TARGET SQL PROJECT

1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset
    1. Data type of columns in a table
       Below is a list of data types of all the columns of 6 tables being used for our analysis of the TARGET SQL project

Ans) *Customers table*

customer_id – STRING
customer_unique_id – STRING
customer_zip_code_prefix – INTEGER
customer_city – STRING
customer_state – string

Field name	Type	Mode	Collation	Default value	Policy tags
customer_id	STRING	NULLABLE			
customer_unique_id	STRING	NULLABLE			
customer_zip_code_prefix	INTEGER	NULLABLE			
customer_city	STRING	NULLABLE			
customer_state	STRING	NULLABLE			

Order_items

order_id – STRING
order_item_id – INTEGER
product_id – STRING
seller_id – STRING
shipping_limit_date – TIMESTAMP
price – FLOAT
freight_value – FLOAT

Field name	Type	Mode	Collation	Default value	Policy tags
order_id	STRING	NULLABLE			
order_item_id	INTEGER	NULLABLE			
product_id	STRING	NULLABLE			
seller_id	STRING	NULLABLE			
shipping_limit_date	TIMESTAMP	NULLABLE			
price	FLOAT	NULLABLE			
freight_value	FLOAT	NULLABLE			

Orders table

order_id – STRING
customer_id - STRING
order_status - STRING
order_purchase_timestamp – TIMESTAMP
order_approved_at - TIMESTAMP
order_delivered_carrier_date - TIMESTAMP
order_delivered_customer_date - TIMESTAMP
order_estimated_delivery_date – TIMESTAMP

SCHEMA DETAILS PREVIEW

Field name	Type	Mode	Collation	Default value
order_id	STRING	NULLABLE		
customer_id	STRING	NULLABLE		
order_status	STRING	NULLABLE		
order_purchase_timestamp	TIMESTAMP	NULLABLE		
order_approved_at	TIMESTAMP	NULLABLE		
order_delivered_carrier_date	TIMESTAMP	NULLABLE		
order_delivered_customer_date	TIMESTAMP	NULLABLE		
order_estimated_delivery_date	TIMESTAMP	NULLABLE		

Payments table

order_id - STRING
payment_sequential - INTEGER
payment_type - STRING
payment_installments - INTEGER
payment_value – FLOAT

Filter Enter property name or value

Field name	Type	Mode	Collation	Default value	Policy tags
order_id	STRING	NULLABLE			
payment_sequential	INTEGER	NULLABLE			
payment_type	STRING	NULLABLE			
payment_installments	INTEGER	NULLABLE			
payment_value	FLOAT	NULLABLE			

Products table

product_id - STRING

product_category_name - STRING
product_name_lenght - INTEGER
product_description_lenght - INTEGER
product_photos_qty - INTEGER
product_weight_g - INTEGER
product_length_cm - INTEGER
product_height_cm - INTEGER
product_width_cm – INTEGER

☐	product_id	STRING	NULLABLE
☐	product_category	STRING	NULLABLE
☐	product_name_length	INTEGER	NULLABLE
☐	product_description_length	INTEGER	NULLABLE
☐	product_photos_qty	INTEGER	NULLABLE
☐	product_weight_g	INTEGER	NULLABLE
☐	product_length_cm	INTEGER	NULLABLE
☐	product_height_cm	INTEGER	NULLABLE
☐	product_width_cm	INTEGER	NULLABLE

Sellers table
seller_id - STRING
seller_zip_code_prefix - INTEGER
seller_city - STRING
seller_state – STRING

	Field name	Type	Mode	Collation	Default value	Policy tags
☐	seller_id	STRING	NULLABLE			
☐	seller_zip_code_prefix	INTEGER	NULLABLE			
☐	seller_city	STRING	NULLABLE			
☐	seller_state	STRING	NULLABLE			

2. Time period for which the data is given

Ans) Total time period for which the data is given is 2 years

Query :

```
Select
date_diff(max(date(order_purchase_timestamp)),min(date(order_purchase_timesta
mp)), YEAR) as time_period
from `target_sql.Orders`
```

```
1  Select date_diff(max(date(order_purchase_timestamp)),min(date(order_purchase_timestamp)),
   YEAR) as time_period
2  from `target_sql.Orders`
```

Press Alt+F1 for accessibility options

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

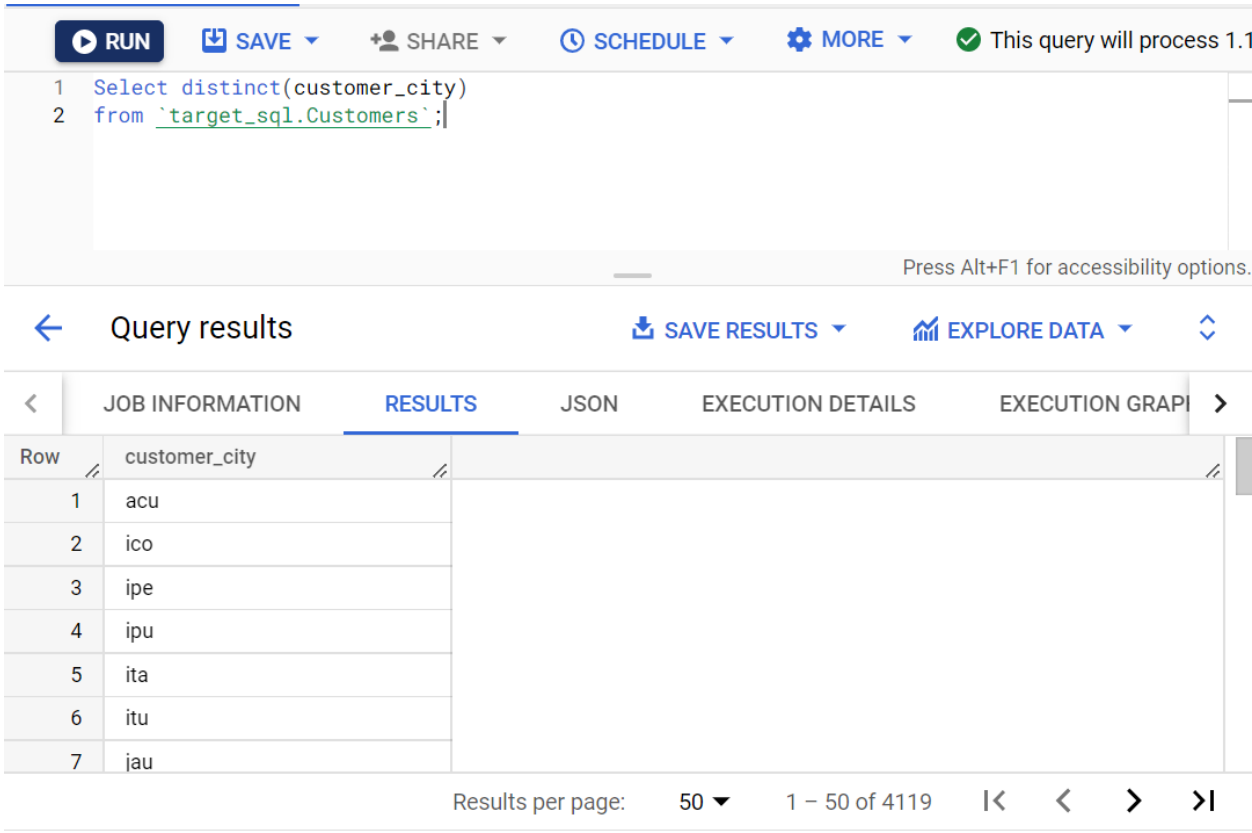
Row	time_period
1	2

3. Cities and states covered in the dataset

Ans) We have both customers and sellers cities & states we will need for our analysis. All of them are listed below

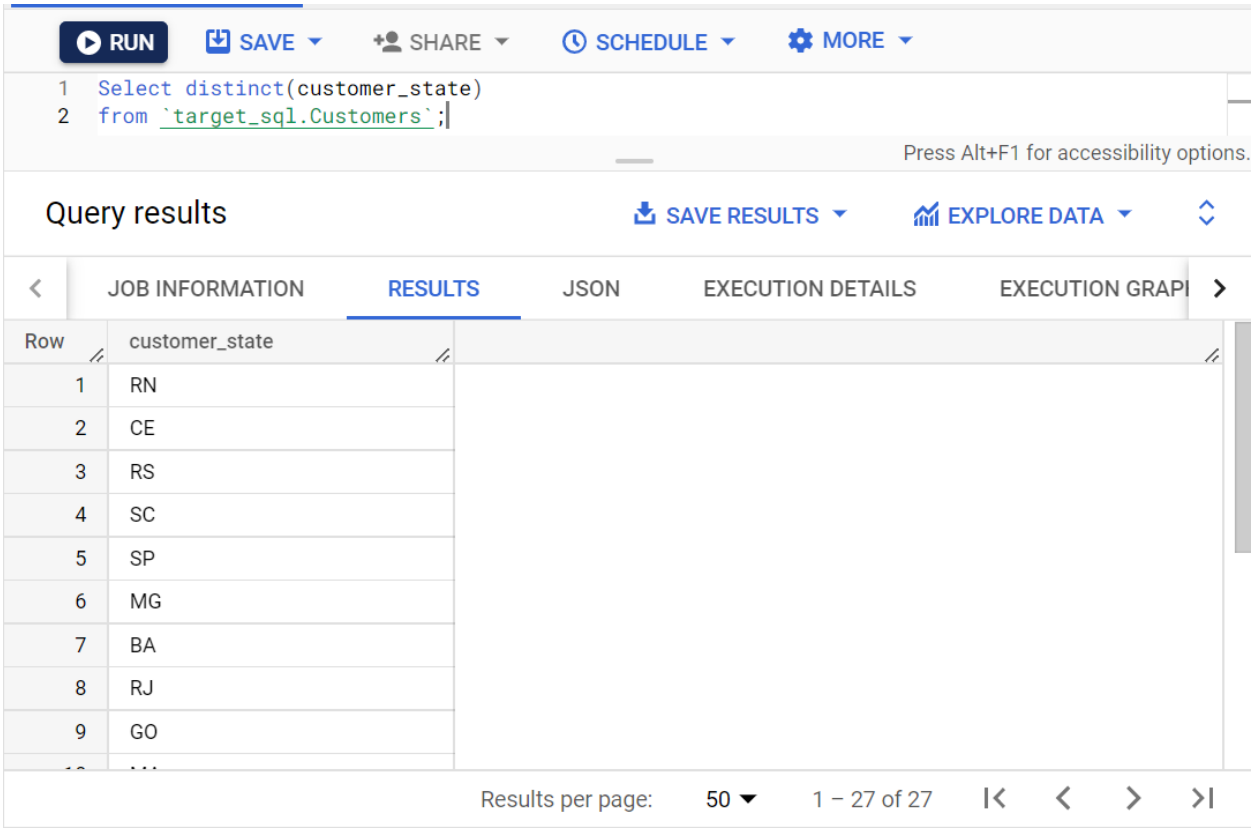
Customers cities

```
Select distinct(customer_city)
from `target_sql.Customers`;
```



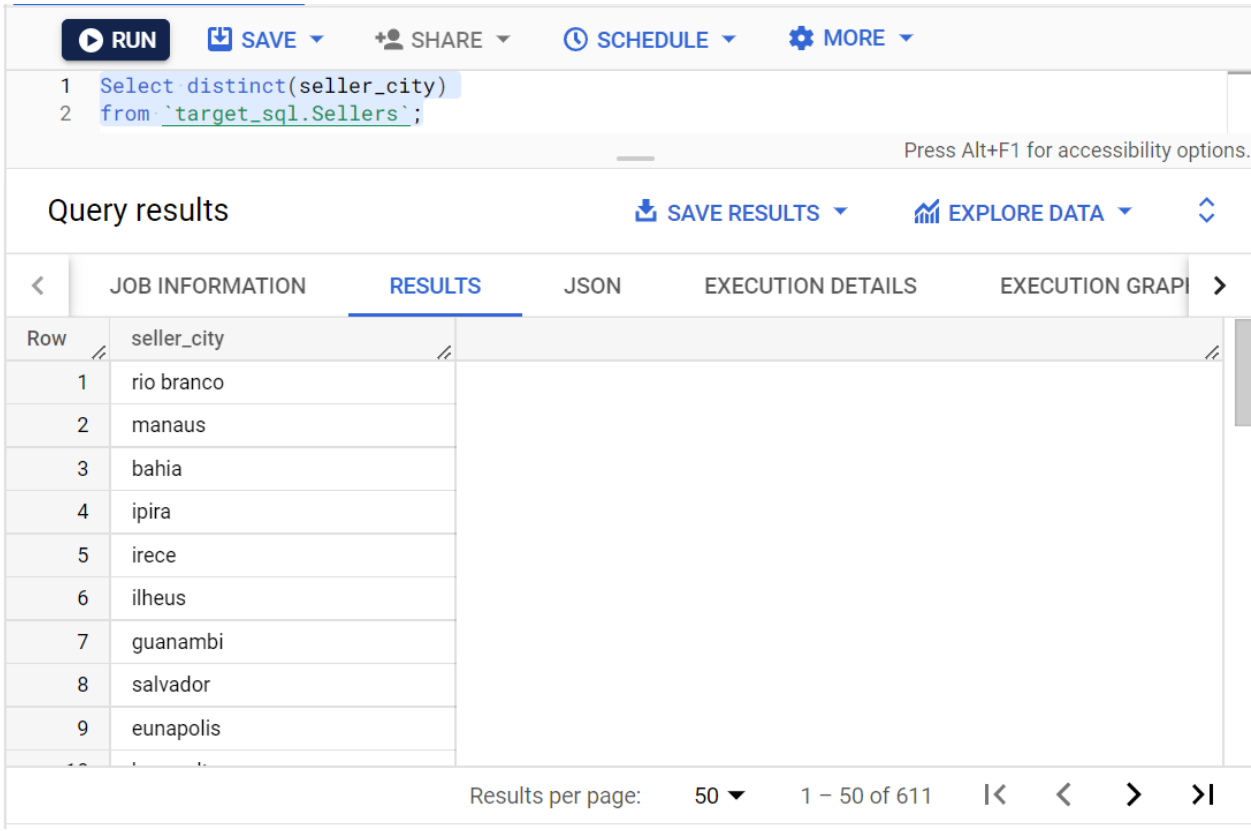
Customer states

```
Select distinct(customer_state)
from `target_sql.Customers`;
```



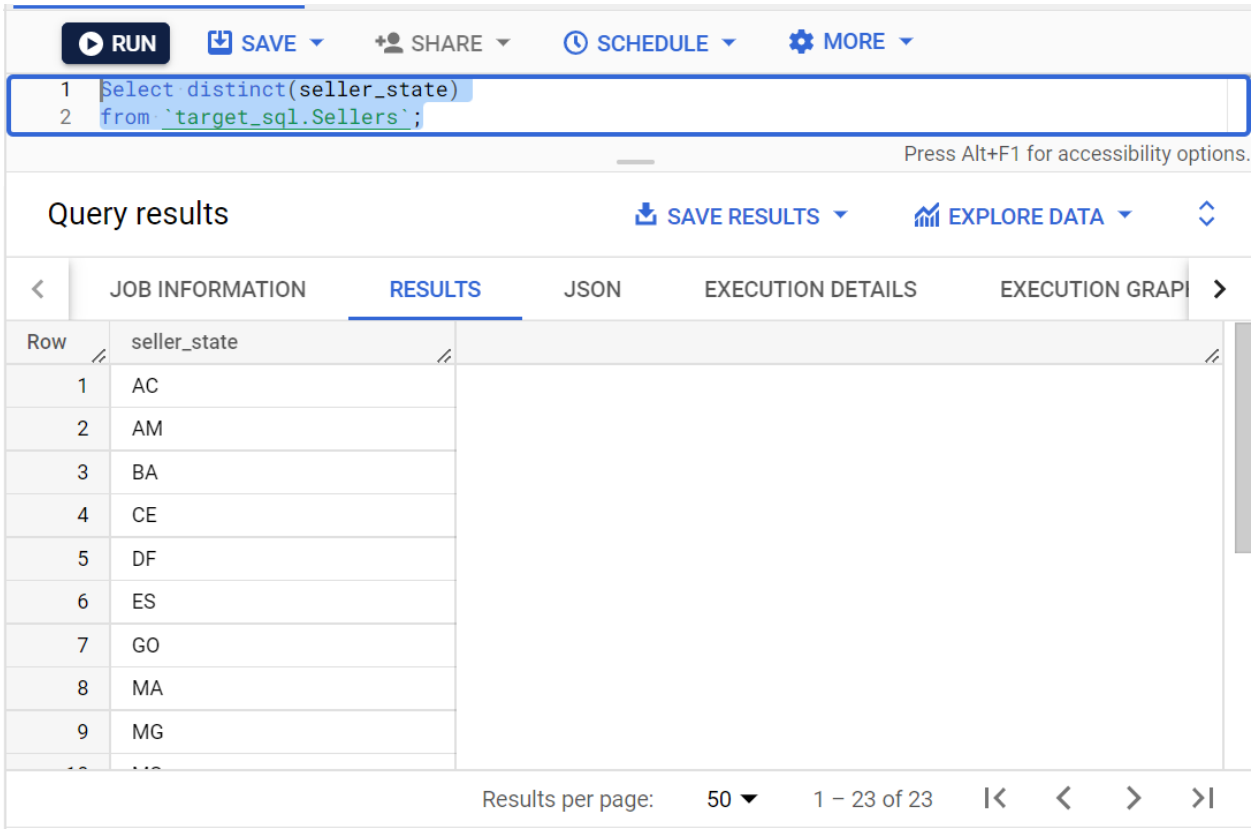
Seller cities

```
Select distinct(seller_city)
from `target_sql.Sellers`;
```



Seller states

```
Select distinct(seller_state)
from `target_sql.Sellers`;
```



2. In-depth Exploration :

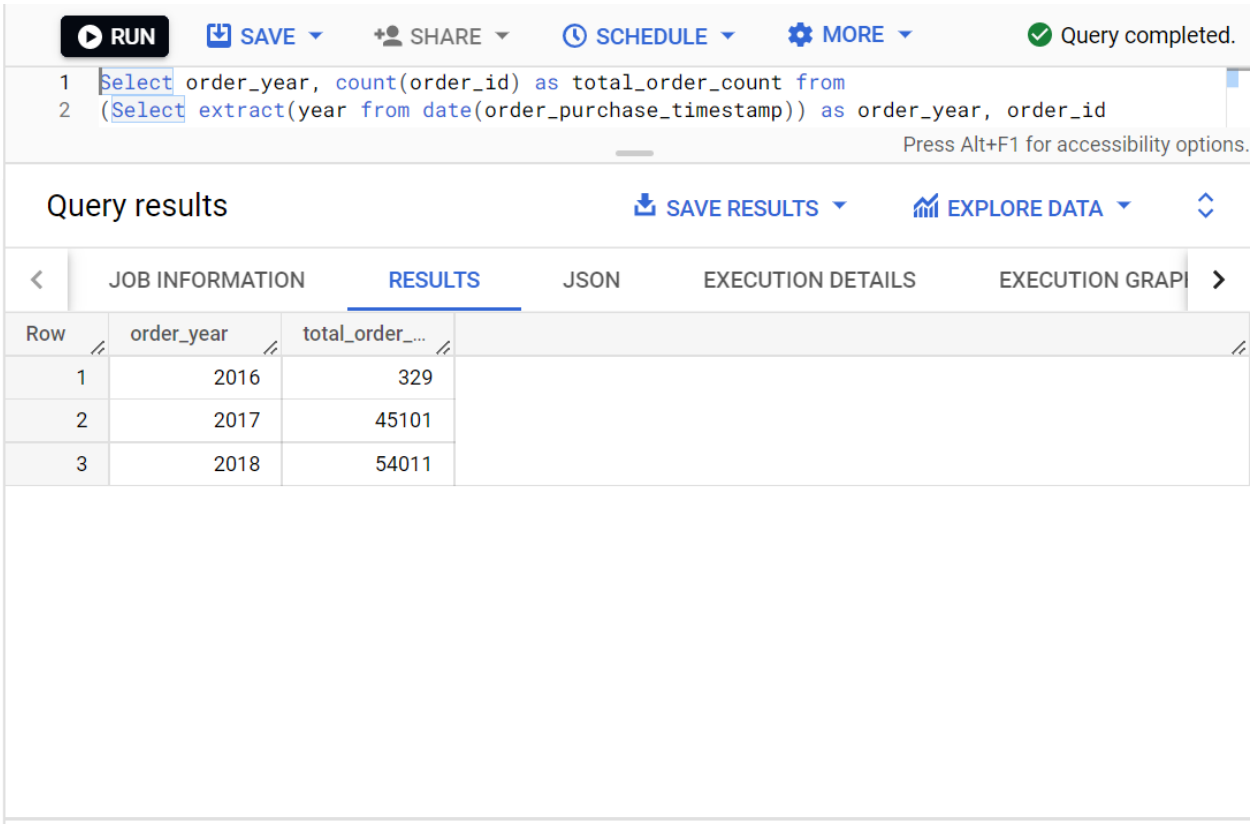
1. Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?

Ans) To analyze the growing trend on e-commerce in Brazil we can analyze the number of orders placed each year to see if there is an increase in online purchases. Although the no. of months in the year 2016 is only 3 months, it can still provide data for some growth trend analysis

Query :

```
Select order_year, count(order_id) as total_order_count from
(Select extract(year from date(order_purchase_timestamp)) as order_year, order_id
from `target_sql.Orders`) t
group by order_year
order by order_year
```

Result :



Inference : Here above in the results we can clearly see that there has been a very good increase in the number of orders year on year. This means the e-commerce trend is growing and people are buying more and more every year online.

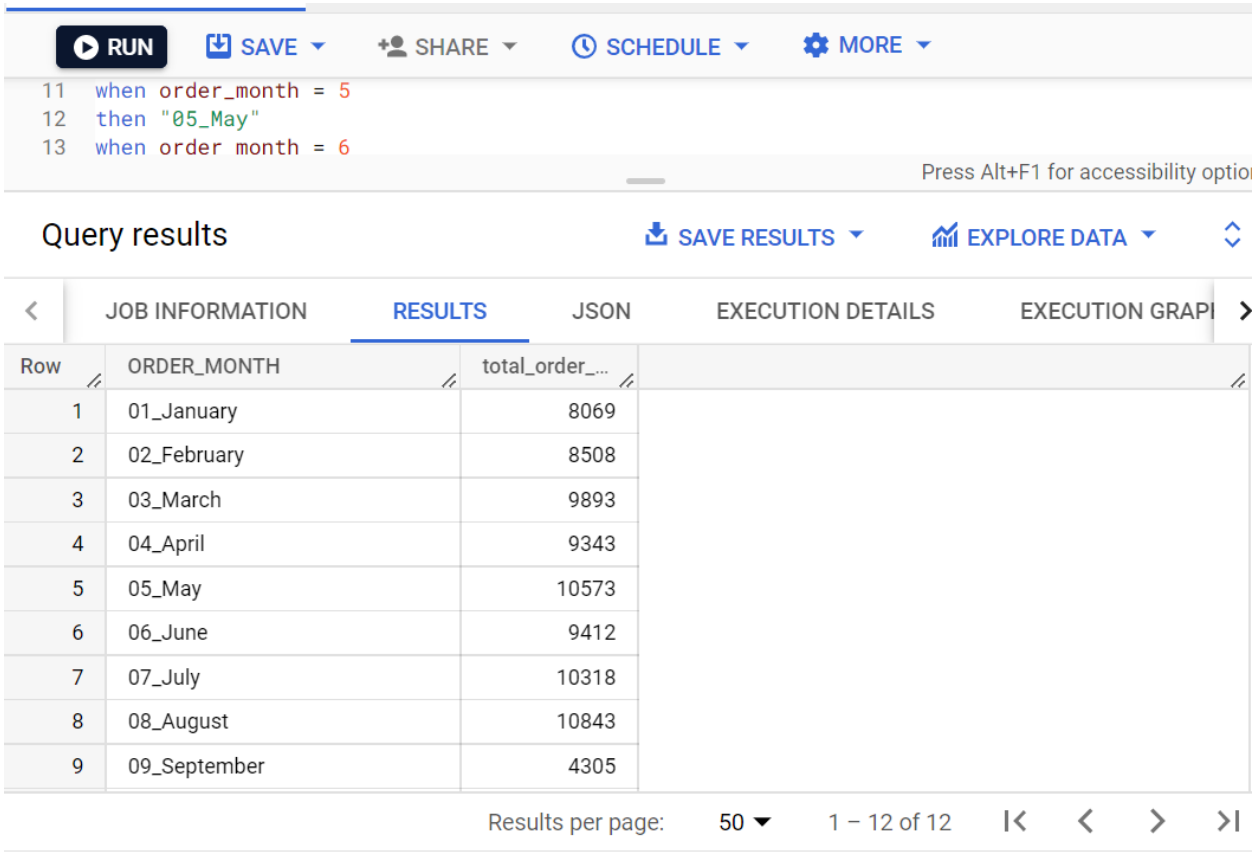
The month-wise data from Orders table can be used to observe which month/s have the highest/lowest purchases.

Query :

```
Select
case
when order_month = 1
then "01_January"
when order_month = 2
then "02_February"
when order_month = 3
then "03_March"
when order_month = 4
then "04_April"
when order_month = 5
then "05_May"
when order_month = 6
then "06_June"
when order_month = 7
```

```
then "07_July"
when order_month = 8
then "08_August"
when order_month = 9
then "09_September"
when order_month = 10
then "10_October"
when order_month = 11
then "11_November"
when order_month = 12
then "12_December"
end
as ORDER_MONTH, count(distinct(order_id)) as total_order_count from
(Select extract(month from date(order_purchase_timestamp)) as order_month,
order_id
from `target_sql.Orders`) t
group by ORDER_MONTH
order by ORDER_MONTH
```

Result:



```
when order_month = 12
then "12_December"
end
```

Press Alt+F1 for accessibility options.

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	ORDER_MONTH	total_order_...
4	04_April	9343
5	05_May	10573
6	06_June	9412
7	07_July	10318
8	08_August	10843
9	09_September	4305
10	10_October	4959
11	11_November	7544
12	12_December	5674

Results per page: 50 1 – 12 of 12

Inference : From the above result we can see that the months of May, July and August have the **top** 3 no. of orders count and September, November and December have the **lowest** 3 no. of orders count. Which means the e-commerce trend is the most active during the middle of the year and the lowest towards the end of the year

2. What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?

   Ans) Query:

```
Select count(order_id), time_of_the_day from
(
  Select
  case
  when extract(time from order_purchase_timestamp)
  between '05:30:00' and '06:59:00'
  then "Dawn"
  when extract(time from order_purchase_timestamp)
  between '07:00:00' and '11:59:00'
  then "Morning"
  when extract(time from order_purchase_timestamp)
  between '12:00:00' and '15:59:00'
  then "afternoon"
  when extract(time from order_purchase_timestamp)
```

```
  between '16:00:00' and '20:59:00'
  then "evening"
  else "night"
  end
  as time_of_the_day,
  order_id
  from target_sql.Orders
) t
group by time_of_the_day
```

RESULT:

```
1  Select count(order_id) as order_count, time_of_the_day from
2  (
3    Select
```

Press Alt+F1 for accessibility options.

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPI

Row	order_count	time_of_the_day
1	21619	Morning
2	21133	night
3	30684	evening
4	25409	afternoon
5	596	Dawn

Inference: From the results we can conclude that the e-commerce/online shoppers in Brazil are **most active** and tend to buy the most during the **evening** time and **very less active** during the **dawn**.

3. Evolution of E-commerce orders in the Brazil region:
   1. Get month on month orders by region, states

   Ans) QUERY:

```
 Select
case
when order_month = 1
```

```
then "01_January"
when order_month = 2
then "02_February"
when order_month = 3
then "03_March"
when order_month = 4
then "04_April"
when order_month = 5
then "05_May"
when order_month = 6
then "06_June"
when order_month = 7
then "07_July"
when order_month = 8
then "08_August"
when order_month = 9
then "09_September"
when order_month = 10
then "10_October"
when order_month = 11
then "11_November"
when order_month = 12
then "12_December"
end
as ORDER_MONTH,
count(distinct(order_id)) as total_order_count,
cust.customer_state
from
(Select extract(month from date(order_purchase_timestamp)) as order_month,
order_id, customer_id
from `target_sql.Orders`) t,
`target_sql.Customers` cust
where t.customer_id = cust.customer_id
group by cust.customer_state, ORDER_MONTH
order by cust.customer_state, ORDER_MONTH
```

RESULT:

```
then "09_September"
when order_month = 10
```

Press Alt+F1 for accessibility options.

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	ORDER_MONTH	total_order_count	customer_state
1	01_January	8	AC
2	02_February	6	AC
3	03_March	4	AC
4	04_April	9	AC
5	05_May	10	AC
6	06_June	7	AC
7	07_July	9	AC
8	08_August	7	AC
9	09_September	5	AC

Results per page: 50 1 – 50 of 322

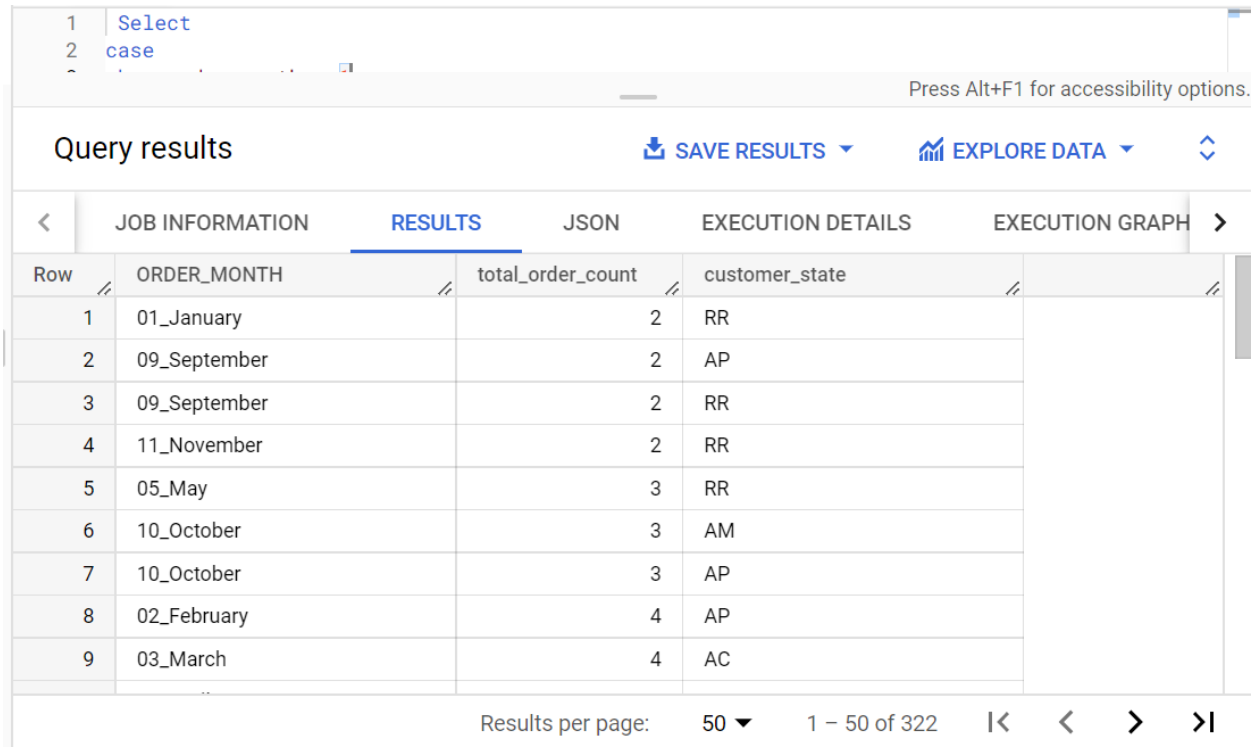
Inference : In the above results we can see the month and month no. of orders count from customers in each state in Brazil

Below query can be used to analyze the lowest sales/order count by month and state

```
Select
case
when order_month = 1
then "01_January"
when order_month = 2
then "02_February"
when order_month = 3
then "03_March"
when order_month = 4
then "04_April"
when order_month = 5
then "05_May"
when order_month = 6
then "06_June"
when order_month = 7
then "07_July"
when order_month = 8
```

```
then "08_August"
when order_month = 9
then "09_September"
when order_month = 10
then "10_October"
when order_month = 11
then "11_November"
when order_month = 12
then "12_December"
end
as ORDER_MONTH,
count(distinct(order_id)) as total_order_count,
cust.customer_state
from
(Select extract(month from date(order_purchase_timestamp)) as order_month,
order_id, customer_id
from `target_sql.Orders`) t,
`target_sql.Customers` cust
where t.customer_id = cust.customer_id
group by cust.customer_state, ORDER_MONTH
order by total_order_count,  ORDER_MONTH, cust.customer_state
```

RESULT :



2. How are customers distributed in Brazil

Ans) Query:

```
Select count(distinct(customer_id)) as customer_count, customer_state
from target_sql.Customers
group by customer_state
order by customer_count desc
```

Result:

```
1  Select count(distinct(customer_id)) as customer_count, customer_state
2  from target_sql.Customers
```

Press Alt+F1 for accessibility options.

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_c...	customer_state
1	41746	SP
2	12852	RJ
3	11635	MG
4	5466	RS
5	5045	PR
6	3637	SC
7	3380	BA
8	2140	DF
9	2033	ES

Inference: In the above result we can see the no. of customers distributed in each and every state in Brazil from highest to lowest

States with lowest customers in Brazil

QUERY

```
Select count(distinct(customer_id)) as customer_count, customer_state
from target_sql.Customers
group by customer_state
order by customer_count
```

RESULT:

```
4  order by customer_count
5
```

Press Alt+F1 for accessibility options.

Query results SAVE RESULTS EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_c...	customer_state
1	46	RR
2	68	AP
3	81	AC
4	148	AM
5	253	RO
6	280	TO
7	350	SE
8	413	AL
9	485	RN

Results per page: 50 1 – 27 of 27

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

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

Ans) Query:

```
Select sum(price + freight_value)/(select sum(price + freight_value)
from target_sql.Order_items ot,
target_sql.Orders o
where ot.order_id = o.order_id
and extract(month from date(o.order_purchase_timestamp)) in (1,2,3,4,5,6,7,8)
and extract(year from date(o.order_purchase_timestamp)) in (2018)) as
pct_increase
from target_sql.Order_items ot,
target_sql.Orders o
where ot.order_id = o.order_id
and extract(month from date(o.order_purchase_timestamp)) in (1,2,3,4,5,6,7,8)
and extract(year from date(o.order_purchase_timestamp)) in (2017)
```

RESULT:

```
1  Select sum(price + freight_value)/(select sum(price + freight_value)
2  from target_sql.Order_items ot,
3  target_sql.Orders o
```

Press Alt+F1 for accessibility options.

Query results SAVE RESULTS EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	pct_increase
1	0.41768463...

Inference : From the above result we can see that there is a **41.77%** increase in the cost of orders from year 2017 to year 2018. Price and freight value for the orders have been considered to calculate the same

2. Mean & Sum of price and freight value by customer state

Ans) QUERY:

```
Select round(AVG(price + freight_value),2) as Mean_cost,
round(sum(price + freight_value),2) as Total_cost,
cust.customer_state
from target_sql.Order_items ot,
target_sql.Customers cust,
target_sql.Orders o
where ot.order_id = o.order_id
and o.customer_id = cust.customer_id
group by cust.customer_state
order by Total_cost desc
```

RESULT:

```
9  group by cust.customer_state
10 order by Total_cost desc
11
```

Press Alt+F1 for accessibility options.

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	Mean_cost	Total_cost	customer_state
1	124.8	5921678.12	SP
2	146.08	2129681.98	RJ
3	141.38	1856161.49	MG
4	142.07	885826.76	RS
5	139.54	800935.44	PR
6	160.97	611506.67	BA
7	146.12	610213.6	SC
8	146.81	353229.44	DF
9	149.04	347706.93	GO

Results per page: 50 1 – 27 of 27

5. Analysis on sales, freight and delivery time
    1. Calculate days between purchasing, delivering and estimated delivery
    2. Create columns:

- time_to_delivery = order_purchase_timestamp-order_delivered_customer_date
- diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

QUERY:

```
SELECT order_id,
date_diff(date(order_delivered_customer_date),date(order_purchase_timestamp),DAY)
as time_to_delivery,
date_diff(date(order_estimated_delivery_date),date(order_delivered_customer_date),DAY)
as diff_estimated_delivery
from target_sql.Orders
```

RESULT:

```
SELECT order_id,
date_diff(date(order_delivered_customer_date),date(order_purchase_timestamp),DAY)
as time_to_delivery,
```

Press Alt+F1 for accessibility options.

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	order_id	time_to_delivery	diff_estimated_delivery
1	1950d777989f6a877539f5379...	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	31	29
3	65d1e226dfaeb8cdc42f66542...	36	17
4	635c894d068ac37e6e03dc54e...	31	2
5	3b97562c3aee8bdedcb5c2e45...	33	1
6	68f47f50f04c4cb6774570cfde...	30	2
7	276e9ec344d3bf029ff83a161c...	44	-4
8	54e1a3c2b97fb0809da548a59...	41	-4
9	fd04fa4105ee8045f6a0139ca5...	37	-1

Results per page: 50 1 – 50 of 99441

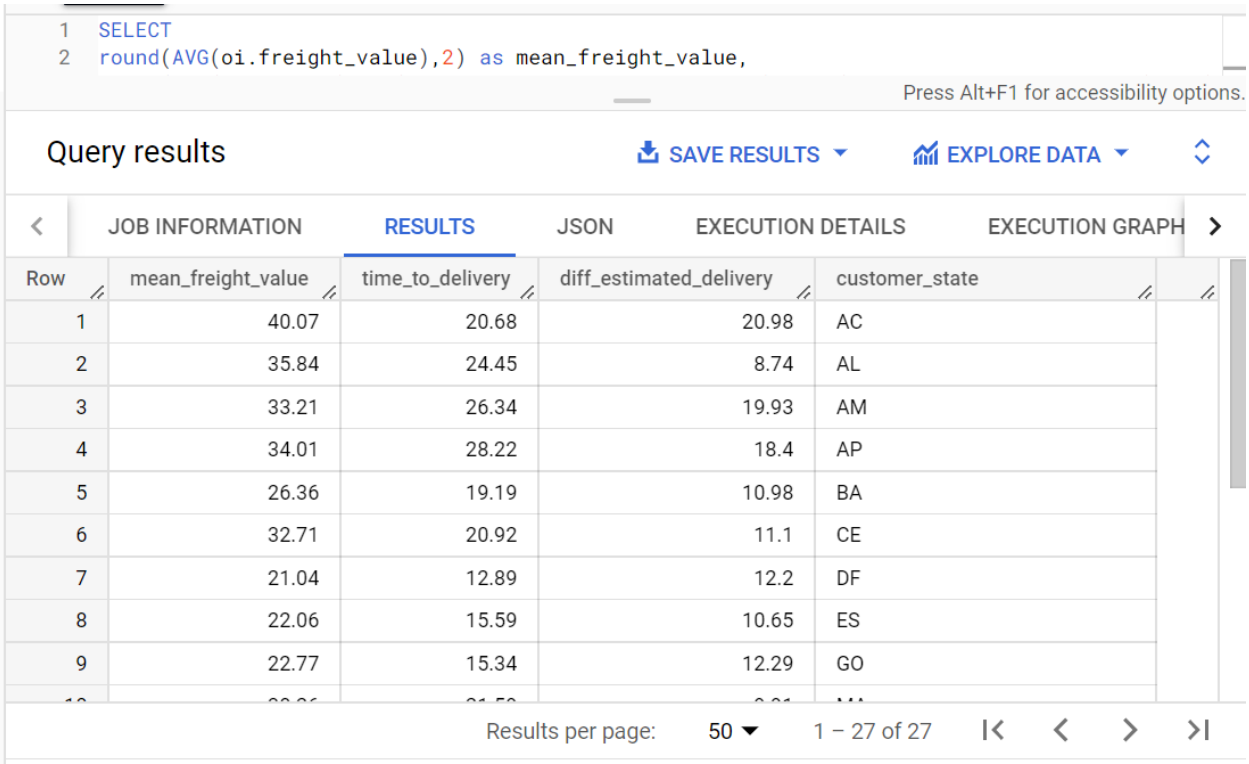
3. Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

Ans) QUERY:

```
SELECT
round(AVG(oi.freight_value),2) as mean_freight_value,
round(AVG(date_diff(date(order_delivered_customer_date),date(order_purchase_timestamp),DAY)),2)
as time_to_delivery,
round(AVG(date_diff(date(order_estimated_delivery_date),date(order_delivered_customer_date),DAY)),2)
as diff_estimated_delivery,
cust.customer_state
from target_sql.Orders o,
target_sql.Customers cust,
target_sql.Order_items oi
where o.customer_id = cust.customer_id
and o.order_id = oi.order_id
```

```
GROUP BY cust.customer_state
order by cust.customer_state
```

RESULT:



```
1  SELECT
2  round(AVG(oi.freight_value),2) as mean_freight_value,
```

Press Alt+F1 for accessibility options.

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	mean_freight_value	time_to_delivery	diff_estimated_delivery	customer_state
1	40.07	20.68	20.98	AC
2	35.84	24.45	8.74	AL
3	33.21	26.34	19.93	AM
4	34.01	28.22	18.4	AP
5	26.36	19.19	10.98	BA
6	32.71	20.92	11.1	CE
7	21.04	12.89	12.2	DF
8	22.06	15.59	10.65	ES
9	22.77	15.34	12.29	GO

Results per page: 50 1 – 27 of 27

4. Sort the data to get the following:

1. Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5

QUERY:

Top 5 states with highest freight value

```
SELECT
round(AVG(oi.freight_value),2) as mean_freight_value,
round(AVG(date_diff(date(order_delivered_customer_date),date(order_purchase
_timestamp),DAY)),2)
as time_to_delivery,
round(AVG(date_diff(date(order_estimated_delivery_date),date(order_delivered_
customer_date),DAY)),2)
as diff_estimated_delivery,
cust.customer_state
```

```
from target_sql.Orders o,
target_sql.Customers cust,
target_sql.Order_items oi
where o.customer_id = cust.customer_id
and o.order_id = oi.order_id
GROUP BY cust.customer_state
order by mean_freight_value desc
limit 5
```

RESULT :

```
1  SELECT
2  round(AVG(oi.freight_value),2) as mean_freight_value,
3  round(AVG(date_diff(date(order_delivered_customer_date),date(order_purchase_timestamp),DAY)
   ),2)
```

Press Alt+F1 for accessibility options.

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	mean_freight_value	time_to_delivery	diff_estimated_delivery	customer_state
1	42.98	28.17	18.33	RR
2	42.72	20.55	13.04	PB
3	41.07	19.66	20.04	RO
4	40.07	20.68	20.98	AC
5	39.15	19.32	11.53	PI

Top 5 states with lowest freight value

```
SELECT
round(AVG(oi.freight_value),2) as mean_freight_value,
round(AVG(date_diff(date(order_delivered_customer_date),date(order_purchase
_timestamp),DAY)),2)
as time_to_delivery,
round(AVG(date_diff(date(order_estimated_delivery_date),date(order_delivered_
customer_date),DAY)),2)
as diff_estimated_delivery,
cust.customer_state
```

```
from target_sql.Orders o,
target_sql.Customers cust,
target_sql.Order_items oi
where o.customer_id = cust.customer_id
and o.order_id = oi.order_id
GROUP BY cust.customer_state
order by mean_freight_value
limit 5
```

RESULT :

```
1  SELECT
2  round(AVG(oi.freight_value),2) as mean_freight_value,
3  round(AVG(date_diff(date(order_delivered_customer_date),date(order_purchase_timestamp),DAY)
   ),2)
4  as time_to_delivery
```

Press Alt+F1 for accessibility options.

Query results SAVE RESULTS EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	mean_freight_value	time_to_delivery	diff_estimated_delivery	customer_state
1	15.15	8.66	11.21	SP
2	20.53	11.89	13.49	PR
3	20.63	11.92	13.34	MG
4	20.96	15.07	12.01	RJ
5	21.04	12.89	12.2	DF

2. Top 5 states with highest average time to delivery

QUERY:

```
SELECT
round(AVG(oi.freight_value),2) as mean_freight_value,
round(AVG(date_diff(date(order_delivered_customer_date),date(order_purchase_timesta
mp),DAY)),2)
as time_to_delivery,
round(AVG(date_diff(date(order_estimated_delivery_date),date(order_delivered_custom
er_date),DAY)),2)
as diff_estimated_delivery,
cust.customer_state
from target_sql.Orders o,
target_sql.Customers cust,
```

```
target_sql.Order_items oi
where o.customer_id = cust.customer_id
and o.order_id = oi.order_id
GROUP BY cust.customer_state
order by time_to_delivery desc
limit 5
```

RESULT:

```
1  SELECT
2  round(AVG(oi.freight_value),2) as mean_freight_value,
```

Press Alt+F1 for accessibility options.

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	mean_freight_value	time_to_delivery	diff_estimated_delivery	customer_state
1	34.01	28.22	18.4	AP
2	42.98	28.17	18.33	RR
3	33.21	26.34	19.93	AM
4	35.84	24.45	8.74	AL
5	35.83	23.7	14.25	PA

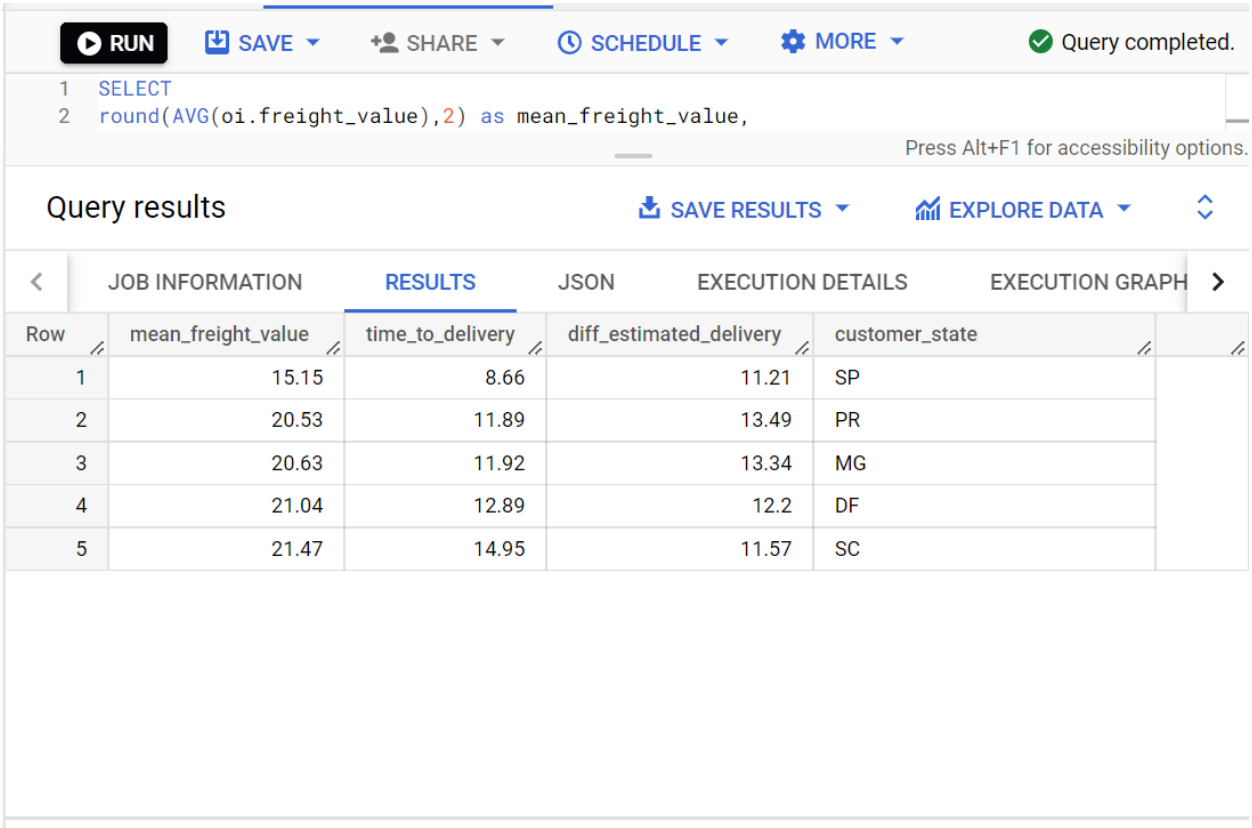
Top 5 states with lowest average time to delivery

QUERY:

```
SELECT
round(AVG(oi.freight_value),2) as mean_freight_value,
round(AVG(date_diff(date(order_delivered_customer_date),date(order_purchase_timesta
mp),DAY)),2)
as time_to_delivery,
round(AVG(date_diff(date(order_estimated_delivery_date),date(order_delivered_custom
er_date),DAY)),2)
as diff_estimated_delivery,
cust.customer_state
from target_sql.Orders o,
target_sql.Customers cust,
```

```
target_sql.Order_items oi
where o.customer_id = cust.customer_id
and o.order_id = oi.order_id
GROUP BY cust.customer_state
order by time_to_delivery
limit 5
```

RESULT:



Row	mean_freight_value	time_to_delivery	diff_estimated_delivery	customer_state
1	15.15	8.66	11.21	SP
2	20.53	11.89	13.49	PR
3	20.63	11.92	13.34	MG
4	21.04	12.89	12.2	DF
5	21.47	14.95	11.57	SC

3. Top 5 states where delivery is fastest compared to estimated date

Ans) Here if the estimated delivery date – actual delivery date is more it means it is a faster delivery compared to estimated delivery date

QUERY:

```
SELECT
round(AVG(oi.freight_value),2) as mean_freight_value,
round(AVG(date_diff(date(order_delivered_customer_date),date(order_purchase_timesta
mp),DAY)),2)
as time_to_delivery,
round(AVG(date_diff(date(order_estimated_delivery_date),date(order_delivered_custom
er_date),DAY)),2)
as diff_estimated_delivery,
```

```
cust.customer_state
from target_sql.Orders o,
target_sql.Customers cust,
target_sql.Order_items oi
where o.customer_id = cust.customer_id
and o.order_id = oi.order_id
GROUP BY cust.customer_state
order by diff_estimated_delivery desc
limit 5
```

RESULT:

```
1  SELECT
2  round(AVG(oi.freight_value),2) as mean_freight_value,
```

Press Alt+F1 for accessibility options.

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	mean_freight_value	time_to_delivery	diff_estimated_delivery	customer_state
1	40.07	20.68	20.98	AC
2	41.07	19.66	20.04	RO
3	33.21	26.34	19.93	AM
4	34.01	28.22	18.4	AP
5	42.98	28.17	18.33	RR

Top 5 states where delivery is not so fast compared to estimated date

Ans) Here if the estimated delivery date – actual delivery date is less it means it is a not so fast/slower delivery compared to estimated delivery date

QUERY:

```
SELECT
round(AVG(oi.freight_value),2) as mean_freight_value,
round(AVG(date_diff(date(order_delivered_customer_date),date(order_purchase_timestamp),DAY)),2)
as time_to_delivery,
```

```
round(AVG(date_diff(date(order_estimated_delivery_date),date(order_delivered_custom
er_date),DAY)),2)
as diff_estimated_delivery,
cust.customer_state
from target_sql.Orders o,
target_sql.Customers cust,
target_sql.Order_items oi
where o.customer_id = cust.customer_id
and o.order_id = oi.order_id
GROUP BY cust.customer_state
order by diff_estimated_delivery
limit 5
```

RESULT:

```
1  SELECT
2  round(AVG(oi.freight_value),2) as mean_freight_value,
```

Press Alt+F1 for accessibility options.

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	mean_freight_value	time_to_delivery	diff_estimated_delivery	customer_state
1	35.84	24.45	8.74	AL
2	38.26	21.59	9.91	MA
3	36.65	21.42	10.0	SE
4	22.06	15.59	10.65	ES
5	26.36	19.19	10.98	BA

6. Payment type analysis:

1. Month over Month count of orders for different payment types

Ans) QUERY:

```
SELECT
case
when order_month = 1
then "01_January"
when order_month = 2
```

```
then "02_February"
when order_month = 3
then "03_March"
when order_month = 4
then "04_April"
when order_month = 5
then "05_May"
when order_month = 6
then "06_June"
when order_month = 7
then "07_July"
when order_month = 8
then "08_August"
when order_month = 9
then "09_September"
when order_month = 10
then "10_October"
when order_month = 11
then "11_November"
when order_month = 12
then "12_December"
end
as ORDER_MONTH,
count(distinct(o.order_id)) as total_order_count,
payment_type
from
(Select extract(month from date(order_purchase_timestamp)) as
order_month,
order_id, customer_id
from `target_sql.Orders`) t,
`target_sql.Customers` cust,
`target_sql.Orders` o,
`target_sql.Payments` pay
where t.customer_id = cust.customer_id
and o.customer_id = cust.customer_id
and o.order_id = pay.order_id
group by ORDER_MONTH,payment_type
order by ORDER_MONTH, payment_type
```

RESULT :

```
23  when order_month = 11
24  then "11_November"
```

Press Alt+F1 for accessibility options.

Query results SAVE RESULTS EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	ORDER_MONTH	total_order_count	payment_type
1	01_January	1715	UPI
2	01_January	6093	credit_card
3	01_January	118	debit_card
4	01_January	337	voucher
5	02_February	1723	UPI
6	02_February	6582	credit_card
7	02_February	82	debit_card
8	02_February	288	voucher
9	03_March	1942	UPI

Results per page: 50 1 – 50 of 50

2. Distribution of payment installments and count of orders

Ans) QUERY:

```
SELECT
count(distinct(o.order_id)) as total_orders,
payment_installments
from
`target_sql.Customers` cust,
`target_sql.Orders` o,
`target_sql.Payments` pay
where o.customer_id = cust.customer_id
and o.order_id = pay.order_id
group by payment_installments
order by payment_installments
```

RESULT:

```
1  Select
2  count(distinct(o.order_id)) as total_orders,
```

Press Alt+F1 for accessibility options.

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	total_orders	payment_installments
1	2	0
2	49060	1
3	12389	2
4	10443	3
5	7088	4
6	5234	5
7	3916	6
8	1623	7
9	4253	8

Results per page: 50 1 – 24 of 24

SORT on no.of orders for each payment installment

QUERY:

```
SELECT
count(distinct(o.order_id)) as total_orders,
payment_installments
from
`target_sql.Customers` cust,
`target_sql.Orders` o,
`target_sql.Payments` pay
where o.customer_id = cust.customer_id
and o.order_id = pay.order_id
group by payment_installments
order by total_orders
```

RESULT:

```
11  order by total_orders
12
```

Press Alt+F1 for accessibility options.

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	total_orders	payment_in...
1	1	23
2	1	22
3	2	0
4	3	21
5	5	16
6	8	17
7	15	14
8	16	13
9	17	20

Results per page: 50 1 – 24 of 24