Case Study: Target (SQL)

JUNE 2023

BY
Unnati Athwani
Email: unnatiathwani@gmail.com

Contents

Points to Note

1. The SQL queries are executed in Google BigQuery.
2. The project name is 'target-sql'.
3. The dataset name is 'TargetDATA'.

1. Initial exploration of dataset like checking the characteristics of data

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**

   The query to find the data types of the tables is given below:

```
SELECT column_name, data_type
FROM `target--sql.TargetDATA.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'customers'
```

   The table name can be replaced by other tables.

   The data types of the columns of the tables (got as a result of the above query) are:

   1. customers:

Column	Data Type
customer_id	STRING
customer_unique_id	STRING
customer_zip_code_prefix	INT64
customer_city	STRING
customer_state	STRING

   2. sellers:

Column	Data Type

seller_id	STRING
seller_zip_code_prefix	INT64
seller_city	STRING
seller_state	STRING

3. order_items

Column	**Data Type**
order_id	STRING
order_item_id	STRING
product_id	STRING
seller_id	STRING
shipping_limit_date	TIMESTAMP
price	FLOAT64
freight_value	FLOAT64

4. geolocations

Column	**Data Type**
geolocation_zip_code_prefix	INT64
geolocation_lat	FLOAT64
geolocation_lng	FLOAT64
geolocation_city	STRING
geolocation_state	STRING

5. payments

Column	**Data Type**
order_id	STRING

payment_sequential	INT64
payment_type	STRING
payment_installments	INT64
payment_value	FLOAT64

6. orders

Column	**Data Type**
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

7. reviews

Column	**Data Type**
review_id	STRING
order_id	STRING
review_score	INT64
review_comment_title	STRING
review_creation_date	TIMESTAMP
review_answer_timestamp	TIMESTAMP

8. products

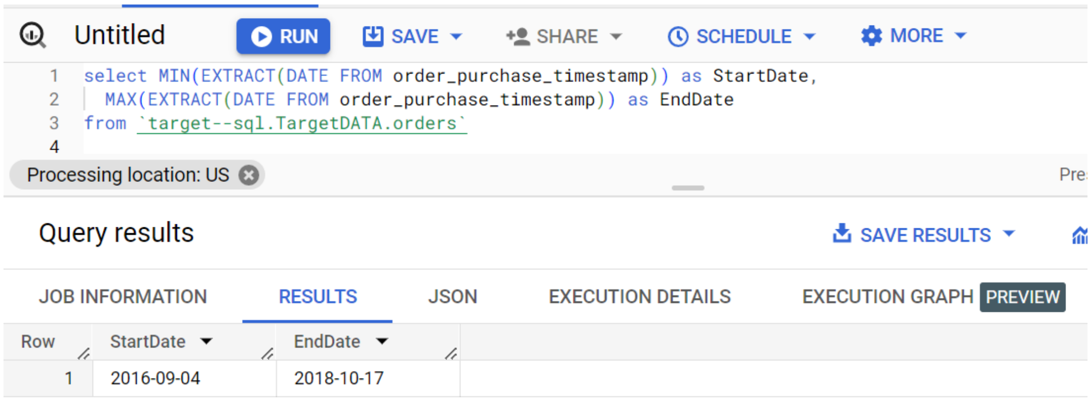
Column	Data Type
product_id	STRING
product_category	STRING
product_name_length	INT64
product_description_length	INT64
product_photos_qty	INT64
product_weight_g	INT64
product_length_cm	INT64
product_height_cm	INT64
product_width_cm	INT64

2. **Time period for which the data is given**
   The time period of the dataset is from the first day of the order to the last day of it. Hence, this can be found by finding the maximum and minimum of order_purchase_timestamp. This is considering the active days of the pilot testing. Undoubtedly, the timestamp of review_answer_timestamp in the 'reviews' table will be greater.

   Query:

```
SELECT MIN(EXTRACT(DATE FROM order_purchase_timestamp)) AS StartDate,
  MAX(EXTRACT(DATE FROM order_purchase_timestamp)) AS EndDate
FROM `target--sql.TargetDATA.orders`
```

   Result:



```
select MIN(EXTRACT(DATE FROM order_purchase_timestamp)) as StartDate,
  MAX(EXTRACT(DATE FROM order_purchase_timestamp)) as EndDate
from `target--sql.TargetDATA.orders`

```

Query results

Row	StartDate	EndDate
1	2016-09-04	2018-10-17

3. Cities and States of customers ordered during the given period

Query:

```
SELECT customer_state, count(DISTINCT customer_city) as customer_cities
FROM `target--sql.TargetDATA.orders` o
  LEFT JOIN `target--sql.TargetDATA.customers` c ON o.customer_id =
c.customer_id
GROUP BY customer_state
ORDER BY customer_state
```

The first 10 rows are shown below:

Row	customer_state	customer_cities
1	AC	8
2	AL	68
3	AM	5
4	AP	6
5	BA	353
6	CE	161
7	DF	6
8	ES	95
9	GO	178
10	MA	122

The number of states participating in Brazil are found by:

```
SELECT COUNT(DISTINCT customer_state)
FROM `target--sql.TargetDATA.customers`
```

The result comes out to be 27, which is total number of states in Brazil (26 states + 1 federal district). Thus, the testing was done in the whole of Brazil.

2. In-depth Exploration

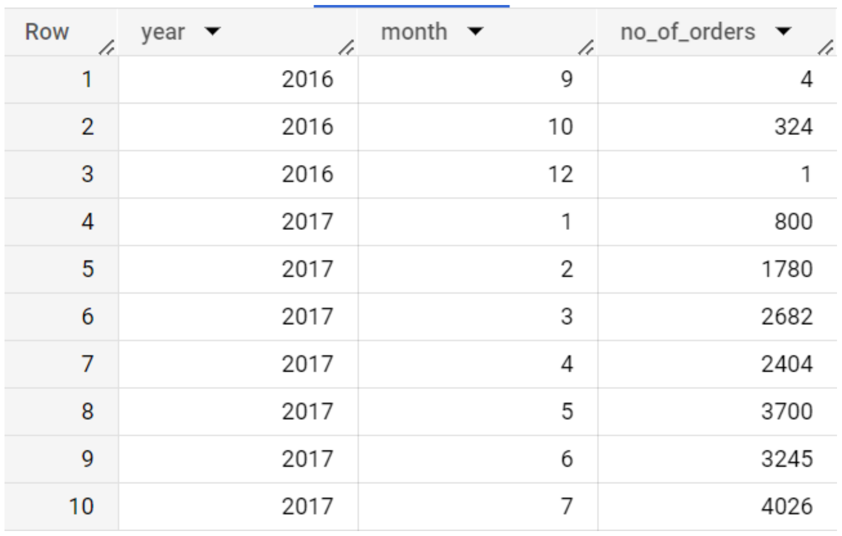
1. **Is there a growing trend in e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?**

   To find the trend in e-commerce in Brazil, we can see the number of orders placed across months and look at the percentage change of these numbers.

   Query:

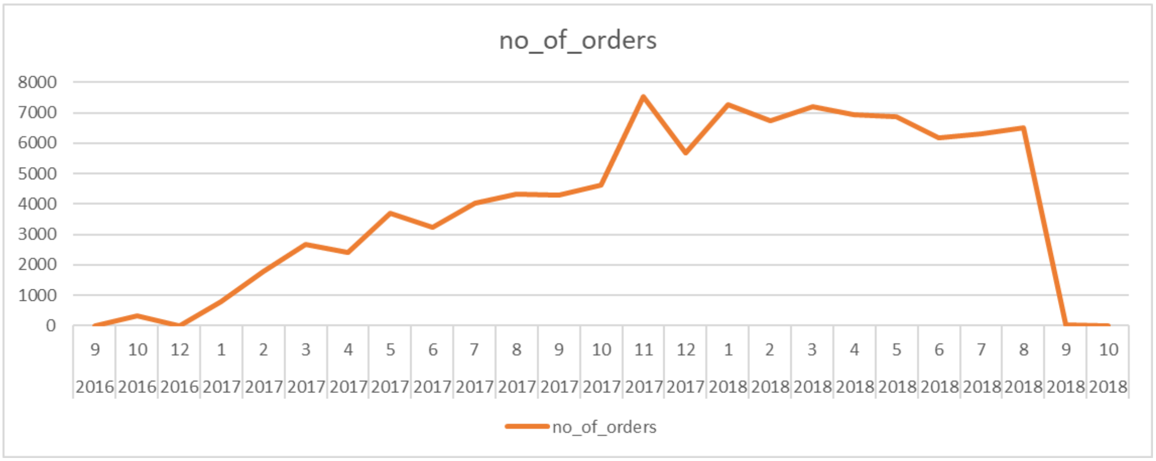
```
SELECT EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
  COUNT(order_id) AS no_of_orders
FROM `target--sql.TargetDATA.orders`
GROUP BY year, month
ORDER BY year, month
```

The first 10 rows of the result are:



Row	year	month	no_of_orders
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026

We plot a graph for a meaningful outcome of the above data.
The below graph plots the number of orders over time (in months and years):

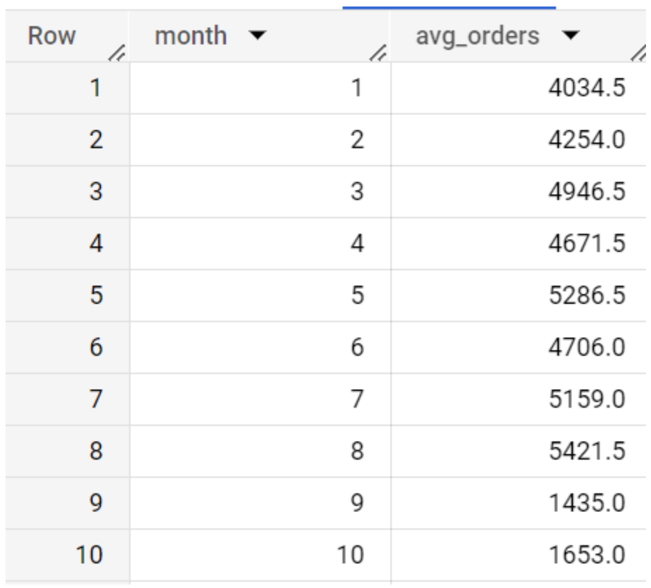


The number of orders has increased over the months. Thus, we can be sure that **the trend of e-commerce is growing in Brazil**. The data of the first and the last months of the testing phase would reasonably be less due to the limited number of order items and sellers listing. But, the graph in most of the testing period gives the essence that the sales on the Target platform have increased.

For seasonality, we can look at the number of orders over the months. We divide the data by the number of years for each month to take an average of number of orders. This is given by:

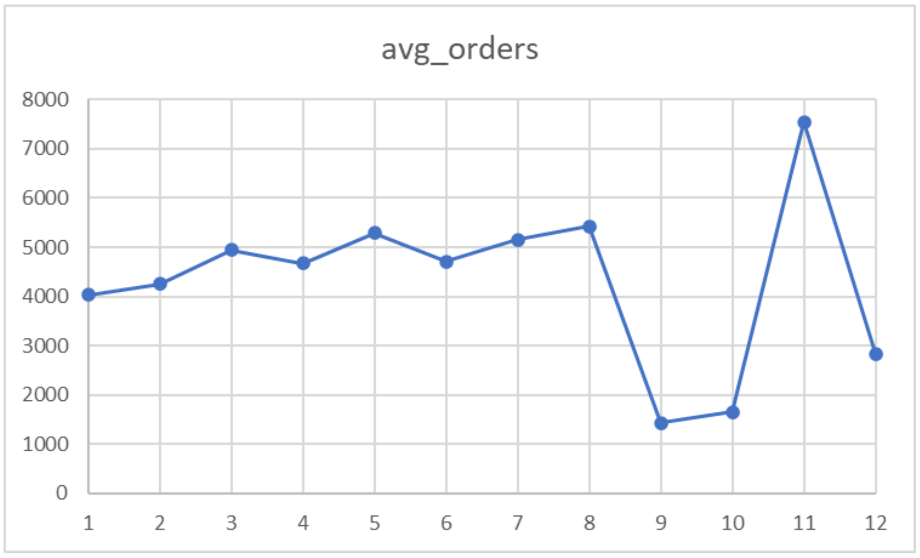
```
WITH cte1 AS (
  SELECT EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
    COUNT(DISTINCT EXTRACT(YEAR FROM order_purchase_timestamp)) as no_of_years
  FROM `target--sql.TargetDATA.orders`
GROUP BY month
), cte2 AS (
  SELECT EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
    COUNT(order_id) as no_of_orders
  FROM `target--sql.TargetDATA.orders`
  GROUP BY month
)
SELECT cte1.month, no_of_orders / no_of_years AS avg_orders
FROM cte1
  INNER JOIN cte2 ON cte1.month = cte2.month
ORDER BY month
```

The first 10 rows of the query are:



Row	month	avg_orders
1	1	4034.5
2	2	4254.0
3	3	4946.5
4	4	4671.5
5	5	5286.5
6	6	4706.0
7	7	5159.0
8	8	5421.5
9	9	1435.0
10	10	1653.0

We plot a graph for a meaningful outcome of the above data.
The below graph plots the number of orders over months to depict the seasonality, if any:



The graph shows that the orders rise in November, with September and October seeing the least interest by the customers.

2. **What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?**
   To find out when the Brazilians are ordering the most, we can divide the 24 hours into a few bins and find the number of orders placed during these times.

The times considered here are:

1. 0000 - 0400 hrs: Late Midnight
2. 0400 - 0800 hrs: Early Morning
3. 0800 - 1200 hrs: Late Morning
4. 1200 - 1600 hrs: Afternoon
5. 1600 - 2000 hrs: Evening
6. 2000 - 0000 hrs: Night

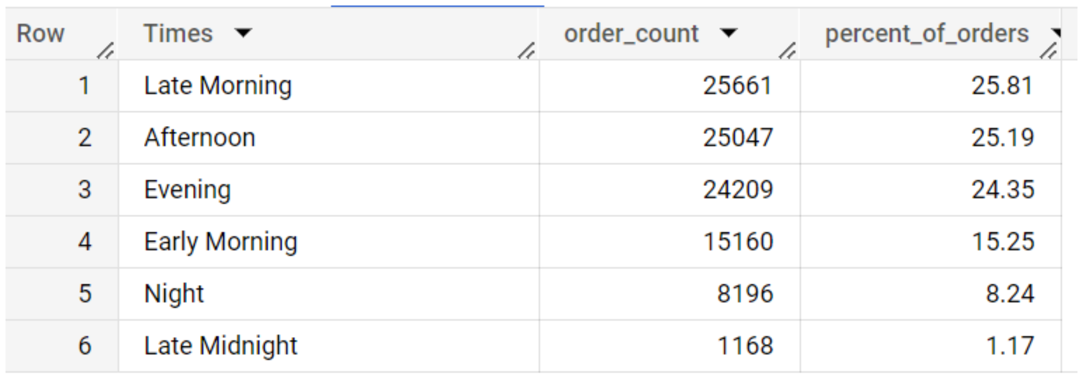
In addition, since timestamps are in UTC, we would need to adjust them with the Brazilian time zones. There are 4 time zones in Brazil. For simplicity, 'UTC-3:00' is considered for the whole of Brazil, since 93% of Brazil's population resides in this time zone.

Query:

```
SELECT Times,
  COUNT(order_id) AS order_count,
  ROUND(COUNT(order_id) / (SELECT COUNT(order_id) FROM
`target--sql.TargetDATA.orders`) * 100, 2) AS percent_of_orders
FROM (
  SELECT
    CASE
      WHEN EXTRACT(TIME FROM order_purchase_timestamp AT TIME ZONE 'UTC-3:00')
BETWEEN '00:00:00' and '04:00:00'
      THEN 'Late Midnight'
      WHEN EXTRACT(TIME FROM order_purchase_timestamp AT TIME ZONE 'UTC-3:00')
BETWEEN '04:00:01' and '08:00:00'
      THEN 'Early Morning'
      WHEN EXTRACT(TIME FROM order_purchase_timestamp AT TIME ZONE 'UTC-3:00')
BETWEEN '08:00:01' and '12:00:00'
      THEN 'Late Morning'
      WHEN EXTRACT(TIME FROM order_purchase_timestamp AT TIME ZONE 'UTC-3:00')
BETWEEN '12:00:01' and '16:00:00'
      THEN 'Afternoon'
      WHEN EXTRACT(TIME FROM order_purchase_timestamp AT TIME ZONE 'UTC-3:00')
BETWEEN '16:00:01' and '20:00:00'
      THEN 'Evening'
      WHEN EXTRACT(TIME FROM order_purchase_timestamp AT TIME ZONE 'UTC-3:00')
BETWEEN '20:00:01' and '23:59:59'
      THEN 'Night'
    END AS Times,
    order_id
  FROM `target--sql.TargetDATA.orders`
) AS T1
```

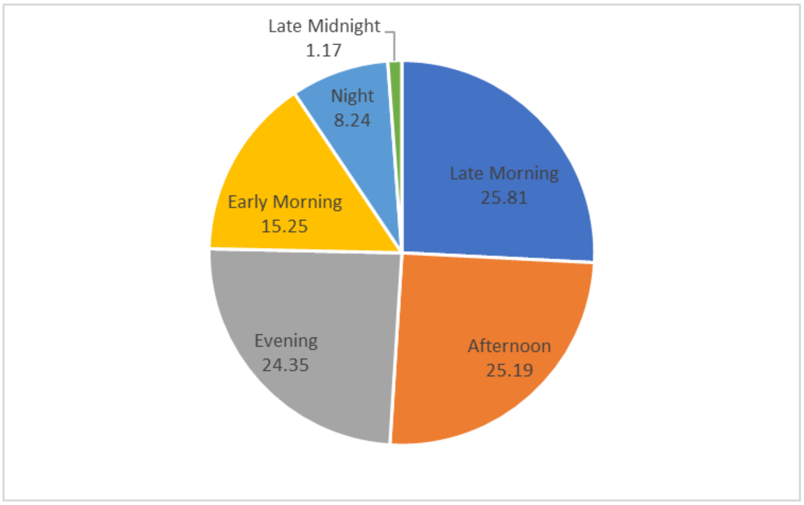
```
GROUP BY Times
ORDER BY order_count DESC
```

Result:



Row	Times	order_count	percent_of_orders
1	Late Morning	25661	25.81
2	Afternoon	25047	25.19
3	Evening	24209	24.35
4	Early Morning	15160	15.25
5	Night	8196	8.24
6	Late Midnight	1168	1.17

A graph of the result is shown below for convenience:



Insights:

❖ We find that most of the orders are placed in the Late Morning and Afternoon, which is between 0800 - 1600 hrs BRT, with the evening times not lagging far behind.

❖ The late midnight hours showed the least engagement in the order transactions, which is justifiable and also needed for server maintenance.

❖ There is a significant rise in orders placed in November, probably due to the festival season.

Recommendations:

- We shall make sure that the servers are up and running at all times, especially in these blocks of time when customer engagement shoots up.
- The notifications of the items nearing their expiration dates or life can be sent at high engagement times (time of the day and month of the year) so that the stock that is to be cleared has a high chance of being bought. In addition, the sale can be kept for a limited time (2 hours only, for example) so that the customers buy the products on impulse.
- The sellers can be suggested to add their items before these times so that the products are in stock when the bulk of customers would go for buying them.
- The server maintenance, when required, can easily be done between 0000 - 0400 hrs BRT, as it accounts for no more than 1.17% of orders.
- There can be events like lucky draw, quizzes etc organised in September or October to engage customers and have increased sales.

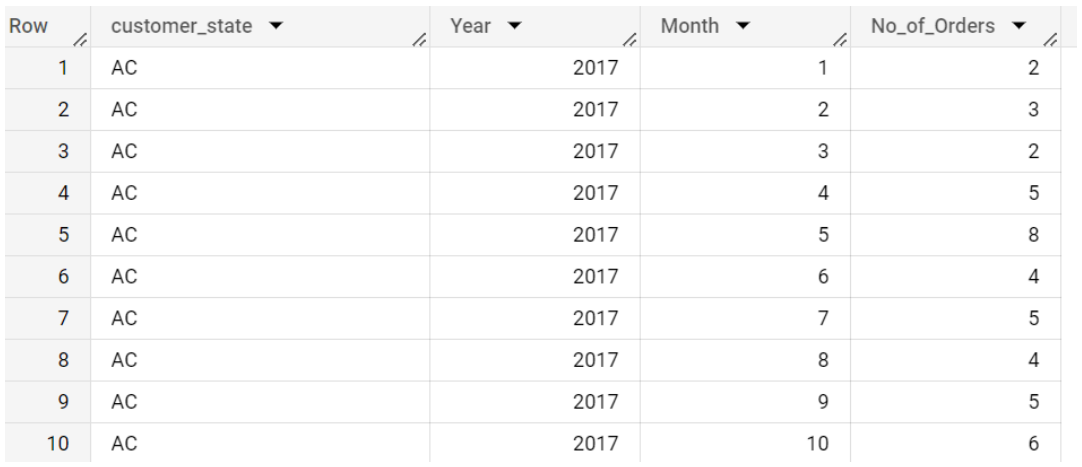
3. Evolution of E-commerce orders in the Brazil region

1. **Get month-on-month orders by states**

   Query:

```
SELECT customer_state, Year, Month, SUM(IF(order_id IS NULL, 0, 1)) AS
No_of_Orders
FROM (
  SELECT c.customer_id,
    order_id,
    customer_state,
    EXTRACT(YEAR FROM order_purchase_timestamp) AS Year,
    EXTRACT(MONTH FROM order_purchase_timestamp) AS Month
  FROM `target--sql.TargetDATA.orders` o
    RIGHT JOIN `target--sql.TargetDATA.customers` c ON o.customer_id =
c.customer_id
) AS T1
GROUP BY customer_state, Year, Month
ORDER BY customer_state, Year, Month
```

The first 10 rows of the result are tabulated below:



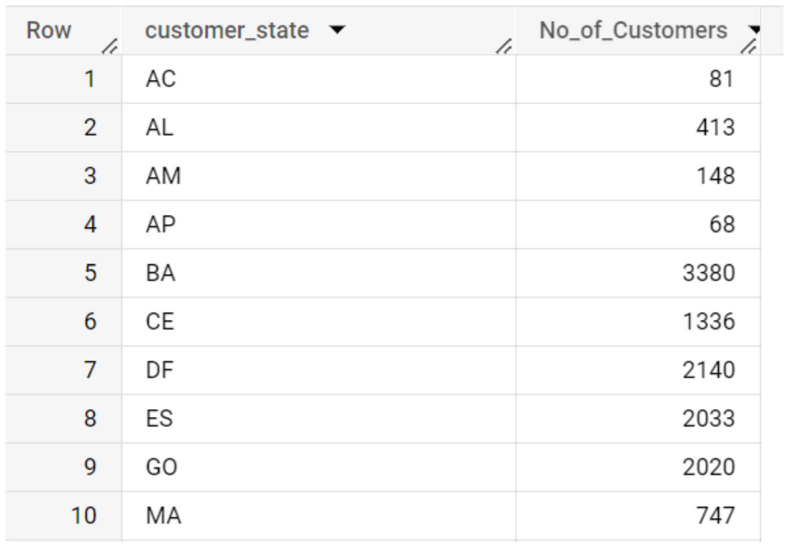
Row	customer_state	Year	Month	No_of_Orders
1	AC	2017	1	2
2	AC	2017	2	3
3	AC	2017	3	2
4	AC	2017	4	5
5	AC	2017	5	8
6	AC	2017	6	4
7	AC	2017	7	5
8	AC	2017	8	4
9	AC	2017	9	5
10	AC	2017	10	6

2. **Distribution of customers across the states in Brazil**

Query:

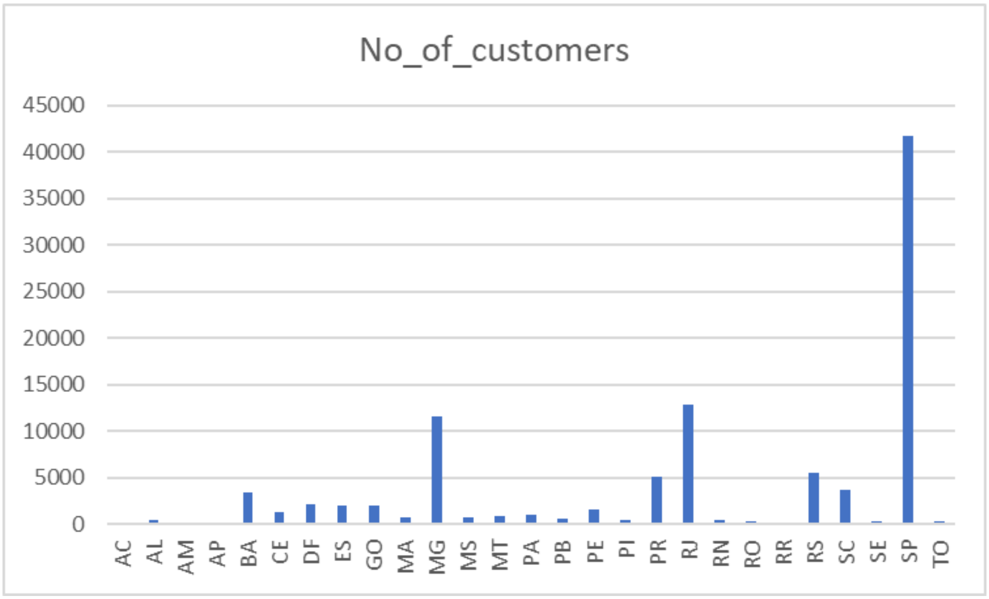
```
SELECT customer_state, COUNT(customer_id) as No_of_Customers
FROM `target--sql.TargetDATA.customers`
GROUP BY customer_state
ORDER BY customer_state
```

The first 10 rows of the result are shown below:



Row	customer_state	No_of_Customers
1	AC	81
2	AL	413
3	AM	148
4	AP	68
5	BA	3380
6	CE	1336
7	DF	2140
8	ES	2033
9	GO	2020
10	MA	747

A graph of the result is shown below for convenience:



Insights:

- The states of SP, RJ, MG, PR and RS have shown a greater customer engagement.
- Customers have shown good interest in the testing phase itself, which is a positive note for the business.

Recommendations:

- The month-on-month order data shows promising results for the business if marketing is pursued further.
- The brand recognition among the customers is already good. With more products and sellers in the app, customer participation can be increased.

4. Impact on Economy

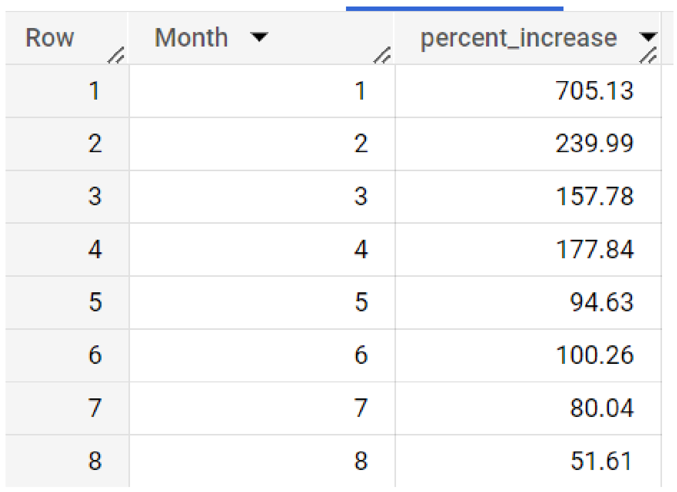
Analyze the money movement by e-commerce by looking at order prices, freight and others.

1. **Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use "payment_value" column in payments table**

   Query:

```
WITH cte1 AS (
  SELECT
    EXTRACT(YEAR FROM order_purchase_timestamp) AS Year,
    EXTRACT(MONTH FROM order_purchase_timestamp) AS Month,
    ROUND(SUM(payment_value), 2) AS total_payments
  FROM `target--sql.TargetDATA.payments` p
    INNER JOIN `target--sql.TargetDATA.orders` o on p.order_id = o.order_id
  GROUP BY EXTRACT(MONTH FROM order_purchase_timestamp),
    EXTRACT(YEAR FROM order_purchase_timestamp)
), cte2 AS (
  SELECT *,
    LAG(total_payments, 1, 1) OVER (PARTITION BY Month ORDER BY Year) AS
lag_data
    FROM cte1
), cte3 AS (
  SELECT *,
    ROUND((total_payments - lag_data) / lag_data * 100, 2) AS percent_increase
    FROM cte2
    WHERE lag_data != 1 and Year = 2018 and Month BETWEEN 1 and 8
)
SELECT Month, percent_increase
FROM cte3
ORDER BY Month
```

Result:



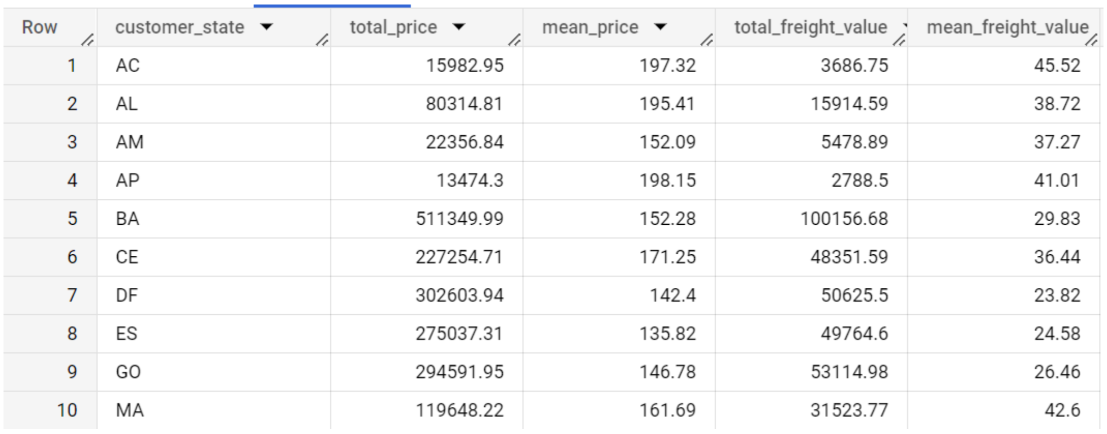
Row	Month	percent_increase
1	1	705.13
2	2	239.99
3	3	157.78
4	4	177.84
5	5	94.63
6	6	100.26
7	7	80.04
8	8	51.61

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

Query:

```
WITH cte1 AS (
  SELECT c.customer_id, customer_state, order_id
  FROM `target--sql.TargetDATA.customers` c
  LEFT JOIN `target--sql.TargetDATA.orders` o ON c.customer_id =
o.customer_id
), cte2 AS (
 SELECT customer_id, customer_state, cte1.order_id,
  IFNULL(price, 0) as price,
  IFNULL(freight_value, 0) as freight_value
 FROM cte1
  INNER JOIN `target--sql.TargetDATA.order_items` oi ON cte1.order_id =
oi.order_id
)
SELECT customer_state,
 ROUND(SUM(price), 2) AS total_price,
 ROUND(SUM(price) / COUNT(DISTINCT order_id), 2) AS mean_price,
 ROUND(SUM(freight_value), 2) AS total_freight_value,
 ROUND(SUM(freight_value) / COUNT(DISTINCT order_id), 2) AS
mean_freight_value,
FROM cte2
GROUP BY customer_state
ORDER BY customer_state
```

The first 10 rows of the result are tabulated below:



Row	customer_state	total_price	mean_price	total_freight_value	mean_freight_value
1	AC	15982.95	197.32	3686.75	45.52
2	AL	80314.81	195.41	15914.59	38.72
3	AM	22356.84	152.09	5478.89	37.27
4	AP	13474.3	198.15	2788.5	41.01
5	BA	511349.99	152.28	100156.68	29.83
6	CE	227254.71	171.25	48351.59	36.44
7	DF	302603.94	142.4	50625.5	23.82
8	ES	275037.31	135.82	49764.6	24.58
9	GO	294591.95	146.78	53114.98	26.46
10	MA	119648.22	161.69	31523.77	42.6

Insights:

❖ The cost of orders is constantly increasing, which shows more customers are trusting and buying products from the platform.

❖ Considering the total cost of orders, SP has contributed the most and RR the least. But the mean price of RR is higher. This could mean that in proportion to the customers in SP, the orders from the platform are less.

❖ The freight value varies greatly (from 17.37 for SP to 48.59 for RR).

Recommendations:

❖ As the platform grows in size, the freight value can be further reduced and the booked profits will increase.

❖ The customers are willing to spend more, which can be taken as a positive sign.

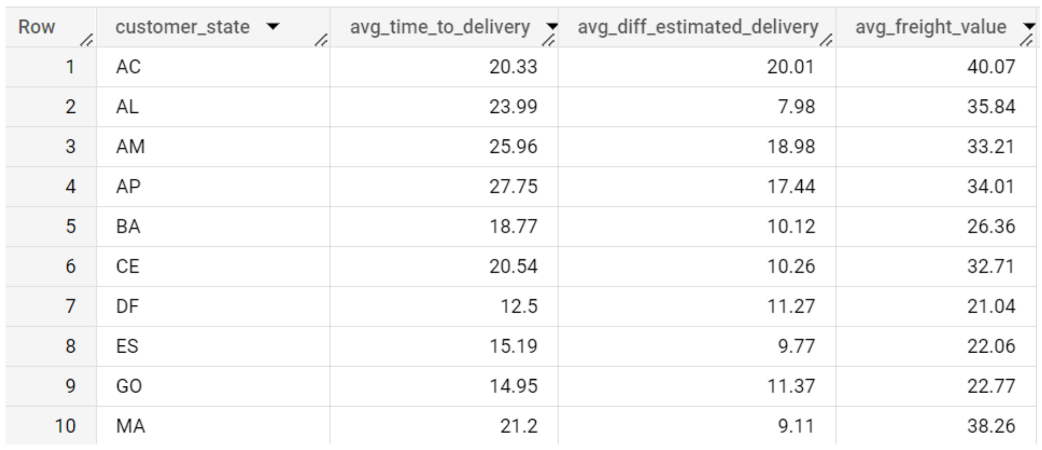
5. Analysis on sales, freight and delivery time

1. **Calculate days between purchasing, delivering and estimated delivery**
2. **Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:**
   a. **time_to_delivery = order_delivered_customer_date - order_purchase_timestamp**
   b. **diff_estimated_delivery = order_estimated_delivery_date - order_delivered_customer_date**
3. **Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery**

   Query:

```
SELECT customer_state,
  ROUND(AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,
DAY)), 2) AS avg_time_to_delivery,
  ROUND(AVG(DATE_DIFF(order_estimated_delivery_date,
order_delivered_customer_date, DAY)), 2) AS avg_diff_estimated_delivery,
  ROUND(AVG(freight_value), 2) as avg_freight_value
FROM `target--sql.TargetDATA.orders` o
  INNER JOIN `target--sql.TargetDATA.customers` c ON c.customer_id =
o.customer_id
  INNER JOIN `target--sql.TargetDATA.order_items` oi ON oi.order_id =
o.order_id
GROUP BY customer_state
ORDER BY customer_state
```

   Result:



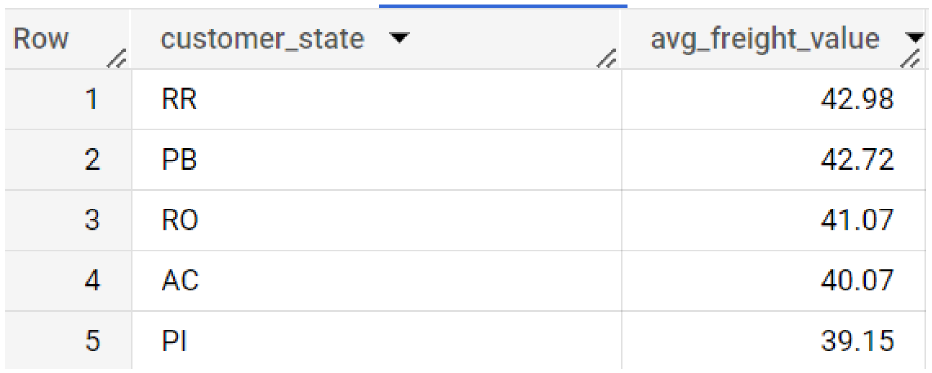
Row	customer_state	avg_time_to_delivery	avg_diff_estimated_delivery	avg_freight_value
1	AC	20.33	20.01	40.07
2	AL	23.99	7.98	35.84
3	AM	25.96	18.98	33.21
4	AP	27.75	17.44	34.01
5	BA	18.77	10.12	26.36
6	CE	20.54	10.26	32.71
7	DF	12.5	11.27	21.04
8	ES	15.19	9.77	22.06
9	GO	14.95	11.37	22.77
10	MA	21.2	9.11	38.26

4. **Sort the data to get the following:**
    a. **Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5**

Query for top 5 states with highest average freight value:

```
SELECT customer_state,
  ROUND(AVG(freight_value), 2) as avg_freight_value
FROM `target--sql.TargetDATA.orders` o
  INNER JOIN `target--sql.TargetDATA.customers` c ON c.customer_id =
o.customer_id
  INNER JOIN `target--sql.TargetDATA.order_items` oi ON oi.order_id =
o.order_id
GROUP BY customer_state
ORDER BY avg_freight_value DESC
LIMIT 5
```

Result:



Row	customer_state	avg_freight_value
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15

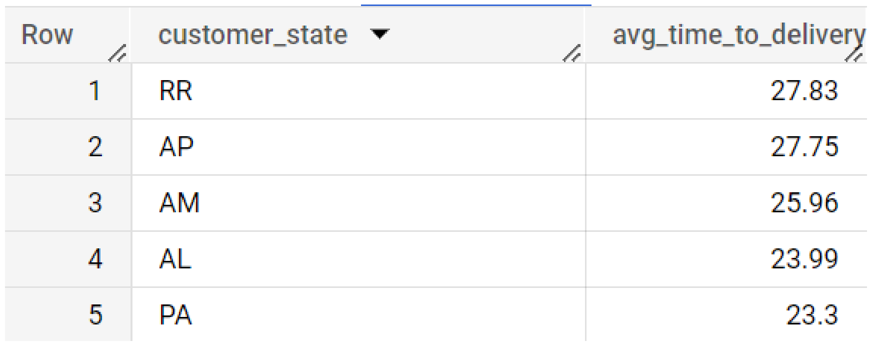
A similar query would run for the top 5 states with lowest average freight value, only the 'ORDER BY' clause would contain ASC instead of DESC.

b. **Top 5 states with highest/lowest average time to delivery**

Query for top 5 states with highest average time to delivery:

```
SELECT customer_state,
  ROUND(AVG(DATE_DIFF(order_delivered_customer_date,
order_purchase_timestamp, DAY)), 2) AS avg_time_to_delivery
FROM `target--sql.TargetDATA.orders` o
  INNER JOIN `target--sql.TargetDATA.customers` c ON c.customer_id =
o.customer_id
  INNER JOIN `target--sql.TargetDATA.order_items` oi ON oi.order_id =
o.order_id
GROUP BY customer_state
ORDER BY avg_time_to_delivery DESC
LIMIT 5
```

Result:



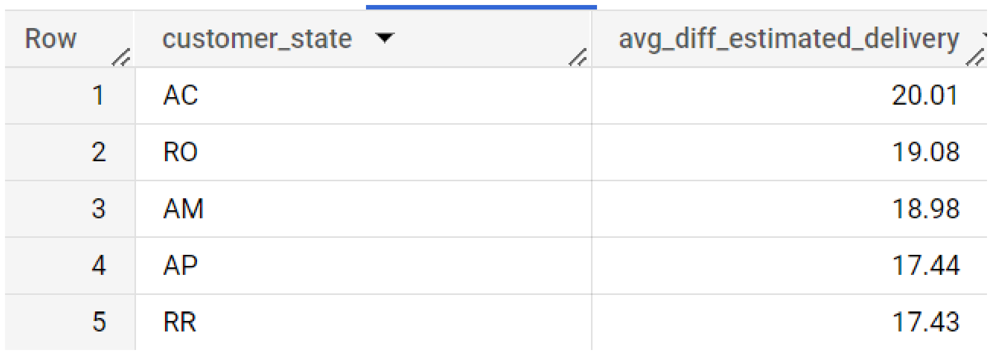
Row	customer_state	avg_time_to_delivery
1	RR	27.83
2	AP	27.75
3	AM	25.96
4	AL	23.99
5	PA	23.3

c. **Top 5 states where delivery is really fast/ not so fast compared to estimated date**

Query for fast delivery:

```
SELECT customer_state,
  ROUND(AVG(DATE_DIFF(order_estimated_delivery_date,
order_delivered_customer_date, DAY)), 2) AS avg_diff_estimated_delivery
FROM `target--sql.TargetDATA.orders` o
  INNER JOIN `target--sql.TargetDATA.customers` c ON c.customer_id =
o.customer_id
  INNER JOIN `target--sql.TargetDATA.order_items` oi ON oi.order_id =
o.order_id
GROUP BY customer_state
ORDER BY avg_diff_estimated_delivery DESC
LIMIT 5
```

Result:



Row	customer_state	avg_diff_estimated_delivery
1	AC	20.01
2	RO	19.08
3	AM	18.98
4	AP	17.44
5	RR	17.43

Insights:

- ❖ The calculation of estimated time to delivery is heavily inaccurate.
- ❖ Although it is good that the company delivers the orders days in advance, but that could easily turn to be a hassle for customers as they cannot plan (for example, their outings or when to start usage of the product delivered, etc) accordingly. Plus, it leaves a good opportunity for the competitors to capture the market.

Recommendations:

- ❖ It could be analysed if it is profitable to source local sellers more, as this would further reduce both freight value and time to delivery.
- ❖ The customer demands can be studied according to the location and types of items ordered and the data can then be used to stock those estimated products to the nearby warehouse. This would reduce the customer wait-time and also the freight value, since the products will be brought in bulk and the cost per item will decrease.
- ❖ The calculation for the estimated time to delivery needs to be precise. Else, this can be frustrating for the customers.

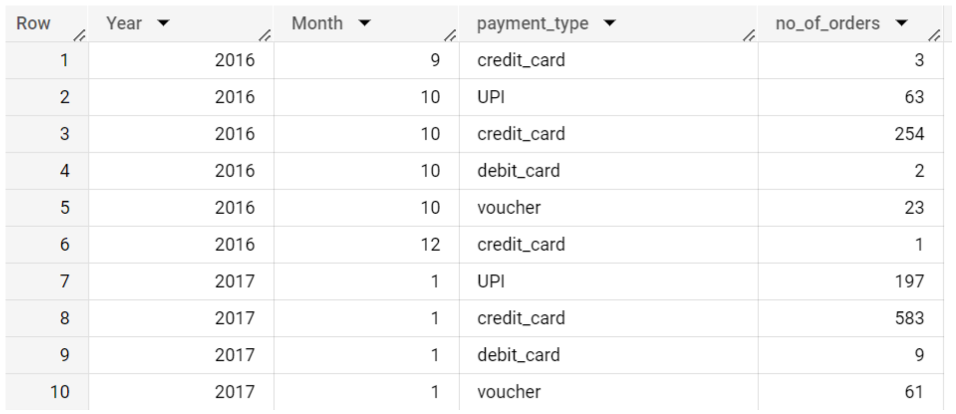
6. Payment type analysis

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

   Query:

```
WITH cte1 as (
  SELECT payment_type,
    EXTRACT(YEAR FROM order_purchase_timestamp) AS Year,
    EXTRACT(MONTH FROM order_purchase_timestamp) AS Month,
    p.order_id
  FROM `target--sql.TargetDATA.payments` p
    INNER JOIN `target--sql.TargetDATA.orders` o on o.order_id = p.order_id
)
SELECT Year, Month, payment_type, COUNT(order_id) AS no_of_orders
FROM cte1
GROUP BY Year, Month, payment_type
ORDER BY Year, Month, payment_type
```

The first 10 rows of the result are:



Row	Year	Month	payment_type	no_of_orders
1	2016	9	credit_card	3
2	2016	10	UPI	63
3	2016	10	credit_card	254
4	2016	10	debit_card	2
5	2016	10	voucher	23
6	2016	12	credit_card	1
7	2017	1	UPI	197
8	2017	1	credit_card	583
9	2017	1	debit_card	9
10	2017	1	voucher	61

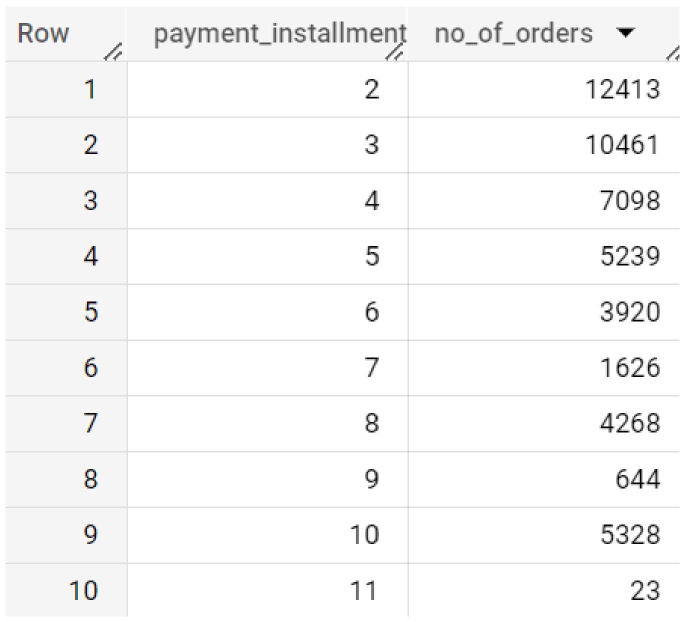
2. **Count of orders based on the no. of payment installments**

<u>Query:</u>

```
SELECT payment_installments, COUNT(order_id) AS no_of_orders
FROM `target--sql.TargetDATA.payments`
WHERE payment_installments > 1
GROUP BY payment_installments
ORDER BY payment_installments
```

The payment installments of 0 and 1 are excluded as 0 could involve voucher payments and 1 means one-time payment.

<u>Result:</u>



Row	payment_installment	no_of_orders
1	2	12413
2	3	10461
3	4	7098
4	5	5239
5	6	3920
6	7	1626
7	8	4268
8	9	644
9	10	5328
10	11	23

Insights:

- Credit cards are definitely the most frequently used payment type. This can be verified using a quick query on the number of orders for each payment type.
- Also, in credits, people are willing to opt for EMI options for buying. This suggests that the consumer market in Brazil is flourishing.

Recommendations:

- The EMIs can be facilitated by the platform itself. Options like EMI payments, Buy Now Pay Later, etc can be introduced.