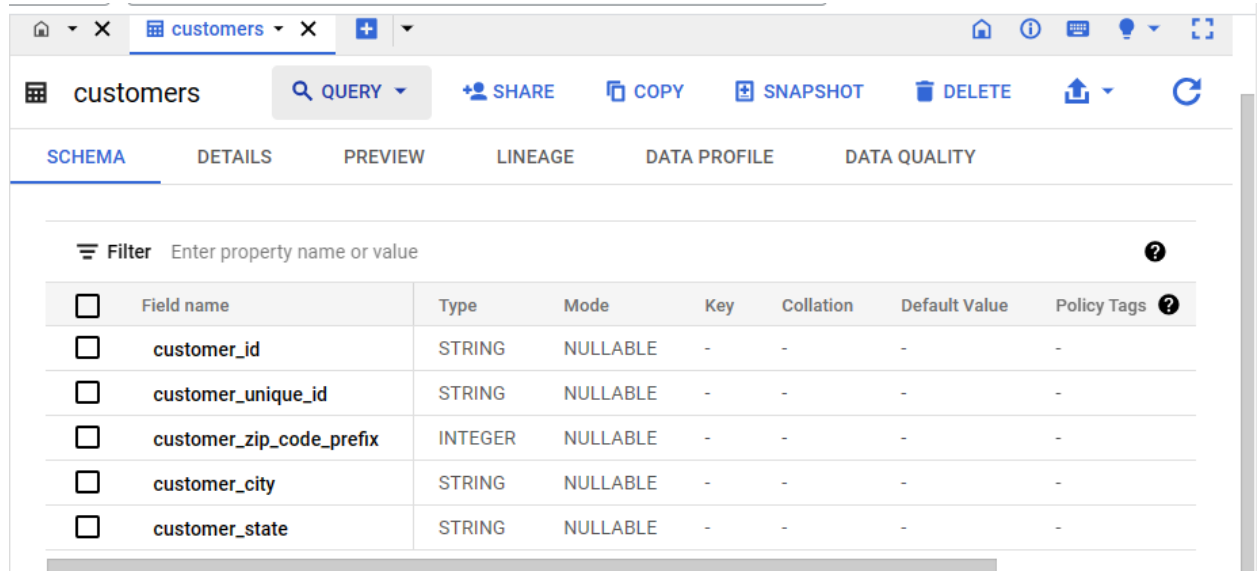


TARGET BUSINESS CASE

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.



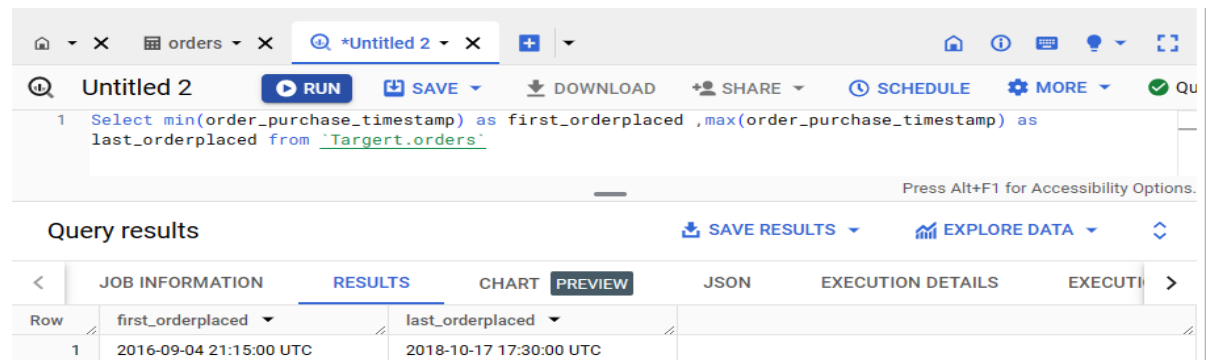
The screenshot shows a database interface with a tab for the 'customers' table. Below the tab, there are buttons for 'QUERY', 'SHARE', 'COPY', 'SNAPSHOT', 'DELETE', and a refresh icon. A navigation bar includes 'SCHEMA', 'DETAILS', 'PREVIEW', 'LINEAGE', 'DATA PROFILE', and 'DATA QUALITY'. The 'SCHEMA' tab is active, displaying a table with columns: Field name, Type, Mode, Key, Collation, Default Value, and Policy Tags. The table lists five fields: customer_id (STRING, NULLABLE), customer_unique_id (STRING, NULLABLE), customer_zip_code_prefix (INTEGER, NULLABLE), customer_city (STRING, NULLABLE), and customer_state (STRING, NULLABLE).

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.



The screenshot shows a database interface with a query editor. The query is: `Select min(order_purchase_timestamp) as first_orderplaced ,max(order_purchase_timestamp) as last_orderplaced from `Targert.orders``. The query is executed, and the results are displayed in a table. The results show the first order placed on 2016-09-04 21:15:00 UTC and the last order placed on 2018-10-17 17:30:00 UTC.

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

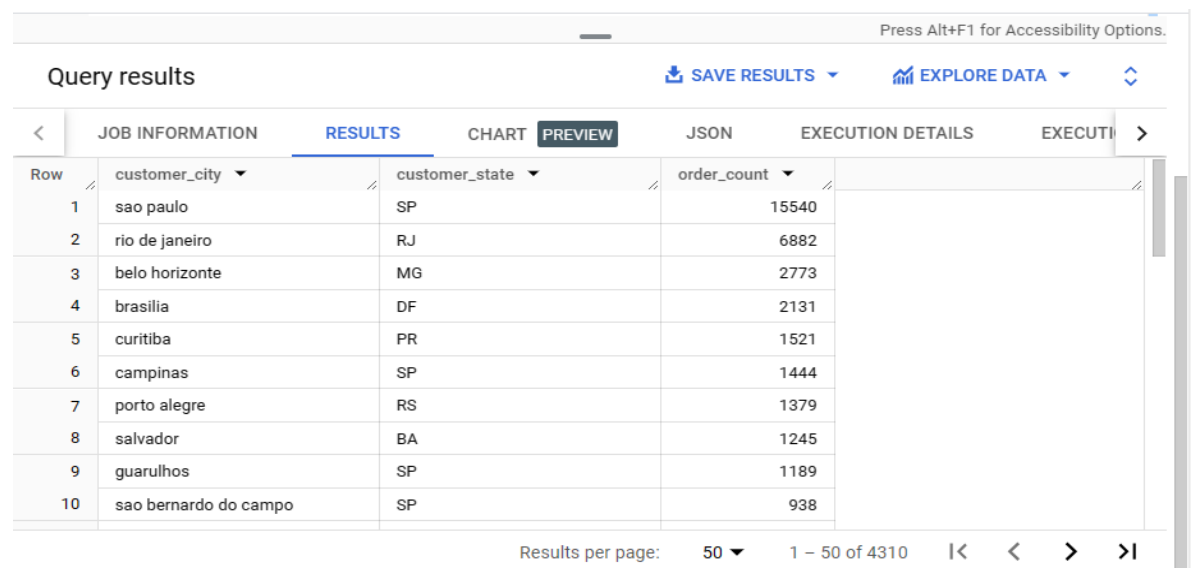
1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset.

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



```
1 SELECT DISTINCT c.customer_city,c.customer_state,COUNT(o.customer_id) order_count
2 FROM
3 Target.orders o
4 JOIN
5 | Target.customers c
6 ON
7 | o.customer_id = c.customer_id
8 GROUP BY
9 | c.customer_city,
10 | c.customer_state
11 ORDER BY
12 | order_count DESC
```

Press Alt+F1 for Accessibility Options.



Query results

SAVE RESULTS EXPLORE DATA

	JOB INFORMATION	RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION
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				

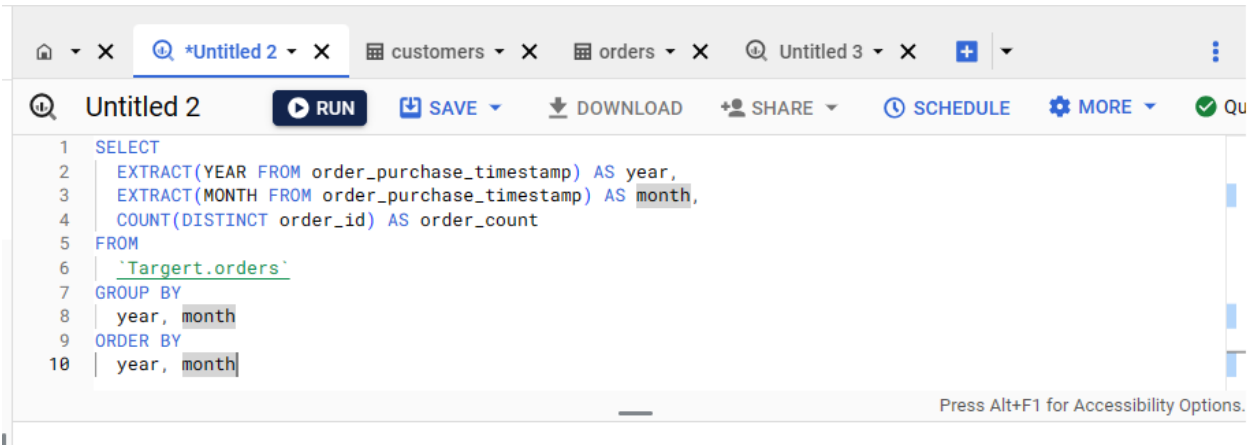
Results per page: 50 1 - 50 of 4310

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?

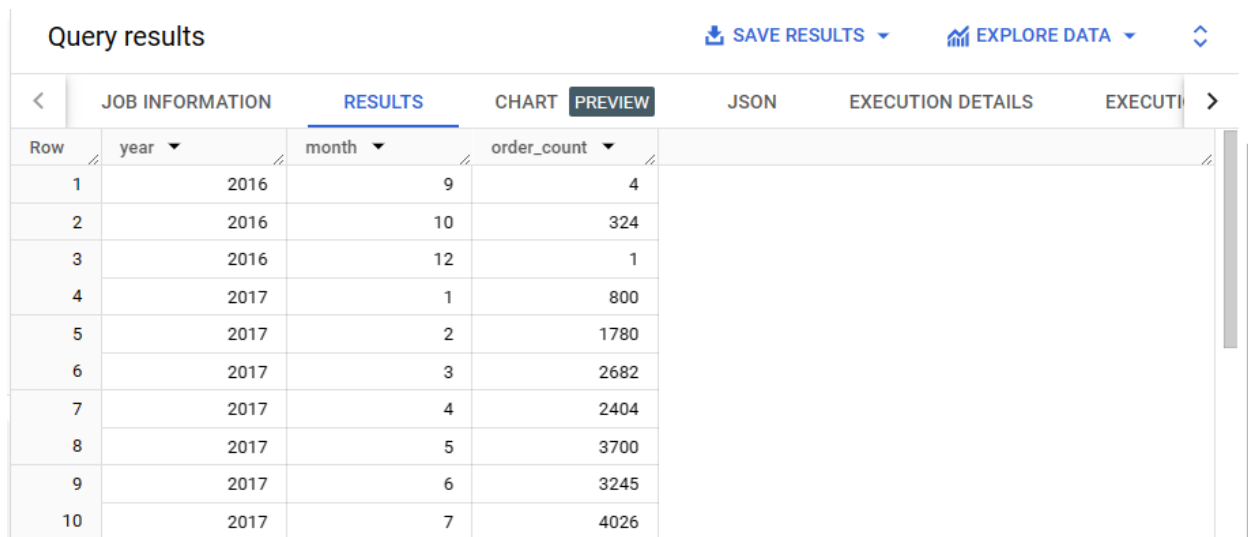
Query



```
1 SELECT
2   EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
3   EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
4   COUNT(DISTINCT order_id) AS order_count
5 FROM
6   `Targert.orders`
7 GROUP BY
8   year, month
9 ORDER BY
10  year, month
```

Press Alt+F1 for Accessibility Options.

Result



Query results

[SAVE RESULTS](#) [EXPLORE DATA](#)

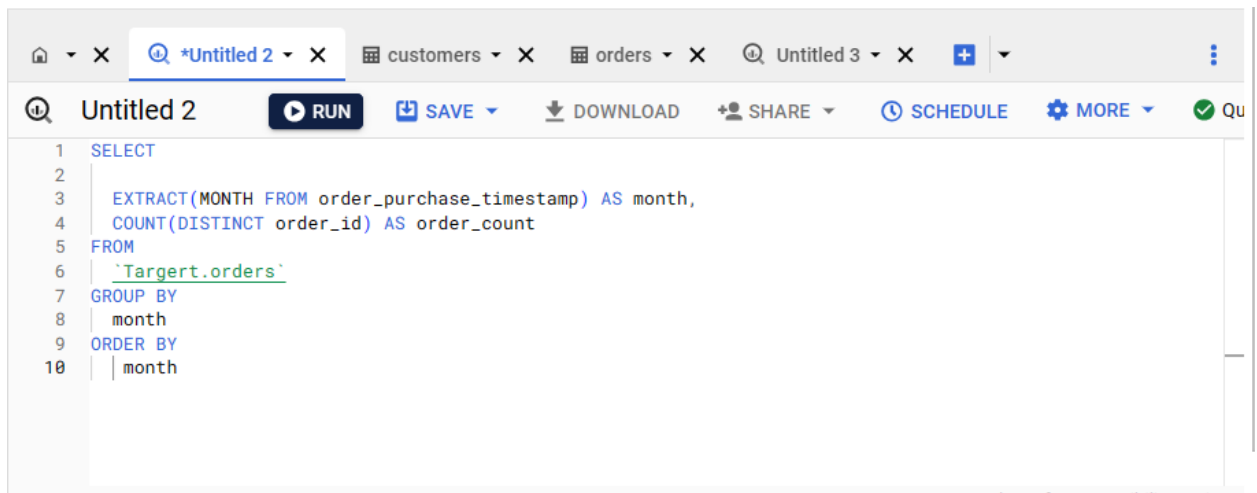
	JOB INFORMATION	RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION
Row	year	month	order_count				
1	2016	9	4				
2	2016	10	324				
3	2016	12	1				
4	2017	1	800				
5	2017	2	1780				
6	2017	3	2682				
7	2017	4	2404				
8	2017	5	3700				
9	2017	6	3245				
10	2017	7	4026				

Observation: It was observed 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



```
1 SELECT
2
3   EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
4   COUNT(DISTINCT order_id) AS order_count
5 FROM
6   `Targert.orders`
7 GROUP BY
8   month
9 ORDER BY
10  month
```

Result

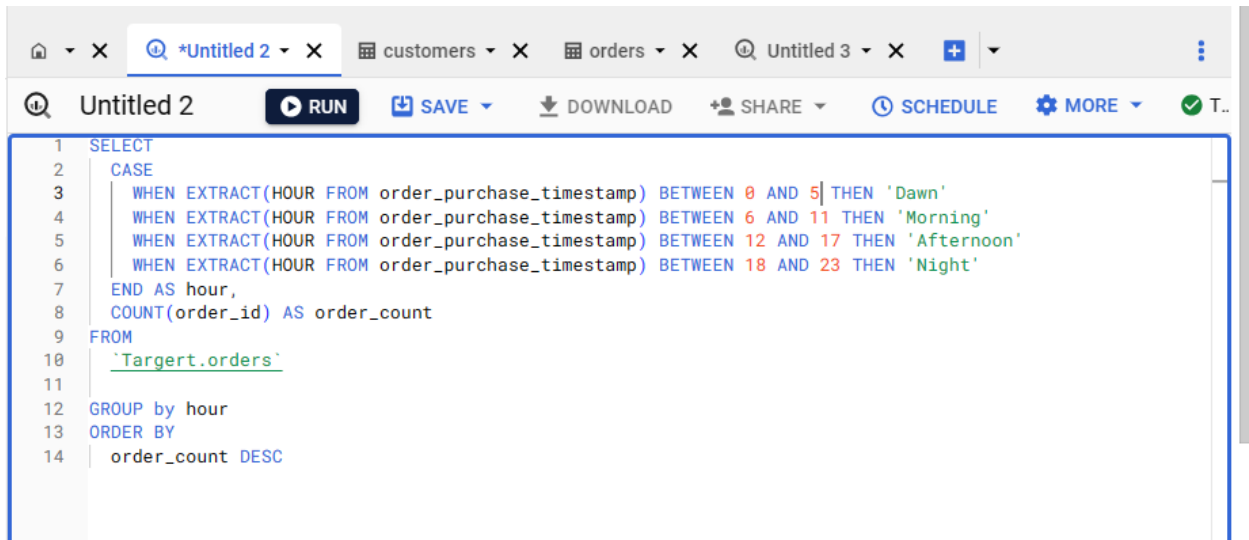
Row	month	order_count
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318
8	8	10843
9	9	4305
10	10	4959

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 their orders? (Dawn, Morning, Afternoon or Night)

Query



```
1 SELECT
2   CASE
3     WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 5 THEN 'Dawn'
4     WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 6 AND 11 THEN 'Morning'
5     WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 12 AND 17 THEN 'Afternoon'
6     WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 18 AND 23 THEN 'Night'
7   END AS hour,
8   COUNT(order_id) AS order_count
9 FROM
10  'Targert.orders'
11
12 GROUP BY hour
13 ORDER BY
14  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

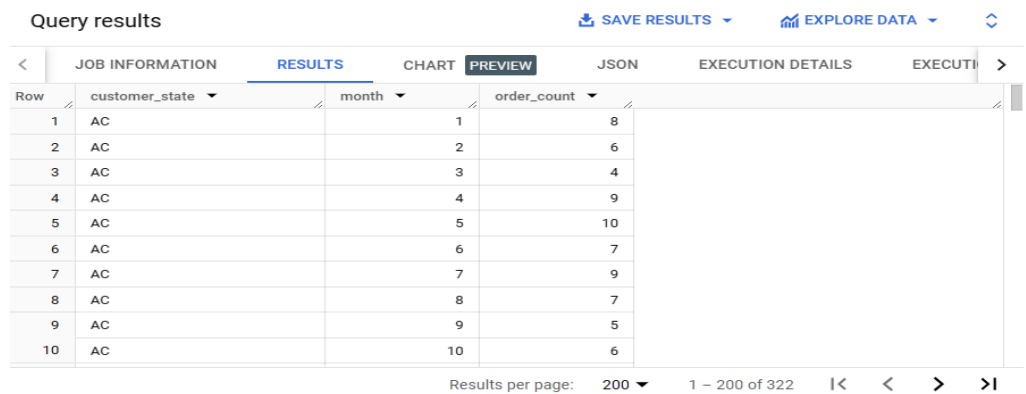


The screenshot shows a SQL query editor with a query titled "Untitled". The query is as follows:

```
1 SELECT
2   c.customer_state,
3   EXTRACT(month FROM o.order_purchase_timestamp) AS month,
4   COUNT(o.order_purchase_timestamp) AS order_count
5 FROM
6   `Targert.orders` o
7 JOIN
8   `Targert.customers` c
9 ON
10  o.customer_id = c.customer_id
11 GROUP BY
12  c.customer_state, month
13 ORDER BY
14  c.customer_state, month
```

Below the query editor, there is a "Query results" section with tabs for "JOB INFORMATION", "RESULTS", "CHART", "PREVIEW", "JSON", "EXECUTION DETAILS", and "EXECUTION". The "RESULTS" tab is selected.

Result



The screenshot shows the "Query results" section with the "RESULTS" tab selected. The table displays the following data:

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

At the bottom of the table, there is a "Results per page:" dropdown set to "200" and a "1 - 200 of 322" indicator.

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](#)

DISMISS

UPGRADE

Explorer

+ ADD

IK

Type to search

?

Viewing resources.

SHOW STARRED ONLY

business-case-scanner

☆

!

Q *Untitled

X

orders

X

customers

X

+

▼

Q

Untitled

RUN

⋮

✓ This query will process 3.6 MB when run.

1 Select customer_state, count(customer_unique_id) as Unique_customers

2 from `Target.customers`

3 group by customer_state

4 order by Unique_customers Desc

Desc

Press Alt+F1 for Accessibility Options.

Result

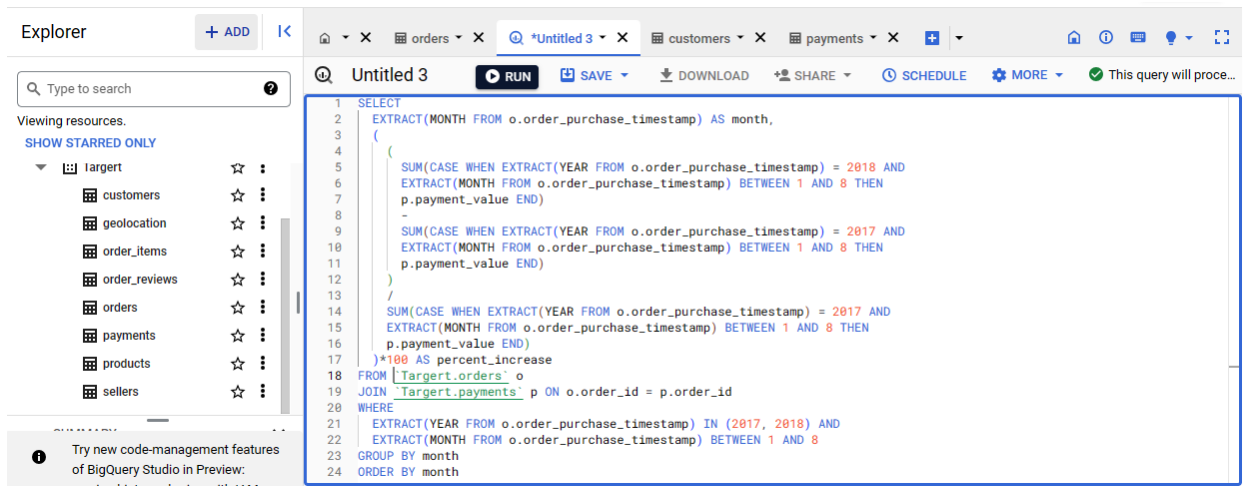
< JOB INFORMATION RESULTS CHART PREVIEW JSON EXECUTION DETAILS EXECUTING >		
Row	customer_state ▼	Unique_customers
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	PA	3380

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.

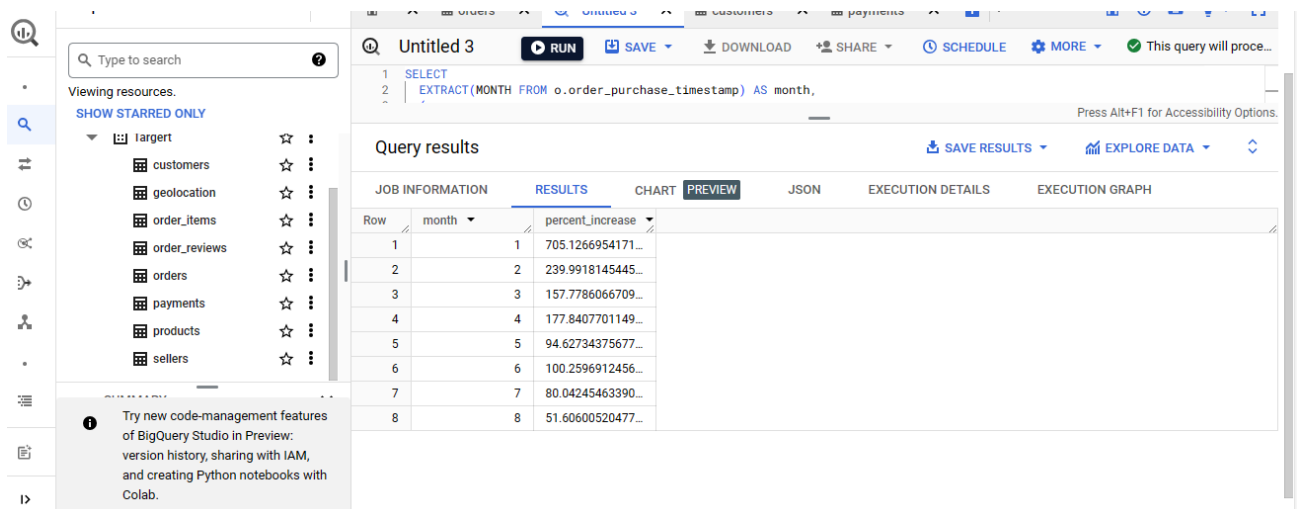
- Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

QUERY



```
1 SELECT
2   EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
3   (
4     SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2018 AND
5       EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8 THEN
6         p.payment_value END)
7     -
8     SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017 AND
9       EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8 THEN
10        p.payment_value END)
11   ) /
12   SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017 AND
13     EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8 THEN
14       p.payment_value END)
15   ) * 100 AS percent_increase
16 FROM `target.orders` o
17 JOIN `target.payments` p ON o.order_id = p.order_id
18 WHERE
19   EXTRACT(YEAR FROM o.order_purchase_timestamp) IN (2017, 2018) AND
20   EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
21 GROUP BY month
22 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 %

- Calculate the Total & Average value of order price for each state.
- Calculate the Total & Average value of order freight for each state.

Bussiness case scaler

Search (/) for resources, docs, products, and...

Search

Untitled

RUN

This query will process 15.42 MB when run.

```

1 select
2   c.customer_state,
3   ROUND(AVG(i.price), 2) as Avg_price,
4   ROUND(SUM(i.price), 2) as total_price,
5   ROUND(AVG(i.freight_value), 2) as Avg_value,
6   ROUND(SUM(i.freight_value), 2) as total_freight_value
7 from
8   `Targert.orders` o
9 join
10  `Targert.order_items` as i
11  on o.order_id = i.order_id
12 join
13  `Targert.customers` c ON o.customer_id = c.customer_id
14 group by
15   c.customer_state
16   order by
17   c.customer_state

```

Query results

[SAVE RESULTS](#)
[EXPLORE DATA](#)

	JOB INFORMATION	RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION LOG
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.

- 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

```
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

Query results

[SAVE RESULTS](#) [EXPLORE DATA](#)

	JOB INFORMATION	RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION	
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

The screenshot shows a SQL query editor with a query to find the top 5 states with the highest average freight value. The query is as follows:

```
1 select
2   c.customer_state,
3   ROUND(AVG(i.freight_value), 2) AS Highest_five_freight_value
4 FROM
5   `Targert.orders` o
6 JOIN
7   `Targert.order_items` i ON o.order_id = i.order_id
8 JOIN
```

The query results are displayed in a table with the following data:

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

The screenshot shows a SQL query editor with a query to find the top 5 states with the lowest average freight value. The query is as follows:

```
1 select
2   c.customer_state,
3   ROUND(AVG(i.freight_value), 2) AS Highest_five_freight_value_fromBottom
4 FROM
5   `Targert.orders` o
6 JOIN
7   `Targert.order_items` i ON o.order_id = i.order_id
8 JOIN
9   `Targert.customers` c ON o.customer_id = c.customer_id
10 GROUP BY c.customer_state order by Highest_five_freight_value_fromBottom asc limit 5
11
12
```

The query results are displayed in a table with the following data:

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.

```
1 SELECT
2   c.customer_state,
3   ROUND(AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)), 2)
4   AS AVG_Delivery_Time,
5
6 FROM
7   `Target.orders` o
8 JOIN
9   `Target.customers` c ON o.customer_id = c.customer_id
10 WHERE
11   DATE_DIFF(order_purchase_timestamp, order_delivered_customer_date, DAY) IS NOT NULL
12 GROUP BY
13   c.customer_state
14 ORDER BY
15   AVG_Delivery_Time desc
16   limit 5
17
```

RESULT

Query results			SAVE RESULTS	EXPLORE DATA	
JOB INFORMATION			RESULTS	CHART	PREVIEW
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

```
1 SELECT
2   c.customer_state,
3   ROUND(AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)), 2)
4   AS AVG_Delivery_Time,
5
6 FROM
7   `Target.orders` o
8 JOIN
9   `Target.customers` c ON o.customer_id = c.customer_id
10 WHERE
11   DATE_DIFF(order_purchase_timestamp, order_delivered_customer_date, DAY) IS NOT NULL
12 GROUP BY
13   c.customer_state
14 ORDER BY
15   AVG_Delivery_Time asc
16   limit 5
17
```

RESULT

Query results

SAVE RESULTSEXPLORE DATA

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
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

orders *Untitled 3 customers

Untitled 3RUNSAVEDOWNLOADSHARESCHEDULEMOREThis query will proce...

```
1 SELECT customer_state,AVG_Delivery_Time, avg_diff_estimated_delivery,
2 ( AVG_Delivery_Time - avg_diff_estimated_delivery) AS ACTUAL_DELIVERY_TIME
3 FROM
4 (SELECT
5   c.customer_state,
6   ROUND(AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)), 2)
7   AS AVG_Delivery_Time,
8   ROUND(AVG(DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY)), 2)
9   AS avg_diff_estimated_delivery,
10  FROM
11   `Targert.orders` o
12  JOIN
13   `Targert.customers` c ON o.customer_id = c.customer_id
14  WHERE
15   DATE_DIFF(order_purchase_timestamp, order_delivered_customer_date, DAY) IS NOT NULL
16   AND
17   DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY) IS NOT NULL
18  GROUP BY
19   c.customer_state
20  ORDER BY
21   AVG_Delivery_Time DESC
22  limit 5)
23
```

RESULT

Query results

SAVE RESULTSEXPLORE DATA

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	customer_state	AVG_Delivery_Time	avg_diff_estimated_delivery	ACTUAL_DELIVERY			
1	RR	28.98	16.41	12.57			
2	AP	26.73	18.73	8.0			
3	AM	25.99	18.61	7.379999999999999...			
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:

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

QUERY

```
1 SELECT
2   p.payment_type,
3   EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
4   COUNT(DISTINCT o.order_id) AS order_count
5 FROM
6   `Targert.orders` o
7 JOIN
8   `Targert.payments` p
9 ON
10  o.order_id = p.order_id
11 GROUP BY
12  p.payment_type, month
13 ORDER BY
14  p.payment_type, month
15
```

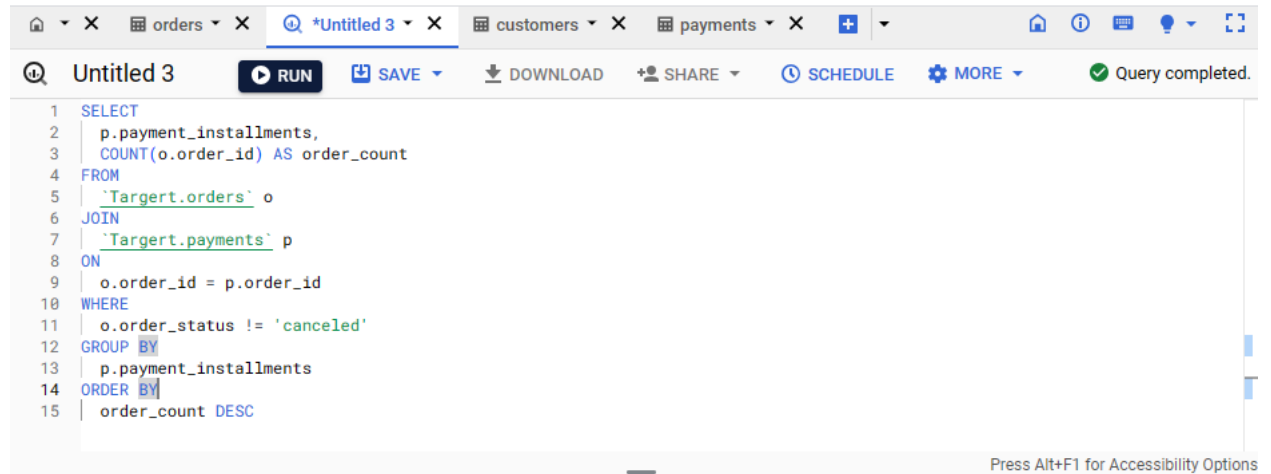
Result

Query results				SAVE RESULTS	EXPLORE DATA	
JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS
				EXECUTION GRAPH		
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

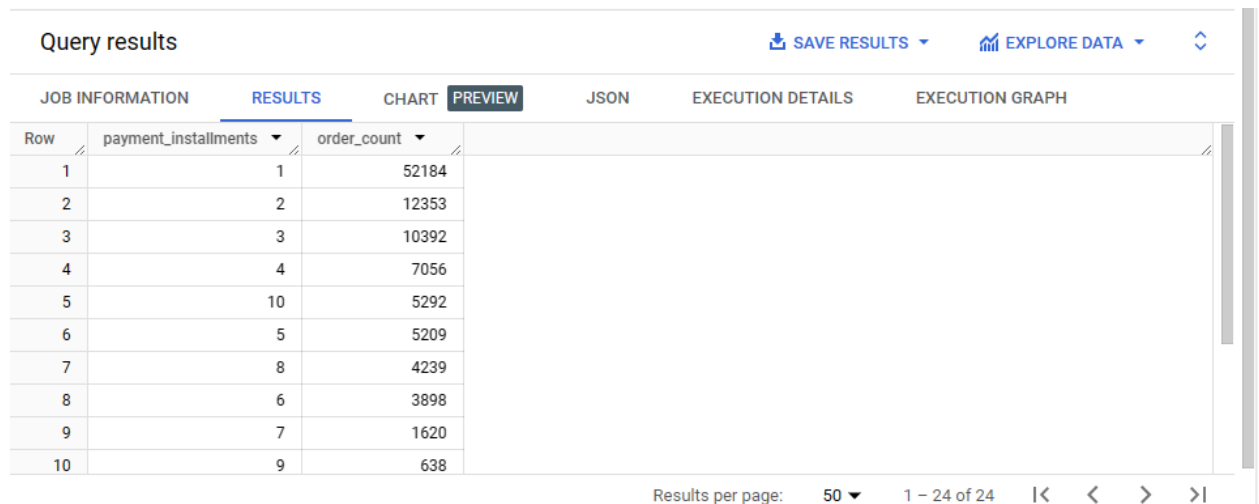


The screenshot shows a SQL query editor with a toolbar at the top containing icons for home, search, and various database actions. The query is as follows:

```
1 SELECT
2   p.payment_installments,
3   COUNT(o.order_id) AS order_count
4 FROM
5   `Target.orders` o
6 JOIN
7   `Target.payments` p
8 ON
9   o.order_id = p.order_id
10 WHERE
11   o.order_status != 'canceled'
12 GROUP BY
13   p.payment_installments
14 ORDER BY
15   order_count DESC
```

The editor interface includes a 'RUN' button, 'SAVE', 'DOWNLOAD', 'SHARE', 'SCHEDULE', and 'MORE' options. A status bar at the bottom right indicates 'Query completed.' and 'Press Alt+F1 for Accessibility Options'.

RESULT



The screenshot displays the 'Query results' section of a database interface. It includes tabs for 'JOB INFORMATION', 'RESULTS' (selected), 'CHART', 'PREVIEW', 'JSON', 'EXECUTION DETAILS', and 'EXECUTION GRAPH'. The 'RESULTS' tab shows a table with two columns: 'payment_installments' and 'order_count'. The table contains 10 rows of data. At the bottom right, there is a 'Results per page' dropdown set to 50, and a pagination indicator '1 - 24 of 24'.

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