

## TARGET BUSINESS CASE STUDY

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

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

Field name	Type
customer_id	STRING
customer_unique_id	STRING
customer_zip_code_prefix	INTEGER
customer_city	STRING
customer_state	STRING

### 2.Get the time range between which the orders were placed.

Query :

```
SELECT  
min(order_purchase_timestamp) as range_from,  
max(order_purchase_timestamp) as range_to  
FROM `circular-matrix-406015.Target.orders`
```

Output:

<pre>1 SELECT min(order_purchase_timestamp) AS range_from, 2          max(order_purchase_timestamp) AS range_to 3 FROM `circular-matrix-406015.Target.orders`</pre>			Press Alt+F1 for Accessibility Options		
Query results			SAVE RESULTS ▾ EXPLORE DATA ▾ ↕		
<	JOB INFORMATION	RESULTS	CHART	PREVIEW	JSON
	EXECUTION DETAILS	ED	>		
Row	range_from ▾	range_to ▾			
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC			

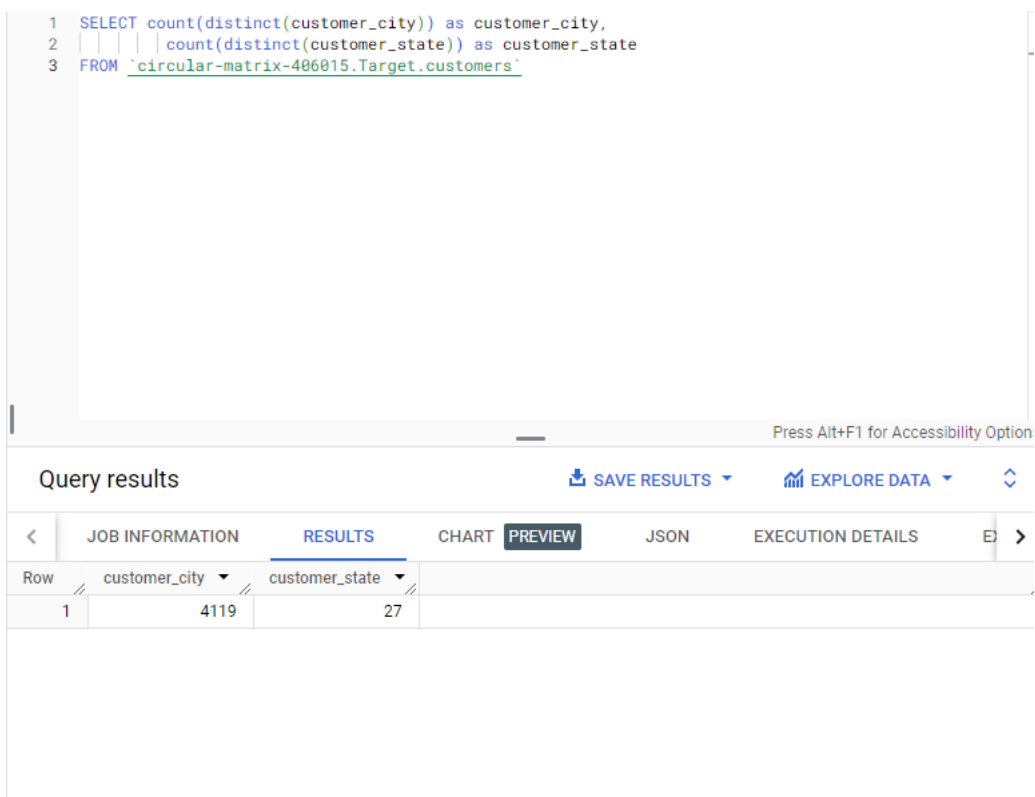
## TARGET BUSINESS CASE STUDY

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

#### Query :

```
SELECT count(distinct(customer_city)) as customer_city,  
       count(distinct(customer_state)) as customer_state  
FROM `circular-matrix-406015.Target.customers`
```

#### Output:



The screenshot shows a SQL query execution interface. The query is displayed in a text area at the top. Below the query, there is a section titled "Query results" with a table of results. The table has two columns: "customer\_city" and "customer\_state". The first row shows the counts for these columns.

```
1 SELECT count(distinct(customer_city)) as customer_city,  
2 | | | | count(distinct(customer_state)) as customer_state  
3 FROM `circular-matrix-406015.Target.customers`
```

Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS CHART PREVIEW JSON EXECUTION DETAILS

Row	customer_city	customer_state
1	4119	27

#### Insights

***There are over 5000+ cities in brazil, but order is coming from only 4119 cities***

## TARGET BUSINESS CASE STUDY

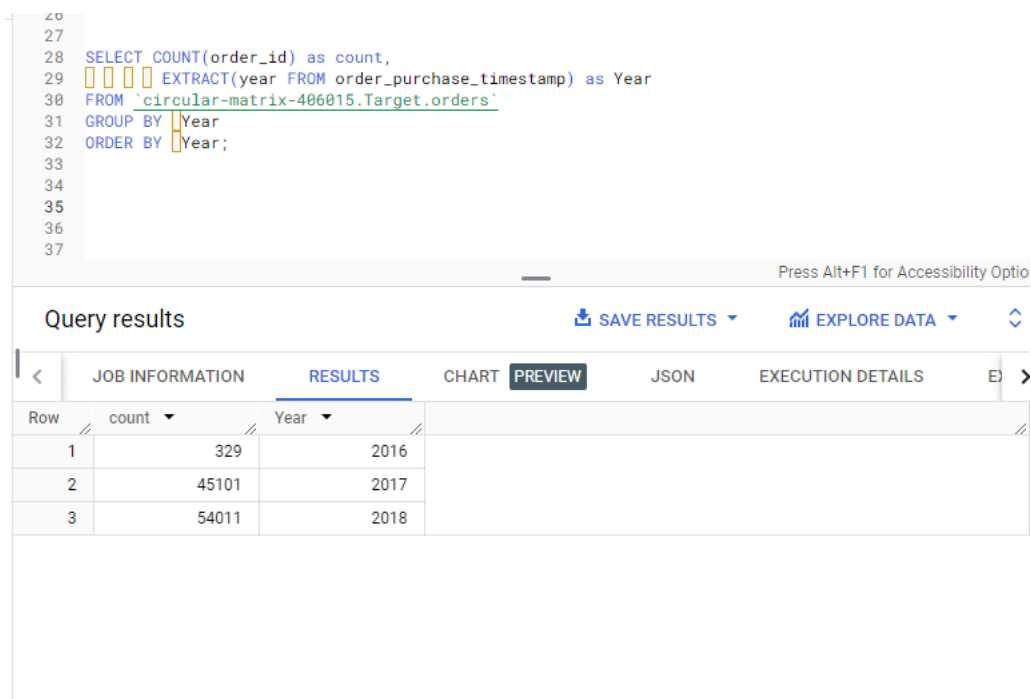
### In-depth Exploration

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

#### Query

```
SELECT COUNT(order_id) as count,  
       EXTRACT(year FROM order_purchase_timestamp) as Year  
FROM `circular-matrix-406015.Target.orders`  
GROUP BY Year  
ORDER BY Year;
```

#### Output:



```
27  
28 SELECT COUNT(order_id) as count,  
29      EXTRACT(year FROM order_purchase_timestamp) as Year  
30 FROM `circular-matrix-406015.Target.orders`  
31 GROUP BY Year  
32 ORDER BY Year;
```

Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS CHART PREVIEW JSON EXECUTION DETAILS

Row	count	Year
1	329	2016
2	45101	2017
3	8910	2018

#### Insights

On, 2016 target has only 329 in order count, but in next year, it has been drastically increased to 45k order count, then again it was a drop to 8910 order in 2018.

May be because of offers, promotion etc.

## TARGET BUSINESS CASE STUDY

**2.Can we see some kind of monthly seasonality in terms of the no. of orders being placed?**


### Query




```
SELECT
EXTRACT(month FROM order_purchase_timestamp) as month,
EXTRACT(year FROM order_purchase_timestamp) as year,
COUNT(order_id) as count

FROM `circular-matrix-406015.Target.orders`
GROUP BY month,year
order by year,month;
```





### Output

```
1 SELECT EXTRACT(month FROM order_purchase_timestamp) as month,
2 EXTRACT(year FROM order_purchase_timestamp) as year,
3 COUNT(order_id) as count
4
5 FROM `circular-matrix-406015.Target.orders`
6 GROUP BY month,year
7 order by year,month;
8 |
```

Processing location: asia-south1  Press Alt+F1 for Accessibility Opt

Query results  SAVE RESULTS  EXPLORE DATA 

	month	year	count
1	9	2016	4
2	10	2016	324
3	12	2016	1
4	1	2017	800
5	2	2017	1780
6	3	2017	2682
7	4	2017	2404
8	5	2017	3700
9	6	2017	3245

Results per page: 50 1 – 25 of 25    

### Insights

Every quarter, order has been increasing steadily, but in every last quarter, there is a fall of order. Best quarter will be 3<sup>rd</sup>.

## TARGET BUSINESS CASE STUDY

3. 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

**Query :**

```
SELECT
CASE
  WHEN EXTRACT(hour FROM order_purchase_timestamp) < 7 THEN
'Dawn'
  WHEN EXTRACT(hour FROM order_purchase_timestamp) between 7
and 12 THEN 'Mornings'
  WHEN EXTRACT(hour FROM order_purchase_timestamp) between 13
and 18 THEN 'Afternoon'
  ELSE 'Night'
END AS Time_of_day, count(order_id) as count_of_order
from `circular-matrix-406015.Target.orders`
group by time_of_day
order by count_of_order desc ;
```

**Output:**

```
1 SELECT
2 CASE
3   WHEN EXTRACT(hour FROM order_purchase_timestamp) < 7 THEN 'Dawn'
4   WHEN EXTRACT(hour FROM order_purchase_timestamp) between 7 and 12 THEN 'Mornings'
5   WHEN EXTRACT(hour FROM order_purchase_timestamp) between 13 and 18 THEN 'Afternoon'
6   ELSE 'Night'
7 END AS Time_of_day, count(order_id) as count_of_order
8 from `circular-matrix-406015.Target.orders`
9 group by time_of_day
10 order by count_of_order desc
11 ;
12
```

Processing location: asia-south1

Query results

SAVE RESULTS EXPLORE DATA

	JOB INFORMATION	RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS
Row	Time_of_day	count_of_order				
1	Afternoon	38135				
2	Night	28331				
3	Mornings	27733				
4	Dawn	5242				

**Order has getting higher only in afternoon, followed by night and morning**

## TARGET BUSINESS CASE STUDY

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


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

Query :

```
SELECT
EXTRACT(month FROM o.order_purchase_timestamp) as month,
COUNT(o.order_id) as count,c.customer_state as state
FROM `circular-matrix-406015.Target.orders` as o join
`circular-matrix-406015.Target.customers` as c on
o.customer_id=c.customer_id
GROUP BY month, state
order by month,state;
```

Output

```
1 SELECT EXTRACT(month FROM o.order_purchase_timestamp) as month,
2 COUNT(o.order_id) as count,c.customer_state as state
3 FROM `circular-matrix-406015.Target.orders` as o join
4 `circular-matrix-406015.Target.customers` as c
5 on
6 o.customer_id=c.customer_id
7 GROUP BY month, state
8 order by month,state;
9
10
11 |
```

Processing location: asia-south1  Press Alt+F1 for Accessibility C

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

	JOB INFORMATION	RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	E
w	month	count	state				
1		1	8	AC			
2		1	39	AL			
3		1	12	AM			
4		1	11	AP			
5		1	264	BA			
6		1	99	CE			
7		1	151	DF			
8		1	159	ES			
9		1	164	GO			

Results per page: 50 1 - 50 of 322 < >

SP state has the highest no.of order in every month

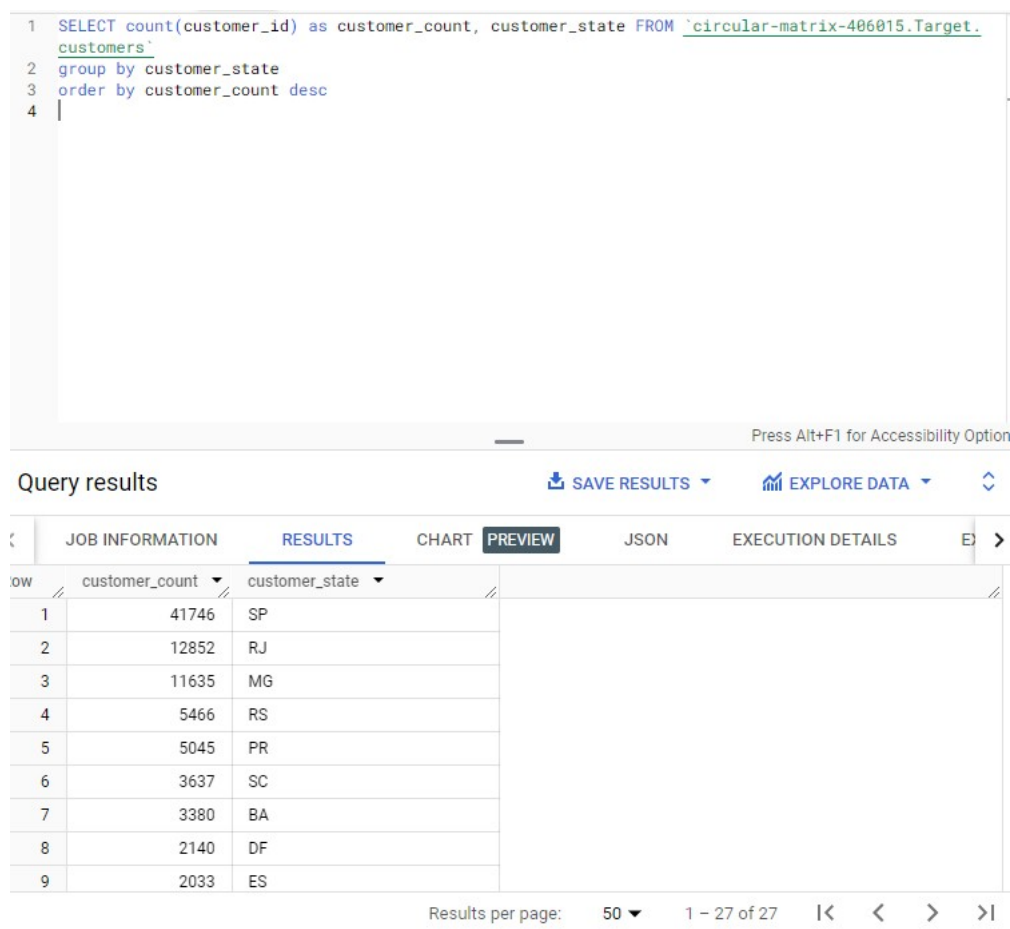
## TARGET BUSINESS CASE STUDY

2.How are the customers distributed across all the states?

Query

```
SELECT  
count(customer_id) as customer_count, customer_state FROM  
`circular-matrix-406015.Target.customers`  
group by customer_state  
order by customer_count desc
```

Output



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

```
1 SELECT count(customer_id) as customer_count, customer_state FROM `circular-matrix-406015.Target.  
2 customers`  
3 group by customer_state  
4 order by customer_count desc
```

Below the query, the results are displayed in a table. The table has two columns: **customer\_count** and **customer\_state**. The results are ordered by **customer\_count** in descending order.

row	customer_count	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

At the bottom of the interface, there is a pagination bar showing "Results per page: 50" and "1 - 27 of 27".

**Top 3 customer\_count state is SP,RJ,&MG**

## TARGET BUSINESS CASE STUDY

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

Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only). You can use the "payment\_value" column in the payments table to get the cost of orders.

Query

```
with refcol AS
(
  SELECT sum(
    case
      WHEN extract(year
        FROM o.order_purchase_timestamp) = 2017 THEN
        p.payment_value
      ELSE 0 end) AS sum2017, sum(
    case
      WHEN extract(year
        FROM o.order_purchase_timestamp) = 2018 THEN
        p.payment_value
      ELSE 0 end) AS sum2018,
  FROM `circular-matrix-406015.Target.orders` AS o
  JOIN `circular-matrix-406015.Target.payments` AS p
    ON o.order_id = p.order_id
  WHERE extract(month
    FROM o.order_purchase_timestamp) < 9 )
SELECT ((sum2018-sum2017)/sum2017) *100 AS percentage
FROM refcol;
```

Output

**136% has been increased in the cost of orders from year 2017 to 2018**

```
1
2
3 with refcol AS
4   (SELECT sum(case
5     WHEN extract(year
6       FROM o.order_purchase_timestamp) = 2017 THEN
7       p.payment_value
8     ELSE 0 end) AS sum2017, sum(case
9     WHEN extract(year
10      FROM o.order_purchase_timestamp) = 2018 THEN
11      p.payment_value
12     ELSE 0 end) AS sum2018,
13   FROM `circular-matrix-406015.Target.orders` AS o
14   JOIN `circular-matrix-406015.Target.payments` AS p
15     ON o.order_id = p.order_id
16   WHERE extract(month
17     FROM o.order_purchase_timestamp) < 9 )
18   SELECT ((sum2018-sum2017)/sum2017) *100 AS percentage
19 FROM refcol;
```

Query results

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXI
Row	percentage					
1	136.9768716466...					



## TARGET BUSINESS CASE STUDY

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

### Query

```
SELECT c.customer_state,  
       sum(oi.price) AS sum_price,  
       avg(oi.price) AS avg_price  
FROM circular-matrix-406015.Target.order_items AS oi  
JOIN circular-matrix-406015.Target.orders AS o  
  ON o.order_id = oi.order_id  
JOIN circular-matrix-406015.Target.customers AS c  
  ON c.customer_id = o.customer_id  
GROUP BY c.customer_state
```

### Output

1	SELECT c.customer_state,
2	sum(oi.price) AS sum_price,
3	avg(oi.price) AS avg_price
4	FROM circular-matrix-406015.Target.order_items AS oi
5	JOIN circular-matrix-406015.Target.orders AS o
6	ON o.order_id = oi.order_id
7	JOIN circular-matrix-406015.Target.customers AS c
8	ON c.customer_id = o.customer_id
9	GROUP BY c.customer_state;
10	

Query results				
JOB INFORMATION		RESULTS	CHART	PREVIEW
JSON				
Row	customer_state	sum_price	avg_price	
1	MT	156453.5299999...	148.2971848341...	
2	MA	119648.2199999...	145.2041504854...	
3	AL	80314.81	180.8892117117...	
4	SP	5202955.050001...	109.6536291597...	
5	MG	1585308.029999...	120.7485741488...	
6	PE	262788.0299999...	145.5083222591...	
7	RJ	1824092.669999...	125.1178180945...	
8	DF	302603.9399999...	125.7705486284...	
9	RS	750304.0200000...	120.3374530874...	
10	SE	58920.85000000...	153.0411688311...	
11	PR	683083.7600000...	119.0041393728...	

SP state has the avg price is 109, and has the highest order value

## TARGET BUSINESS CASE STUDY

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

Query

```
SELECT c.customer_state,  
       sum(oi.freight_value) AS sum_freight_value,  
       avg(oi.freight_value) AS avg_freight_value  
FROM circular-matrix-406015.Target.order_items AS oi  
JOIN circular-matrix-406015.Target.orders AS o  
  ON o.order_id = oi.order_id  
JOIN circular-matrix-406015.Target.customers AS c  
  ON c.customer_id = o.customer_id  
GROUP BY c.customer_state;
```

Output

```
1 SELECT c.customer_state,  
2      sum(oi.freight_value) AS sum_freight_value,  
3      avg(oi.freight_value) AS avg_freight_value  
4 FROM circular-matrix-406015.Target.order_items AS oi  
5 JOIN circular-matrix-406015.Target.orders AS o  
6   ON o.order_id = oi.order_id  
7 JOIN circular-matrix-406015.Target.customers AS c  
8   ON c.customer_id = o.customer_id  
9 GROUP BY c.customer_state;  
10
```

Query results [SAVE RESULTS](#)

	JOB INFORMATION	RESULTS	CHART	PREVIEW	JSON
Row	customer_state	sum_freight_value	avg_freight_value		
1	SP	718723.0699999...	15.14727539041...		
2	RJ	305589.3100000...	20.96092393168...		
3	PR	117851.6800000...	20.53165156794...		
4	SC	89660.26000000...	21.47036877394...		
5	DF	50625.49999999...	21.04135494596...		
6	MG	270853.4600000...	20.63016680630...		
7	PA	38699.30000000...	35.83268518518...		
8	BA	100156.6799999...	26.36395893656...		
9	GO	53114.97999999...	22.76681525932...		
10	RS	135522.7400000...	21.73580433039...		
11	TO	11732.67999999...	37.24660317460...		

Freight charges is between 15- 42, more order went from the state is SP

## TARGET BUSINESS CASE STUDY

### Analysis based on sales, freight and delivery time.

1. 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:
2. **time\_to\_deliver** = order\_delivered\_customer\_date - order\_purchase\_timestamp  
**diff\_estimated\_delivery** = order\_delivered\_customer\_date - order\_estimated\_delivery\_date

```
SELECT order_id,  
       date_diff(o.order_delivered_customer_date,  
o.order_purchase_timestamp, day) AS time_to_deliver,  
       date_diff(o.order_delivered_customer_date,  
o.order_estimated_delivery_date, day) AS  
diff_estimated_delivery  
FROM circular-matrix-406015.Target.orders AS o  
ORDER BY time_to_deliver DESC ;
```

```
1 SELECT order_id,  
2         date_diff(o.order_delivered_customer_date,  
3         o.order_purchase_timestamp,  
4         day) AS time_to_deliver,  
5         date_diff(o.order_delivered_customer_date,  
6         o.order_estimated_delivery_date,  
7         day) AS diff_estimated_delivery  
8 FROM circular-matrix-406015.Target.orders AS o  
9 ORDER BY time_to_deliver DESC ;
```

Query results [SAVE RESULTS](#)

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS
Row	order_id	time_to_deliver	diff_estimated_delivery			
1	ca07593549f1816d26a572e06...	209	181			
2	1b3190b2dfa9d789e1f14c05b...	208	188			
3	440d0d17af552815d15a9e41a...	195	165			
4	0f4519c5f1c541ddec9f21b3bd...	194	161			
5	285ab9426d6982034523a855f...	194	166			
6	2fb597c2f772eca01b1f5c561b...	194	155			
7	47b40429ed8cce3aee9199792...	191	175			
8	2fe324feb907e3ea3f2aa9650...	189	167			
9	2d7561026d542c8dbd8f0daea...	188	159			
10	437222e3fd1b07396f1d9ba8c...	187	144			
11	c27815f7e3dd0b926b5855262...	187	162			

## TARGET BUSINESS CASE STUDY

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

```

SELECT t.customer_state,
       t.avgfirt,
       CASE
         WHEN t.rn_up <= 5 THEN 'Low_avg'
         WHEN t.rn_down <= 5 THEN 'High_avg'
         END AS freight_range
FROM
  (SELECT c.customer_state,
         avg(oi.freight_value) AS avgfirt ,
         DENSE_RANK()
           OVER (ORDER BY avg(oi.freight_value) ASC) AS rn_up,
         DENSE_RANK()
           OVER (ORDER BY avg(oi.freight_value) DESC) AS rn_down
    FROM circular-matrix-406015.Target.order_items AS oi
   JOIN circular-matrix-406015.Target.orders AS o
     ON o.order_id = oi.order_id
   JOIN circular-matrix-406015.Target.customers AS c
     ON c.customer_id = o.customer_id
   GROUP BY c.customer_state ) AS t
WHERE t.rn_up <= 5    OR t.rn_down <= 5
ORDER BY T.AVGFIRT;

```

```

1  SELECT t.customer_state,
2      | | | t.avgfirt,
3      | | |
4      | | |
5      | | | CASE
6      | | | WHEN t.rn_up <= 5 THEN
7      | | | 'Low_avg'
8      | | | WHEN t.rn_down <= 5 THEN
9      | | | 'High_avg'
10     | | | END AS freight_range
11 FROM
12     | | | (SELECT c.customer_state,
13     | | |         avg(oi.freight_value) AS avgfirt ,
14     | | |         DENSE_RANK()
15     | | |         OVER (ORDER BY avg(oi.freight_value) ASC) AS rn_up, DENSE_RANK()
16     | | |         OVER (ORDER BY avg(oi.freight_value) DESC) AS rn_down
17     | | | FROM circular-matrix-406015.Target.order_items AS oi
18     | | | JOIN circular-matrix-406015.Target.orders AS o
19     | | | | ON o.order_id = oi.order_id

```

Query results

[SAVE RESULTS](#) [E](#)

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXECUT
Row	customer_state	avgfirt	freight_range				
1	SP	15.14727539041...	Low_avg				
2	PR	20.53165156794...	Low_avg				
3	MG	20.63016680630...	Low_avg				
4	RJ	20.96092393168...	Low_avg				
5	DF	21.04135494596...	Low_avg				
6	PI	39.14797047970...	High_avg				
7	AC	40.07336956521...	High_avg				
8	RO	41.06971223021...	High_avg				
9	PB	42.72380398671...	High_avg				
10	RR	42.98442307692...	High_avg				

## TARGET BUSINESS CASE STUDY

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

```

SELECT t.customer_state,
       t.avgdeli,
       CASE
         WHEN t.rn_up <= 5 THEN
           'Low_avg'
         WHEN t.rn_down <= 5 THEN
           'High_avg'
         END AS delivery_time
FROM
  (SELECT c.customer_state,
         avg(date_diff(o.order_delivered_customer_date,
                       o.order_purchase_timestamp,
                       day)) AS avgdeli ,
         DENSE_RANK() OVER (ORDER BY
avg(date_diff(o.order_delivered_customer_date,
o.order_purchase_timestamp, day)) ASC) AS rn_up, DENSE_RANK()
OVER (ORDER BY
avg(date_diff(o.order_delivered_customer_date,
o.order_purchase_timestamp, day)) DESC) AS rn_down
FROM circular-matrix-406015.Target.orders AS o
JOIN circular-matrix-406015.Target.customers AS c
ON c.customer_id =o.customer_id
GROUP BY c.customer_state ) AS t
WHERE t.rn_up <= 5 OR t.rn_down <= 5
ORDER BY t.avgdeli;

```

```

1 SELECT t.customer_state,
2       t.avgdeli,
3
4       CASE
5         WHEN t.rn_up <= 5 THEN
6           'Low_avg'
7         WHEN t.rn_down <= 5 THEN
8           'High_avg'
9         END AS delivery_time
10 FROM
11   (SELECT c.customer_state,
12         avg(date_diff(o.order_delivered_customer_date,
13                       o.order_purchase_timestamp,
14                       day)) AS avgdeli ,
15         DENSE_RANK() OVER (ORDER BY avg(date_diff(o.order_delivered_customer_date, o.order_purchase
16         rn_up, DENSE_RANK() OVER (ORDER BY avg(date_diff(o.order_delivered_customer_date, o.order_purchase
17         rn_down

```

Query results

[SAVE RESULTS](#)

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS
row	customer_state	avgdeli	delivery_time			
1	SP	8.298061489072...	Low_avg			
2	PR	11.52671135486...	Low_avg			
3	MG	11.54381329810...	Low_avg			
4	DF	12.50913461538...	Low_avg			
5	SC	14.47956019171...	Low_avg			
6	PA	23.31606765327...	High_avg			
7	AL	24.04030226700...	High_avg			
8	AM	25.98620689655...	High_avg			
9	AP	26.73134328358...	High_avg			
10	RR	28.97560975609...	High_avg			

## TARGET BUSINESS CASE STUDY

- 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 c.customer_state,  
       avg(date_diff(o.order_delivered_customer_date,  
                     o.order_estimated_delivery_date,  
                     day)) AS diff_estimated_delivery  
FROM circular-matrix-406015.Target.orders AS o  
JOIN circular-matrix-406015.Target.customers AS c  
  ON c.customer_id = o.customer_id  
  
GROUP BY c.customer_state  
ORDER BY diff_estimated_delivery desc  
  
limit 5;
```

```
1 SELECT c.customer_state,  
2       avg(date_diff(o.order_delivered_customer_date, o.order_estimated_delivery_date,  
3                     day)) AS diff_estimated_delivery  
4 FROM circular-matrix-406015.Target.orders AS o  
5 JOIN circular-matrix-406015.Target.customers AS c  
6   ON c.customer_id = o.customer_id  
7  
8 GROUP BY c.customer_state  
9 ORDER BY diff_estimated_delivery desc  
10  
11 limit 5;  
12  
13 |
```

Query results [SAVE RESULTS](#)

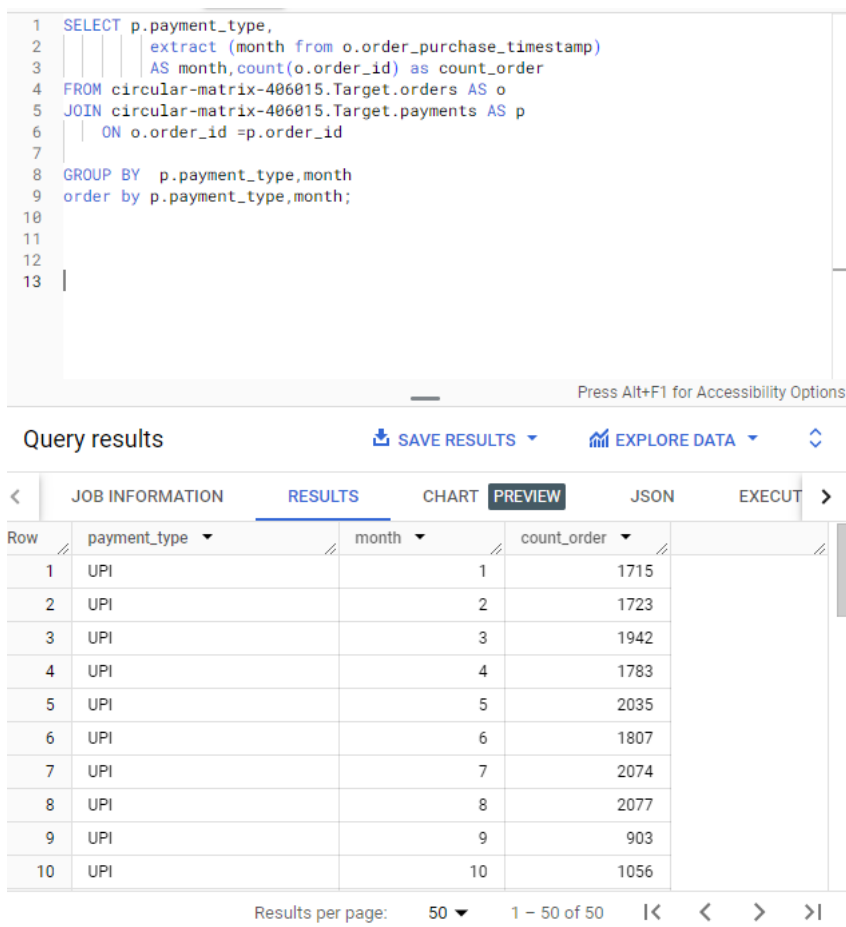
JOB INFORMATION	RESULTS	CHART	PREVIEW	JSON	EXECUTION DETAILS	EXE
Row	customer_state	diff_estimated_delive				
1	AL	-7.94710327455...				
2	MA	-8.76847977684...				
3	SE	-9.17313432835...				
4	ES	-9.61854636591...				
5	BA	-9.93488943488...				

## TARGET BUSINESS CASE STUDY

### Analysis based on the payments:

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

```
SELECT p.payment_type,  
       extract (month from o.order_purchase_timestamp)  
       AS month,count(o.order_id) as count_order  
FROM circular-matrix-406015.Target.orders AS o  
JOIN circular-matrix-406015.Target.payments AS p  
  ON o.order_id =p.order_id  
  
GROUP BY  p.payment_type,month  
order by p.payment_type,month;
```



1 SELECT p.payment\_type,  
2 | | | | | extract (month from o.order\_purchase\_timestamp)  
3 | | | | | AS month,count(o.order\_id) as count\_order  
4 FROM circular-matrix-406015.Target.orders AS o  
5 JOIN circular-matrix-406015.Target.payments AS p  
6 | | ON o.order\_id =p.order\_id  
7  
8 GROUP BY p.payment\_type,month  
9 order by p.payment\_type,month;  
10  
11  
12  
13 |

Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS CHART PREVIEW JSON EXECUT

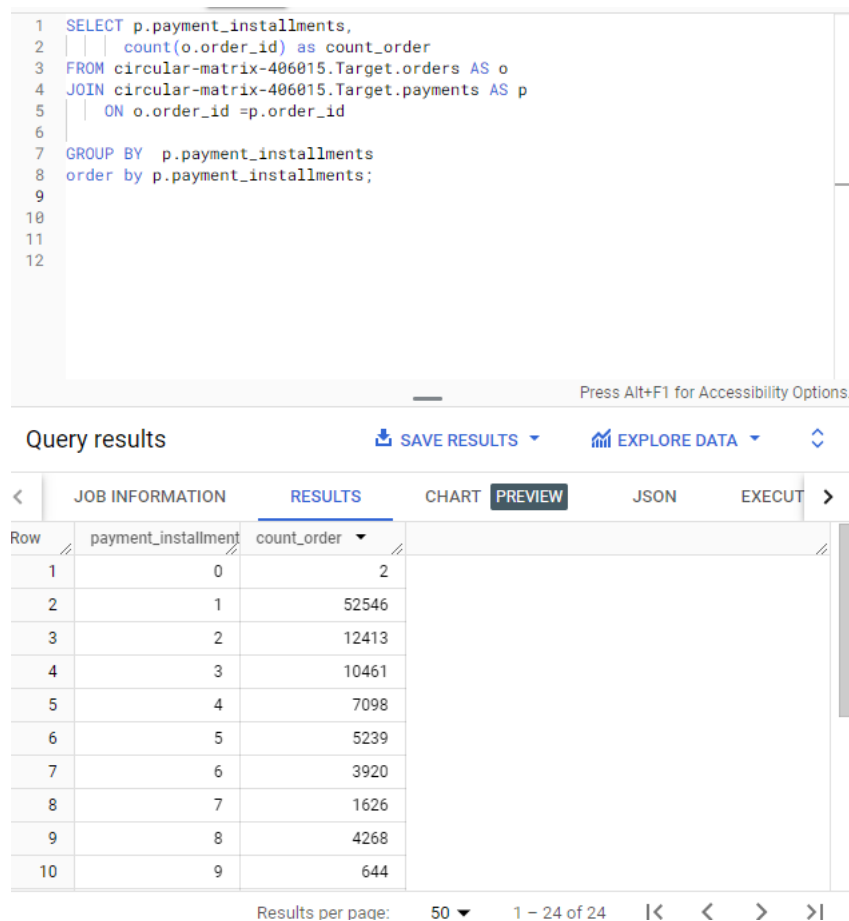
Row	payment_type	month	count_order
1	UPI	1	1715
2	UPI	2	1723
3	UPI	3	1942
4	UPI	4	1783
5	UPI	5	2035
6	UPI	6	1807
7	UPI	7	2074
8	UPI	8	2077
9	UPI	9	903
10	UPI	10	1056

Results per page: 50 1 - 50 of 50

## TARGET BUSINESS CASE STUDY

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

```
SELECT p.payment_installments,  
       count(o.order_id) as count_order  
FROM circular-matrix-406015.Target.orders AS o  
JOIN circular-matrix-406015.Target.payments AS p  
  ON o.order_id =p.order_id  
  
GROUP BY p.payment_installments  
order by p.payment_installments;
```



```
1 SELECT p.payment_installments,  
2     count(o.order_id) as count_order  
3 FROM circular-matrix-406015.Target.orders AS o  
4 JOIN circular-matrix-406015.Target.payments AS p  
5     ON o.order_id =p.order_id  
6  
7 GROUP BY p.payment_installments  
8 order by p.payment_installments;  
9  
10  
11  
12
```

Query results

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

[JOB INFORMATION](#) **[RESULTS](#)** [CHART](#) [PREVIEW](#) [JSON](#) [EXECUT](#)

Row	payment_installment	count_order
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626
9	8	4268
10	9	644

Results per page: 50 1 - 24 of 24