

Project Introduction:

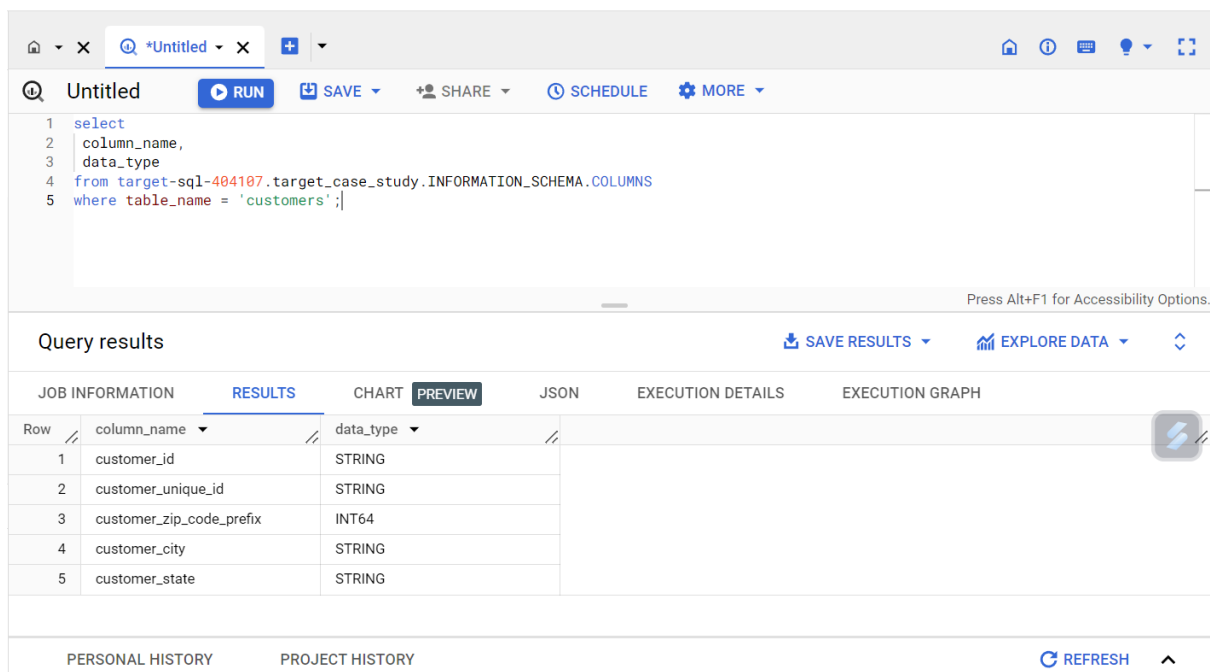
Target is a globally renowned brand and a prominent retailer in the United States. Target makes itself a preferred shopping destination by offering outstanding value, inspiration, innovation and an exceptional guest experience that no other retailer can deliver.

This project analyzes Target's operations in Brazil, analyzing 100,000 orders placed between 2016 and 2018. The dataset provides a comprehensive view of various dimensions, including order status, pricing, payment and freight performance, customer locations, product attributes, and customer reviews. Through this analysis, valuable insights can be gained into Target's business operations in Brazil, including order processing, pricing strategies, payment and shipping efficiency, customer demographics, product characteristics, and customer satisfaction levels. These insights will help understand and improve Target's performance in the Brazilian market.

Q. 1 a)

Data type of all columns in the "customers" table.

```
select
  column_name,
  data_type
from target-sql-
404107.target_case_study.INFORMATION_SCHEMA.COLUMNS
where table_name = 'customers';
```



Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS CHART PREVIEW JSON EXECUTION DETAILS EXECUTION GRAPH

Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

PERSONAL HISTORY PROJECT HISTORY REFRESH

❖ Insights

All columns have same data type except customer_zip_code_prefix.

Q.1

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

`select`

```
min(order_purchase_timestamp) as first_order,  
max(order_purchase_timestamp) as last_order  
from `target_case_study.orders`
```

The screenshot shows a SQL query editor interface. The query is: `select min(order_purchase_timestamp) as first_order, max(order_purchase_timestamp) as last_order from `target_case_study.orders``. Below the editor, the 'Query results' section is active, showing a table with two columns: 'first_order' and 'last_order'. The first row shows the minimum order timestamp as '2016-09-04 21:15:19 UTC' and the maximum as '2018-10-17 17:30:18 UTC'.

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

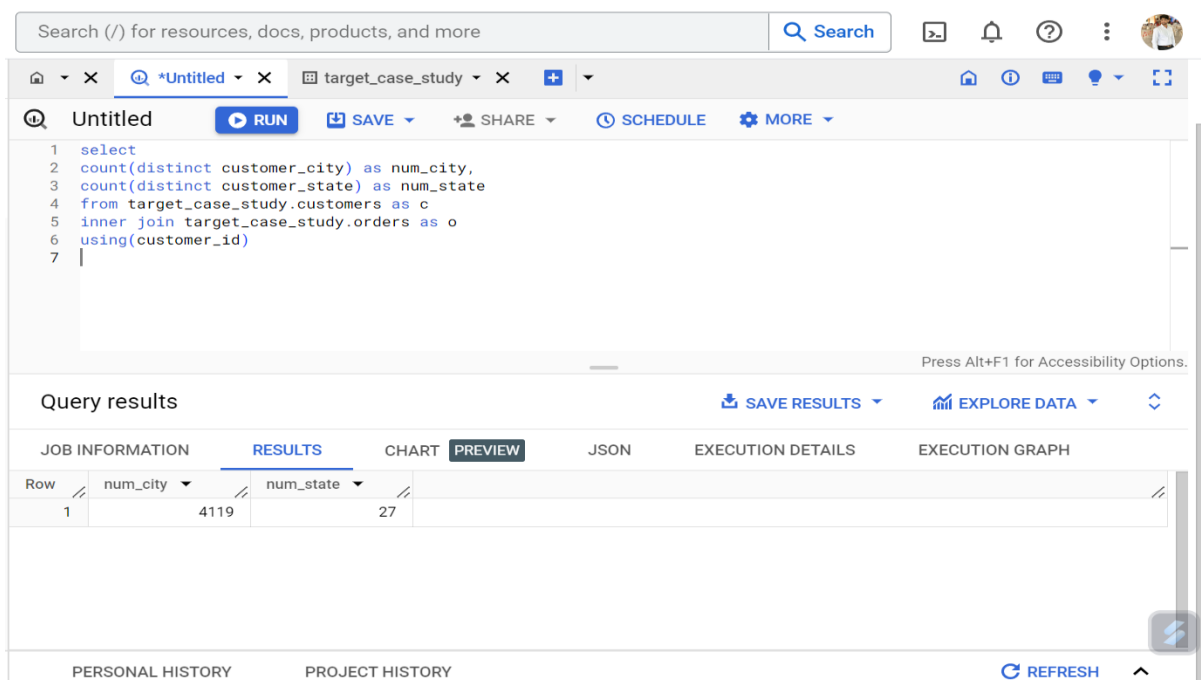
❖ Insights

Difference between first_order and last_order is almost 2 Years.

Q.1

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

```
select
count(distinct customer_city) as num_city,
count(distinct customer_state) as num_state
from target_case_study.customers as c
inner join target_case_study.orders as o
using(customer_id);
```



Search (/) for resources, docs, products, and more [Search](#)

target_case_study

Untitled [RUN](#) [SAVE](#) [SHARE](#) [SCHEDULE](#) [MORE](#)

```
1 select
2 count(distinct customer_city) as num_city,
3 count(distinct customer_state) as num_state
4 from target_case_study.customers as c
5 inner join target_case_study.orders as o
6 using(customer_id)
7
```

Press Alt+F1 for Accessibility Options.

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

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	num_city	num_state					
1	4119	27					

PERSONAL HISTORY PROJECT HISTORY [REFRESH](#)

❖ Insights

Orders were received from all states and cities with few exceptions within the given time

Q.2 In-depth Exploration

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

```
select
extract(year from order_purchase_timestamp) as year,
count(*) as order_count
from target_case_study.orders
group by year
order by year
```

The screenshot shows a web-based data analytics tool. At the top, there is a search bar and navigation icons. Below the search bar, there are tabs for 'Untitled' and 'target_case_study'. The 'target_case_study' tab is active, showing a SQL query in a text editor. The query is:
1 select
2 extract(year from order_purchase_timestamp) as year,
3 count(*) as order_count
4 from target_case_study.orders
5 group by year
6 order by year
Below the query editor, there is a 'RUN' button. Below the 'RUN' button, there is a 'Query results' section. This section has tabs for 'JOB INFORMATION', 'RESULTS', 'CHART', 'PREVIEW', 'JSON', 'EXECUTION DETAILS', and 'EXECUTION GRAPH'. The 'RESULTS' tab is selected, showing a table with 3 rows and 2 columns: 'year' and 'order_count'. The data is as follows:
Row 1: year 2016, order_count 329
Row 2: year 2017, order_count 45101
Row 3: year 2018, order_count 54011
At the bottom of the interface, there are tabs for 'PERSONAL HISTORY' and 'PROJECT HISTORY', and a 'REFRESH' button.

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

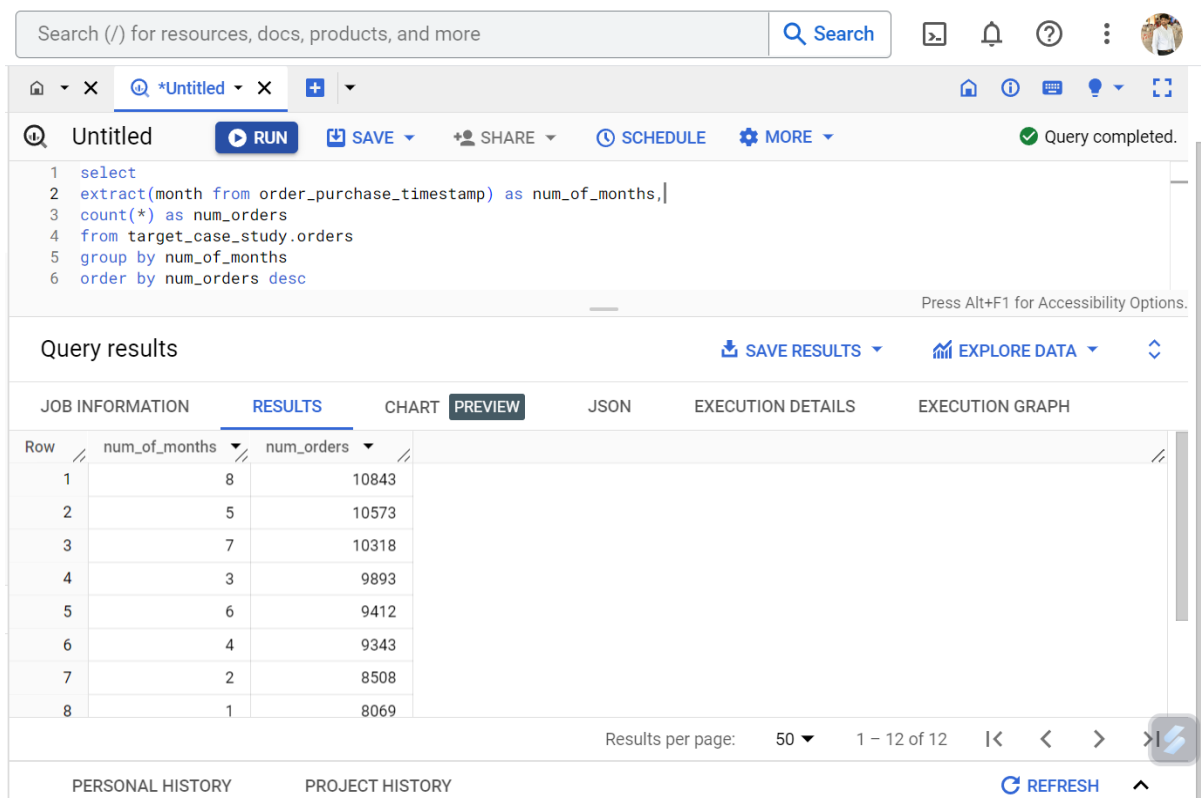
❖ Insights

YES, We can see that orders have been increased over the years

Q. 2

b) 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 num_of_months,
count(*) as num_orders
from target_case_study.orders
group by num_of_months
order by num_orders desc
```



The screenshot shows a web-based query editor. At the top, there is a search bar and a 'Search' button. Below that, a toolbar contains buttons for 'RUN', 'SAVE', 'SHARE', 'SCHEDULE', and 'MORE'. The SQL query is entered in a text area. Below the query, a 'Query results' section shows a table with two columns: 'num_of_months' and 'num_orders'. The table has 8 rows of data. At the bottom, there are tabs for 'PERSONAL HISTORY' and 'PROJECT HISTORY', and a 'REFRESH' button.

Row	num_of_months	num_orders
1	8	10843
2	5	10573
3	7	10318
4	3	9893
5	6	9412
6	4	9343
7	2	8508
8	1	8069

❖ Insights

Yes, we can see that the num_of_months decreases the num_of_orders also decreases.

Q.2

c) During what time of the day, do the Brazilian customers mostly place the 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 order_purchase_timestamp) between 0 and 6 then 'dawn'
when extract(hour from order_purchase_timestamp) between 7 and 12 then 'morning'
when extract (hour from order_purchase_timestamp) between 13 and 18 then
'afternoon'
when extract(hour from order_purchase_timestamp) between 19 and 23 then 'night'
end as time_of_day,
count(distinct order_id) as order_count
from target_case_study.orders
group by time_of_day
order by order_count desc
```

The screenshot shows a SQL query editor with a search bar at the top. The query is as follows:

```
1 select
2 case
3   when extract(hour from order_purchase_timestamp) between 0 and 6 then 'dawn'
4   when extract(hour from order_purchase_timestamp) between 7 and 12 then 'morning'
5   when extract (hour from order_purchase_timestamp) between 13 and 18 then 'afternoon'
6   when extract(hour from order_purchase_timestamp) between 19 and 23 then 'night'
7 end as time_of_day,
8 count(distinct order_id) as order_count
9 from target_case_study.orders |
10 group by time_of_day
11 order by order_count desc
```

Below the query editor, the 'Query results' section is displayed. It includes tabs for 'JOB INFORMATION', 'RESULTS', 'CHART', 'PREVIEW', 'JSON', 'EXECUTION DETAILS', and 'EXECUTION GRAPH'. The 'RESULTS' tab is active, showing a table with the following data:

Row	time_of_day	order_count
1	afternoon	38135
2	night	28331
3	morning	27733
4	dawn	5242

At the bottom of the interface, there are sections for 'PERSONAL HISTORY' and 'PROJECT HISTORY', along with a 'REFRESH' button.

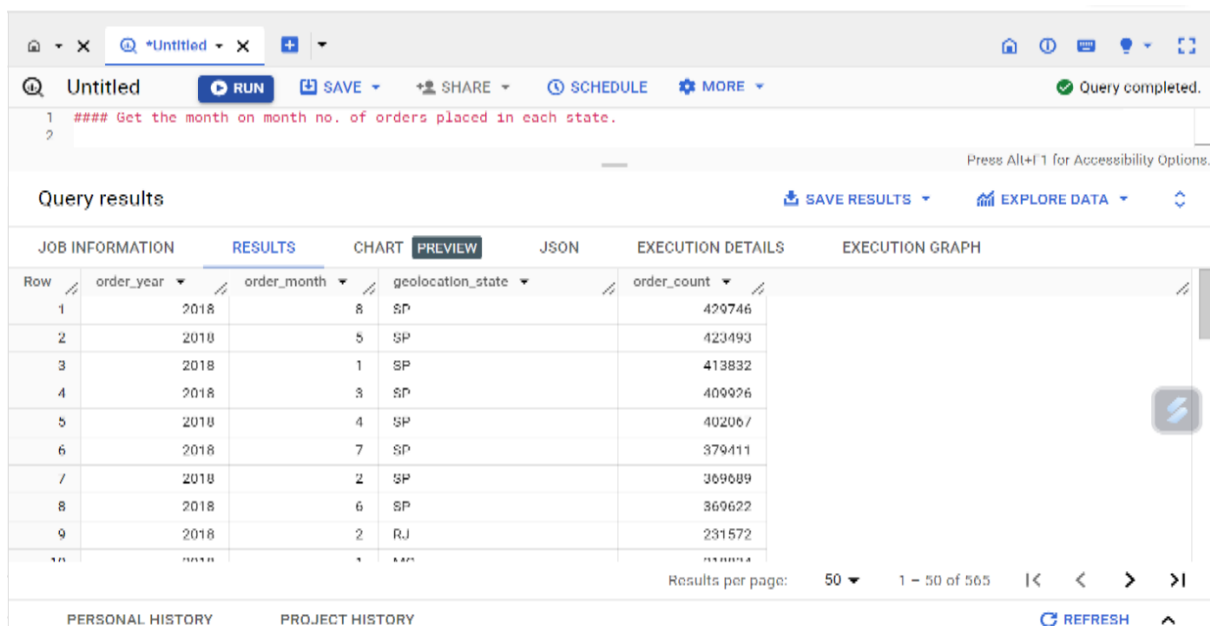
❖ Insights

At afternoon Brazilian peoples places highest amout orders followed by night and morning very few orders places at dawn.

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

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

```
SELECT
EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
g.geolocation_state,
COUNT(o.order_id) AS order_count
FROM target_case_study.orders as o
inner join target_case_study.customers as c on o.customer_id =
c.customer_id
inner join target_case_study.geolocation as g on
c.customer_zip_code_prefix = g.geolocation_zip_code_prefix
GROUP BY
    order_year,
    order_month,
    geolocation_state
order by
    order_count desc,
    order_year desc
```



Query results

Row	order_year	order_month	geolocation_state	order_count
1	2018	8	SP	429746
2	2018	5	SP	423493
3	2018	1	SP	413832
4	2018	3	SP	409926
5	2018	4	SP	402067
6	2018	7	SP	379411
7	2018	2	SP	369689
8	2018	6	SP	369622
9	2018	2	RJ	231572

Results per page: 50 1 - 50 of 565

PERSONAL HISTORY PROJECT HISTORY REFRESH

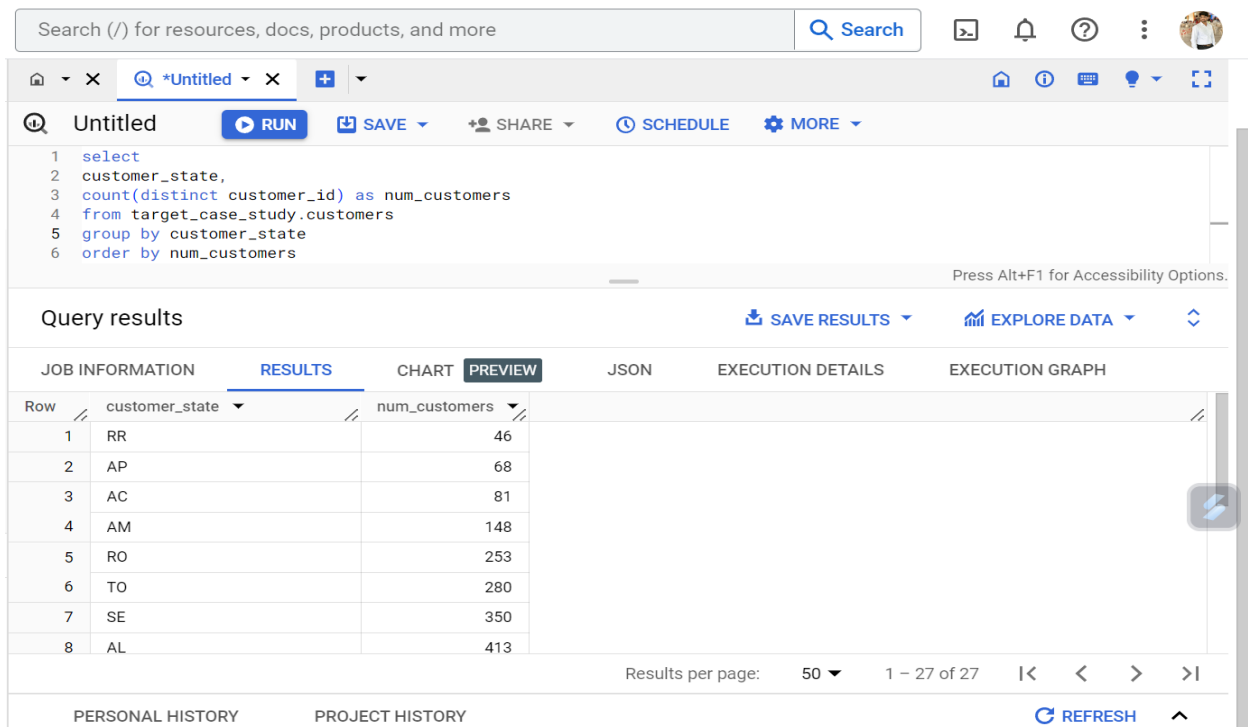
Insights

In 2018 SP is 8 times out of top 10 by order_count .

Q.3

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

```
select
customer_state,
count(distinct customer_id) as num_customers
from target_case_study.customers
group by customer_state
order by num_customers
```



Search (/) for resources, docs, products, and more [Search](#)

Untitled [RUN](#) [SAVE](#) [SHARE](#) [SCHEDULE](#) [MORE](#)

```
1 select
2 customer_state,
3 count(distinct customer_id) as num_customers
4 from target_case_study.customers
5 group by customer_state
6 order by num_customers
```

Press Alt+F1 for Accessibility Options.

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

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

Row	customer_state	num_customers
1	RR	46
2	AP	68
3	AC	81
4	AM	148
5	RO	253
6	TO	280
7	SE	350
8	AL	413

Results per page: 50 1 – 27 of 27 [|<](#) [<](#) [>](#) [>|](#)

PERSONAL HISTORY PROJECT HISTORY [REFRESH](#)

❖ Insights

The distribution of the customers is not uniform, there is major variation present between state wise data.

Q.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).

```
with cte_2017 as
(select extract(year FROM o.order_purchase_timestamp) as year,
sum(payment_value) as pur_1
from target_case_study.orders o INNER JOIN target_case_study.payments p
USING(order_id)
where extract(year FROM o.order_purchase_timestamp) = 2017
AND extract(month FROM o.order_purchase_timestamp) BETWEEN 1 and 8
group by year),
cte_2018 as
(select
extract(year FROM o.order_purchase_timestamp) as year,
sum(payment_value) as pur_2
from target_case_study.orders o INNER JOIN target_case_study.payments p
USING(order_id)
where extract(year FROM o.order_purchase_timestamp) = 2018
AND extract(month FROM o.order_purchase_timestamp) BETWEEN 1 and 8
group by year)
select
ROUND((pur_2 - pur_1) / pur_1 * 100,2) as percent_increase
from cte_2017,cte_2018
```

10 | extract(year FROM o.order_purchase_timestamp) as year,

11 | sum(payment_value) as pur_2

12 | from target_case_study.orders o INNER JOIN target_case_study.payments p USING(order_id)

Press Alt+F1 for Accessibility Opti

Query results

SAVE RESULTS

EXPLORE DATA

JOB INFORMATION

RESULTS

CHART

PREVIEW

JSON

EXECUTION DETAILS

EXECUTION GRAPH

Row	percent_increase
1	136.98

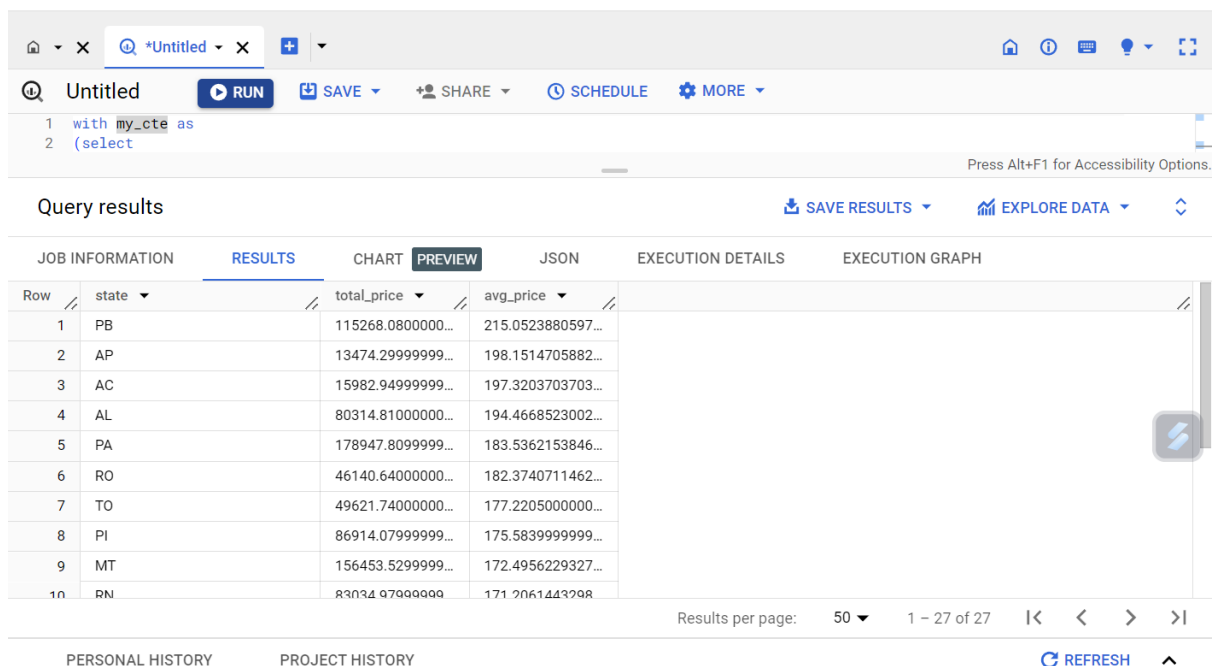
❖ Insights

The % increase in the the cost of orders from year 2017 to 2018 include Months between Jan to Aug) is **136.98%**

Q.4

b) Calculate the Total & Average value of order price for each state.

```
with my_cte as
(select
c.customer_state as state,
SUM(price) as total_price,
COUNT(distinct o.order_id) as num_orders
from target_case_study.customers c LEFT JOIN target_case_study.orders o
USING(customer_id)
LEFT JOIN target_case_study.order_items oi USING(order_id)
group by state)
select
state, my_cte.total_price,
(my_cte.total_price/num_orders) as avg_price
from my_cte
order by avg_price desc
```



Query results

Row	state	total_price	avg_price
1	PB	115268.0800000...	215.0523880597...
2	AP	13474.29999999...	198.1514705882...
3	AC	15982.94999999...	197.3203703703...
4	AL	80314.81000000...	194.4668523002...
5	PA	178947.8099999...	183.5362153846...
6	RO	46140.64000000...	182.3740711462...
7	TO	49621.74000000...	177.2205000000...
8	PI	86914.07999999...	175.5839999999...
9	MT	156453.5299999...	172.4956229327...
10	RN	83034.97000000...	171.2061443208...

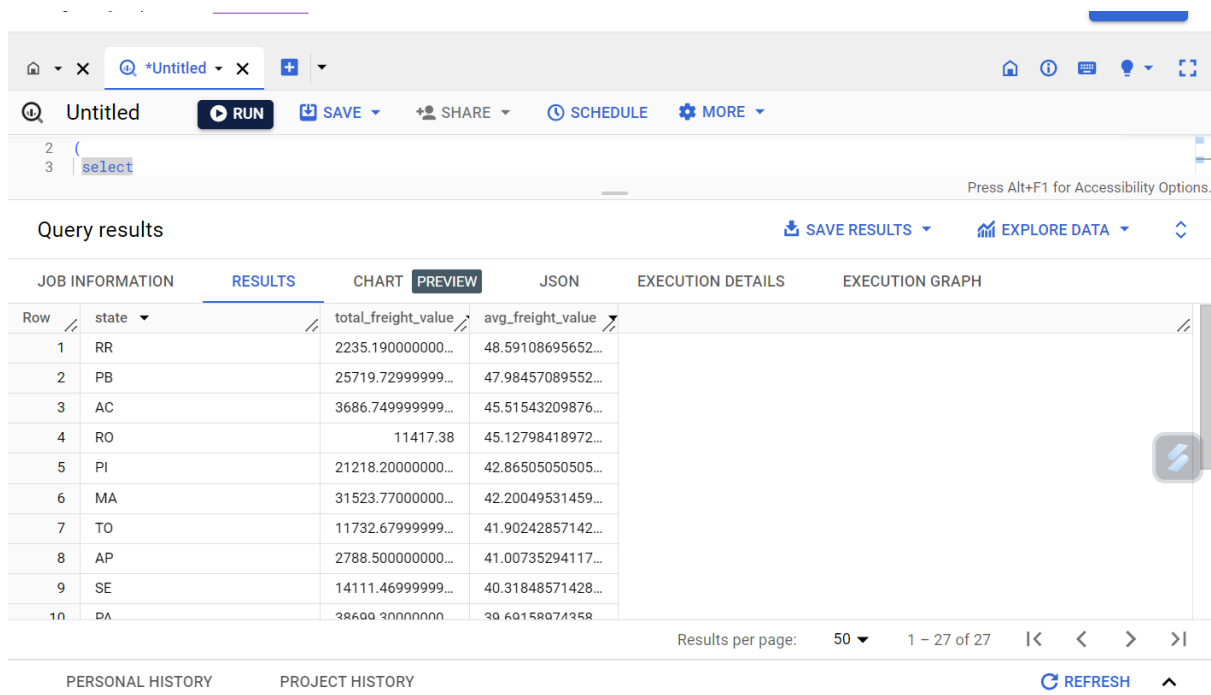
❖ Insights

The state with the highest average cost is the twice the state with the lowest cost. Max avg_price = 215.05 and Min avg_price = 124.63

Q.4

C) Calculate the Total & Average value of order freight for each state.

```
with my_cte as (  
  select  
    c.customer_state as state,  
    SUM(freight_value) as total_freight_value,  
    COUNT(distinct o.order_id) as num_orders  
  from target_case_study.customers c LEFT JOIN  
target_case_study.orders o USING(customer_id)  
  LEFT JOIN target_case_study.order_items oi USING(order_id)  
  group by state)  
select state,  
  my_cte.total_freight_value,  
  (my_cte.total_freight_value/num_orders) as avg_freight_value  
from my_cte  
order by avg_freight_value desc
```



Query results

Row	state	total_freight_value	avg_freight_value
1	RR	2235.190000000...	48.59108695652...
2	PB	25719.72999999...	47.98457089552...
3	AC	3686.749999999...	45.51543209876...
4	RO	11417.38	45.12798418972...
5	PI	21218.20000000...	42.86505050505...
6	MA	31523.77000000...	42.20049531459...
7	TO	11732.67999999...	41.90242857142...
8	AP	2788.500000000...	41.00735294117...
9	SE	14111.46999999...	40.31848571428...
10	PA	38600.30000000...	30.60158071358...

❖ Insights

The PR state has heights average fright value.

Q.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.

Do this in a single query.

You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

- **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
- **diff_estimated_delivery** = order_estimated_delivery_date - order_delivered_customer_date
-

```
select order_id,  
       timestamp_diff(order_delivered_customer_date,order_purchase_t  
imestamp, DAY) as time_to_del,  
       timestamp_diff(order_delivered_customer_date,order_estimated_  
delivery_date, DAY)  
as diff_estimated_dil  
from target_case_study.orders where order_status = 'delivered'
```

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

```
1 select  
2 order_id,  
3 timestamp_diff(order_delivered_customer_date,order_purchase_timestamp, DAY) as  
time_to_del,  
timestamp_diff(order_delivered_customer_date,order_estimated_delivery_date, DAY)  
as diff_estimated_dil  
from target_case_study.orders where order_status = 'delivered'
```

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

Row	order_id	time_to_del	diff_estimated_dil
1	635c894d068ac37e6e03dc54e...	30	-1
2	3b97562c3aee8bdedcb5c2e45...	32	0
3	68f47f50f04c4cb6774570cfde...	29	-1
4	276e9ec344d3bf029ff83a161c...	43	4
5	54e1a3c2b97fb0809da548a59...	40	4
6	fd04fa4105ee8045f6a0139ca5...	37	1
7	302bb8109d097a9fc6e9cefc5...	33	5
8	66057d37308e787052a32828...	38	6
9	19135c945c554eebfd7576c73...	36	2

The table has 4 columns: Row, order_id, time_to_del, and diff_estimated_dil. The data is sorted by Row. The 'time_to_del' column shows the number of days taken to deliver each order, and the 'diff_estimated_dil' column shows the difference between the estimated and actual delivery dates.

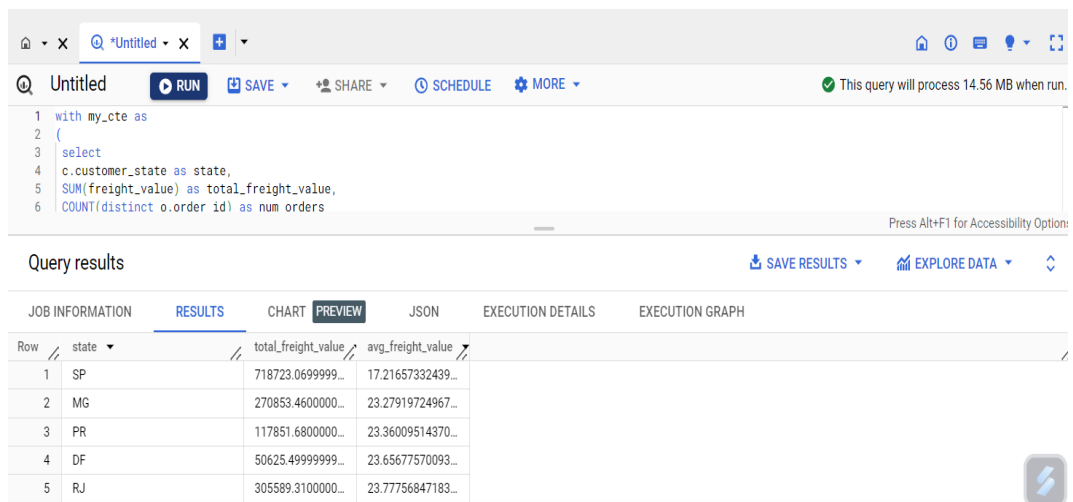
❖ Insights :

Delivery is on time except few orders.

Q.5

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

```
with my_cte as (select
  c.customer_state as state,
  SUM(freight_value) as total_freight_value,
  COUNT(distinct o.order_id) as num_orders
from target_case_study.customers c LEFT JOIN
target_case_study.orders o USING(customer_id)
LEFT JOIN target_case_study.order_items oi USING(order_id)
group by state)
select
state,
  my_cte.total_freight_value,
  (my_cte.total_freight_value/num_orders) as avg_freight_value
from my_cte
order by avg_freight_value asc
limit 5;
```



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

```
1 with my_cte as
2 (
3   select
4     c.customer_state as state,
5     SUM(freight_value) as total_freight_value,
6     COUNT(distinct o.order_id) as num_orders
```

Below the query editor, the 'Query results' section is displayed, showing a table with 5 rows and 4 columns: Row, state, total_freight_value, and avg_freight_value. The results are sorted by avg_freight_value in ascending order.

Row	state	total_freight_value	avg_freight_value
1	SP	718723.0699999...	17.21657332439...
2	MG	270853.4600000...	23.27919724967...
3	PR	117851.6800000...	23.36009514370...
4	DF	50625.499999999...	23.65677570093...
5	RJ	305589.3100000...	23.77756847183...

❖ Insights

Top 5 state with the highest and lowest average freight value is SP, MG, PR, DF, RJ.

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

```
SELECT
customer_state AS state,
ROUND(SUM(TIMESTAMP_DIFF(order_delivered_customer_date,
order_purchase_timestamp,
DAY))/COUNT(ORDER_ID), 2) AS average_time_for_del,
ROUND(SUM(TIMESTAMP_DIFF(order_estimated_delivery_date,
order_purchase_timestamp,
DAY))/COUNT(ORDER_ID), 2) AS average_est_dil_time,
FROM `target_case_study.orders` o
INNER JOIN `target_case_study.customers` c
ON o.customer_id=c.customer_id
WHERE order_status='delivered'
GROUP BY customer_state
ORDER BY average_time_for_del asc/des      #### asc for lowest and desc for
highest
limit 5
```

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON
Row	state		average_time_for_de	average_est_dil_time	
1	RR		28.98	45.63	
2	AP		26.73	45.87	
3	AM		25.99	44.92	
4	AL		24.04	32.21	
5	PA		23.32	36.79	

Query results

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON
Row	state		average_time_for_de	average_est_dil_time	
1	SP		8.3	18.78	
2	PR		11.53	24.25	
3	MG		11.54	24.19	
4	DF		12.51	23.95	
5	SC		14.48	25.41	

❖ Insights

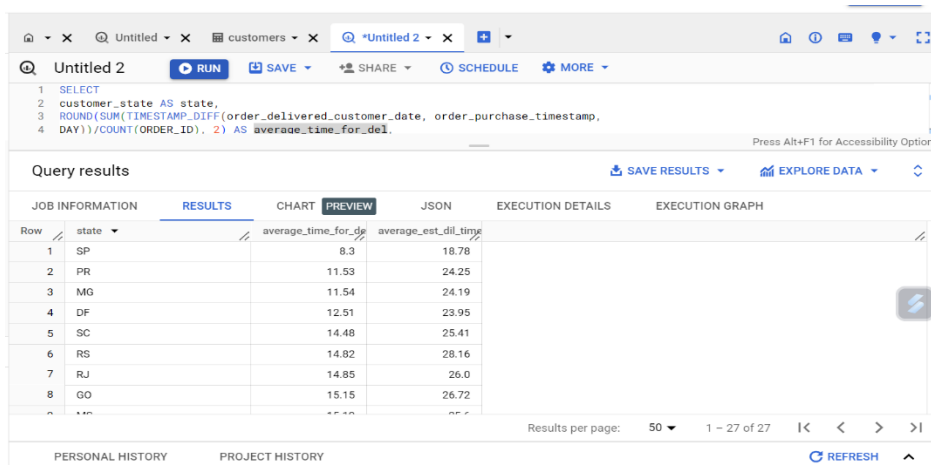
Average time for delivery is half of average estimate delivery time.

Q.5

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

You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

```
SELECT
customer_state AS state,
ROUND(SUM(TIMESTAMP_DIFF(order_delivered_customer_date,
order_purchase_timestamp,
DAY))/COUNT(ORDER_ID), 2) AS average_time_for_del,
ROUND(SUM(TIMESTAMP_DIFF(order_estimated_delivery_date,
order_purchase_timestamp,
DAY))/COUNT(ORDER_ID), 2) AS average_est_dil_time,
FROM `target_case_study.orders` o
INNER JOIN `target_case_study.customers` c
ON o.customer_id=c.customer_id
WHERE order_status='delivered'
GROUP BY customer_state
ORDER BY average_time_for_del
```



Query results

Row	state	average_time_for_del	average_est_dil_time
1	SP	8.3	18.78
2	PR	11.53	24.25
3	MG	11.54	24.19
4	DF	12.51	23.95
5	SC	14.48	25.41
6	RS	14.82	28.16
7	RJ	14.85	26.0
8	GO	15.15	26.72

❖ Insights

The average time for delivery and average estimate delivery time in SP state is less as compare to other state.

Q.6

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

```
SELECT
EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
p.payment_type,
COUNT(o.order_id) AS order_count
FROM target_case_study.orders as o
inner join target_case_study.payments as p on o.order_id = p.order_id
GROUP BY
    order_year,
    order_month,
    payment_type
ORDER BY
    order_year ,
    order_month desc,
    order_count desc;
```

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

```
SELECT
EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
p.payment_type,
COUNT(o.order_id) AS order_count
FROM target_case_study.orders as o
inner join target_case_study.payments as p on o.order_id = p.order_id
GROUP BY
    order_year,
    order_month,
    payment_type
ORDER BY
    order_year ,
    order_month desc,
    order_count desc;
```

The query results are displayed in a table with the following columns: Row, order_year, order_month, payment_type, and order_count. The results are sorted by order_year, order_month (descending), and order_count (descending).

Row	order_year	order_month	payment_type	order_count
1	2016	12	credit_card	1
2	2016	10	credit_card	254
3	2016	10	UPI	63
4	2016	10	voucher	23
5	2016	10	debit_card	2
6	2016	9	credit_card	3
7	2017	12	credit_card	4377
8	2017	12	UPI	1160

The interface also includes a 'Load more' button and a 'Results per page' dropdown set to 50. The total number of results is 90.

❖ Insights

The most favorite payment type of customers is credit_card .

Q.6

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

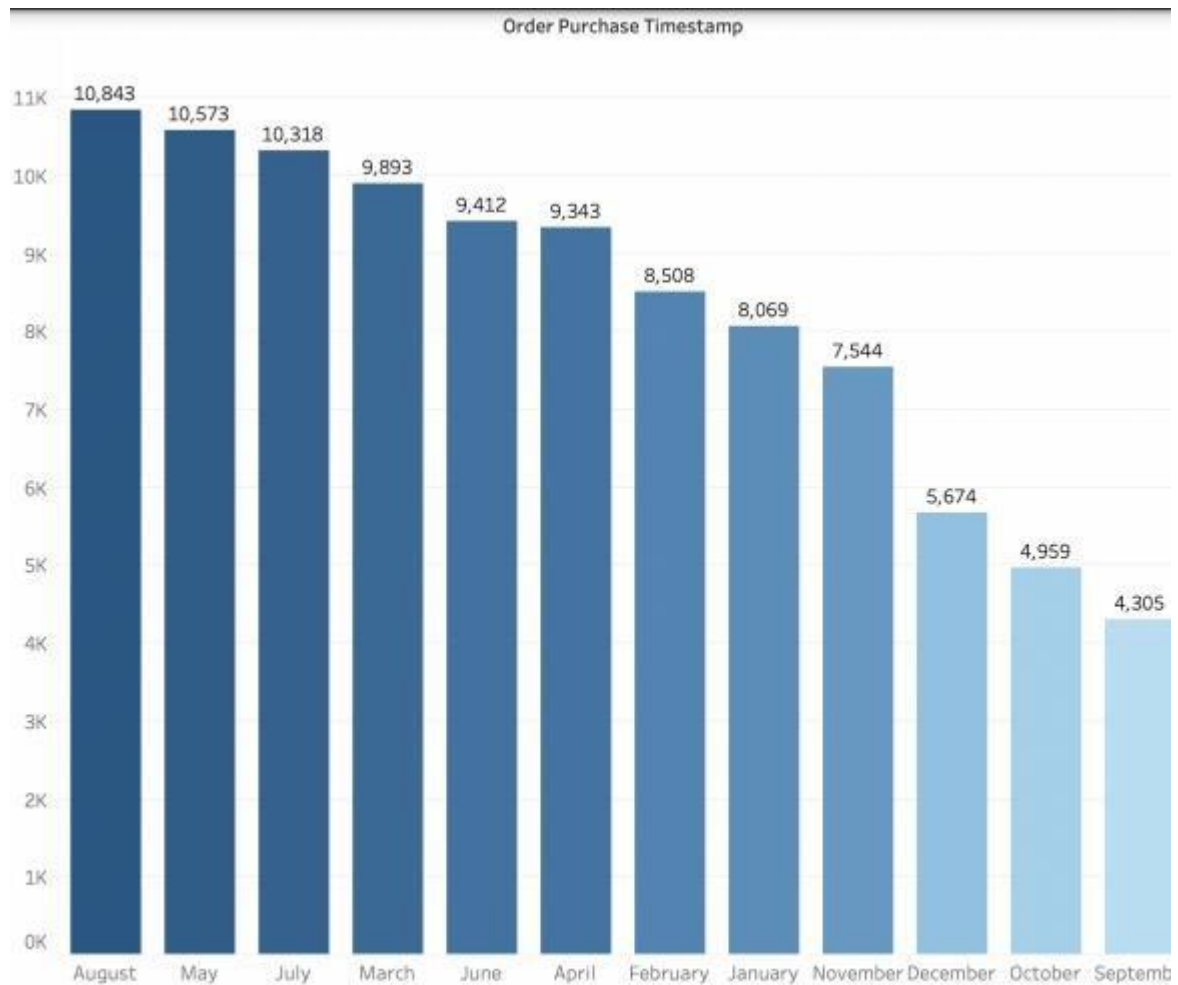
```
select
payment_installments as installments,
count(order_id) as num_of_orders
from target_case_study.payments
where payment_installments >= 1
group by payment_installments
order by num_of_orders desc
```

Query results

Row	installments	num_of_orders
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	10	5328
6	5	5239
7	8	4268
8	6	3920
9	7	1626
10	9	644

❖ Insights

With few exceptions, installments decrease while num_of_order increase.



Recommendation: The month of August shows the highest sale in the entire year, that shows we can plan more ideas to increase the sales bar. We can provide them some good offers or we can try mid sale and discounts to increase the sale. We can provide some vouchers or gift cards to the customers who come under medium range buyers category. We can Send additional small gifts to boost the customer buying interest like some freebies.