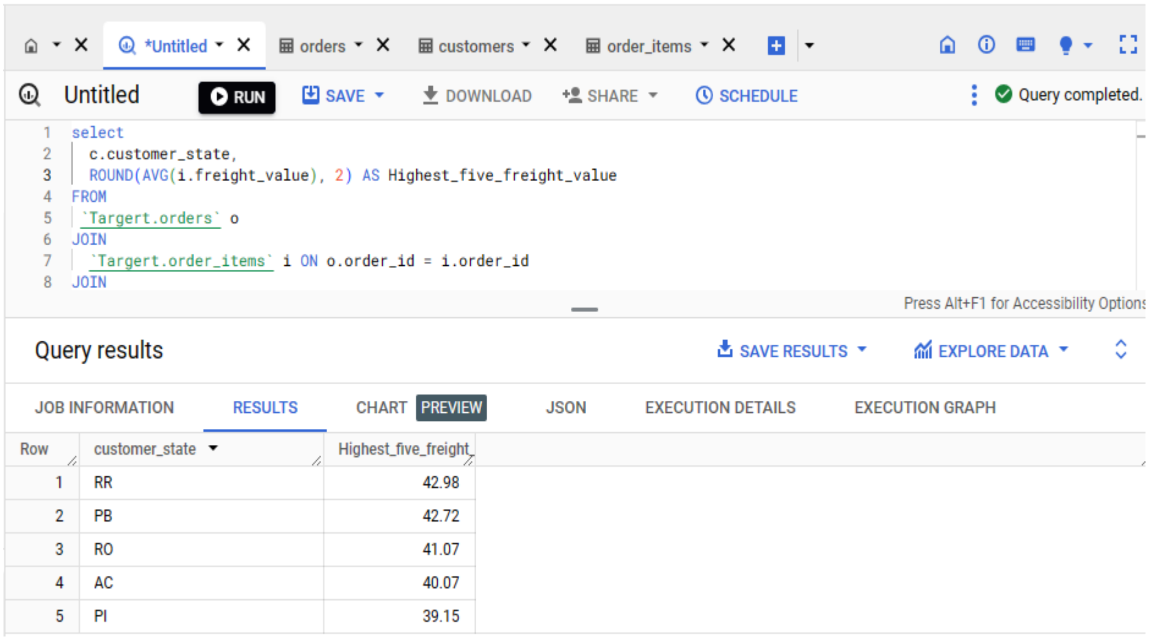


Row	order_id	delivered_in_days	estimated_delivery_i	estimated_minus_ac
1	e65f1eeee1f52024ad1dcd034...	0	10	9
2	bb5a519e352b45b714192a02f...	0	26	25
3	434cecee7d1a65fc65358a632...	0	20	19
4	d3ca7b82c922817b06e5ca211...	0	12	11
5	1d893dd7ca5f77ebf5f59f0d20...	0	10	10
6	d5fbeedc85190ba88580d6f82...	0	8	7
7	79e324907160caea526fd8b94...	0	9	8
8	38c1e3d4ed6a13cd0cf612d4c...	0	17	16
9	8339b608be0d84fca9d8da68b...	0	28	27
10	f349cdb62f69c3fae5c4d7d3f3...	0	13	12

5. **Analysis based on sales, freight and delivery time.**

   b. Find out the top 5 states with the highest & lowest average freight value.

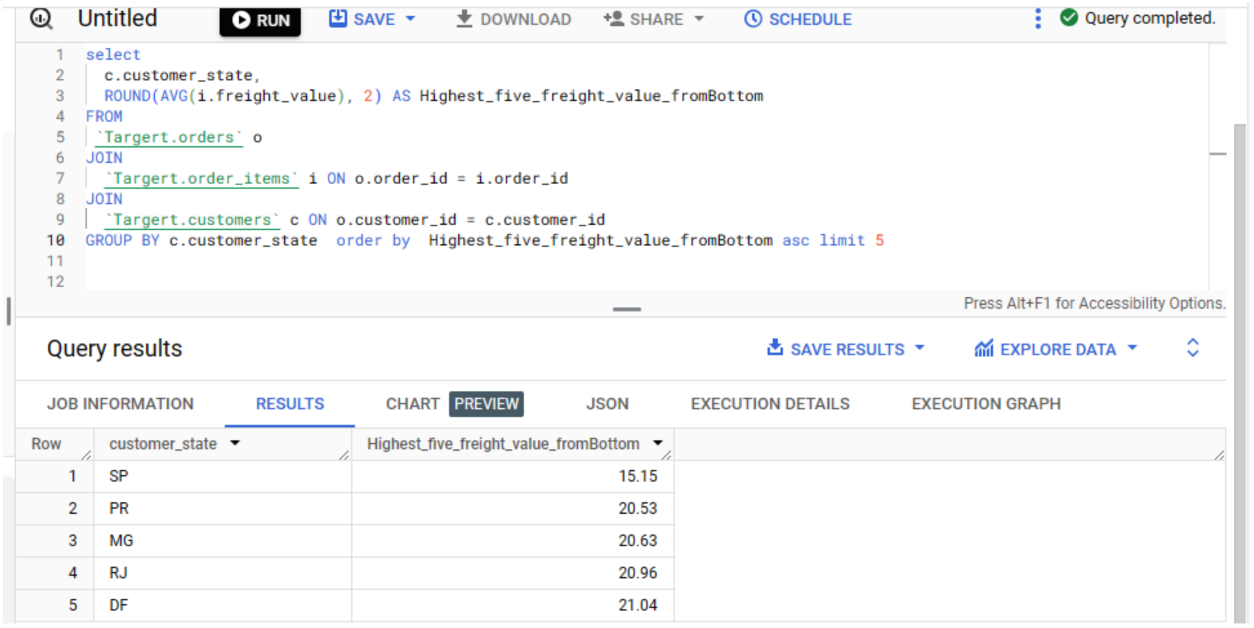
      i. Highest average freight value from top



```
select
  c.customer_state,
  ROUND(AVG(i.freight_value), 2) AS Highest_five_freight_value
FROM
 `Targert.orders` o
JOIN
  `Targert.order_items` i ON o.order_id = i.order_id
JOIN
```

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

      ii. Highest average freight value from Bottom



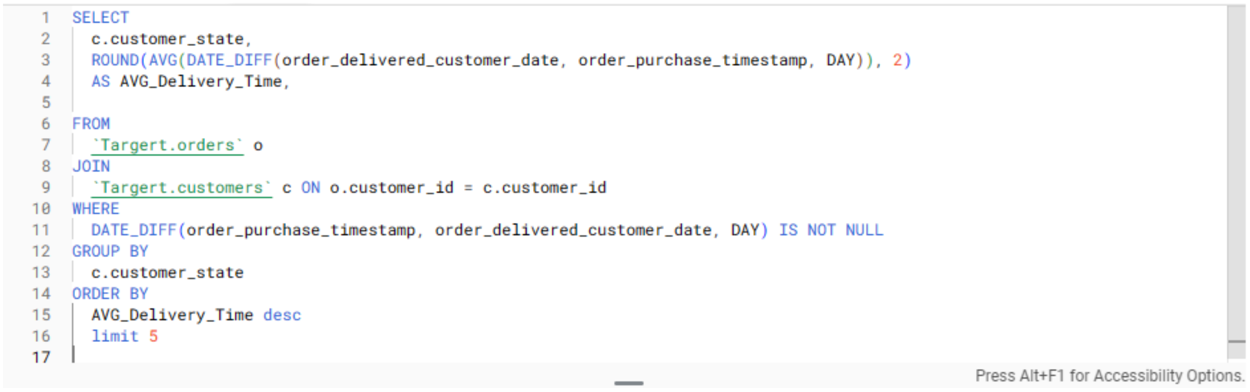
```
select
  c.customer_state,
  ROUND(AVG(i.freight_value), 2) AS Highest_five_freight_value_fromBottom
FROM
 `Targert.orders` o
JOIN
  `Targert.order_items` i ON o.order_id = i.order_id
JOIN
  `Targert.customers` c ON o.customer_id = c.customer_id
GROUP BY c.customer_state  order by  Highest_five_freight_value_fromBottom asc limit 5
```

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

5. **Analysis based on sales, freight and delivery time.**

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

      i. Highest average delivery time.



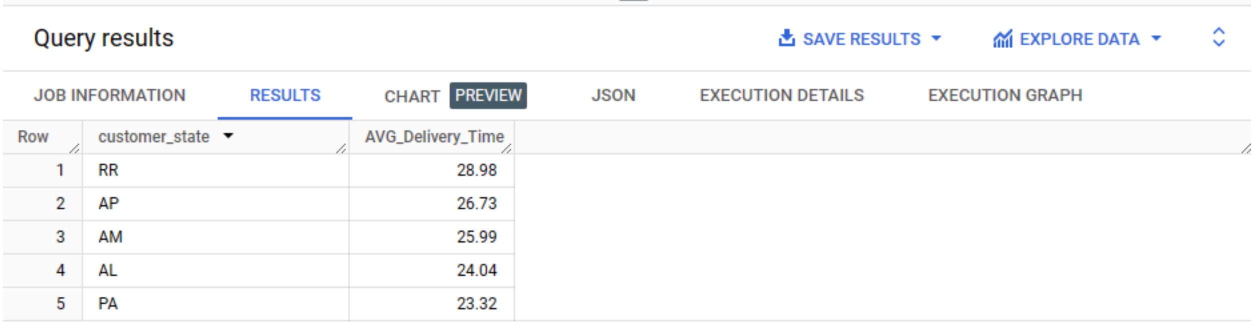
```
SELECT
  c.customer_state,
  ROUND(AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)), 2)
  AS AVG_Delivery_Time,

FROM
  `Targert.orders` o
JOIN
  `Targert.customers` c ON o.customer_id = c.customer_id
WHERE
  DATE_DIFF(order_purchase_timestamp, order_delivered_customer_date, DAY) IS NOT NULL
GROUP BY
  c.customer_state
ORDER BY
  AVG_Delivery_Time desc
  limit 5

```

Press Alt+F1 for Accessibility Options.

RESULT

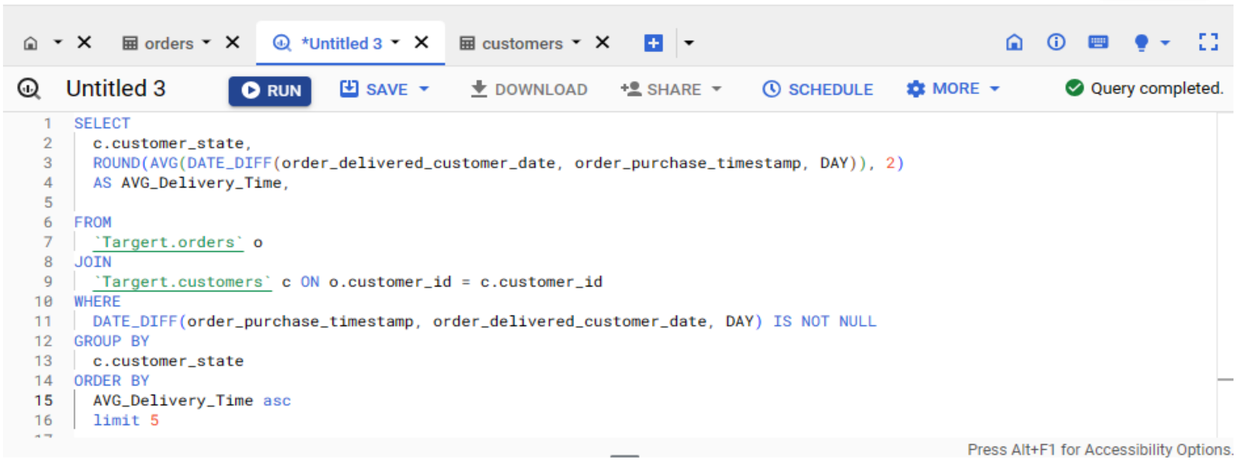


Query results

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

      ii. Lowest average delivery time.

QUERY



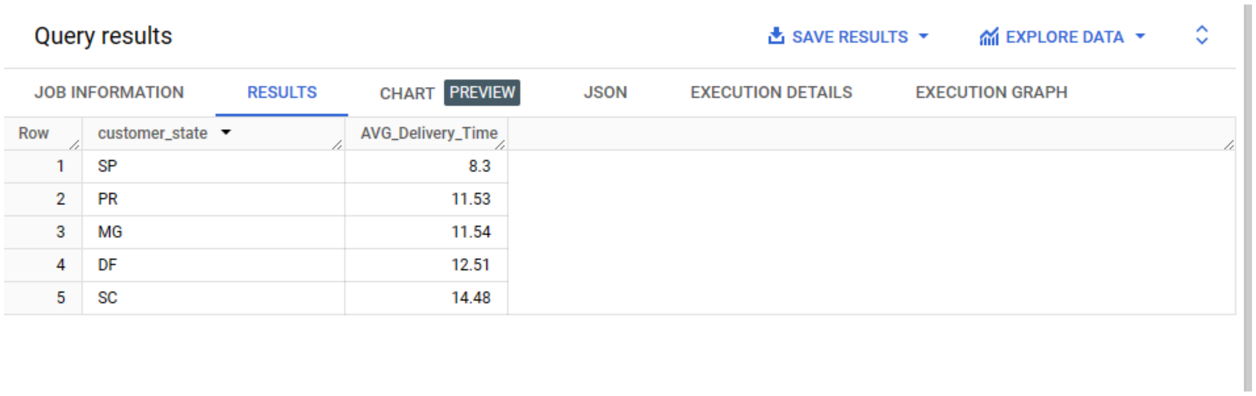
Untitled 3 — Query completed.

```
SELECT
  c.customer_state,
  ROUND(AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)), 2)
  AS AVG_Delivery_Time,

FROM
  `Targert.orders` o
JOIN
  `Targert.customers` c ON o.customer_id = c.customer_id
WHERE
  DATE_DIFF(order_purchase_timestamp, order_delivered_customer_date, DAY) IS NOT NULL
GROUP BY
  c.customer_state
ORDER BY
  AVG_Delivery_Time asc
  limit 5
```

Press Alt+F1 for Accessibility Options.

RESULT

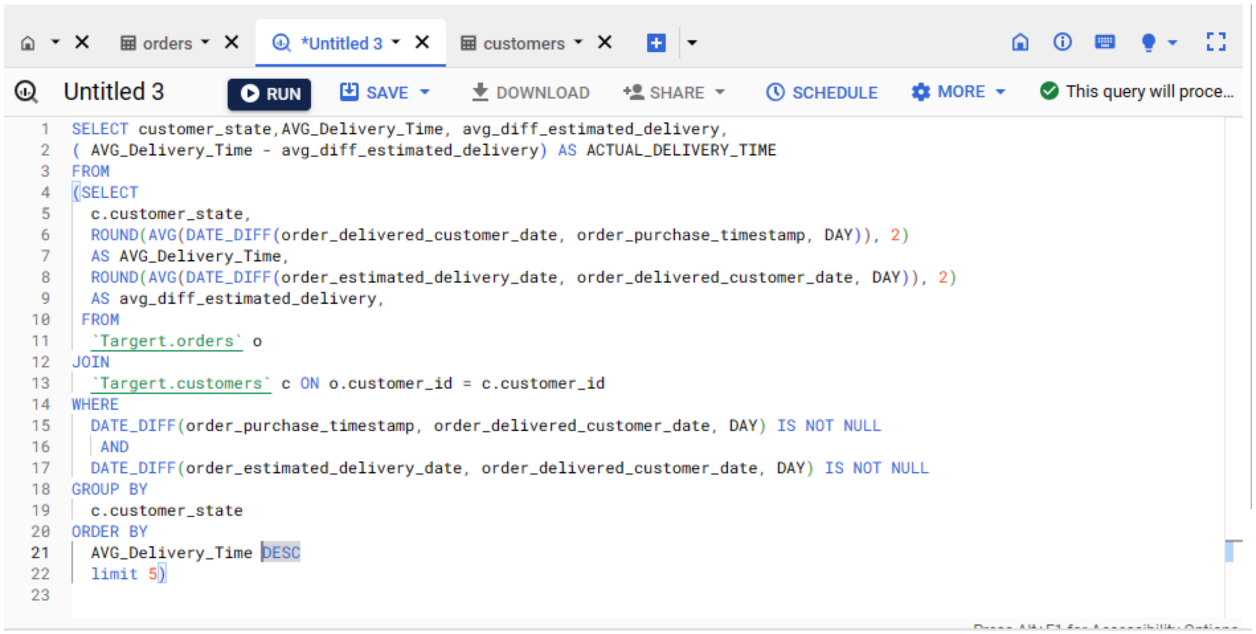


Row	customer_state	AVG_Delivery_Time
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48

5. Analysis based on sales, freight and delivery time.

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

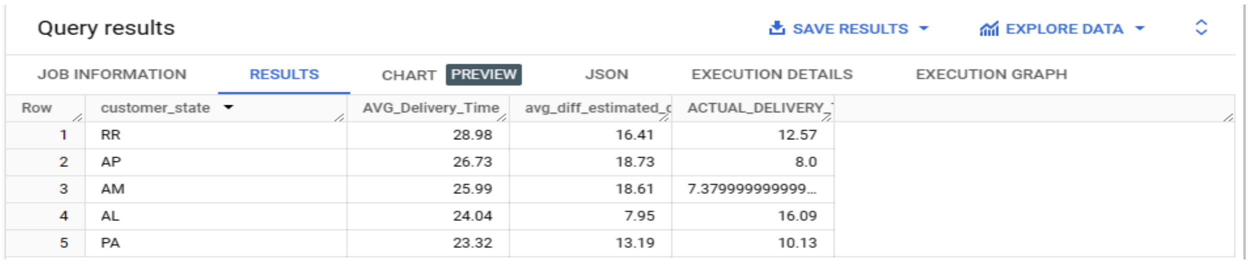
QUERY



```sql
SELECT customer_state,AVG_Delivery_Time, avg_diff_estimated_delivery,
( AVG_Delivery_Time - avg_diff_estimated_delivery) AS ACTUAL_DELIVERY_TIME
FROM
(SELECT
  c.customer_state,
  ROUND(AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)), 2)
  AS AVG_Delivery_Time,
  ROUND(AVG(DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY)), 2)
  AS avg_diff_estimated_delivery,
 FROM
  `Targert.orders` o
JOIN
  `Targert.customers` c ON o.customer_id = c.customer_id
WHERE
  DATE_DIFF(order_purchase_timestamp, order_delivered_customer_date, DAY) IS NOT NULL
   AND
  DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY) IS NOT NULL
GROUP BY
  c.customer_state
ORDER BY
  AVG_Delivery_Time DESC
  limit 5)

```

RESULT



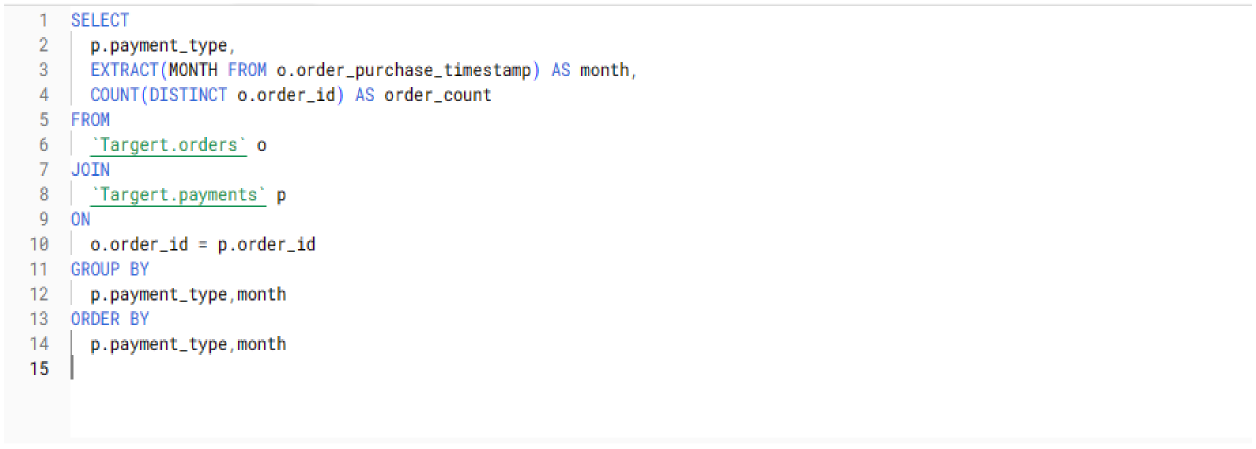
Row	customer_state	AVG_Delivery_Time	avg_diff_estimated_c	ACTUAL_DELIVERY_
1	RR	28.98	16.41	12.57
2	AP	26.73	18.73	8.0
3	AM	25.99	18.61	7.379999999999...
4	AL	24.04	7.95	16.09
5	PA	23.32	13.19	10.13

Observation:- RR state has most fastest rate of delivery in Brazil

6. Analysis based on the payments:

a. Find the month-on-month no. of orders placed using different payment types.

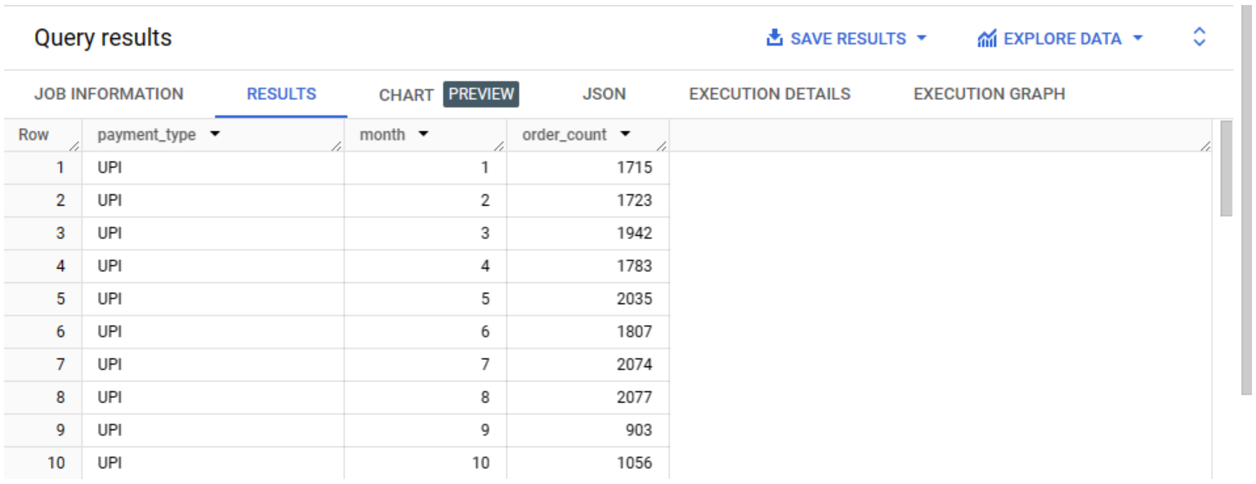
QUERY



```sql
SELECT
  p.payment_type,
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
  COUNT(DISTINCT o.order_id) AS order_count
FROM
  `Targert.orders` o
JOIN
  `Targert.payments` p
ON
  o.order_id = p.order_id
GROUP BY
  p.payment_type,month
ORDER BY
  p.payment_type,month

```

Result

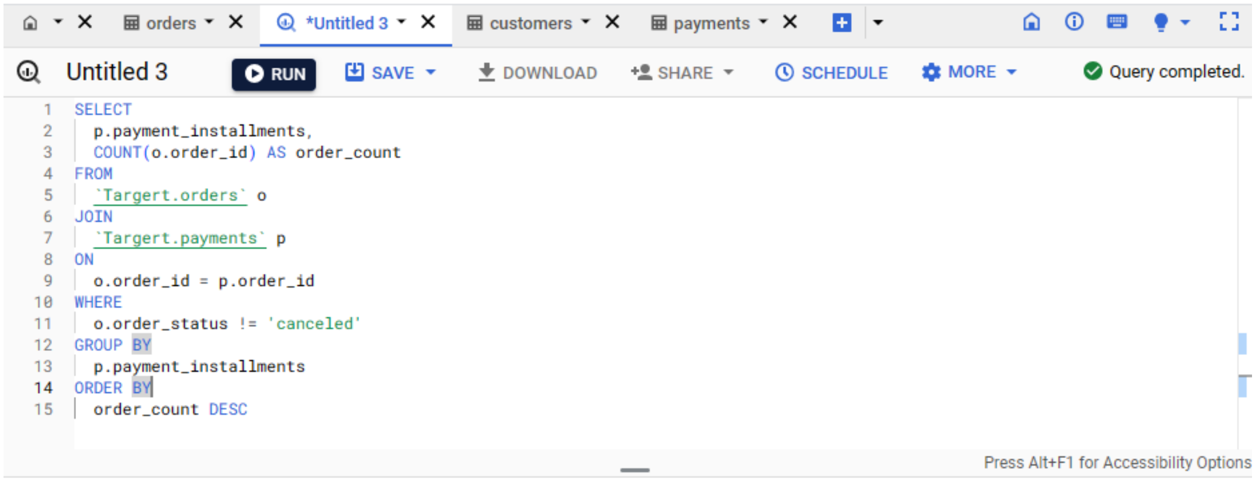


Row	payment_type	month	order_count
1	UPI	1	1715
2	UPI	2	1723
3	UPI	3	1942
4	UPI	4	1783
5	UPI	5	2035
6	UPI	6	1807
7	UPI	7	2074
8	UPI	8	2077
9	UPI	9	903
10	UPI	10	1056

6. Analysis based on the payments:

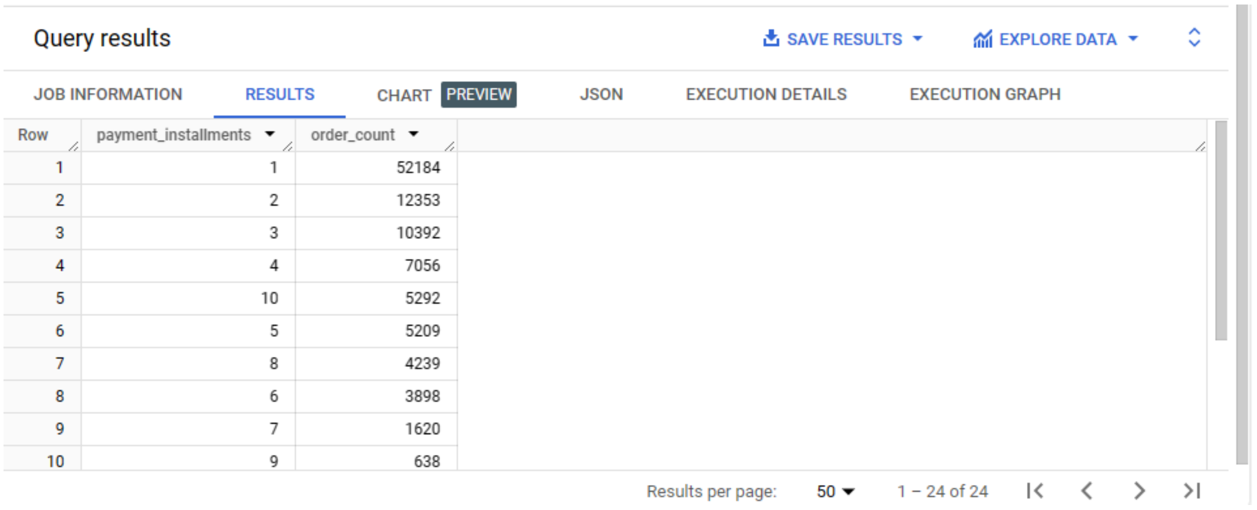
b. Find the no. of orders placed on the basis of the payment installments that have been paid.

QUERY



```sql
SELECT
  p.payment_installments,
  COUNT(o.order_id) AS order_count
FROM
  `Targert.orders` o
JOIN
  `Targert.payments` p
ON
  o.order_id = p.order_id
WHERE
  o.order_status != 'canceled'
GROUP BY
  p.payment_installments
ORDER BY
  order_count DESC
```

RESULT



Row	payment_installments	order_count
1	1	52184
2	2	12353
3	3	10392
4	4	7056
5	10	5292
6	5	5209
7	8	4239
8	6	3898
9	7	1620
10	9	638