

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:

Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS CHART PREVIEW JSON EXECUTION DETAILS EXECUTION GRAPH

Row	customer_id	customer_unique_id	customer_zip_code_prefix	customer_city	customer_state
1	0735e7e4298a2ebbb46649346...	fcb003b1bdc0df64b4d065d9b...	59650	acu	RN
2	903b3d86e3990db01619a4ebe...	46824822b15da44e983b021d...	59650	acu	RN
3	38c97666e962d4fea7fd6a83e...	b6108acc674ae5c99e29adc10...	59650	acu	RN
4	77c2f46cf580f4874c9a5751c2...	402cce5c0509000eed9e77fec...	63430	ico	CE
5	4d3ef4cfff8ad4767c199c36a...	6ba00666ab7eada5ceec279b2...	63430	ico	CE
6	3000841b86e1fbe9493b52324...	796a0b1a21f597704057184a1...	63430	ico	CE
7	3c325415ccc7e622c66dec4bc...	05d1d2d9f0161c5f397ce7fc77...	63430	ico	CE

Results per page: 50 1 – 50 of 99441

Job history REFRESH

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:

The screenshot shows a web-based SQL interface. At the top, there's a toolbar with buttons for 'RUN', 'SAVE', 'DOWNLOAD', 'SHARE', 'SCHEDULE', and 'MORE'. Below the toolbar is a text area containing the SQL query: `select min(order_purchase_timestamp) as start_time,max(order_purchase_timestamp) as End_time from Target_database.orders;`. Below the query editor, the 'Query results' section is active, displaying a table with two columns: 'start_time' and 'End_time'. The table has one row of data. At the bottom, there's a 'Job history' section with a 'REFRESH' button.

Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS CHART PREVIEW JSON EXECUTION DETAILS EXECUTION GRAPH

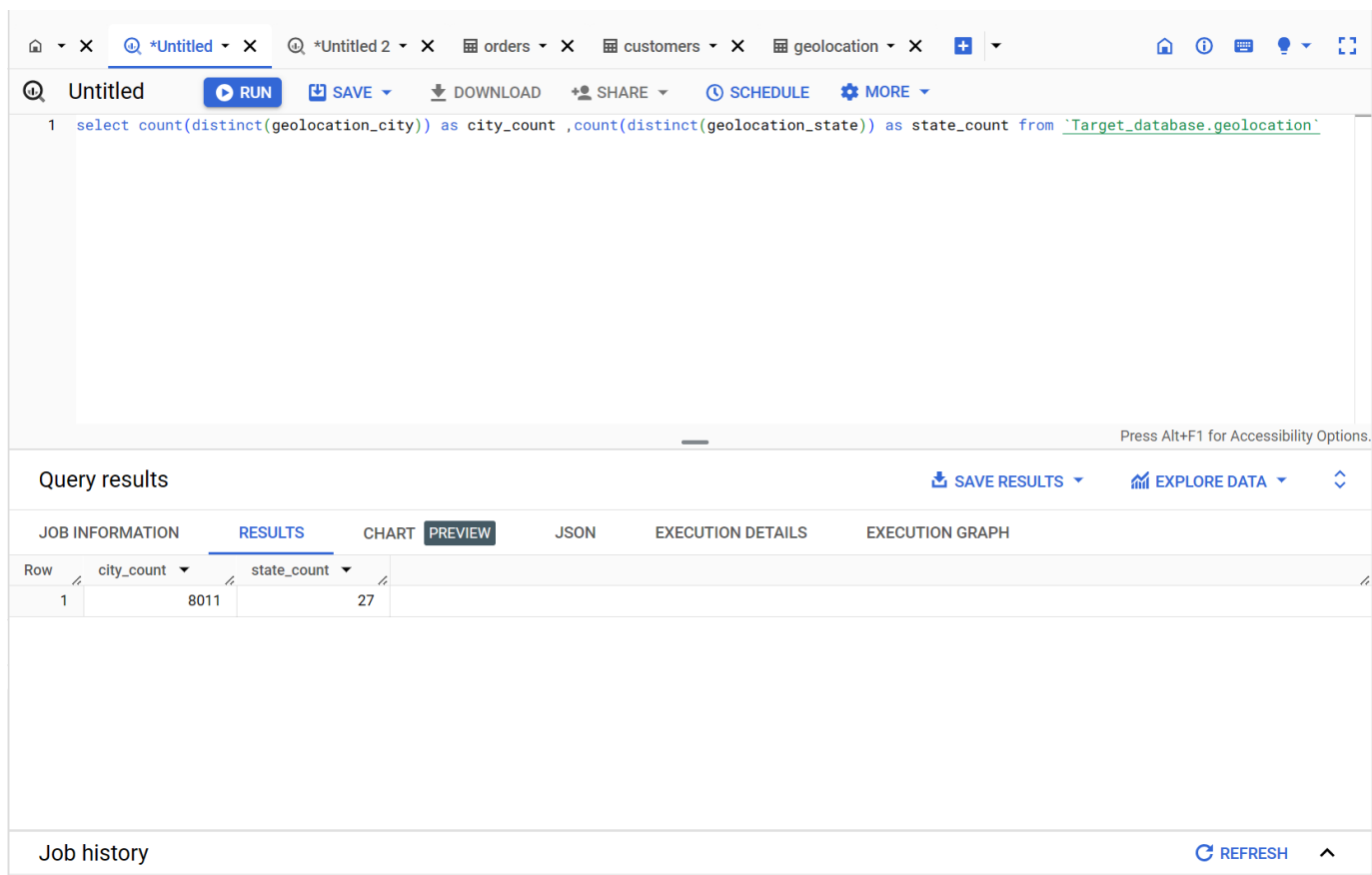
Row	start_time	End_time
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

Job history REFRESH

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:



The screenshot shows a web-based SQL interface. At the top, there's a toolbar with icons for home, info, help, and a search icon. Below the toolbar, the query editor shows the SQL query: `select count(distinct(geolocation_city)) as city_count ,count(distinct(geolocation_state)) as state_count from `Target_database.geolocation``. The query is highlighted in blue. Below the query editor, there's a section for "Query results" with tabs for "JOB INFORMATION", "RESULTS", "CHART", "PREVIEW", "JSON", "EXECUTION DETAILS", and "EXECUTION GRAPH". The "RESULTS" tab is selected, showing a table with two columns: "city_count" and "state_count". The table has one row with values 8011 and 27. At the bottom, there's a "Job history" section with a "REFRESH" button.

Row	city_count	state_count
1	8011	27

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:

Untitled 2		RUN		SAVE	DOWNLOAD	SHARE	SCHEDULE	MORE
1	SELECT							
2	EXTRACT(MONTH FROM order_purchase_timestamp) AS order_month,							
3	EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,							
4	FROM `Target_database.orders` GROUP BY order_month, order_year							
5	ORDER BY order_year, order_month							
6								
7								
8								
9								
10								
11								

Query results

JOB INFORMATION

RESULTS

CHART

PREVIEW

JSON

EXECUTION DETAILS

EXECUTION GRAPH

Row	order_month	order_year
1	9	2016
2	10	2016
3	12	2016
4	1	2017
5	2	2017
6	3	2017
7	4	2017

Results per page: 50 1 – 25 of 25

Job history

REFRESH

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:

The screenshot shows a SQL query editor interface. At the top, there are tabs for 'orders', 'order_items', and 'products'. Below the tabs, there's a toolbar with buttons for 'RUN', 'SAVE', 'DOWNLOAD', 'SHARE', 'SCHEDULE', and 'MORE'. A status message indicates 'This query will process 3.98 MB when run.' The query editor contains the following SQL code:

```
1 select EXTRACT(MONTH FROM order_purchase_timestamp) as order_month , COUNT(DISTINCT order_id) as order_count from Target_database.orders GROUP
2 BY order_month ORDER BY order_month
3
```

Below the query editor, the 'Query results' section is visible. It includes a 'SAVE RESULTS' button and an 'EXPLORE DATA' button. The results are displayed in a table with the following columns: 'Row', 'order_month', and 'order_count'. The table contains 7 rows of data, showing a clear upward trend in the number of orders over the months.

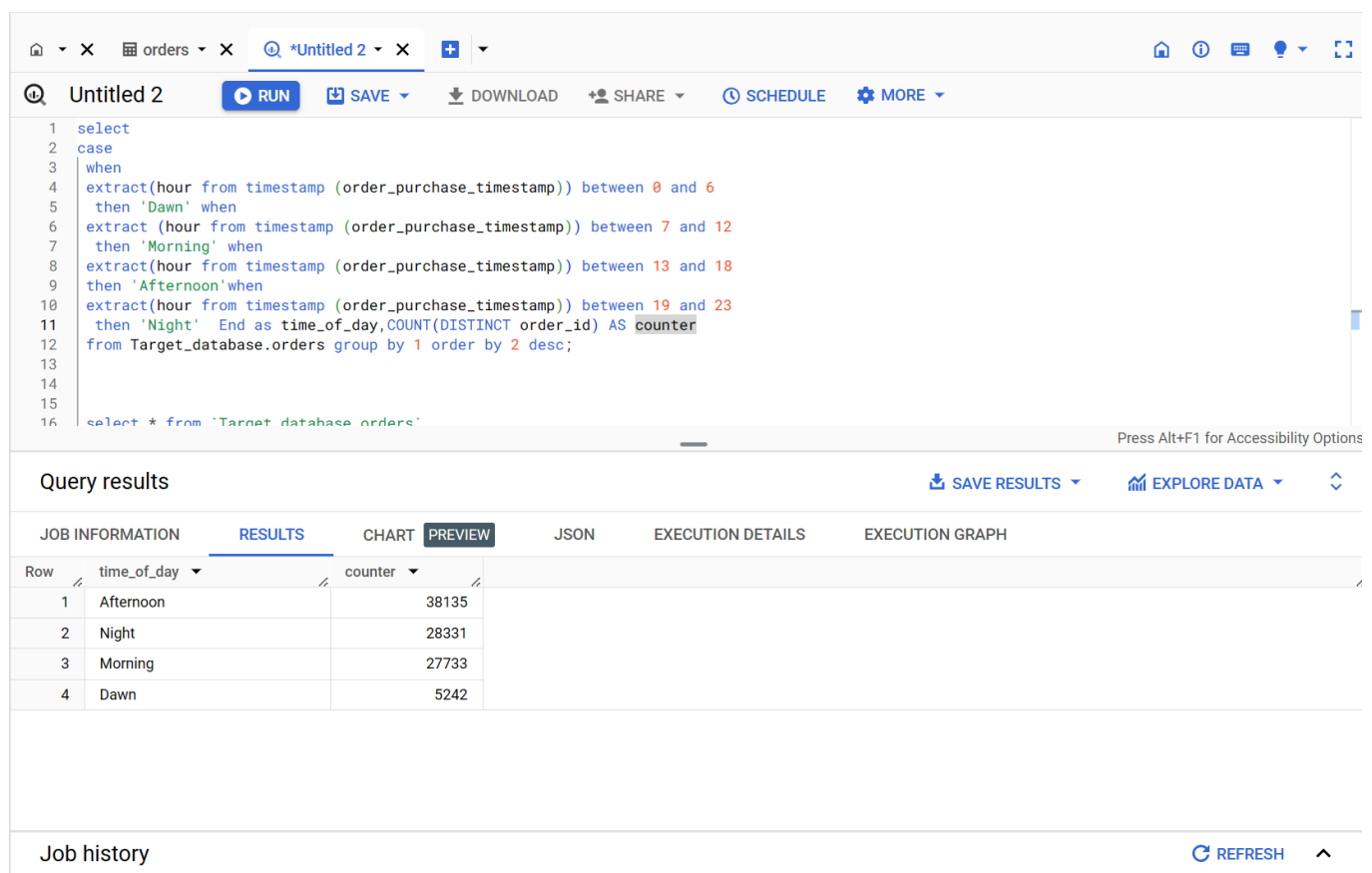
Row	order_month	order_count
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318

At the bottom of the results section, there's a 'Job history' section with a 'REFRESH' button. The results per page are set to 50, and the current view shows 1 - 12 of 12 results.

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:



Press Alt+F1 for Accessibility Options

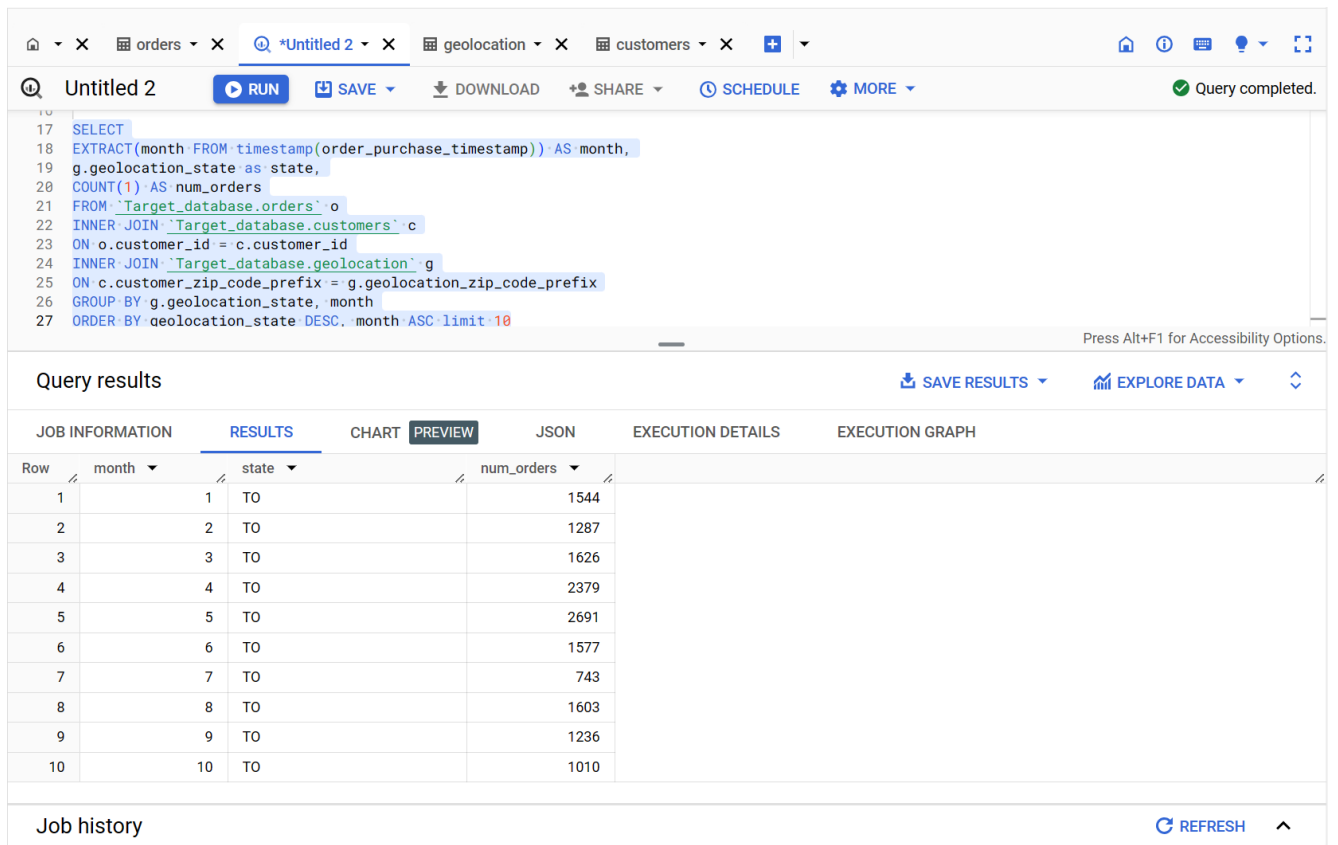
Query results		SAVE RESULTS	EXPLORE DATA
JOB INFORMATION		RESULTS	CHART
RESULTS		PREVIEW	JSON
EXECUTION DETAILS		EXECUTION GRAPH	
Row	time_of_day	counter	
1	Afternoon	38135	
2	Night	28331	
3	Morning	27733	
4	Dawn	5242	

Job history REFRESH

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:



The screenshot shows a SQL query editor interface. The query is as follows:

```
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
```

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

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

The interface also includes a 'Job history' section at the bottom with a 'REFRESH' button.

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:

Untitled 3		RUN	SAVE	DOWNLOAD	SHARE	SCHEDULE	MORE
1 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 ;							
Query results		SAVE RESULTS EXPLORE DATA					
JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	geolocation_state	order_count					
1	RR	45					
2	AP	67					
3	AC	116					
4	AM	143					
5	RO	243					
6	TO	272					
7	SE	341					
Job history		RESULTS per page: 50 1 - 27 of 27 REFRESH					

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

✕

+

▼

🏠

ℹ️

📄

💡

🗨️

🔍Untitled

▶️ RUN

💾 SAVE

⬇️ DOWNLOAD

👤 SHARE

🕒 SCHEDULE

⚙️ MORE

1 with demo as(
2 select extract(year from timestamp(o.order_purchase_timestamp)) as year,extract(month from timestamp(o.order_purchase_timestamp))as month, sum
3 (price)/count(distinct(o.order_id)) as order_count,(sum(freight_value) / COUNT(distinct o.order_id)) AS freight_value from Target_database.
4 orders as o inner join Target_database.order_items as i on o.order_id=i.order_id GROUP BY year, month)
5
6 SELECT
7 month,
8 price_2017,
9 price_2018,
10

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:

2024-01-09 13:11:21ordersorder_items*Untitledcustomers+▼

UntitledRUNSAVE▼DOWNLOADSHARE▼SCHEDULEMORE▼

```
1 with demo1 as(
2 select c.customer_state as state,
3 sum(price) as total_price,
4 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
5 Target_database.customers as c on o.customer_id=c.customer_id
6 group by state)
7 select state,total_price,(total_price/num_orders)as avgg from demo1
8 order by total_price desc limit 10;
```

Query resultsSAVE RESULTS▼EXPLORE DATA▼

JOB INFORMATIONRESULTSCHARTPREVIEWJSONEXECUTION DETAILSEXECUTION GRAPH

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...

Job historyREFRESH^

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_2018 / 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:

2024-01-10 00:00:31

RUN

SAVE QUERY

SHARE

SCHEDULE

MORE

Query completed.

```
1 with demo3 as
2 (
3   select c.customer_state as state , extract(year from timestamp(o.order_purchase_timestamp)) as year,extract(month from timestamp(o.
4     order_purchase_timestamp))as month,(sum(freight_value)/ COUNT(distinct o.order_id)) AS freight_value from Target_database.orders as o inner
5     join Target_database.order_items
6     as i
7     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 )
8 SELECT
9   month,state,
10  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_p
urchase_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:

2024-01-10 00:28:13

RUN

SAVE QUERY

SHARE

SCHEDULE

MORE

Query completed.

```
1 select order_id,date_diff(date(order_delivered_customer_date),date(order_purchase_timestamp),day) as time_to_deliver,
2 date_diff(date(order_delivered_customer_date),date(order_estimated_delivery_date), day) as diff_estimated_delivery
3 from Target_database.orders where order_status='delivered' limit 10
```

Query results

SAVE RESULTSEXPLORE DATA

JOB INFORMATIONRESULTSCHARTPREVIEWJSONEXECUTION DETAILSEXECUTION GRAPH

Row	order_id	time_to_deliver	diff_estimated_delivery
1	c158e9806f85a33877bdfd4f60...	24	-10
2	b60b53ad0bb7dacaf2989fe2...	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

Job history

REFRESH

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(ord  
er_purchase_timestamp),day)) as time_to_deliver,  
avg(date_diff(date(order_delivered_customer_date),date(order_estimated_deliv  
ery_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:

10:31 x 2024-01-09 13:11:21 x *Untitled 8 x customers x order_items x					
Untitled 8 RUN SAVE DOWNLOAD SHARE SCHEDULE MORE					
<pre>1 2 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, 3 avg(date_diff(date(order_delivered_customer_date),date(order_estimated_delivery_date), day)) as diff_estimated_delivery 4 from Target_database.orders as o inner join Target_database.customers as c on o.customer_id = c.customer_id inner join 5 Target_database.order_items as i on i.order_id = o.order_id WHERE order_status = 'delivered' 6 GROUP BY state 7 ORDER BY avg_freight_value ASC 8 LIMIT 5</pre>					
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_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...	
Job history REFRESH					

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:

🏠 X 🔍 Untitled X 🗺️ geolocation X 🔍 top 5 states with the high... ime X 🔍 2024-01-10 00:00:31 X +

🔍 top 5 states with the high...ime

RUN

SAVE QUERY

SHARE

SCHEDULE

MORE

🟢 This query will process 24.44 MB

1 (with demo as (
2 select g.geolocation_state as state, SUM(TIMESTAMP_DIFF(TIMESTAMP(order_estimated_delivery_date), TIMESTAMP(order_purchase_timestamp),
3 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.
4 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 RESULTSEXPLORE DATA

JOB INFORMATIONRESULTSCHARTPREVIEWJSONEXECUTION DETAILSEXECUTION 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_delivery_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:

The screenshot shows a SQL query editor with the following query:

```
1 select g.geolocation_state as state,AVG(TIMESTAMP_DIFF(TIMESTAMP(order_estimated_delivery_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
```

Below the query editor, the 'Query results' section displays the following table:

Row	state	diff_estimated_delivery
1	SP	19.04831245981...
2	PR	23.96693288404...
3	DF	24.08380053436...
4	MG	24.11110763087...
5	ES	25.00719754878...
6	SC	25.34949075640...
7	MS	25.75134393039...
8	RJ	26.08429630862...
9	GO	26.87842279050...
10	TO	27.90429275158...

The interface also includes a 'Job history' section at the bottom with a 'REFRESH' button.

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:

top 5 states with the high... ime

e top 5 states where the o...ast

order_items

*Untitled 2

payments

>

+

⋮

Untitled 2

RUN

SAVE

DOWNLOAD

SHARE

SCHEDULE

MORE

1with demo as(
2select 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
3|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)
4|SELECT year,month, total_count, payment_type
5FROM demo
6order by payment_type,total_count, year asc, month asc ;

Press Alt+F1 for Accessibility Options.

Query results

SAVE RESULTS

EXPLORE DATA

JOB INFORMATION

RESULTS

CHART

PREVIEW

JSON

EXECUTION DETAILS

EXECUTION GRAPH

Row	year	month	total_count	payment_type	
1	2016	10	71	UPI	
2	2017	1	232	UPI	
3	2017	2	438	UPI	
4	2017	4	567	UPI	
5	2017	3	667	UPI	
6	2017	6	810	UPI	
7	2017	5	870	UPI	
8	2017	7	970	UPI	
9	2017	9	1041	UPI	
10	2017	8	1067	UPI	
11	2017	10	1169	UPI	

Results per page:501 – 50 of 87

<<<>>>

Job history

REFRESH

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:

e top 5 states where the o...ast			
order_items			
month on month no. of or... pes			
payments			
*Untitled 2			
Query completed.			
Untitled 2			
1 select payment_installments,count(order_id) as total_count, from Target_database.payments			
2 GROUP BY			
3 payment_installments limit 10			
Press Alt+F1 for Accessibility Options.			
Query results			
SAVE RESULTS			
EXPLORE DATA			
JOB INFORMATION			
RESULTS			
CHART			
PREVIEW			
JSON			
EXECUTION DETAILS			
EXECUTION GRAPH			
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	
Job history			
REFRESH			