Sree Mrudhula . B

Initial analysis of the e-commerce data of Target from Brazil from the period 2016-2018.

TARGET SQL

Analysis of the dataset through Bigquery

Contents

[1 Introduction 3](#_Toc162962596)

[1.1 Datasets 3](#_Toc162962597)

[2 Import the dataset and do the usual exploratory analysis steps like checking the structure & characteristics of the dataset 4](#_Toc162962598)

[2.1 Data type of all columns in the "customers" table 4](#_Toc162962599)

[2.2 Get the time range between which the orders were placed. 6](#_Toc162962600)

[2.3 Count the Cities & States of customers who ordered during the given period 6](#_Toc162962601)

[3 In-depth Exploration: 8](#_Toc162962602)

[3.1 Is there a growing trend in the no. of orders placed over the past years? 8](#_Toc162962603)

[3.1.1 Growing trend in the no. of orders placed over the past years (all orders) 8](#_Toc162962604)

[3.1.2 The trend of the only successfully delivered orders 8](#_Toc162962605)

[3.2 Can we see some kind of monthly seasonality in terms of the no. of orders being placed? 10](#_Toc162962606)

[3.3 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) 12](#_Toc162962607)

[3.3.1 Creating a temporary table with a new column that represents the part of the day based on the purchase time 12](#_Toc162962608)

[3.3.2 Using CTE using the temporary table created and grouping the orders based on the part of the day 12](#_Toc162962609)

[4 Evolution of E-commerce orders in the Brazil region 14](#_Toc162962610)

[4.1 Get the month on month no. of orders placed in each state 14](#_Toc162962611)

[4.1.1 Calculating the number of orders wrt ach city and state 14](#_Toc162962612)

[4.1.2 Comparing and Calculating the % difference in the number of orders with orders of the next month of each state 15](#_Toc162962613)

[4.1.3 Month on month orders and the revenue generated for each state 16](#_Toc162962614)

[4.2 How are the customers distributed across all the states? 18](#_Toc162962615)

[4.2.1 Checking for outliers .i.e: if there are any orders without the customer details or any customers who have placed no orders 18](#_Toc162962616)

[4.2.2 Number of customers across each state 18](#_Toc162962617)

[5 Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight, and others 20](#_Toc162962618)

[5.1 Get the % increase in the cost of orders from the 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. 20](#_Toc162962619)

[5.1.1 Getting the orders from 2017 to 2018 from months January to August only 20](#_Toc162962620)

[5.1.2 Getting the cost of orders wrt year-month 21](#_Toc162962621)

[5.1.3 Displaying the cost of orders from 2017 to 2018(include months between JAN and AUG only) 21](#_Toc162962622)

[**5.1.4** Calculating %increase in the cost of orders **2017 month on 2018 month** 23](#_Toc162962623)

[5.1.5 Calculating the % increase in the cost of orders from year 2017 to 2018 **YEAR-WISE** (include months between Jan to Aug only) 24](#_Toc162962624)

[5.2 Calculate the Total & Average value of the order price for each state. 25](#_Toc162962625)

[5.3 Calculate the Total & Average value of order freight for each state. 28](#_Toc162962626)

[6 Analysis based on sales, freight, and delivery time. 31](#_Toc162962627)

[6.1 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. 31](#_Toc162962628)

[6.2 Find out the top 5 states with the highest & lowest average freight value 33](#_Toc162962629)

[6.2.1 Top 5 states with the highest average freight value 33](#_Toc162962630)

[6.2.2 Top 5 states with the lowest average freight value 34](#_Toc162962631)

[6.3 Find out the top 5 states with the highest & lowest average delivery time. 35](#_Toc162962632)

[6.3.1 Top 5 states with the highest average delivery time 35](#_Toc162962633)

[6.3.2 Top 5 states a lowest average delivery time 36](#_Toc162962634)

[6.4 Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery. 37](#_Toc162962635)

[7 Analysis based on the payments 39](#_Toc162962636)

[7.1 Find the month-on-month no. of orders placed using different payment types 39](#_Toc162962637)

[7.2 Find the no. of orders placed based on the payment instalments that have been paid. 41](#_Toc162962638)

# Introduction

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

This particular business case focuses on the operations of Target in Brazil and provides insightful information about 100,000 orders placed between 2016 and 2018. The dataset offers a comprehensive view of various dimensions including the order status, price, payment and freight performance, customer location, product attributes, and customer reviews.

By analyzing this extensive dataset, it becomes possible 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.

## Datasets

1. Customers Table: It has details of all the customers with their customer id, a unique customer id assigned by Target(probably) which acts as a Primary key(PK). Along with those, there is information about the customer's city, state, and zip code.
2. Sellers Table: It has a unique seller id issued by Target which is the Primary key(PK), there are details of the zip code of the sellers’ city, city and state.
3. Order\_items Table: There is information about the order id, id for each order item, product id, seller id, shipping limit date, price and freight details. The Primary key will combination of the order id, and order item id.
4. Geolocation Table: This has information about the location of the orders the latitude and longitude values along with their zip codes. There is no Primary key(PK) in this table, if required one has to generate it.
5. Payments Table: This table has information about the payment of each order. It also has information of payment type, installments, and payment sequential. The Primary key(PK) here is the combination of order id and payment sequential.
6. Orders Table: This table has information about the orders of each customer. It has the customer id, order id, time details, and delivery details. The Primary key (PK) is the order id.
7. Reviews Table: This has information about the review of the order given by each customer. It has the comment title, message, date and time details. The Primary key here is the combination of review id and order id.
8. Products Table: This has information about each product. It has unique product id, category details, descriptions, and attributes of the products. The Primary key(PK) is the product id.

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

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

Query:

SELECT

       COLUMN\_NAME, DATA\_TYPE

FROM

       targetsql.INFORMATION\_SCHEMA.COLUMNS

WHERE

       TABLE\_NAME = 'customers';

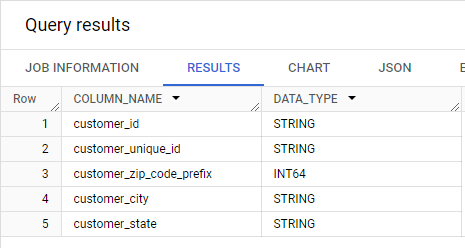


Figure 2—1: Data types of customers’ table

Similarly exploring other tables of the dataset data types

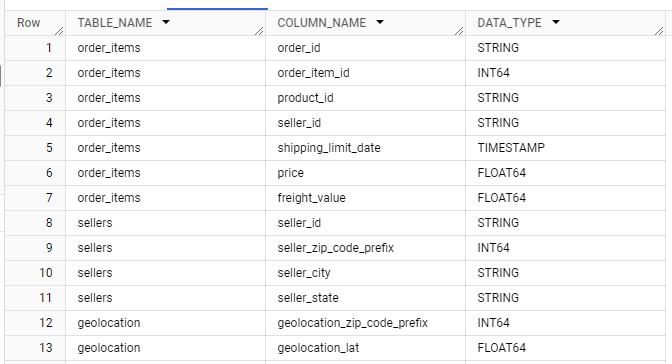
Query:

SELECT

       TABLE\_NAME, COLUMN\_NAME, DATA\_TYPE

FROM

       targetsql.INFORMATION\_SCHEMA.COLUMNS



With the datatypes thus obtained the detailed SCHEMA of the TARGET DATA SET

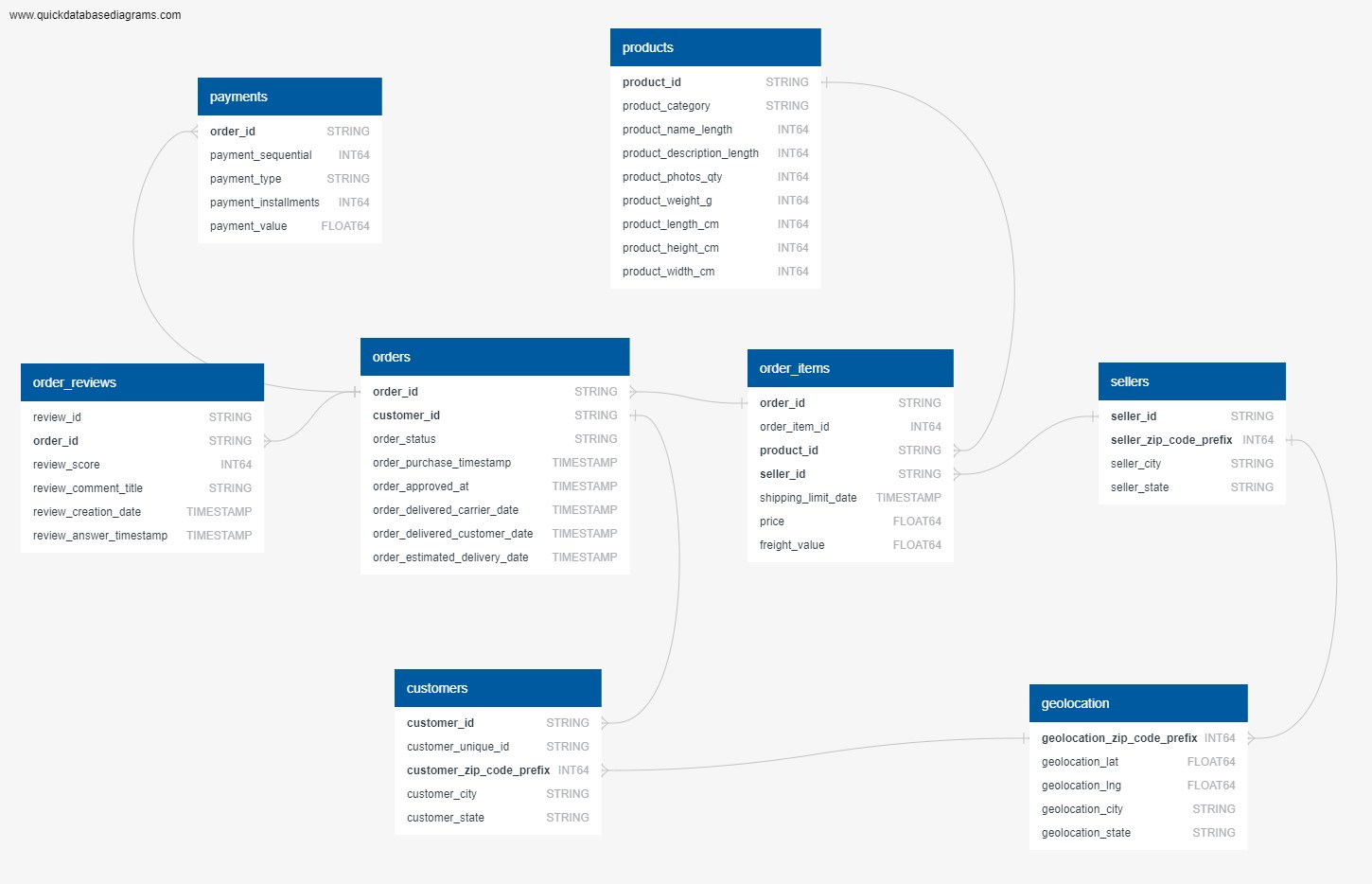


Figure 2—2: Schema of the dataset

INSIGHTS/RECOMMENDATIONS:

1. From the data it could be derived that the customer id, city, and state are all in STRING format.
2. The customer zip code is in INT i.e.; integer format.
3. Also all other data that contains text as input in is STRING format, numbers are in INT64 format and decimals are in FLOAT64 format.

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

Query:

SELECT MIN(order\_purchase\_timestamp) AS start\_date,

       MAX(order\_purchase\_timestamp) AS end\_date,

       DATE\_DIFF(MAX(order\_purchase\_timestamp), MIN(order\_purchase\_timestamp), DAY) AS time\_range

FROM

`targetsql.orders`;



Figure 2—3: Time range of the orders placed

INSIGHTS/RECOMMENDATIONS:

1. The start date of the dataset is 2016 9th of April and the end date is 2018 17th of October.
2. The dataset is over a period of 772 days.

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

Query:

-- Getting a list of all cities and countries of customers who placed orders during the period of 772 days.

SELECT

       distinct c.customer\_city, c.customer\_state

FROM

       `targetsql.customers` c

JOIN

`targetsql.orders` o ON c.customer\_id = o.customer\_id

order by

       c.customer\_state;

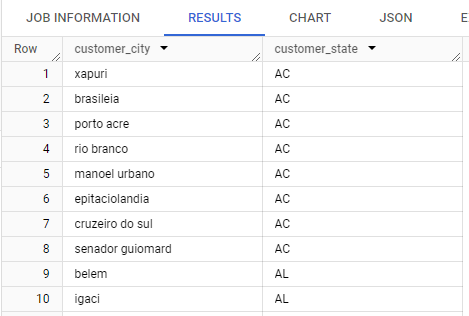


Figure 2—4:List of all cities and states

--Getting the count of cities and countries

SELECT count(distinct c.customer\_city) as city\_count,

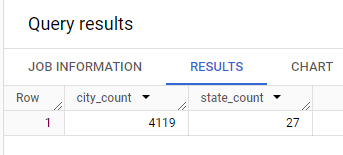
       count(distinct c.customer\_state) as state\_count

FROM

       `targetsql.customers` c

JOIN

       `targetsql.orders` o ON c.customer\_id = o.customer\_id;



INSIGHTS/RECOMMENDATIONS:

1. Customers from 4119 distinct cities and 27 states had ordered from the Target store.
2. This gives us the information of the areas where the orders are placed, giving us the scope to increase the e-commerce from the areas where there are few or no orders.

# ****In-depth Exploration:****

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

### Growing trend in the no. of orders placed over the past years (all orders)

Query:

SELECT EXTRACT(YEAR FROM order\_purchase\_timestamp) as years,

       EXTRACT(MONTH FROM order\_purchase\_timestamp) as months,

       count(\*) as num\_of\_orders

FROM

       `targetsql.orders`

GROUP BY

       1,2

ORDER BY

       years, months;

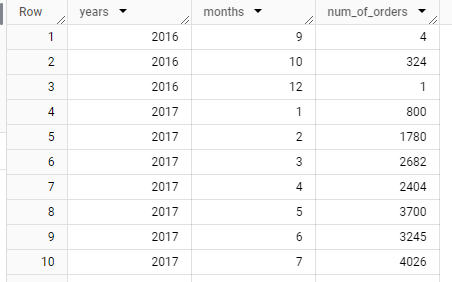


Figure 3—1: Trend in the orders placed over the years

### The trend of the only successfully delivered orders

Query:

SELECT EXTRACT(YEAR FROM order\_purchase\_timestamp) as years,

       EXTRACT(MONTH FROM order\_purchase\_timestamp) as months,

       count(\*) as num\_of\_orders

FROM

       `targetsql.orders`

WHERE

       lower(order\_status) = 'delivered'

GROUP BY

       1,2

ORDER BY

       1,2;

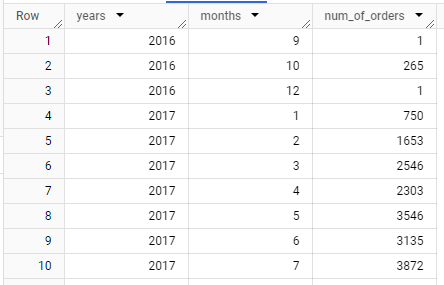


Figure 3—2: Month on Month of num of orders delivered

Figure 3—3: Month on month graph of orders delivered

INSIGHTS:

1. The numbers of orders per month is nothing but the sales that Target has made in that particular month
2. From the results it can be observed that the maximum sales happened in the year 2017 in the month of November, the next highest in the year 2018 Jan probably due to New Year offers.
3. It can be observed that the sales have steadily increased from 2017 January to December.
4. A similar case is noticed in the 2018 year.

RECOMMENDATIONS:

1. An increase in e-commerce has been noticed from the period of 2017-2018.
2. To keep the rate of sales at a steady rate, recommendations/suggestions for the orders regularly purchased can be shown to customers.

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

Query:

SELECT EXTRACT(YEAR FROM order\_purchase\_timestamp) as years,

       EXTRACT(MONTH FROM order\_purchase\_timestamp) as months,

       count(\*) as num\_orders

FROM

       `targetsql.orders`

GROUP BY

       1,2

ORDER BY

       3 DESC;



Figure 3—4: Peaks in the sales

--For orders the orders that are sucessfully delivered

SELECT EXTRACT(YEAR FROM order\_purchase\_timestamp) as years,

       EXTRACT(MONTH FROM order\_purchase\_timestamp) as months,

       count(\*) as num\_orders

FROM

       `targetsql.orders`

WHERE

       lower(order\_status) = 'delivered'

GROUP BY

       1,2

ORDER BY

       3 DESC;

Figure 3—5: Growth trend of orders delivered

INSIGHTS:

1. From the graph based on the result it can be observed that there has been an increase in sales from 2016- 2018.
2. In 2016 the sales were low might be due to the limited data available, a few other reasons could be the recession, political reforms, or less reach.
3. Peaks in sales can be observed in months October, and December months wrt the given data.
4. Peak season is observed from September to February in the above graph.

RECOMMENDATIONS:

1. Offers: To increase sales in other seasons, offers such as combo, sales could bring up the number of sales
2. Seasonal Products: Including seasonal products in the recommendation of regular customers and other outreach programs might also help increase the numbers in sales.

## 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:

### Creating a temporary table with a new column that represents the part of the day based on the purchase time

--Adding a new column that gives info about the part of the day

select order\_id, order\_purchase\_timestamp, time(order\_purchase\_timestamp) as order\_time,

       CASE

       WHEN time(order\_purchase\_timestamp) between '00:00:00' and '06:59:59' then 'Dawn'

       WHEN time(order\_purchase\_timestamp) between '07:00:00' and '12:59:59' then 'Mornings'

       WHEN time(order\_purchase\_timestamp) between '13:00:00' and '18:59:59' then 'Afternoon'

       WHEN time(order\_purchase\_timestamp) between '19:00:00' and '23:59:59' then 'Night'

       end as time\_of\_day

from

       `targetsql.orders`

order by

       order\_purchase\_timestamp;

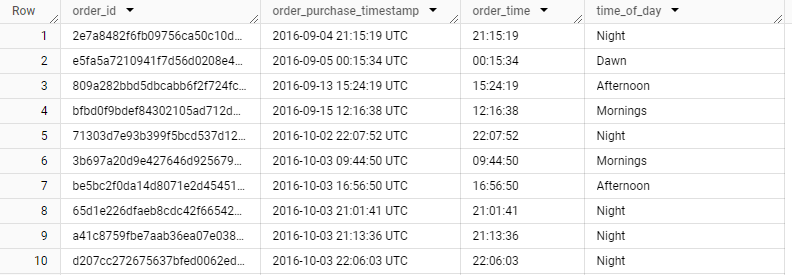


Figure 3—6: Orders table along with part of the day column

### Using CTE using the temporary table created and grouping the orders based on the part of the day

--count of orders during the part of the day

WITH orders\_time\_day as (select order\_id, order\_purchase\_timestamp, time(order\_purchase\_timestamp) as order\_time,

       CASE

       WHEN time(order\_purchase\_timestamp) between '00:00:00' and '06:59:59' then 'Dawn'

       WHEN time(order\_purchase\_timestamp) between '07:00:00' and '12:59:59' then 'Mornings'

       WHEN time(order\_purchase\_timestamp) between '13:00:00' and '18:59:59' then 'Afternoon'

       WHEN time(order\_purchase\_timestamp) between '19:00:00' and '23:59:59' then 'Night'

       end as time\_of\_day

from

       `targetsql.orders`

order by

       order\_purchase\_timestamp)

SELECT o.time\_of\_day,

       COUNT(\*) as num\_orders\_prt\_day

FROM

       orders\_time\_day o

GROUP BY

       o.time\_of\_day

ORDER BY

       num\_orders\_prt\_day DESC;

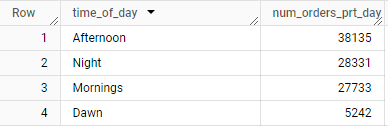


Figure 3—7: Number of orders based on part of the day

INSIGHTS:

1. From the result it is observed that a maximum number of orders are placed in the afternoon time.
2. The least orders are placed during dawn.
3. A similar kind of sales is seen in the mornings and afternoons based on the given data

RECOMMENDATIONS:

1. Sales Offers: Early bird offers during the early hours, and midnight sales offers could boost the sales at dawn.
2. Discount on Fresh Produce during early hours could increase the sales in the 0-6 hrs.

# ****Evolution of E-commerce orders in the Brazil region****

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

Query:

### Calculating the number of orders wrt ach city and state

--Displaying the number of orders placed by each state wrt each year and month

SELECT

       c.customer\_state, t.years, t.months, count(\*) as orders\_per\_state

FROM

       `targetsql.customers` c

JOIN (SELECT EXTRACT(YEAR FROM order\_purchase\_timestamp) as years,

              EXTRACT(MONTH FROM order\_purchase\_timestamp) as months,

              customer\_id

       FROM `targetsql.orders`) t

ON

       c.customer\_id = t.customer\_id

GROUP BY

       c.customer\_state, t.years, t.months

order by

       t.years, t.months;

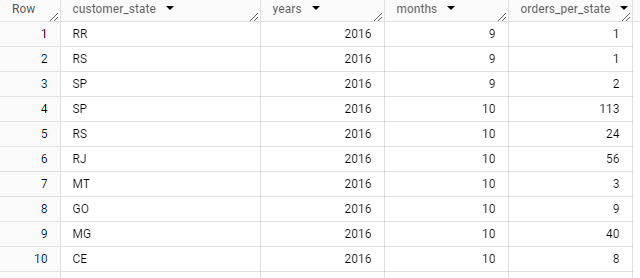


Figure 4—1: Number of orders of each state per month

--comparison based on months only

SELECT

       c.customer\_state, t.months,

       count(\*) as orders\_per\_state

FROM

       `targetsql.customers` c

JOIN (

       SELECT EXTRACT(YEAR FROM order\_purchase\_timestamp) as years,

              EXTRACT(MONTH FROM order\_purchase\_timestamp) as months,

              customer\_id

              FROM `targetsql.orders`) t

ON

       c.customer\_id = t.customer\_id

GROUP BY

       c.customer\_state, t.months

order by

       c.customer\_state, t.months;

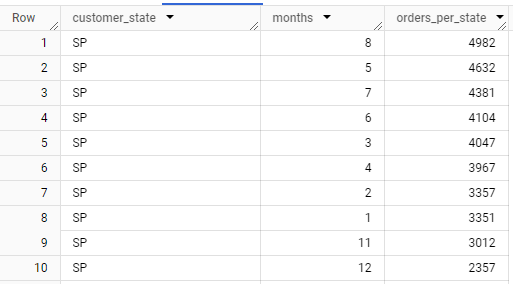


Figure 4—2: Month on month orders

Figure 4—3: Graph of orders wrt each state

### Comparing and Calculating the % difference in the number of orders with orders of the next month of each state

Query:

WITH month\_wise as (

SELECT

       c.customer\_state as state, t.years as years, t.months as months, count(\*) as orders\_per\_state

FROM

       `targetsql.customers` c

JOIN

       (SELECT

              EXTRACT(YEAR FROM order\_purchase\_timestamp) as years,

              EXTRACT(MONTH FROM order\_purchase\_timestamp) as months,

              customer\_id

       FROM

              `targetsql.orders`) t ON c.customer\_id = t.customer\_id

GROUP BY

       c.customer\_state, t.years, t.months

)

SELECT

       \*,

       ROUND((nm.orders\_per\_state -pm.orders\_per\_state)/pm.orders\_per\_state \* 100, 2) as prec\_diff

FROM

       month\_wise pm

JOIN

       month\_wise nm on pm.state = nm.state

WHERE

       pm.years = nm.years and nm.months = pm.months+1

ORDER BY

       pm.state, pm.years, pm.months;

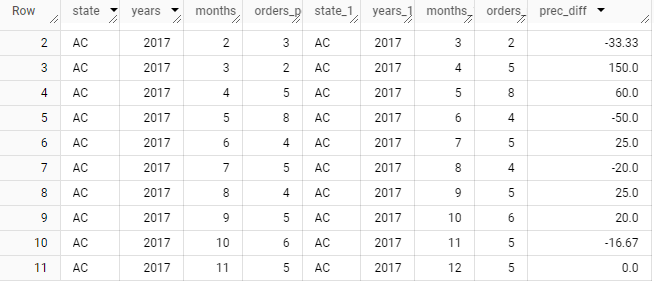


Figure 4—4; Percentage difference in order wrt next month

### Month on month orders and the revenue generated for each state

Query:

--Month on month orders vs the revenue it generated for each state

SELECT

       c.customer\_state, t.years, t.months,

       count(\*) as orders\_per\_state,

       ROUND(SUM(p.payment\_value),3) as total\_revenue\_per\_month

FROM

       `targetsql.customers` c

JOIN (

       SELECT EXTRACT(YEAR FROM order\_purchase\_timestamp) as years,

              EXTRACT(MONTH FROM order\_purchase\_timestamp) as months,

              customer\_id, order\_id

              FROM `targetsql.orders`) t ON c.customer\_id = t.customer\_id

JOIN

       `targetsql.payments` p ON p.order\_id = t.order\_id

GROUP BY

       c.customer\_state, t.years, t.months

order by

       3 DESC, 4 DESC;

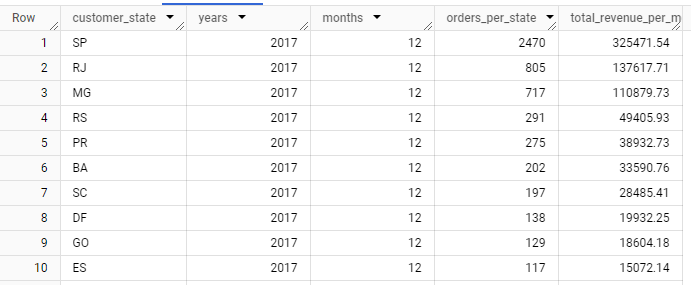


Figure 4—5: Month on month orders vs revenue it generated

INSIGHT:

1. From the data it can be observed that the state (SP) São Paulo has recorded the maximum number of sales as well as revenue
2. Followed by the states Rio de Janeiro (RJ) and Minas Gerais (MG).
3. It can also be observed that the highest sales happened in the year 2017 of the Month of December from SP state.
4. If observed highest sales had been recorded in many states in December, this may be due to the Christmas and New Year sales and offers.

RECOMMENDATIONS:

1. Lowest sales are observed in the states RR, AP and AC with a total of 46,68,81 orders respectively, Outreach programs could help increase sales in these places
2. Advertising and Discounts in membership could help increase the customer base in such areas.

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

### Checking for outliers .i.e: if there are any orders without the customer details or any customers who have placed no orders

Query:

SELECT \*

FROM `targetsql.customers` c full outer join `targetsql.orders` o

on c.customer\_id = o.customer\_id

where o.order\_id is null or c.customer\_id is null;

### Number of customers across each state

Query:

select

       customer\_state, count(\*) as no\_of\_cust

from

       `targetsql.customers`

group by

       customer\_state

order by

       2 DESC;



Figure 4—6: No of customers per each state

Figure 4—7: Graph of state vs customers

INSIGHTS:

1. From the result it can be observed that in the state (SP) São Paulo, TARGET has a large customer base. Due to this large sales were generated which resulted in a high % revenue from the state.
2. The next highest is RJ and MG which has produced the next highest sales and revenue.
3. TARGET has a low customer base in the states of RR, AP, and AC.

RECOMMENDATIONS:

1. Region-specific offers: To focus on low customer base areas, offering discounts on the products that are most consumed in their areas such as staples and other appliances might increase the customer base and sales

# ****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 the 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.

### Getting the orders from 2017 to 2018 from months January to August only

Query:

SELECT \*

FROM (SELECT order\_id,

       EXTRACT(YEAR FROM order\_purchase\_timestamp) as years,

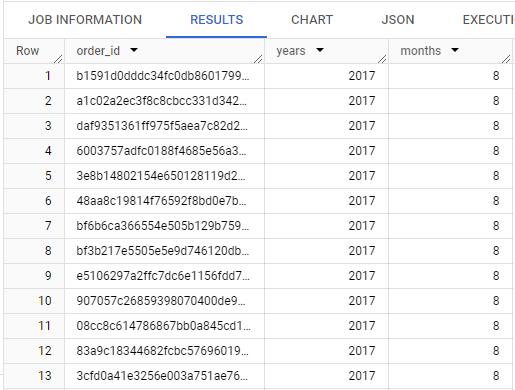
       EXTRACT(MONTH FROM order\_purchase\_timestamp) as months

       FROM `targetsql.orders`) tbl

WHERE (tbl.years = 2017 or tbl.years = 2018)

      AND tbl.months between 01 and 08

ORDER BY tbl.years, tbl.months DESC;



### Getting the cost of orders wrt year-month

Query:

--Getting the cost of orders wrt year, month.

WITH orders\_YM as (SELECT order\_id,

       EXTRACT(YEAR FROM order\_purchase\_timestamp) as years,

       EXTRACT(MONTH FROM order\_purchase\_timestamp) as months,

       order\_status

FROM

`targetsql.orders`)

SELECT

o.years,o.months,

       ROUND(SUM(payment\_value),3) as cost\_of\_orders\_mnthwise

FROM

orders\_YM o

JOIN

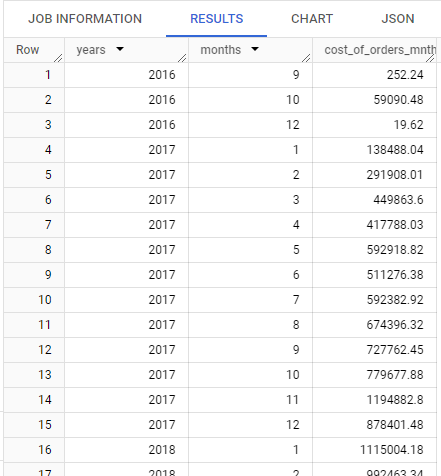
`targetsql.payments` p ON o.order\_id = p.order\_id

GROUP BY

o.years,o.months

ORDER BY

o.years,o.months;



### Displaying the cost of orders from 2017 to 2018(include months between JAN and AUG only)

Query:

-- The cost of orders from year 2017 to 2018 (include months between Jan to Aug only)

WITH orders\_YM as (SELECT order\_id,

                          EXTRACT(YEAR FROM order\_purchase\_timestamp) as years,

                          EXTRACT(MONTH FROM order\_purchase\_timestamp) as months,

                          FORMAT\_DATETIME("%Y - %m", order\_purchase\_timestamp) as year\_month,

                          order\_purchase\_timestamp

                   FROM

`targetsql.orders`)

SELECT

o.year\_month,

       ROUND(SUM(payment\_value),3) as cost\_of\_orders\_mnthwise

FROM

orders\_YM o

JOIN `targetsql.payments` p ON o.order\_id = p.order\_id

WHERE

(o.years = 2017 or o.years = 2018)

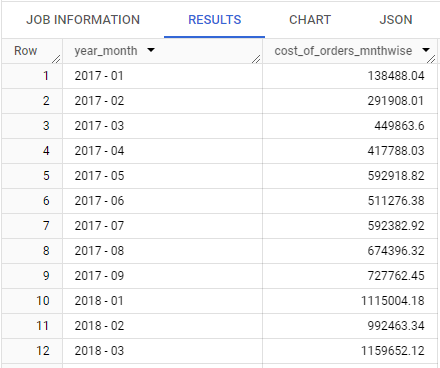
      AND o.months between 01 and 09

GROUP BY

o.year\_month

ORDER BY

o.year\_month;



### Calculating %increase in the cost of orders **2017 month on 2018 month**

Query:

--% of increase in cost of orders month-wise from 2017 to 2018

WITH orders\_YM as (SELECT

                          order\_id,

                          EXTRACT(YEAR FROM order\_purchase\_timestamp) as years,

                          EXTRACT(MONTH FROM order\_purchase\_timestamp) as months,

                          FORMAT\_DATETIME("%Y - %m", order\_purchase\_timestamp) as year\_month,

                          order\_purchase\_timestamp

                   FROM

                     `targetsql.orders`),

cost\_of\_order\_1718 as (SELECT

                            o.years as YEAR, o.months as MONTH,

                            ROUND(SUM(payment\_value),3) as cost\_of\_orders\_mnthwise

                       FROM

                            orders\_YM o

                       JOIN

                            `targetsql.payments` p  ON o.order\_id = p.order\_id

                       WHERE

                            (o.years = 2017 or o.years = 2018)

                            AND o.months between 01 and 08

                       GROUP BY

                            o.years, o.months

                       ORDER BY

                            o.years, o.months )

SELECT

       p.YEAR, p.MONTH, p.cost\_of\_orders\_mnthwise, n.YEAR as NXT\_YR, n.MONTH as NXT\_MNTH,

n.cost\_of\_orders\_mnthwise,

       ROUND((n.cost\_of\_orders\_mnthwise - p.cost\_of\_orders\_mnthwise)/p.cost\_of\_orders\_mnthwise \* 100,2) as `%increase\_monthwise`

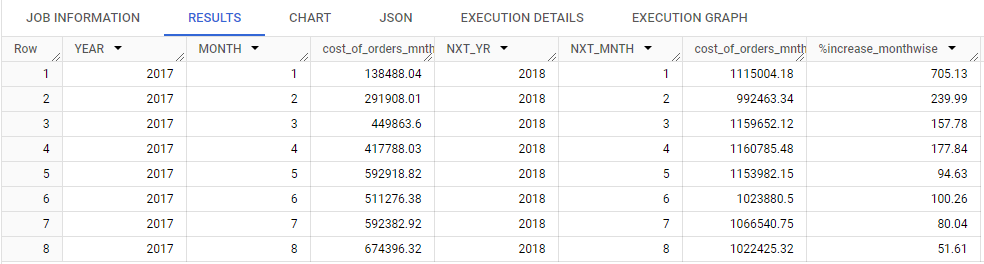
FROM

       cost\_of\_order\_1718 p

JOIN

       cost\_of\_order\_1718 n   ON  p.YEAR = 2017 and n.YEAR = 2018 and n.MONTH = p.MONTH

ORDER BY 2;



### Calculating the % increase in the cost of orders from year 2017 to 2018 **YEAR-WISE** (include months between Jan to Aug only)

Query:

--% of increase in cost of orders year-wise from 2017 to 2018

WITH orders\_YM as

       (SELECT

              order\_id,

              EXTRACT(YEAR FROM order\_purchase\_timestamp) as years,

              EXTRACT(MONTH FROM order\_purchase\_timestamp) as months,

              FORMAT\_DATETIME("%Y - %m", order\_purchase\_timestamp) as year\_month,

              order\_purchase\_timestamp

       FROM

              `targetsql.orders`)

SELECT

       o.years as YEAR,

       ROUND(SUM(payment\_value),3) as cost\_of\_orders,

       CASE

              WHEN o.years = 2017 THEN 0

              ELSE ROUND((SUM(CASE WHEN o.years = 2018 THEN p.payment\_value ELSE 0 END)- LAG(ROUND(SUM(payment\_value),3)) over(order by o.years))/ LAG(ROUND(SUM(payment\_value),3)) over(order by o.years) \*100, 2)

       END as nxt\_yr\_cost

FROM

       orders\_YM o

JOIN

       `targetsql.payments` p  ON o.order\_id = p.order\_id

WHERE

       (o.years = 2017 or o.years = 2018)

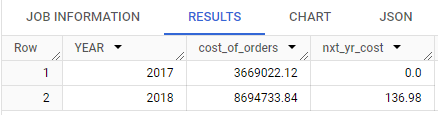
       AND o.months between 01 and 08

GROUP BY

       o.years

ORDER BY

       o.years;



INSIGHTS:

1. From the result it can be observed that the % difference in the orders where high in the first few months.
2. Gradually the %difference started to reduce. Which could mean an increase in the number of orders by the customers.
3. Also a hike of 136.9% is observed in the number of orders from 2017 to 2018 in the asked time range.

RECOMMENDATIONS:

1. Apart from gathering a new customer base, Maintaining the customer base and the sales is also important, providing suggestions and recommendations to the regular customers based on their purchases. It may help in maintaining sales.

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

Query:

--Displaying the Total & Average value of order price for each state and city

SELECT

       c.customer\_state, c.customer\_city,

       ROUND(SUM(p.payment\_value), 2) as total\_price\_city,

       ROUND(AVG(p.payment\_value),2) as avg\_price\_city

FROM

       `targetsql.customers` c

JOIN

       `targetsql.orders` o ON c.customer\_id = o.customer\_id

JOIN

       `targetsql.payments` p ON o.order\_id = p.order\_id

GROUP BY

       c.customer\_state, c.customer\_city

ORDER BY

       total\_price\_city desc, avg\_price\_city;

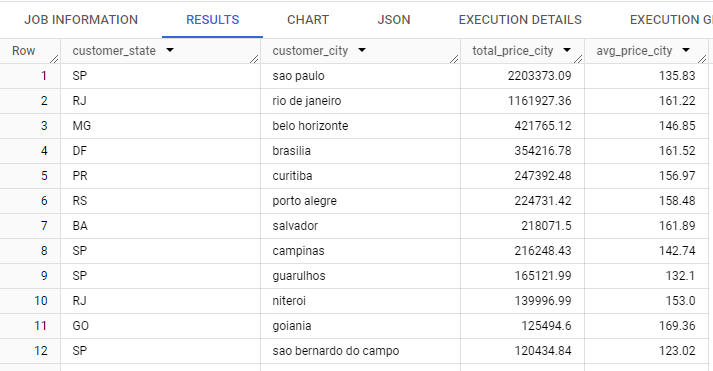


Figure 5—1: Total and average price of orders per city

--State wise

SELECT

       c.customer\_state,

       ROUND(SUM(p.payment\_value), 2) as total\_price,

       ROUND(AVG(p.payment\_value),2) as avg\_price

FROM

       `targetsql.customers` c

JOIN

       `targetsql.orders` o ON c.customer\_id = o.customer\_id

JOIN

       `targetsql.payments` p ON o.order\_id = p.order\_id

GROUP BY

       c.customer\_state

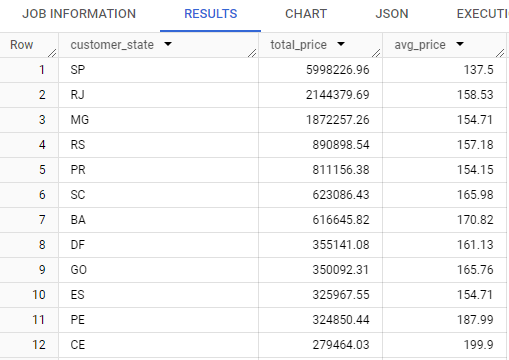


Figure 5—2: Total and average price of orders per state

Figure 5—3: States vs Total price

Figure 5—4: States vs Average price

INSIGHTS:

1. It can be observed from the results that the state SP has the highest total price among all the states but interestingly the average price of the sales is not high for SP.
2. The average price is highest for the state PB, which may indicate that the orders from the state SP might be of higher value but the number of orders is higher in PB.

RECOMMENDATIONS:

1. Encouraging customers to purchase high-value items through e-commerce needs a certain level of trust factor. Engaging resources there might aid the customers in purchasing high-value items.
2. Trust can be gained by guaranteed returns, trusted brand partnering, and seamless service.

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

Query:

--Displaying the Total & Average value of order freight for each state and city

SELECT

       c.customer\_state, customer\_city,

       ROUND(SUM(oi.freight\_value),2) as total\_freight\_city,

       ROUND(AVG(oi.freight\_value),2) as avg\_freight\_city

FROM

       `targetsql.customers` c

JOIN

       `targetsql.orders` o ON c.customer\_id = o.customer\_id

JOIN

       `targetsql.order\_items` oi ON o.order\_id = oi.order\_id

GROUP BY

       c.customer\_state, customer\_city

ORDER BY

       total\_freight\_city desc, avg\_freight\_city;

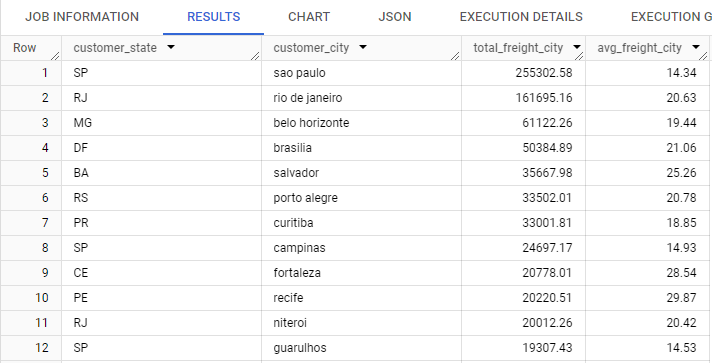


Figure 5—5: Total and mean freight values per city

--Statewise

SELECT

       c.customer\_state,

       ROUND(SUM(oi.freight\_value),2) as total\_freight,

       ROUND(AVG(oi.freight\_value),2) as avg\_freight

FROM

       `targetsql.customers` c

JOIN

       `targetsql.orders` o ON c.customer\_id = o.customer\_id

JOIN

       `targetsql.order\_items` oi ON o.order\_id = oi.order\_id

GROUP BY

       c.customer\_state

ORDER BY

       total\_freight desc, avg\_freight;

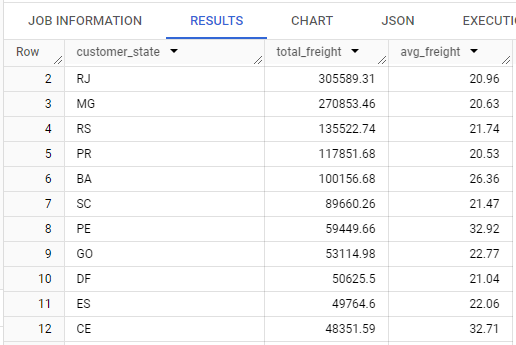


Figure 5—6: Total and Mean freight values per state

Figure 5—7: States vs total freight value

Figure 5—8: States vs mean freight price

INSIGHTS:

1. It can be observed from the results that the state SP has the highest total freight among all the states but interestingly the average freight of the sales is not high for SP.
2. The average freight is highest for the state PB.

RECOMMENDATIONS:

1. Reducing the cost of shipping could incentivize customers residing in remote locations to place more orders, which could ultimately lead to an increase in sales.

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

## 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.

Query:

#5.11.  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.

--Query

SELECT

  order\_id,

  order\_purchase\_timestamp,

  DATE\_DIFF(DATE(order\_delivered\_customer\_date), DATE(order\_purchase\_timestamp), DAY) as act\_time\_delivery,

  DATE\_DIFF(DATE(order\_estimated\_delivery\_date), DATE(order\_purchase\_timestamp), DAY) as est\_time\_delivery,

  DATE\_DIFF(DATE(order\_delivered\_customer\_date), DATE(order\_estimated\_delivery\_date), DAY) as diff\_delivery

FROM

  `targetsql.orders`

WHERE

  lower(order\_status) = 'delivered';

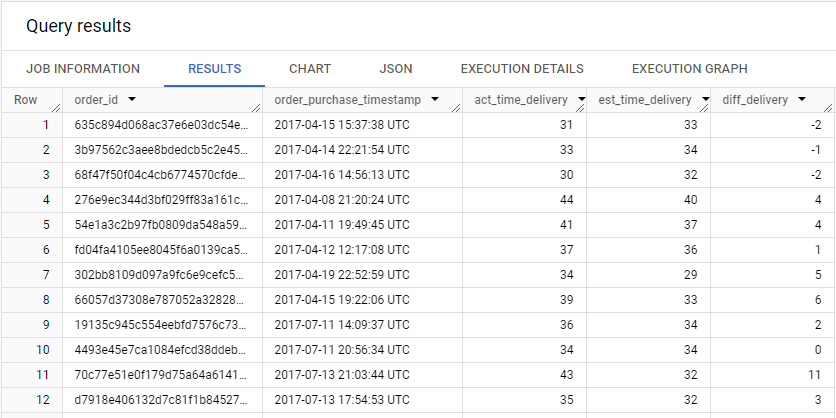


Figure 6—1: Time taken to deliver for each customer.

Ordering the above query in descending orders of the actual delivery time. To know the highest time delay.

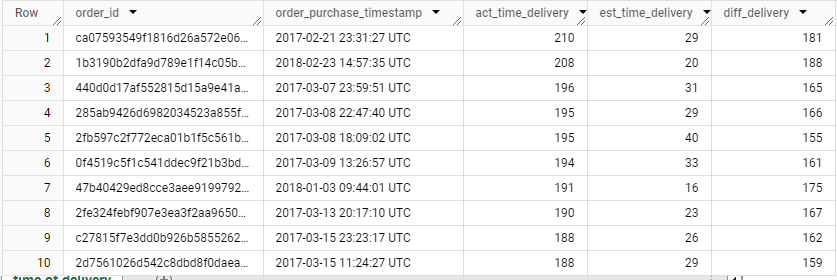


Figure 6—2: Time taken to delivery ordered in DESC manner

Figure 6—3: Time taken to delivery per customer

INSIGHTS:

1. From the result it can be observed that the highest delay observed is 187 days. Although there are many cases where the delivery was quick, there are also considerable cases where there is almost a delay of months.

RECOMMENDATIONS:

1. Ensuring delivery within the estimated delivery will boost the customer experience.
2. Ensuring Same-day delivery/Fast Delivery happens on time will give an inflow of urgent orders, mostly when there is an urgency of the product, even if there is a slight increase in the price will not be considered.

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

### Top 5 states with the highest average freight value

Query:

WITH avg\_fr\_val as

(SELECT

  c.customer\_id, c.customer\_city, c.customer\_state, oi.freight\_value as freight\_value,

  MIN(freight\_value) over(partition by customer\_state) as lowest\_freight\_value,

  MAX(freight\_value) over(partition by customer\_state) as highest\_freight\_value,

  ROUND(AVG(freight\_value) over(partition by customer\_state), 2) as avg\_freight\_value,

FROM

  `targetsql.customers` c

JOIN

  `targetsql.orders` o ON c.customer\_id = o.customer\_id

JOIN

  `targetsql.order\_items` oi ON o.order\_id = oi.order\_id

WHERE

  lower(order\_status) = 'delivered'

)

 -- TOP 5 states

SELECT

  c.customer\_state, a.avg\_freight\_value

FROM

  `targetsql.customers` c

JOIN

  avg\_fr\_val a ON c.customer\_id = a.customer\_id

GROUP BY

  c.customer\_state, a.avg\_freight\_value

ORDER BY

  a.avg\_freight\_value desc

LIMIT 5;

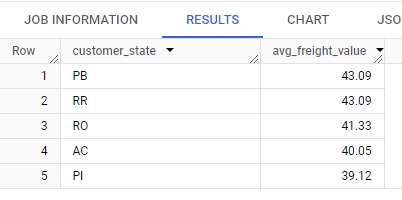


Figure 6—4: Top 5 states with highest avg freight value

### Top 5 states with the lowest average freight value

Query:

--LOWEST 5 states

WITH avg\_fr\_val as

(SELECT

  c.customer\_id, c.customer\_city, c.customer\_state, oi.freight\_value as freight\_value,

  MIN(freight\_value) over(partition by customer\_state) as lowest\_freight\_value,

  MAX(freight\_value) over(partition by customer\_state) as highest\_freight\_value,

  ROUND(AVG(freight\_value) over(partition by customer\_state), 2) as avg\_freight\_value,

FROM

  `targetsql.customers` c

JOIN

  `targetsql.orders` o ON c.customer\_id = o.customer\_id

JOIN

  `targetsql.order\_items` oi ON o.order\_id = oi.order\_id

WHERE

  lower(order\_status) = 'delivered'

)

SELECT

  c.customer\_state, a.avg\_freight\_value

FROM

  `targetsql.customers` c

JOIN

  avg\_fr\_val a ON c.customer\_id = a.customer\_id

GROUP BY

  c.customer\_state, a.avg\_freight\_value

ORDER BY

  a.avg\_freight\_value

LIMIT 5;

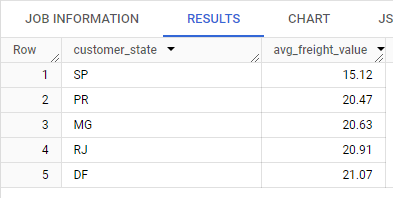


Figure 6—5: Top 5 states with the lowest average freight value

INSIGHTS:

1. PB state has the highest average freight value and SP state has the lowest average freight value.
2. SP state with the highest sales has the lowest average freight value, probably because it has a large customer base, so there are more warehouse facilities which makes the delivery quick and less freight.

RECOMMENDATIONS:

1. For the states that are farther away and where the freight charges are high, Identifying a cluster of customer base areas and then establishing a warehouse near the area will help reduce the freight value.

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

### Top 5 states with the highest average delivery time

Query:

WITH delivery\_time as

(

SELECT

  order\_id, customer\_id,

  order\_purchase\_timestamp,

  DATE\_DIFF(DATE(order\_delivered\_customer\_date), DATE(order\_purchase\_timestamp), DAY) as act\_time\_delivery,

  DATE\_DIFF(DATE(order\_estimated\_delivery\_date), DATE(order\_purchase\_timestamp), DAY) as est\_time\_delivery,

  DATE\_DIFF(DATE(order\_delivered\_customer\_date), DATE(order\_estimated\_delivery\_date), DAY) as diff\_delivery

FROM

  `targetsql.orders`

WHERE

  lower(order\_status) = 'delivered')

--TOP 5

SELECT

  c.customer\_state, ROUND(AVG(d.act\_time\_delivery), 2) as avg\_delivery\_time

FROM

 `targetsql.customers` c

JOIN

  delivery\_time as d ON c.customer\_id = d.customer\_id

GROUP BY

  c.customer\_state

ORDER BY

  avg(d.act\_time\_delivery) DESC

LIMIT 5;

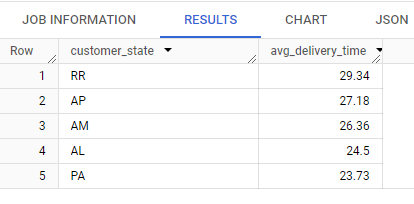


Figure 6—6: Top 5 states with highest average delivery time

### Top 5 states a lowest average delivery time

Query:

WITH delivery\_time as

(

SELECT

  order\_id, customer\_id,

  order\_purchase\_timestamp,

  DATE\_DIFF(DATE(order\_delivered\_customer\_date), DATE(order\_purchase\_timestamp), DAY) as act\_time\_delivery,

  DATE\_DIFF(DATE(order\_estimated\_delivery\_date), DATE(order\_purchase\_timestamp), DAY) as est\_time\_delivery,

  DATE\_DIFF(DATE(order\_delivered\_customer\_date), DATE(order\_estimated\_delivery\_date), DAY) as diff\_delivery

FROM

  `targetsql.orders`

WHERE

  lower(order\_status) = 'delivered')

SELECT

  c.customer\_state, ROUND(AVG(d.act\_time\_delivery), 2) as avg\_delivery\_time

FROM

 `targetsql.customers` c

JOIN

  delivery\_time as d ON c.customer\_id = d.customer\_id

GROUP BY

  c.customer\_state

ORDER BY

  avg(d.act\_time\_delivery)

LIMIT 5;

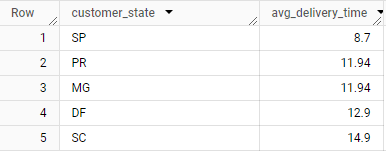


Figure 6—7: Top 5 states with the lowest avg delivery time

INSIGHTS:

1. From the results it can be observed that the state SP has the lowest delivery time, i.e. the state SP has a large customer base which is known wrt to previous queries. Therefore, it can be derived that in states with a large customer base, the average delivery time is less. Maybe because they have established more warehouses to cater to the demand.
2. But the states like RR, AP, and AM with smaller customer bases, the average delivery time is high. This can be because these are farther from the warehouse and therefore take a lot of time to transport, which may also be due to low access to these places as they are remote.

RECOMMENDATIONS:

1. Partnership with delivery agencies might help bring down the freight charges and transportation to remote areas can be made frequent. This may help decrease the delivery time of the products which in turn encourages the customers to place more orders.

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

Query:

WITH delivery\_time as

(

SELECT

  order\_id, customer\_id,

  order\_purchase\_timestamp,

  DATE\_DIFF(DATE(order\_delivered\_customer\_date), DATE(order\_purchase\_timestamp), DAY) as act\_time\_delivery,

  DATE\_DIFF(DATE(order\_estimated\_delivery\_date), DATE(order\_purchase\_timestamp), DAY) as est\_time\_delivery,

  DATE\_DIFF(DATE(order\_delivered\_customer\_date), DATE(order\_estimated\_delivery\_date), DAY) as diff\_delivery

FROM

  `targetsql.orders`

WHERE

  lower(order\_status) = 'delivered' and order\_delivered\_customer\_date is not NULL)

SELECT

  c.customer\_state,

  ROUND(AVG(d.est\_time\_delivery),2) as avg\_est\_delivery,

  ROUND(AVG(d.act\_time\_delivery),2) as avg\_act\_delivery,

  ROUND(AVG(d.diff\_delivery), 2) as avg\_diff\_delivery

FROM

 `targetsql.customers` c

JOIN

  delivery\_time d ON c.customer\_id = d.customer\_id

GROUP BY

  c.customer\_state

ORDER BY

  AVG(d.diff\_delivery)

LIMIT 5

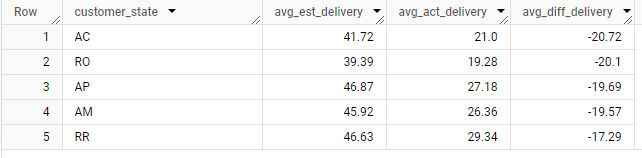


Figure 6—8: Top 5 states with fast delivery when compared to estimate

INSIGHTS:

1. The state AC has fast delivery when compared to the estimated delivery time, whose average estimated delivery days are 41.72 days but the delivery is happening within 20.72 days.
2. Then followed by the state RO, whose average estimated delivery days are 39.39 days but the delivery is happening within 20.72 days

RECOMMENDATIONS:

1. If such delivery could be maintained consistently, the trust could be established and more orders would pour in.

# ****Analysis based on the payments****

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

Query:

WITH orders\_YM as (SELECT order\_id, customer\_id,

       EXTRACT(YEAR FROM order\_purchase\_timestamp) as years,

       EXTRACT(MONTH FROM order\_purchase\_timestamp) as months,

       FORMAT\_DATETIME("%Y - %m",order\_purchase\_timestamp) as year\_month,

       order\_status

FROM `targetsql.orders`

WHERE

  lower(order\_status) = 'delivered'),

--Temporary table for types of payments per order\_id

payment\_type\_order as

(SELECT

  order\_id, payment\_type, COUNT(\*) num\_trans\_per\_paytype

FROM

 `targetsql.payments` p

GROUP BY

  payment\_type, order\_id

)

SELECT

  o.years, o.months, p.payment\_type,

  COUNT(p.num\_trans\_per\_paytype)

FROM

  `targetsql.customers` c

JOIN

 orders\_YM o ON c.customer\_id = o.customer\_id

JOIN

  payment\_type\_order p ON o.order\_id = p.order\_id

GROUP BY

  o.years, o.months, p.payment\_type

ORDER BY

  p.payment\_type;

The screenshot here is taken from row 13 just to show the different payment types in the frame.

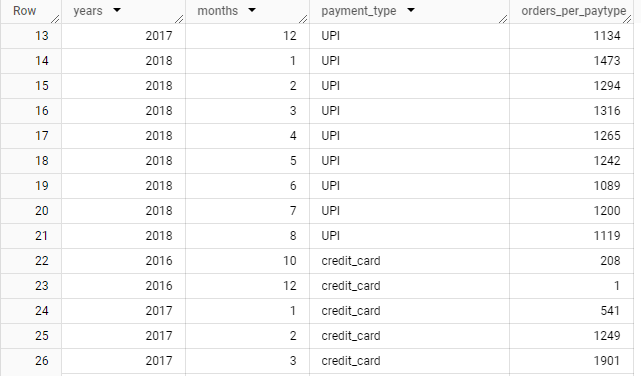


Figure 7—1Month on month orders based on payment types

Figure 7—2: Month vs no of orders wrt payment types

INSIGHTS:

1. From the results it can be observed that the maximum number of customers preferred payment type is through credit cards.
2. The least preferred way is through credit cards.
3. The following may be because of the offers, cashback, and EMI options that are available through credit cards.
4. The next highest payment type is through UPI, which is an emerging type of payment in recent years.

RECOMMENDATIONS:

1. Partnering with various banking agencies and offering cashback, No-cost EMI options, and finance options might help attract customers to buy premium items as well.

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

Query:

SELECT

  p.payment\_installments,

  COUNT(o.order\_id) as num\_orders\_inst

FROM

 `targetsql.orders` o

JOIN

  `targetsql.payments` p ON o.order\_id = p.order\_id

WHERE

  lower(o.order\_status) != 'cancelled'

GROUP BY

  p.payment\_installments

ORDER BY

  2 DESC

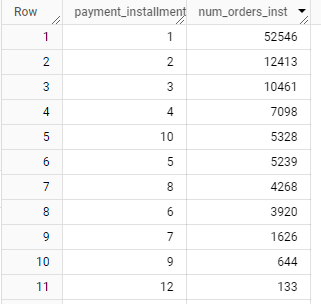


Figure 7—3: Number of orders based on payment installments

Figure 7—4: Installments vs no of orders

INSIGHTS:

1. From the result it can be observed that the maximum customers preferred payment in 1 installment, followed by 2 and 3 installments.
2. The maximum installments offered are 24, availed by 18 orders.
3. It can be observed that a considerable number of orders used up to 10 instalments, beyond that are very few orders. This may be due to high interest rates.

RECOMMENDATIONS:

1. Offering Products of high value that are in demand concerning the area, in instalments preferably for a period ranging from 1-10 months, might push the customers to purchase the product.
2. Offering No-cost EMI options and Finance options for 1-10 months of the period will also help increase the sales.