

# Target (SQL) Business Case

## About the Company:-

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.

**Business Problem:-** To gain valuable insights into Target's operations in Brazil. The information can shed light on various aspects of the business, such as order processing, pricing strategies, payment and shipping efficiency, customer demographics, product characteristics, and customer satisfaction levels.

## I.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.

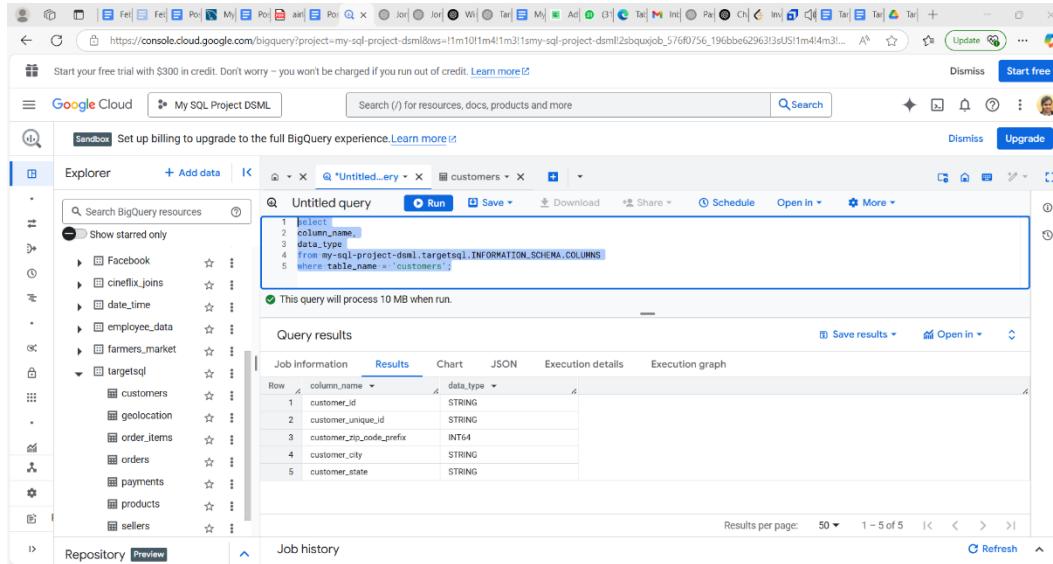
```
select
```

```
column_name,
```

```
data_type
```

```
from my-sql-project-dsml.targetsql.INFORMATION_SCHEMA.COLUMNS
```

```
where table_name = 'customers';
```



The screenshot shows the Google Cloud BigQuery interface. The left sidebar displays a project structure with various datasets and tables. The main area shows a query editor with the following SQL code:

```
1 select
2   column_name,
3   data_type
4 from my-sql-project-dsml.targetsql.INFORMATION_SCHEMA.COLUMNS
5 where table_name = 'customers';
```

The query results pane shows the following data:

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

## Insights & Recommendations:-

As per the above results, customers table have most of the columns of string datatype which we can say is of varchar type.

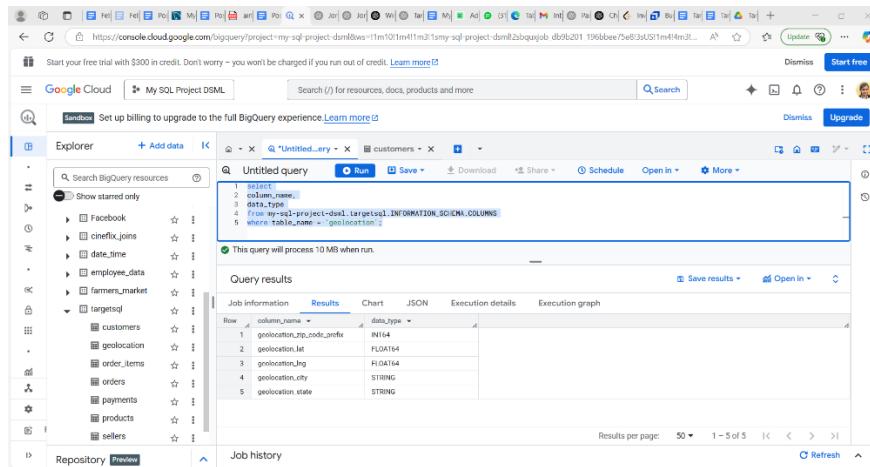
select

column\_name,

data\_type

from my-sql-project-dsml.targetsql.INFORMATION\_SCHEMA.COLUMNS

where table\_name = 'geolocation';



The screenshot shows the Google Cloud BigQuery interface. In the left sidebar, there's an 'Explorer' section with a tree view of datasets like Facebook, employee\_data, farmers\_market, and targetsql, which contains tables such as customers, geolocation, orders, payments, products, and sellers. The main area has a query editor with the following SQL code:

```
1 select
2   column_name,
3   data_type
4 from my-sql-project-dsml.targetsql.INFORMATION_SCHEMA.COLUMNS
5 where table_name = 'geolocation';
```

Below the query editor is a 'Query results' table with the following data:

Row	column_name	data_type
1	geolocation_zip_code_prefix	INT64
2	geolocation_lat	FLOAT64
3	geolocation_lng	FLOAT64
4	geolocation_city	STRING
5	geolocation_state	STRING

### Insights & Recommendation:-

We have mixed data types – lat,lon are of float type, zip code is of int type, city and state is of string type hence this table contents complete location details.

select

column\_name,

data\_type

from my-sql-project-dsml.targetsql.INFORMATION\_SCHEMA.COLUMNS

where table\_name = 'order\_items';

The screenshot shows the Google Cloud BigQuery interface. In the top navigation bar, there are tabs for 'Google Cloud' and 'My SQL Project DSML'. A search bar contains the placeholder 'Search (/) for resources, docs, products and more'. On the far right, there are buttons for 'Dismiss' and 'Start free'.

In the main area, there's a banner that says 'Sandbox Set up billing to upgrade to the full BigQuery experience.' with a 'Learn more' link. Below the banner is the 'Explorer' section, which includes a sidebar with various datasets like Facebook, cineflix\_joins, date\_time, employee\_data, farmers\_market, and targetsql. The 'targetsqli' dataset is expanded, showing tables such as customers, geolocation, order\_items, orders, payments, products, and sellers. A specific query is running in the main pane:

```

1 select
2   column_name,
3   data_type
4 from my-sql-project-dsml.targetsqli.INFORMATION_SCHEMA.COLUMNS
5 where table_name = 'order_items';

```

The query has completed successfully, and the results table shows the following data:

Row	column_name	data_type
1	order_id	STRING
2	order_item_id	INT64
3	product_id	STRING
4	seller_id	STRING
5	shipping_limit_date	TIMESTAMP
6	price	FLOAT64
7	freight_value	FLOAT64

At the bottom of the results pane, there are buttons for 'Save results' and 'Open in'.

## Insights & Recommendations:-

We can see this table contains order\_id, product\_id, order\_id which are string however the date is of timestamp type and the other two price and freight value are float type.

select

column\_name,

data\_type

from my-sql-project-dsml.targetsqli.INFORMATION\_SCHEMA.COLUMNS

where table\_name = 'orders';

This screenshot shows the same BigQuery interface as the previous one, but the results table now displays data for the 'orders' table. The columns are 'order\_id' (STRING), 'customer\_id' (STRING), 'order\_status' (STRING), 'order\_purchase\_timestamp' (TIMESTAMP), 'order\_approved\_at' (TIMESTAMP), 'order\_delivered\_carrier\_date' (TIMESTAMP), and 'order\_delivered\_customer\_date' (TIMESTAMP). The results table shows 8 rows of data.

We can see the `order_id`,`customer_id`,`order_status` are of String type; and the rest three are of timestamp type hence the table `orders` has all the information for an order including the date and time at which the order was placed.

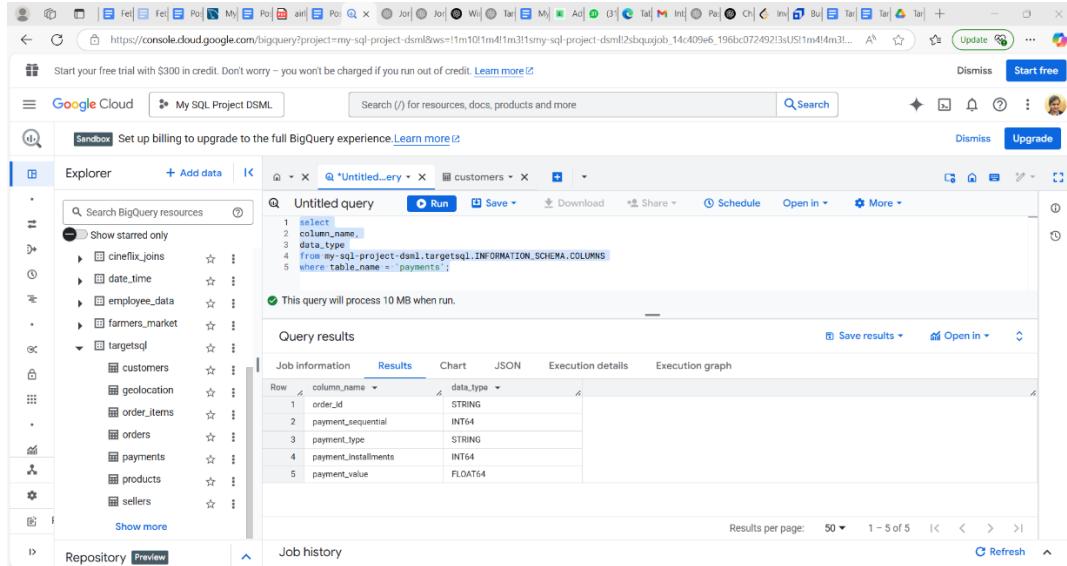
select

`column_name,`

`data_type`

from my-sql-project-dsml.targetsql.INFORMATION\_SCHEMA.COLUMNS

where table\_name = 'payments';



The screenshot shows the Google Cloud BigQuery interface. In the left sidebar, there's an 'Explorer' section with a tree view of datasets and tables. The main area is titled 'Untitled query' and contains the following SQL code:

```
1 select
2   column_name
3   , data_type
4   from my-sql-project-dsml.targetsql.INFORMATION_SCHEMA.COLUMNS
5   where table_name = 'payments';
```

Below the code, a note says 'This query will process 10 MB when run.' The 'Results' tab is selected in the 'Query results' section. The results table has two columns: 'column\_name' and 'data\_type'. The data is as follows:

Row	column_name	data_type
1	order_id	STRING
2	payment_sequential	INT64
3	payment_type	STRING
4	payment_installments	INT64
5	payment_value	FLOAT64

We can see the `order_id` and `payment_type` as string , `payment_sequential` and `payment_installments` as integer, `payment_value` as float; 2 string , 2 integer and 1 float data type.

select

`column_name,`

`data_type`

from my-sql-project-dsml.targetsql.INFORMATION\_SCHEMA.COLUMNS

where table\_name = 'products';

The screenshot shows the Google Cloud BigQuery interface. In the left sidebar, there's an 'Explorer' section with a tree view of datasets and tables. The main area shows a query editor with the following SQL code:

```

1 select
2   column_name,
3   data_type
4 from my-sql-project-dsml.targetsql.INFORMATION_SCHEMA.COLUMNS
5 where table_name = 'products';

```

The 'Results' tab is selected, displaying the following table:

Row	column_name	data_type
1	product_id	STRING
2	product_category	STRING
3	product_name_length	INT64
4	product_description_length	INT64
5	product_photos_qty	INT64
6	product_weight_g	INT64
7	product_length_cm	INT64

We can see here two types of data types – String and Integer. Product\_id and Product category are string type; Product\_name\_length, Product\_description\_length, Product\_photos\_qty, Product\_weight\_g, Product\_length\_cm, Product\_height\_cm, Product\_weight\_cm are Integer type.

select

column\_name,

data\_type

from my-sql-project-dsml.targetsql.INFORMATION\_SCHEMA.COLUMNS

where table\_name = 'sellers';

The screenshot shows the Google Cloud BigQuery interface. In the left sidebar, there's an 'Explorer' section with a tree view of datasets and tables. The main area shows a query editor with the same SQL code as the previous screenshot:

```

1 select
2   column_name,
3   data_type
4 from my-sql-project-dsml.targetsql.INFORMATION_SCHEMA.COLUMNS
5 where table_name = 'sellers';

```

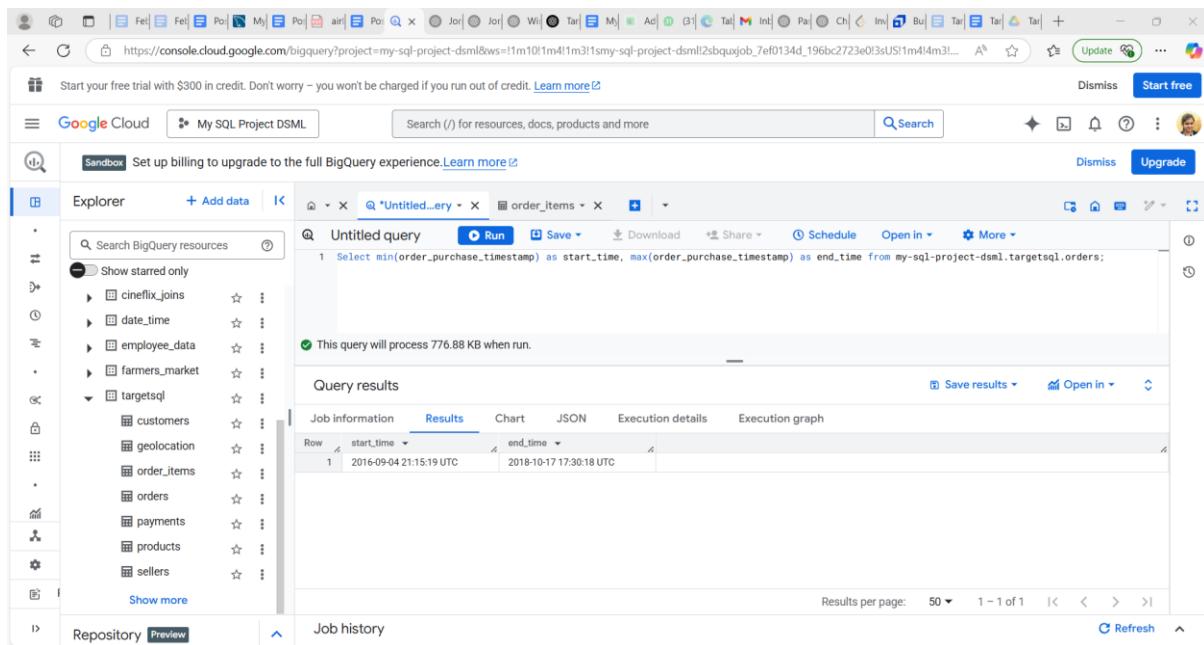
The 'Results' tab is selected, displaying the following table:

Row	column_name	data_type
1	seller_id	STRING
2	seller_zip_code_prefix	INT64
3	seller_city	STRING
4	seller_state	STRING

We can see two types of data types- string and integer. Seller city, state and id are string while seller zip code is integer.

## B. 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 my-sql-project-dsml.targetsql.orders;



The screenshot shows the Google Cloud BigQuery interface. In the center, there is a query editor window titled "Untitled query". The query is:

```
1 Select min(order_purchase_timestamp) as start_time, max(order_purchase_timestamp) as end_time from my-sql-project-dsml.targetsql.orders;
```

Below the query editor, the "Results" tab is selected, showing the following table:

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

At the bottom of the results table, it says "Results per page: 50 1 - 1 of 1".

## Insights:-

First order was placed at 9:59:19 PM UTC on 4th Sept 2016 and Last order was placed at 5:30:18 PM UTC on 17th Oct 2018.

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

Select min(ord.order\_purchase\_timestamp) as start\_time,  
max(ord.order\_purchase\_timestamp) as end\_time, count(distinct cust.customer\_city) as  
number\_of\_cities , count(distinct cust.customer\_state) as number\_of\_states from my-sql-  
project-dsml.targetsql.customers cust join my-sql-project-dsml.targetsql.orders ord using  
(customer\_id) ;

The screenshot shows the Google Cloud BigQuery interface. In the top navigation bar, there are tabs for 'File', 'Edit', 'File', 'Po...', 'M...', 'Po...', 'air...', 'Po...', 'Jor...', 'Jor...', 'Wi...', 'Tar...', 'M...', 'Ad...', 'B1...', 'Ta...', 'M...', 'Int...', 'Pa...', 'Ch...', 'Im...', 'Bu...', 'Ta...', 'Ta...', 'Ta...', 'Ta...' and a 'Start free' button. Below the navigation bar, there's a banner for a free trial and a search bar. The main area has tabs for 'Explorer', 'Untitled query', 'order\_items', 'customers', 'orders', and 'More'. The 'Untitled query' tab contains the following SQL code:

```

1 Select min(ord.order_purchase_timestamp) as start_time, max(ord.order_purchase_timestamp) as end_time, count(distinct cust.customer_city) as
2 number_of_cities , count(distinct cust.customer_state) as number_of_states from my-sql-project-dsml.targetsql.customers cust join
3 my-sql-project-dsml.targetsql.orders ord
        using (customer_id)

```

A note below the code says 'This query will process 8.76 MB when run.' The 'Results' tab is selected, showing a single row of data:

Row	start_time	end_time	number_of_cities	number_of_states
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC	4119	27

At the bottom, there are buttons for 'Save results' and 'Open in', and a message 'Resource ID copied to clipboard.'

## Insights:-

4119 cities from 27 states placed the order during the given time period from 9:59:19 PM UTC on 4th Sept 2016 to 5:30:18 PM UTC on 17th Oct 2018.

### I.In-depth Exploration:

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

Exploration done both year wise and month wise

#### Year-Wise Analysis

Select \*,

case when number\_of\_orders-lag(number\_of\_orders) over (order by year)=0 then 'same'

when number\_of\_orders-lag(number\_of\_orders) over (order by year)>0 then 'increase'

when number\_of\_orders-lag(number\_of\_orders) over (order by year)<0 then 'decrease'

end as trend\_from\_previous\_year

from

(Select FORMAT\_TIMESTAMP('%Y',order\_purchase\_timestamp) AS year,count(order\_id) as  
number\_of\_orders from my-sql-project-dsml.targetsql.orders group by  
FORMAT\_TIMESTAMP('%Y',order\_purchase\_timestamp))x

order by year;

The screenshot shows the Google Cloud BigQuery interface. In the top navigation bar, there are links for 'Start your free trial with \$300 in credit. Don't worry - you won't be charged if you run out of credit. [Learn more](#)' and 'Dismiss' and 'Start free' buttons. Below the navigation bar, there's a search bar with 'Search (/) for resources, docs, products and more' and a 'Search' button. A banner at the top says 'Sandbox Set up billing to upgrade to the full BigQuery experience. [Learn more](#)' with 'Dismiss' and 'Upgrade' buttons.

The main area is titled 'Untitled query' and contains the following SQL code:

```

1 Select *, 
2 case when number_of_orders-lag(number_of_orders) over (order by year)=0 then 'same'
3 when number_of_orders-lag(number_of_orders) over (order by year)>0 then 'increase'
4 when number_of_orders-lag(number_of_orders) over (order by year)<0 then 'decrease'
5 end as trend_from_previous_year
6 from
7 (Select FORMAT_TIMESTAMP('%Y',order_purchase_timestamp) AS year,count(order_id) as number_of_orders from my-sql-project-dsml.targetsqli.orders
group by FORMAT_TIMESTAMP('%Y',order_purchase_timestamp))x
8 order by year;

```

A note below the code says 'This query will process 3.98 MB when run.'

The 'Results' tab is selected in the 'Query results' section. The table has columns: Row, year, number\_of\_orders, and trend\_from\_previous\_month. The data is as follows:

Row	year	number_of_orders	trend_from_previous_month
1	2016	329	null
2	2017	45101	increase
3	2018	54011	increase

At the bottom, there are buttons for 'Save results' and 'Open in'.

## Month-wise analysis

Select \* ,

```

case when number_of_orders-lag(number_of_orders) over (order by month)=0 then 'same'
when number_of_orders-lag(number_of_orders) over (order by month)>0 then 'increase'
when number_of_orders-lag(number_of_orders) over (order by month)<0 then 'decrease'
end as trend_from_previous_month
from
(Select FORMAT_TIMESTAMP( '%Y-%m',order_purchase_timestamp) AS month,count(order_id) as number_of_orders from my-sql-project-dsml.targetsqli.orders group by FORMAT_TIMESTAMP( '%Y-%m',order_purchase_timestamp))x
order by month;

```

The screenshot shows the Google Cloud BigQuery interface. On the left, the 'Explorer' sidebar lists datasets like 'targets' and tables such as 'customers', 'geolocation', 'order\_items', 'orders', 'payments', 'products', and 'sellers'. The main area displays an 'Untitled query' with the following SQL code:

```

1 Select *, 
2 case when number_of_orders-lag(number_of_orders) over (order by month)>0 then 'same'
3 when number_of_orders-lag(number_of_orders) over (order by month)<0 then 'increase'
4 when number_of_orders-lag(number_of_orders) over (order by month)=0 then 'decrease'
5 end as trend_from_previous_month
6 from
7 (Select FORMAT_TIMESTAMP( '%Y-%m',order_purchase_timestamp) AS month,count(order_id) as number_of_orders from my-sql-project-dsml.targets
orders group by FORMAT_TIMESTAMP( '%Y-%m',order_purchase_timestamp))x
8 order by month;

```

A note below the code says: "This query will process 3.98 MB when run."

The 'Results' tab is selected in the 'Query results' section, showing a table with the following data:

Row	month	number_of_orders	trend_from_previous_month
1	2016-09	4	null
2	2016-10	324	increase
3	2016-12	1	decrease
4	2017-01	800	increase
5	2017-02	1780	increase

At the bottom right of the results table, there are buttons for 'Save results' and 'Open in'.

## Insights:-

Overall the number of orders have increased each year but if we see month-wise then in 2018 every month the number of orders are mostly decreasing.

In Year 2016, number of order increased from sept to October but then decreased in December.

In year 2017, number of orders increased every month except from March to April.May to June, August to Sept and then November to December.

In year 2018, number of orders increased in the Month of January but then it started decreasing it only increased from Feb to March and then from June to July.

## 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 month,count(order\_id) as number\_of\_orders from my-sql-project-dsml.targets.orders group by month  
order by number\_of\_orders desc;

The screenshot shows the Google Cloud BigQuery interface. In the top navigation bar, there are tabs for 'File', 'Edit', 'File', 'Po...', 'M...', 'Po...', 'air...', 'Po...', 'Jor...', 'Jor...', 'Wi...', 'Tar...', 'M...', 'Ad...', 'B1...', 'Ta...', 'M...', 'Int...', 'Pa...', 'Ch...', 'Im...', 'Bu...', 'Ta...', 'Ta...', 'Ta...', 'Ta...', 'Start free' (button), 'Dismiss' (button), and 'Upgrade' (button). Below the navigation bar, there is a search bar with the placeholder 'Search (/) for resources, docs, products and more' and a 'Search' button. The main area has a title 'Untitled query' with a 'Run' button, a 'Save' dropdown, a 'Download' button, a 'Share' button, a 'Schedule' button, an 'Open in' dropdown, and a 'More' button. The left sidebar is titled 'Explorer' and contains a tree view of datasets and tables, including 'Facebook', 'cineflix\_joins', 'date\_time', 'employee\_data', 'farmers\_market', 'targetsql' (which contains 'customers', 'geolocation', 'order\_items', 'orders', 'payments', 'products', and 'sellers'), and 'Repository'. The main content area shows the query results for an untitled query:

```
1 Select extract (month from order_purchase_timestamp) AS month, count(order_id) as number_of_orders from my-sql-project-dsml.targetsql.orders
2 group by month
3 order by number_of_orders desc;
```

This query will process 3.98 MB when run.

**Query results**

Row	month	number_of_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
9	11	7544

Results per page: 50 ▾ 1 – 12 of 12 | < < > >|

**Insights:-**

Overall, highest number of orders have been placed during August, May and July Month, means, during mid-winter in Brazil (July, August) and also during cultural holidays (like Mother's Day widely celebrated in Brazil in May).

**C. 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 time\_of\_the\_day, count(order\_id) as number\_of\_orders

from

(Select \*,

case when EXTRACT(HOUR FROM order\_purchase\_timestamp)<=6 then 'dawn'

when EXTRACT(HOUR FROM order\_purchase\_timestamp)<=12 then 'morning'

when EXTRACT(HOUR FROM order\_purchase\_timestamp)<=18 then 'Afternoon'

```

else 'Night'

end as time_of_the_day

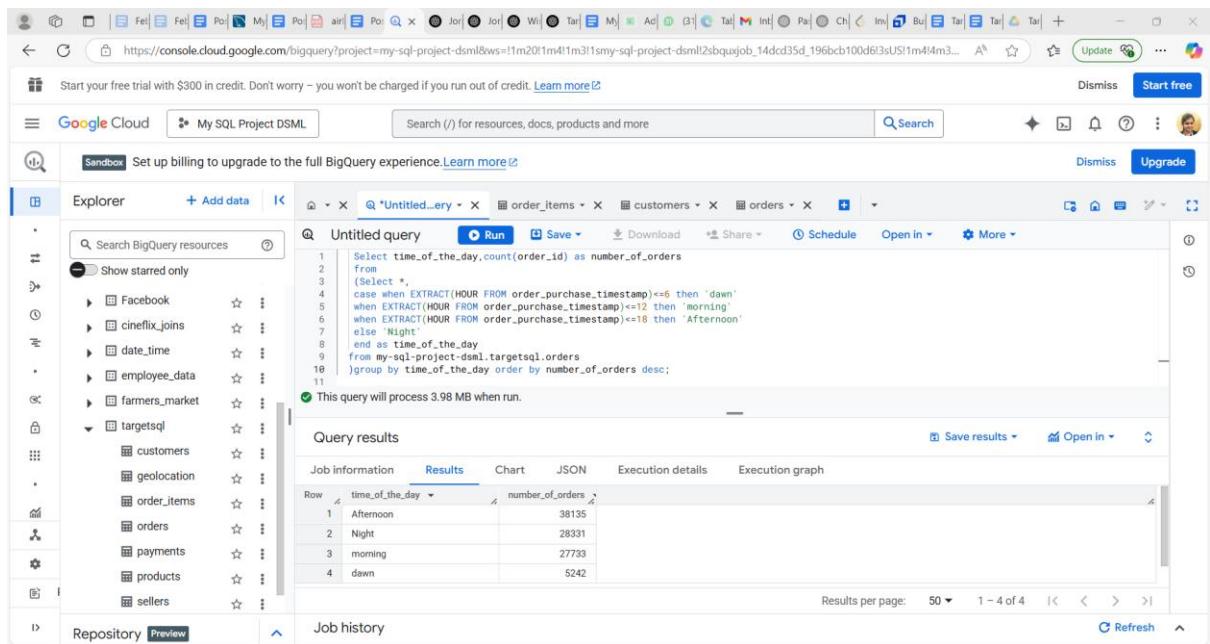
from my-sql-project-dsml.targetsql.orders

)

group by time_of_the_day

order by number_of_orders desc;

```



The screenshot shows the Google Cloud BigQuery interface. On the left, the 'Explorer' sidebar lists various datasets and tables, including 'Facebook', 'cineflix\_joins', 'date\_time', 'employee\_data', 'farmers\_market', 'targetsqli', and 'targetsqli' (which contains 'customers', 'geolocation', 'order\_items', 'orders', 'payments', 'products', and 'sellers'). In the center, an 'Untitled query' tab is open with the following SQL code:

```

1 Select time_of_the_day, count(order_id) as number_of_orders
2   (Select *,
3    case when EXTRACT(HOUR FROM order_purchase_timestamp)<=6 then 'dawn'
4         when EXTRACT(HOUR FROM order_purchase_timestamp)<=12 then 'morning'
5         when EXTRACT(HOUR FROM order_purchase_timestamp)<=18 then 'Afternoon'
6         else 'Night'
7     end as time_of_the_day
8   from my-sql-project-dsml.targetsqli.orders
9   )group by time_of_the_day order by number_of_orders desc;
10
11

```

A note below the code says 'This query will process 3.98 MB when run.' Below the code, the 'Query results' section displays the following table:

Row	time_of_the_day	number_of_orders
1	Afternoon	38135
2	Night	28331
3	morning	27733
4	dawn	5242

## Insights:-

Highest number of orders have been placed in the afternoon (possibly because people take lunch breaks during work) and after 7 PM they are free from work when it's their family time going out during dinner so they are free to do online or offline shopping And making decisions with their families.

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

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

```

Select distinct (cust.customer_state) as state,FORMAT_TIMESTAMP('%B',
order_purchase_timestamp) as Month, count(ord.order_id) as no_of_orders

```

from my-sql-project-dsml.targetsql.customers cust join my-sql-project-dsml.targetsql.orders ord using (customer\_id) group by 1,2 order by no\_of\_orders desc;

Select distinct (cust.customer\_state) as state,FORMAT\_TIMESTAMP('%B',  
order\_purchase\_timestamp) as Month, count(ord.order\_id) as no\_of\_orders  
from my-sql-project-dsml.targetsql.customers cust join my-sql-project-dsml.targetsql.orders  
ord using (customer\_id) group by 1,2 order by no\_of\_orders asc;

The screenshot shows the Google Cloud BigQuery interface. In the top navigation bar, the project is set to "My SQL Project DSML". The main area displays an "Untitled query" with the following SQL code:

```

2 from my-sql-project-dsml.targetsql.customers cust join my-sql-project-dsml.targetsql.orders ord
3 using (customer_id)
4 group by 1,2
5 order by no_of_orders desc;
6

```

A note below the code states: "This query will process 10.81 MB when run." The "Results" tab is selected, showing the following data:

Row	state	Month	no_of_orders
1	RR	January	2
2	AP	September	2
3	RR	September	2
4	RR	November	2
5	RR	May	3
6	AP	October	3
7	AM	October	3
8	RR	October	4

At the bottom of the results table, it says "Results per page: 50 1 – 50 of 322".

The screenshot shows the Google Cloud BigQuery interface. In the top navigation bar, the project is set to "My SQL Project DSML". The main area displays an "Untitled query" with the same SQL code as the previous screenshot:

```

2 from my-sql-project-dsml.targetsql.customers cust join my-sql-project-dsml.targetsql.orders ord
3 using (customer_id)
4 group by 1,2
5 order by no_of_orders desc;
6

```

A note below the code states: "Query completed". The "Results" tab is selected, showing the following data:

Row	state	Month	no_of_orders
1	SP	August	4982
2	SP	May	4632
3	SP	July	4381
4	SP	June	4104
5	SP	March	4047
6	SP	April	3967
7	SP	February	3357
8	SP	January	3351

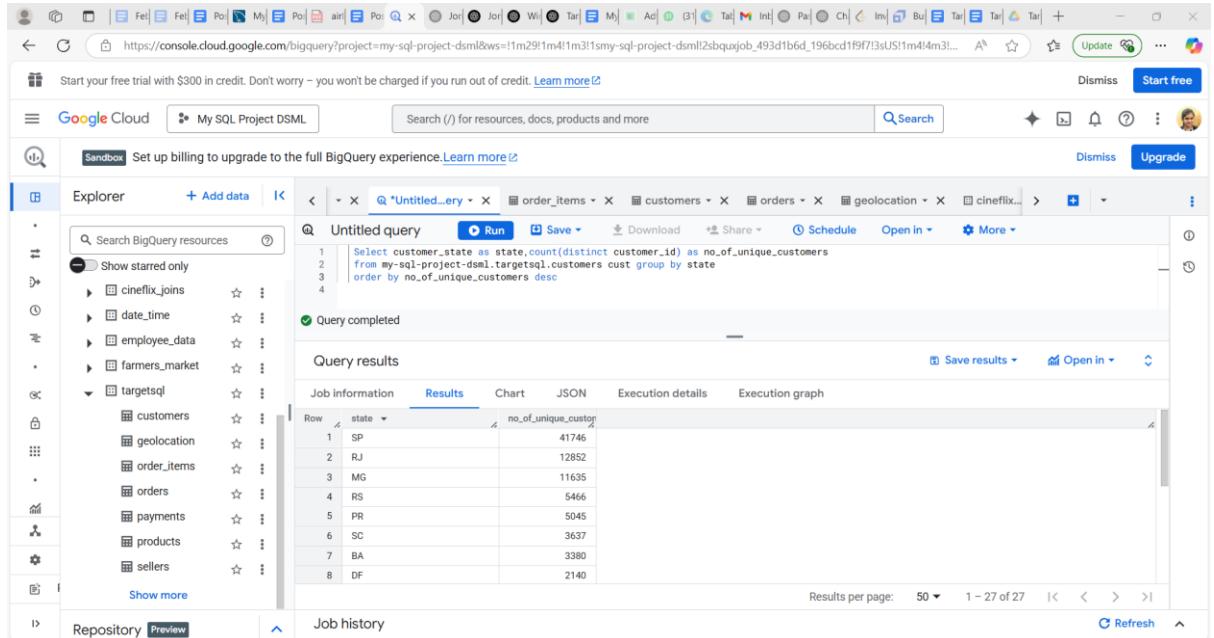
At the bottom of the results table, it says "Results per page: 50 1 – 50 of 322".

## Insights:-

Highest number of orders have been placed in the Month of August in state SP , and lowest number of orders have been placed in the month of January in RR

## B. How are the customers distributed across all the states?

Select customer\_state as state,count(distinct customer\_id) as no\_of\_unique\_customers from my-sql-project-dsml.targetsql.customers cust group by state order by no\_of\_unique\_customers desc



The screenshot shows the Google Cloud BigQuery interface. The left sidebar has an 'Explorer' section with a tree view of datasets like 'cineflix\_joins', 'date\_time', 'employee\_data', 'farmers\_market', and 'targetsqli'. Under 'targetsqli', there are tables: 'customers', 'geolocation', 'order\_items', 'orders', 'payments', 'products', and 'sellers'. A 'Repository' tab is also visible. The main area shows an 'Untitled query' with the following SQL code:

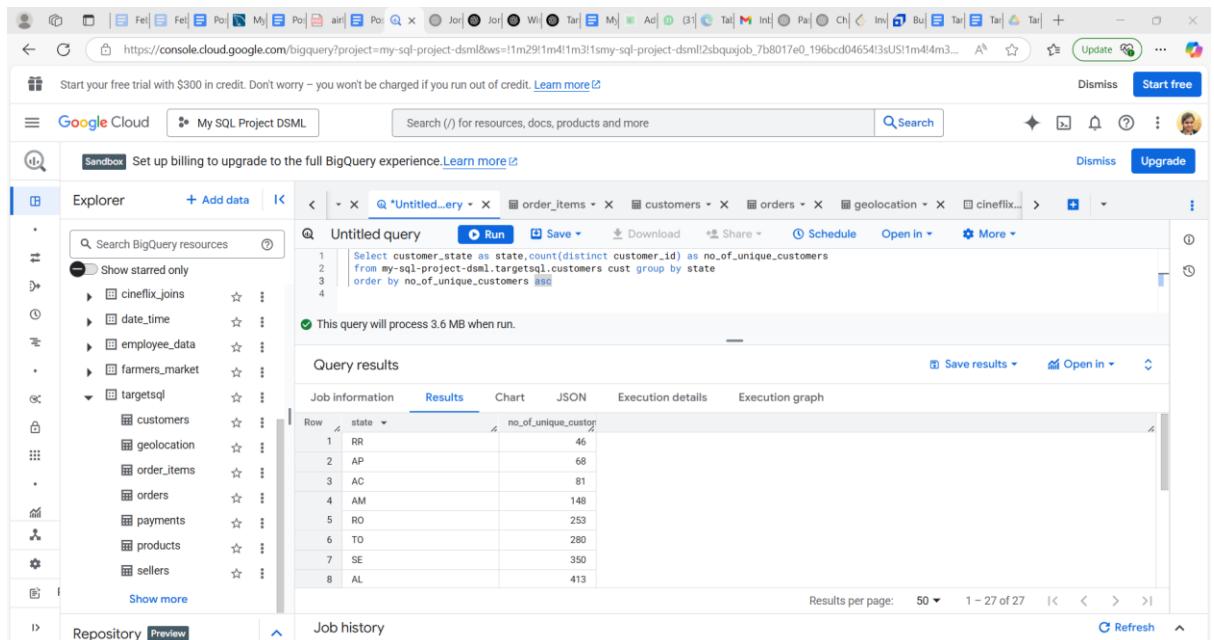
```
1 Select customer_state as state, count(distinct customer_id) as no_of_unique_customers
2   from my-sql-project-dsml.targetsql.customers cust
3   group by state
4   order by no_of_unique_customers desc
```

The status bar indicates 'Query completed'. Below it is the 'Query results' table:

Row	state	no_of_unique_customers
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140

At the bottom, there are buttons for 'Save results' and 'Open in'.

Select customer\_state as state,count(distinct customer\_id) as no\_of\_unique\_customers from my-sql-project-dsml.targetsql.customers cust group by state order by no\_of\_unique\_customers asc



The screenshot shows the Google Cloud BigQuery interface. The left sidebar has an 'Explorer' section with a tree view of datasets like 'cineflix\_joins', 'date\_time', 'employee\_data', 'farmers\_market', and 'targetsqli'. Under 'targetsqli', there are tables: 'customers', 'geolocation', 'order\_items', 'orders', 'payments', 'products', and 'sellers'. A 'Repository' tab is also visible. The main area shows an 'Untitled query' with the following SQL code:

```
1 Select customer_state as state, count(distinct customer_id) as no_of_unique_customers
2   from my-sql-project-dsml.targetsql.customers cust
3   group by state
4   order by no_of_unique_customers asc
```

The status bar indicates 'This query will process 3.6 MB when run.' Below it is the 'Query results' table:

Row	state	no_of_unique_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

At the bottom, there are buttons for 'Save results' and 'Open in'.

**Insights:-** Highest number of customers from state SP while lowest number of customers from state RR

**I.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 year_wise_payments AS (
    SELECT *
        FROM `my-sql-project-dsml.targetsql.payments`
        WHERE order_id IN (
            SELECT order_id
                FROM `my-sql-project-dsml.targetsql.orders`
                WHERE EXTRACT(YEAR FROM order_purchase_timestamp) BETWEEN 2017 AND 2018
                AND EXTRACT(MONTH FROM order_purchase_timestamp) <= 8
        )
),
lagged_payments AS (
    SELECT
        order_id,
        payment_value,
        LAG(payment_value) OVER (ORDER BY payment_value) AS prev_payment_value
    FROM year_wise_payments
)
SELECT
```

```
order_id,  
payment_value,  
prev_payment_value,  
ROUND(  
    100 * (payment_value - prev_payment_value) / prev_payment_value,  
    2  
) AS percentage_increase  
  
FROM lagged_payments  
  
WHERE prev_payment_value <> 0;
```

The screenshot shows the Google Cloud BigQuery interface. At the top, there's a navigation bar with various icons and a search bar. Below it, a banner encourages users to start a free trial with \$300 in credit. The main area has tabs for 'Explorer' and 'Sandbox'. The 'Explorer' tab is active, showing a tree view of datasets like 'cineflix\_joins', 'date\_time', 'employee\_data', 'farmers\_market', and 'targets'. Under 'targets', there are sub-tables: 'customers', 'geolocation', 'order\_items', 'orders', 'payments', 'products', and 'sellers'. A tooltip indicates that this query will process 8.14 MB when run. The 'Results' tab is selected in the 'Query results' section, displaying a table with columns: Row, order\_id, payment\_value, prev\_payment\_value, and percentage\_increase. The table contains 6 rows of data. A message at the bottom says 'Resource ID copied to clipboard.'

Row	order_id	payment_value	prev_payment_value	percentage_increase
1	092a7de4b983f4ee3e77af54...	0.59	0.58	1.72
2	8ce09fab8966711981b3caa...	1.97	1.96	0.51
3	d7c320185a2f9b32f5e2370ff3...	5.91	5.89	0.34
4	4069c489933782af79afcd3a...	7.67	7.66	0.13
5	90d69a292bce7756bbe4485...	8.81	8.8	0.11
6	47d11383b93b217d96defb2e...	8.92	8.91	0.11

**B. Calculate the Total & Average value of order price for each state.**

```
Select customer_state, round(sum(payment_value),2) as  
total_order_price,round(avg(payment_value),2) as avg_price from my-sql-project-  
dsml.targetsql.payments pmt join my-sql-project-dsml.targetsql.orders ord using (order_id)  
join my-sql-project-dsml.targetsql.customers cust using (customer_id) group by  
customer_state
```

order by total\_order\_price desc

```

1 Select customer_state, round(sum(payment_value),2) as total_order_price,round(avg(payment_value),2) as avg_price from my-sql-project-dsml.
targetsqli.payments pnt join my-sql-project-dsml.targetsqli.orders ord using (order_id) join my-sql-project-dsml.targetsqli.customers cust using
(customer_id) group by customer_state
2 order by total_order_price desc

```

customer_state	total_order_price	avg_price
SP	5998226.96	137.5
RJ	2144379.69	158.53
MG	1872257.26	154.71
RS	890898.54	157.18
PR	811156.38	154.15
SC	623086.43	165.98
BA	616645.82	170.82
DF	355141.08	161.13

**Insights:-** Highest order value from State SP while lowest order value from State RR

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

```

Select customer_state, round(sum(freight_value),2) as
total_freight_value,round(avg(freight_value),2) as avg_freight_value from my-sql-project-
dsml.targetsqli.order_items oi join my-sql-project-dsml.targetsqli.orders ord using (order_id)
join my-sql-project-dsml.targetsqli.customers cust using (customer_id) group by
customer_state

```

order by total\_freight\_value desc

**Insights:-** Highest freight value from State SP while lowest freight order value from State RR

### I.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.

```

SELECT
order_id,
DATE(order_purchase_timestamp) AS purchase_date,
DATE(order_delivered_customer_date) AS delivered_date,
DATE(order_estimated_delivery_date) AS estimated_date,

```

```

abs(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)) AS
time_to_deliver,

```

```
abs(DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date, DAY)) AS diff_estimated_delivery
```

```
FROM `my-sql-project-dsml.targetsql.orders`;
```

The screenshot shows the Google Cloud BigQuery interface. In the top navigation bar, there are tabs for 'File', 'Edit', 'Poi', 'My', 'Po', 'air', 'Po', 'Joi', 'Joi', 'Wi', 'Tar', 'Mj', 'Ad', 'G3', 'Ta', 'Int', 'Pa', 'Ch', 'Im', 'Bu', 'Tar', 'Ta', 'Tar'. Below the tabs, there's a message: 'Start your free trial with \$300 in credit. Don't worry – you won't be charged if you run out of credit. [Learn more](#)' with 'Dismiss' and 'Start free' buttons. The main header includes 'Google Cloud' and 'My SQL Project DSML' with a search bar and a 'Search' button. A 'Sandbox' message is displayed: 'Set up billing to upgrade to the full BigQuery experience. [Learn more](#)'. On the left, there's an 'Explorer' sidebar with a tree view of datasets like 'cineflix\_joins', 'date\_time', 'employee\_data', 'farmers\_market', 'targetsqli', 'customers', 'geolocation', 'order\_items', 'orders', 'payments', 'products', and 'sellers'. The central area shows an 'Untitled query' editor with the following SQL code:

```
1 SELECT
2     order_id,
3     DATE(order_purchase_timestamp) AS purchase_date,
4     DATE(order_delivered_customer_date) AS delivered_date,
5     DATE(order_estimated_delivery_date) AS estimated_date,
6     abs(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)) AS time_to_deliver,
7     abs(DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date, DAY)) AS diff_estimated_delivery
8
9 FROM `my-sql-project-dsml.targetsqli.orders`;
```

A note below the code says 'This query will process 5.48 MB when run.' The 'Results' tab is selected in the 'Query results' section. The results table has columns: Row, order\_id, purchase\_date, delivered\_date, estimated\_date, time\_to\_deliver, and diff\_estimated\_delivery. The data is as follows:

Row	order_id	purchase_date	delivered_date	estimated_date	time_to_deliver	diff_estimated_delivery
1	770d331c84e5b214bd9dc70a1...	2016-10-07	2016-10-14	2016-11-29	7	45
2	dabf2b0e35b423f94618bf965f...	2016-10-09	2016-10-16	2016-11-30	7	44
3	8beb59392e21af5eb9547ae1a...	2016-10-08	2016-10-19	2016-11-30	10	41
4	bfb0f9bdef84302105ad712db...	2016-09-15	2016-10-09	2016-10-04	54	36

At the bottom, it says 'Results per page: 50 1 - 50 of 99441' and there are navigation arrows.

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

```
select cust.customer_state,avg(oi.freight_value) as avg_freight_value from my-sql-project-dsml.targetsqli.customers cust join my-sql-project-dsml.targetsqli.orders ord using(customer_id) join my-sql-project-dsml.targetsqli.order_items oi using (order_id)
group by customer_state
order by avg_freight_value asc limit 5;
```

```
select cust.customer_state,avg(oi.freight_value) as avg_freight_value from my-sql-project-dsml.targetsqli.customers cust join my-sql-project-dsml.targetsqli.orders ord using(customer_id) join my-sql-project-dsml.targetsqli.order_items oi
```

```
using (order_id)
group by customer_state
order by avg_freight_value desc limit 5;
```

The screenshot shows the Google Cloud BigQuery interface. On the left, the Explorer sidebar lists various datasets and tables under 'targetsqli'. In the center, an 'Untitled query' window displays a SQL query and its results. The query is:

```

1 select cust.customer_state,avg(oi.freight_value) as avg_freight_value from my-sql-project-dsml.targetsqli.customers cust join my-sql-project-dsml.targetsqli.orders ord using(customer_id) join my-sql-project-dsml.targetsqli.order_items oi
2 using (order_id)
3 group by customer_state
4 order by avg.freight_value asc limit 5;

```

The results table shows the average freight value for five states:

customer_state	avg.freight_value
SP	15.14727539041...
PR	20.53165156794...
MG	20.63016680630...
RJ	20.96092393168...
DF	21.04135494596...

The screenshot shows the Google Cloud BigQuery interface. On the left, the Explorer sidebar lists various datasets and tables under 'targetsqli'. In the center, an 'Untitled query' window displays a SQL query and its results. The query is identical to the one in the previous screenshot:

```

1 select cust.customer_state,avg(oi.freight_value) as avg_freight_value from my-sql-project-dsml.targetsqli.customers cust join my-sql-project-dsml.targetsqli.orders ord using(customer_id) join my-sql-project-dsml.targetsqli.order_items oi
2 using (order_id)
3 group by customer_state
4 order by avg.freight_value desc limit 5;

```

The results table shows the average freight value for five states:

customer_state	avg.freight_value
RR	42.9842307692...
PB	42.7238098671...
RO	41.06971223021...
AC	40.0736956521...
PI	39.14797047970...

**Insights:-** SP is the state with the lowest average freight value and RR is the state with the highest freight value

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

select cust.customer\_state,avg(ord.time\_to\_deliver) as avg\_time\_to\_deliver from my-sql-project-dsml.targetsqli.customers cust join

(SELECT

customer\_id,order\_id,

```

DATE(order_purchase_timestamp) AS purchase_date,
DATE(order_delivered_customer_date) AS delivered_date,
DATE(order_estimated_delivery_date) AS estimated_date,

abs(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)) AS
time_to_deliver,
abs(DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date, DAY)) AS
diff_estimated_delivery

FROM `my-sql-project-dsml.targetsql.orders` ord
using (customer_id)
group by customer_state
order by avg_time_to_deliver desc limit 5;

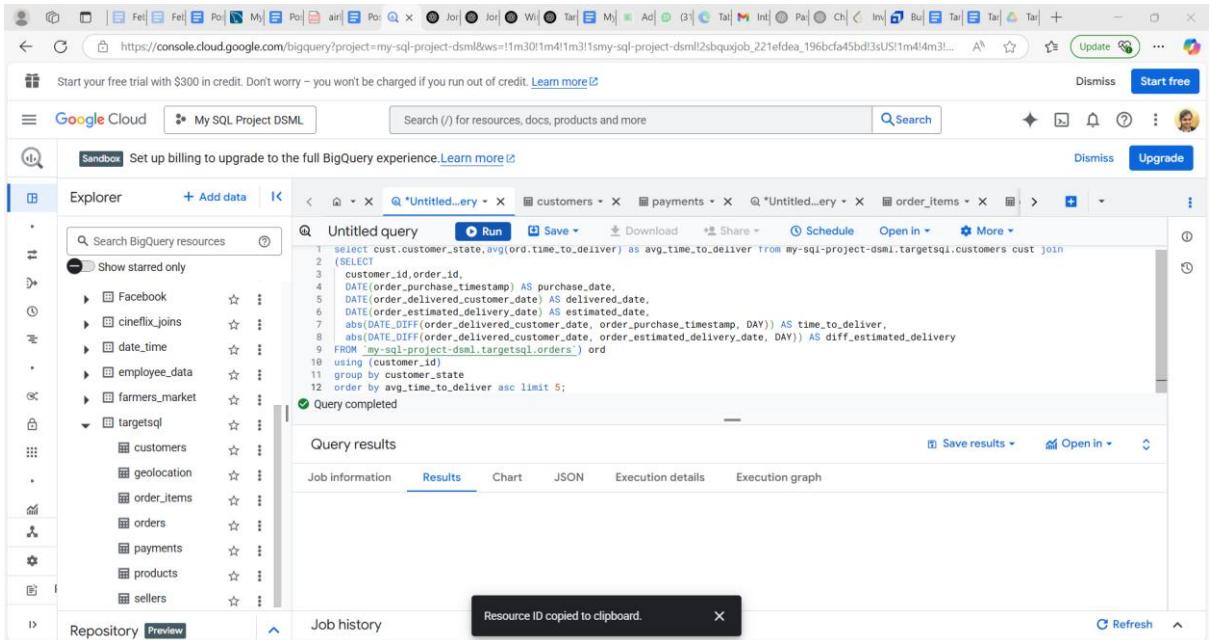
select cust.customer_state,avg(ord.time_to_deliver) as avg_time_to_deliver from my-sql-
project-dsml.targetsql.customers cust join
(SELECT
customer_id,order_id,
DATE(order_purchase_timestamp) AS purchase_date,
DATE(order_delivered_customer_date) AS delivered_date,
DATE(order_estimated_delivery_date) AS estimated_date,
abs(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)) AS
time_to_deliver,
abs(DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date, DAY)) AS
diff_estimated_delivery

FROM `my-sql-project-dsml.targetsql.orders` ord
using (customer_id)

```

group by customer\_state

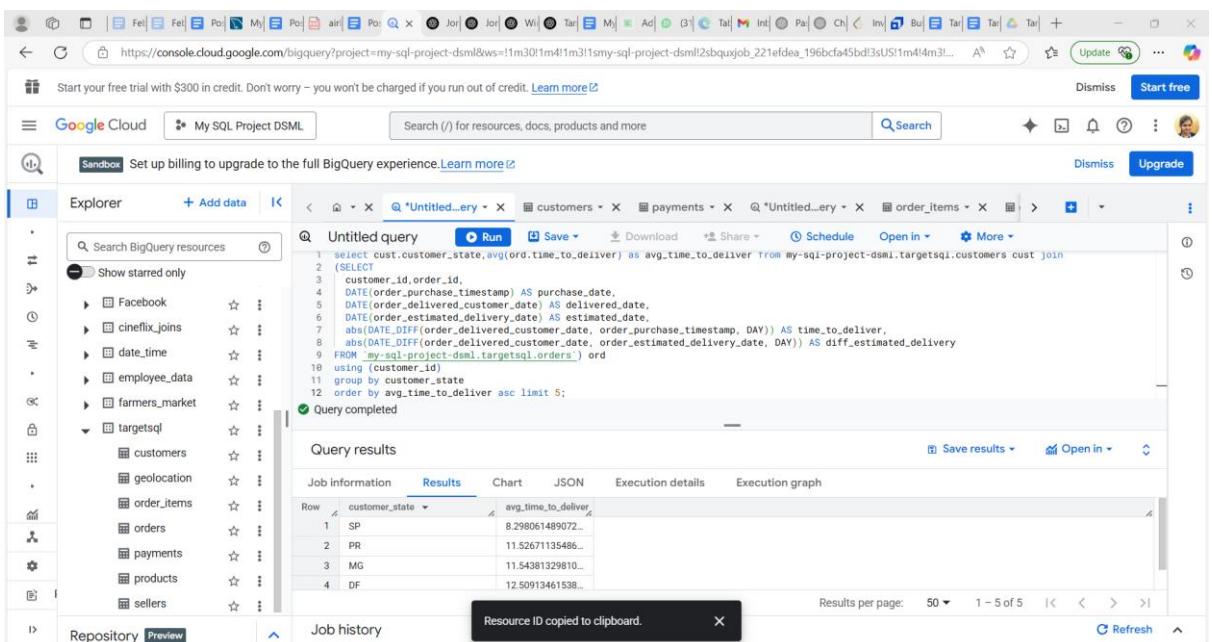
order by avg\_time\_to\_deliver asc limit 5;



The screenshot shows the Google Cloud BigQuery interface. The left sidebar is titled 'Explorer' and lists various datasets and tables under 'targetsqli'. The main area contains an 'Untitled query' window with the following SQL code:

```
1 select cust.customer_state,avg(ord.time_to_deliver) as avg_time_to_deliver from my-sql-project-dsml.targetsqli.customers cust join
2 (SELECT
3     customer_id,order_id,
4     DATE(order_purchase_timestamp) AS purchase_date,
5     DATE(order_delivered_customer_date) AS delivered_date,
6     DATE(order_estimated_delivery_date) AS estimated_date,
7     abs(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)) AS time_to_deliver,
8     abs(DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date, DAY)) AS diff_estimated_delivery
9     FROM `my-sql-project-dsml.orders` ) ord
10 using (customer_id)
11 group by customer_state
12 order by avg_time_to_deliver asc limit 5;
```

The status bar at the bottom indicates 'Query completed'.



The screenshot shows the Google Cloud BigQuery interface with the same setup as the previous one. The 'Results' tab is selected in the 'Query results' section. The results table displays the following data:

customer_state	avg_time_to_deliver
SP	8.298061489072...
PR	11.52671135486...
MG	11.54381329810...
DF	12.50913461538...

The status bar at the bottom indicates 'Results per page: 50 | 1 - 5 of 5'.

**Insights:-** SP is the state with the lowest average time delivery and RR is the state with the highest average time delivery

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

```
WITH delivery_diff AS (
    SELECT
        cust.customer_state AS state,
        DATE_DIFF(ord.order_estimated_delivery_date, ord.order_delivered_customer_date, DAY)
        AS days_early
    FROM `my-sql-project-dsml.targetsql.orders` ord
    JOIN `my-sql-project-dsml.targetsql.customers` cust
        USING (customer_id)
    WHERE ord.order_delivered_customer_date IS NOT NULL
        AND ord.order_estimated_delivery_date IS NOT NULL
)
, avg_early_delivery AS (
    SELECT
        state,
        ROUND(AVG(days_early), 2) AS avg_days_early
    FROM delivery_diff
    GROUP BY state
)
SELECT *
FROM avg_early_delivery
WHERE avg_days_early > 0
ORDER BY avg_days_early DESC
LIMIT 5;
```

The screenshot shows the Google Cloud BigQuery interface. On the left, the 'Explorer' sidebar lists various datasets and tables such as Facebook, cineflix\_joins, date\_time, employee\_data, farmers\_market, and targetsql. The main area displays an 'Untitled query' with the following SQL code:

```

1 WITH delivery_diff AS (
2     SELECT
3         cust.customer_state AS state,
4         DATE_DIFF(ord.order_estimated_delivery_date, ord.order_delivered_customer_date, DAY) AS days_early
5     FROM `my-sql-project-dsml.targetsql.orders` ord
6     JOIN `my-sql-project-dsml.targetsql.customers` cust
7     USING (customer_id)
8     WHERE ord.order_delivered_customer_date IS NOT NULL
9     AND ord.order_estimated_delivery_date IS NOT NULL

```

A note below the code states: "This query will process 8.32 MB when run." Below the code, the 'Query results' section shows a table with the following data:

Row	state	avg_days_early
1	AC	19.76
2	RO	19.13
3	AP	18.73
4	AM	18.61
5	RR	16.41

At the bottom right of the results table, there are buttons for 'Save results' and 'Open in'.

**Insights:-** Order delivery is really fast as compared to estimated delivery date in state AC.

Top 5 States:-

state	avg_days_early
1	AC
2	RO
3	AP
4	AM
5	RR

### I.Analysis based on the payments:

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

```

Select format_timestamp("%B",ord.order_purchase_timestamp) as Month,
pmt.payment_type,count(ord.order_id) as no_of_orders from my-sql-project-dsml.targetsql.orders ord
join my-sql-project-dsml.targetsql.payments pmt
using(order_id)
group by 1,2

```

The screenshot shows the Google Cloud BigQuery interface. On the left is the Explorer sidebar with various datasets and tables listed. The main area is titled "Untitled query" and contains the following SQL code:

```

1 Select format.timestamp('us',ord.order_purchase_timestamp) as Month, pmt.payment_type,count(ord.order_id) as no_of_orders from
my-sql-project-dsml.targetsqli.orders ord
2 join my-sql-project-dsml.targetsqli.payments pmt
3 using(order_id)
4 group by 1,2

```

A note below the code says "This query will process 8.47 MB when run." Below the code is a "Query results" table with the following data:

Month	payment_type	no_of_orders
February	credit_card	6609
April	credit_card	7301
September	credit_card	3286
October	voucher	318
October	credit_card	3778
October	UPI	1056
January	credit_card	6103

At the bottom of the results table, it says "Results per page: 50 1 - 50 of 50".

**Insights:-** Highest number of orders have been placed using credit card in the month of May.

- B.** Find the no. of orders placed on the basis of the payment installments that have been paid.

```

SELECT pmt.payment_installments,count(ord.order_id) FROM `my-sql-project-dsml.targetsqli.payments` pmt join my-sql-project-dsml.targetsqli.orders ord using (order_id) where payment_installments>=1 group by payment_installments

```

The screenshot shows the Google Cloud BigQuery interface. On the left is the Explorer sidebar with various datasets and tables listed. The main area is titled "Untitled query" and contains the following SQL code:

```

1 SELECT pmt.payment_installments,count(ord.order_id) FROM `my-sql-project-dsml.targetsqli.payments` pmt join my-sql-project-dsml.targetsqli.orders ord using (order_id) where payment_installments>=1 group by payment_installments

```

A note below the code says "This query will process 7.39 MB when run." Below the code is a "Query results" table with the following data:

payment_installments	f0_
1	52546
2	5239
3	12413
4	10461
5	7098
6	5328
7	3920
8	4268
9	644

At the bottom of the results table, it says "Results per page: 50 1 - 23 of 23".

**Insights:-** 52,546 orders have been placed where atleast 1 payment installment have been made.

### **Overall Insights:-**

1. The number of orders have increased each year
2. In the month of 2018, the order have mostly decreased over the months.
3. Highest number of orders have been placed during August, May and July Month, means, during mid-winter in Brazil(July,August) and also during cultural holidays (like Mother's Day widely celebrated in Brazil in May).
4. Highest number of orders have been placed in the afternoon (possibly because people take lunch breaks during work) and after 7 PM they are free from work when it's their Family time going out during dinner so they are free to do online or offline shopping and making decisions with their families. Lowest orders in September.
5. Highest number of orders have been placed in the Month of August in state SP , and lowest number of orders have been placed in the month of January in RR
6. Highest Order and Freight Value from State SP while the lowest from state RR.
7. SP is the state with the lowest average time delivery and RR is the state with the highest average time delivery
8. Order delivery is really fast as compared to estimated delivery date in state AC.
9. Highest number of orders have been placed using credit card in the month of May.
10. For 52,546 orders , atleast 1 payment installment has been made.

## **Recommendations:-**

1. Target should launch special discount and offers in the month of May especially on mother's day in Brazil in May.
2. Target should launch special discount and offers in the month of July and August as well when there is winter season especially on winter clothes and accessories .
3. Target should give good discounts on credit cards because it is the mostly used payment method especially in the months of May,July and August.
4. Target should work on reducing the delivery time for state RR because that is possibly the reason why this state has lowest number of orders, should consider launching more stores or warehouses , partner with more local delivery companies.
5. For state SP, Target can launch some rewards for it's most loyal customers as it is the state with highest order .
6. Target should give special discounts specifically between 7 AM-12 PM on online purchases especially on products for housewives or elderly people who don't commute to work to boost sales in the morning time as well.
7. Run digital ads and push notifications between **1–3 PM** (lunch) and **7–10 PM** (family & shopping time).
8. AC is the state with really fast delivery as compared to estimated delivery date, target should promote this as "early delivery guarantee" maximum in this state to increase number of orders.
9. For orders with higher value, credit card with EMI options should be more visible on the application and websites. Also no-cost EMI options should be given during mid year.
10. As there are lowest orders in September Month (when there is spring season in Brazil), more promotions and offers on clothing and accessories (like spring season essentials) or may be pre-holiday kits because from November holiday season starts can boost sales.
11. In January , more promotions and offers on products like new year planners can boost sales. Special Instagram giveaways or flash sales can work.