Business Case Study : Target SQL

Target is a highly recognized brand and a leading retailer in the United States. It strives to differentiate itself by offering value, inspiration, innovation, and a unique guest experience that sets it apart from other retailers.It contains data on 100,000 orders made at Target in Brazil between 2016 and 2018.
The case provides a comprehensive view of each order across various dimensions, including order status, price, payment and freight performance, customer location, product attributes, and customer reviews. This rich dataset enables analysis and evaluation of different aspects of Target's operations in Brazil during that period. It can be leveraged to gain insights into customer behavior, product performance, geographic trends, and customer satisfaction, among other factors. Such insights can inform decision-making, strategy formulation, and improvements in various areas of Target's business in Brazil.

The customers.csv contain following features:

Features	Description
customer_id	Id of the consumer who made the purchase.
customer_unique_id	Unique Id of the consumer.
customer_zip_code_prefix	Zip Code of the location of the consumer.
customer_city	Name of the City from where order is made.
customer_state	State Code from where order is made(Ex- sao paulo-SP).

The sellers.csv contains following features:

Features	Description
seller_id	Unique Id of the seller registered
seller_zip_code_prefix	Zip Code of the location of the seller.
seller_city	Name of the City of the seller.
seller_state	State Code (Ex- sao paulo-SP)

The order_items.csv contain following features:

Features	Description
order_id	A unique id of order made by the consumers.
order_item_id	A Unique id given to each item ordered in the order.
product_id	A unique id given to each product available on the site.
seller_id	Unique Id of the seller registered in Target.
shipping_limit_date	The date before which shipping of the ordered product must be completed.
price	Actual price of the products ordered .
freight_value	Price rate at which a product is delivered from one point to another.

The geolocations.csv contain following features:

Features	Description
geolocation _zip_code_prefix	first 5 digits of zip code
geolocation_lat	latitude
geolocation_lng	longitude
geolocation_city	city name
geolocation_state	state

The payments.csv contain following features:

Features	Description
order_id	A unique id of order made by the consumers.
payment_sequential	sequences of the payments made in case of EMI.
payment_type	mode of payment used.(Ex-Credit Card)
payment_installments	number of installments in case of EMI purchase.
payment_value	Total amount paid for the purchase order.

The orders.csv contain following features

Features	Description
order_id	A unique id of order made by the consumers.
customer_id	Id of the consumer who made the purchase.
order_status	status of the order made i.e delivered, shipped etc
order_purchase_timestamp	Timestamp of the purchase.
order_delivered_carrier_date	delivery date at which carrier made the delivery.
order_delivered_customer_date	date at which customer got the product.
order_estimated_delivery_date	estimated delivery date of the products.

The reviews.csv contain following features:

Features	Description
review_id	Id of the review given on the product ordered by the order id.
order_id	A unique id of order made by the consumers.
review_score	review score given by the customer for each order on the scale of 1–5
review_comment_title	Title of the review
review_comment_message	Review comments posted by the consumer for each order.
review_creation_date	Timestamp of the review when it is created.
review_answer_timestamp	Timestamp of the review answered.

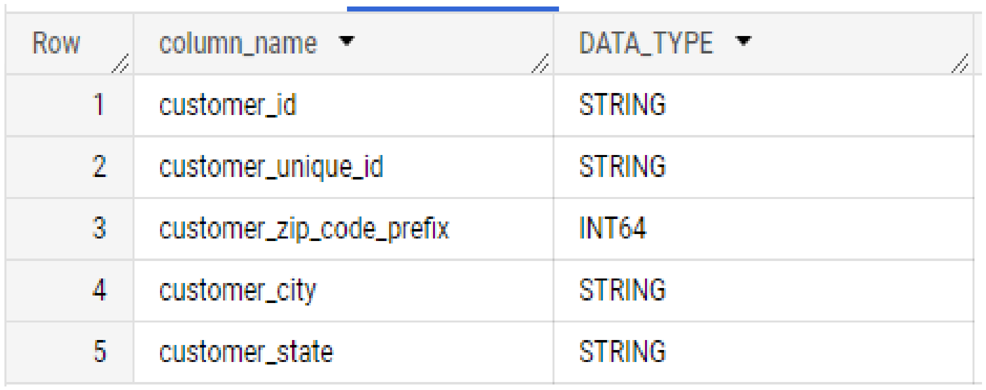
The products.csv contain following features:

Features	Description
product_id	A unique identifier for the proposed project.
product_category_name	Name of the product category
product_name_lenght	length of the string which specifies the name given to the products ordered.
product_description_lenght	length of the description written for each product ordered on the site.
product_photos_qty	Number of photos of each product ordered available on the shopping portal.
product_weight_g	Weight of the products ordered in grams.
product_length_cm	Length of the products ordered in centimeters
product_height_cm	Height of the products ordered in centimeters
product_width_cm	width of the product ordered in centimeters.

Q: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

```
SELECT column_name, DATA_TYPE
FROM `Case_study`.INFORMATION_SCHEMA.COLUMNS
WHERE table_name ="customers"
```



Row	column_name	DATA_TYPE
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

INSIGHT

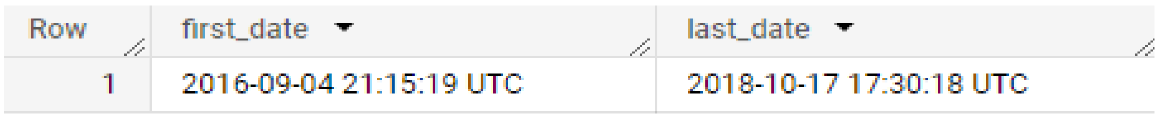
- Data type of a column defines what value the column can store in the table.
- It is defined while creating the tables in the database.

EXPLANATION

Here I have selected the table customers and have found the data type of each column by using “Information_schema.columns”.We can use the same function to get the data type of the columns.

2.Time period for which the data is given.

```
SELECT
min(order_purchase_timestamp) as first_date,
max(order_purchase_timestamp) as last_date,
FROM `Case_study.orders`
```



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

INSIGHT

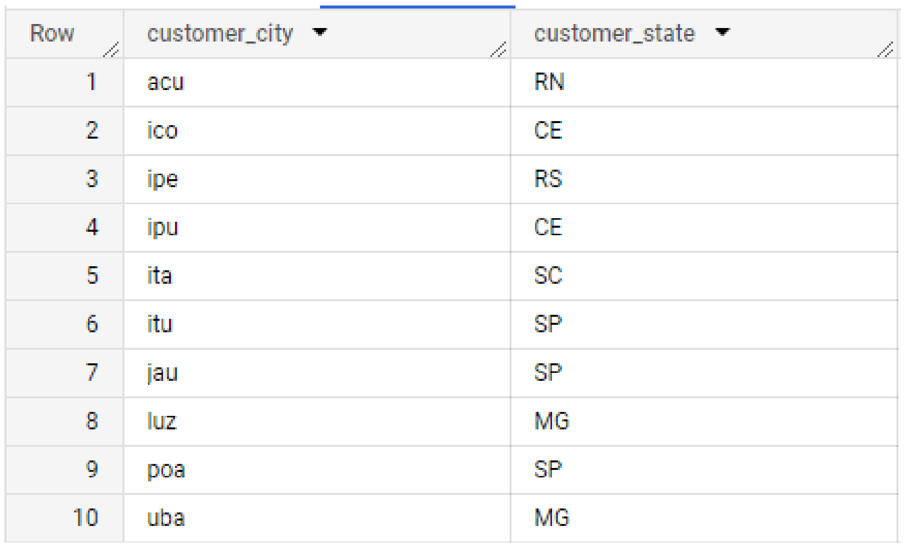
- The data provided is between September 4th 2016 and October 17th 2018.

EXPLANATION

The min and the max functions are to get the Time period of any given data set.The ‘min’ function will give the first date from the data set and ‘max’ will give the last date on which the customers have made their purchase.This will be the range of the orders made by the customers.

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

```
SELECT distinct customer_city,customer_state
FROM `Case_study.customers`
LIMIT 10
```



Row	customer_city	customer_state
1	acu	RN
2	ico	CE
3	ipe	RS
4	ipu	CE
5	ita	SC
6	itu	SP
7	jau	SP
8	luz	MG
9	poa	SP
10	uba	MG

INSIGHT

- There are customers from a wide variety of cities and across various states in Brazil.

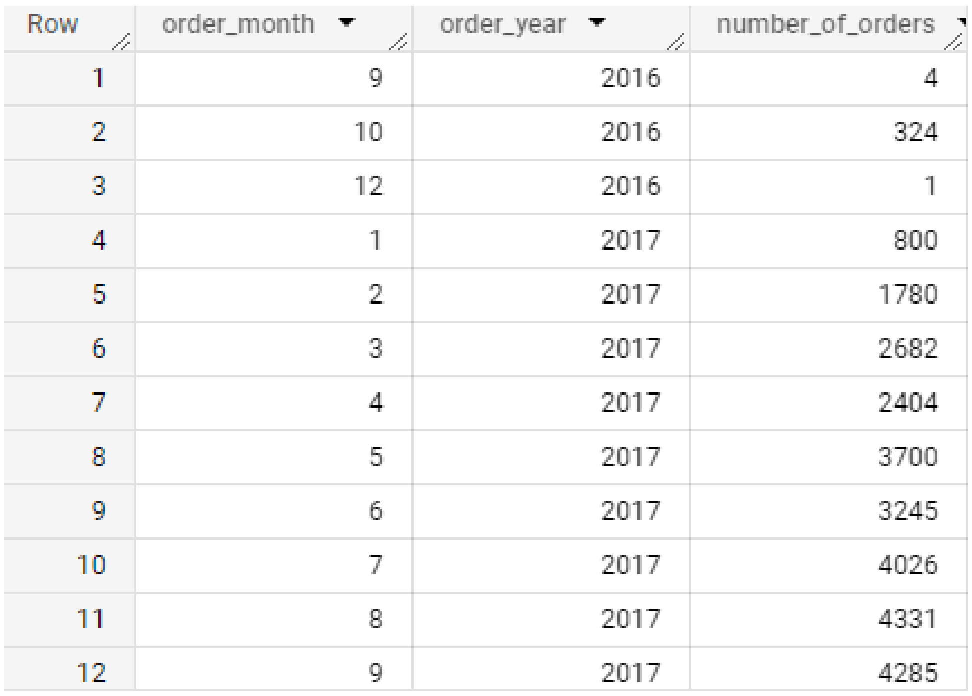
EXPLANATION

We use the SELECT statement to choose the customer state and customer city from the customers table and “distinct customer_city” will get the unique values of customer_city.

2.In-depth Exploration:

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

```
SELECT
EXTRACT (month from order_purchase_timestamp) as order_month,
EXTRACT (year from order_purchase_timestamp) as order_year,
COUNT(order_id) AS number_of_orders,
FROM `Case_study.orders`
GROUP BY order_month ,order_year
ORDER BY order_year, order_month
```



Row	order_month	order_year	number_of_orders
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
10	7	2017	4026
11	8	2017	4331
12	9	2017	4285

INSIGHTS

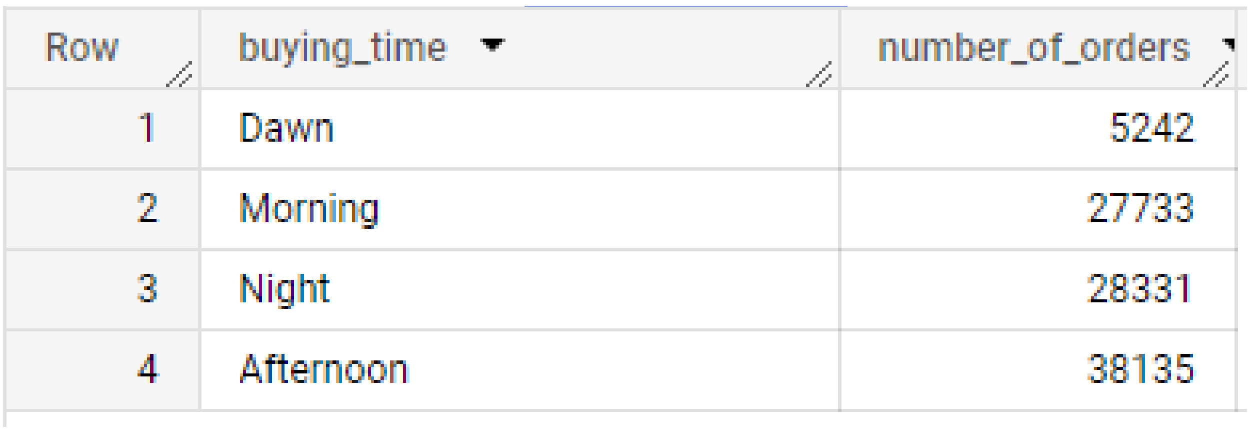
- Yes there is a growing trend of e-commerce in Brazil.
- There is a gradual increase in the sales over the years and peaks at specific months.
- The number_of_orders column gives us clarity about the sales that happened between 2016-2018.
- There has been positive growth
- We can see that the count of orders is gradually increasing every month. It reached 5673 orders in December 2017 and 7269 in January 2018. So it is evident that the count of orders has shown a great growth.

EXPLANATION

The EXTRACT function is used to obtain the year and the month values from the order_purchase_timestamp column in the orders table .The COUNT function is used to count the number of orders for the group of each year and month . The SELECT statement chooses three columns: the count of order_id (also known as number_of_orders), the year and the month extracted from the order_purchase_timestamp column.The Year and Month columns are used to group and order the results using GROUP BY and ORDER BY statement.

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

```
SELECT
CASE
  WHEN EXTRACT (HOUR FROM order_purchase_timestamp) between 0 and 6
THEN "Dawn"
  WHEN EXTRACT (HOUR FROM order_purchase_timestamp) between 6 and 12
THEN "Morning"
  WHEN EXTRACT (HOUR FROM order_purchase_timestamp) between 12 and 18
THEN "Afternoon"
  WHEN EXTRACT (HOUR FROM order_purchase_timestamp) between 18 and 24
THEN "Night"
END AS buying_time,
COUNT(order_id) as number_of_orders
FROM `Case_study.orders`
GROUP BY buying_time
ORDER BY number_of_orders
```



Row	buying_time	number_of_orders
1	Dawn	5242
2	Morning	27733
3	Night	28331
4	Afternoon	38135

INSIGHTS

It can be clearly seen from the output that most of the customers place their order in the afternoon and also very few customers place orders at Dawn .

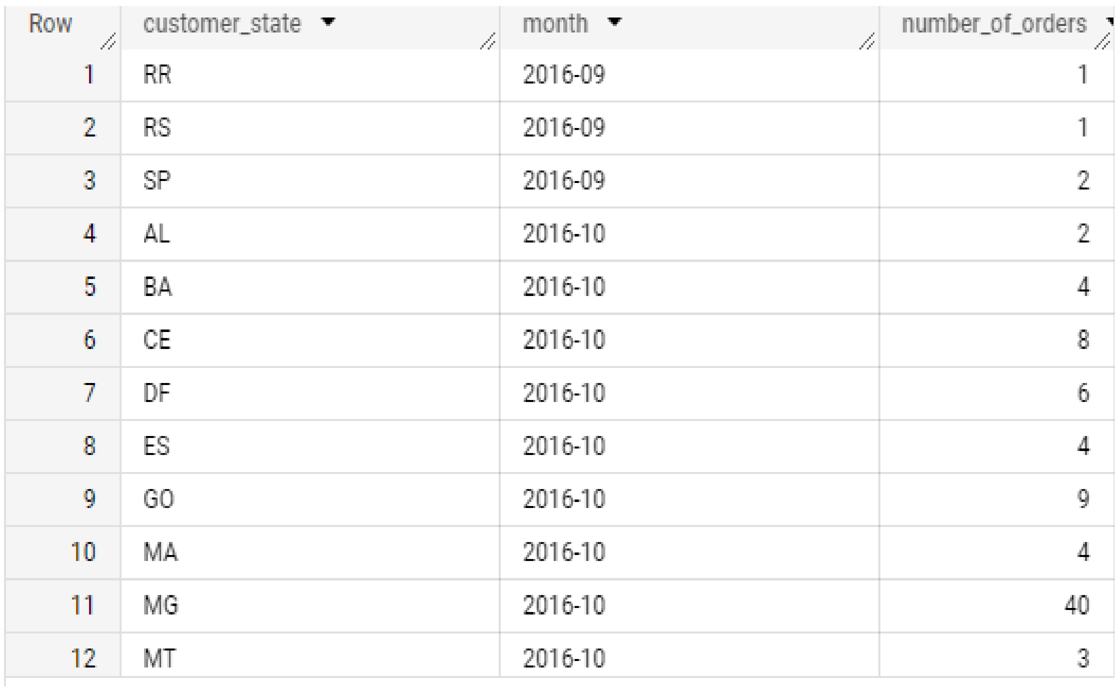
EXPLANATION

The select statement chooses two columns the count of the order id alias number_of_orders and the time bin extracted from the order_purchase_timestamp based on the different bins allotted to the time of the day the order was placed.The 4 different time frames were created using “CASE-WHEN” statement .So if the WHEN clause matches with the Time bin the column is given the name “Dawn”,“Morning”, “Afternoon”, “Night” using the function EXTRACT(EXTRACT (HOUR FROM order_purchase_timestamp) between 0 and 6 THEN "Dawn").For example if the hour value extracted is between 0 and 6 then the name of the column is given as “Dawn” and so on.The result is obtained after grouping the result by “Buying_time” and the ORDER BY statement orders the results by number_of _orders.

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

1.Get month on month orders by states

```
SELECT c.customer_state,
FORMAT_DATE('%Y-%m', o.order_purchase_timestamp) AS month,
COUNT(o.order_id) AS number_of_orders
FROM `Case_study.orders` as o
JOIN `Case_study.customers` as c
ON o.customer_id=c.customer_id
GROUP BY month,c.customer_state
ORDER BY month,customer_state
```



Row	customer_state	month	number_of_orders
1	RR	2016-09	1
2	RS	2016-09	1
3	SP	2016-09	2
4	AL	2016-10	2
5	BA	2016-10	4
6	CE	2016-10	8
7	DF	2016-10	6
8	ES	2016-10	4
9	GO	2016-10	9
10	MA	2016-10	4
11	MG	2016-10	40
12	MT	2016-10	3

INSIGHT

- We can observe that customers from ‘SP’ has placed more number of orders comparatively (In [10/2016: ‘ 113 ’] , [12/2017: ‘ 2357 ’], [8/2018: ‘ 3253 ’])
- The state RR can be seen as one of the states from where the number of orders placed is significantly less (In [2016-9 : ‘1’] ,[2016-10: ‘ 1 ’],[2017-2 : ‘ 2 ’],
[2017-7 : ‘ 2 ’] ,[2018-2 : ‘ 5 ’])

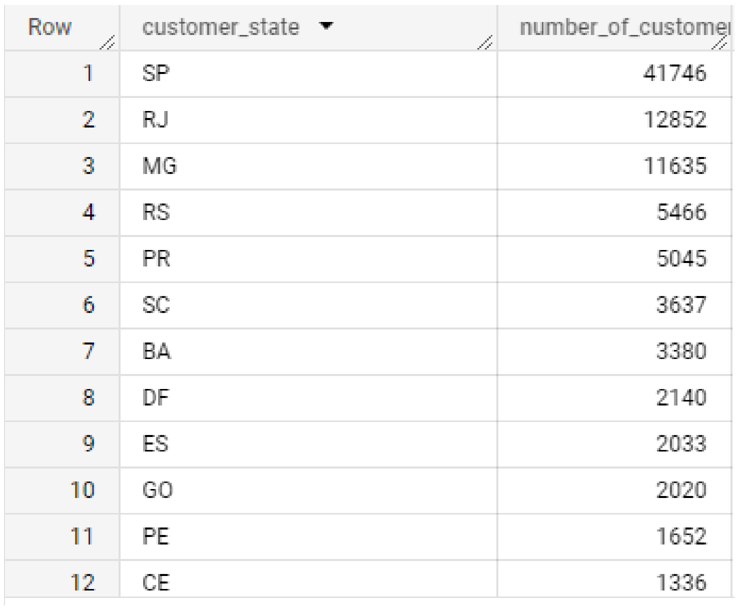
🟨 Number of orders placed

EXPLANATION

The month and year is extracted from the order purchase timestamp by using the FORMAT DATE function.Number of orders is then calculated using the COUNT function.Orders and customers tables are joined since the customer_state column is obtained from customers table. Group the table by month and state name. Order the table with the month and state names.

2.Distribution of customers across the states in Brazil

```
SELECT
customer_state,
COUNT(customer_id) as number_of_customers
FROM `Case_study.customers`
GROUP BY customer_state
ORDER BY number_of_customers desc,customer_state
```



Row	customer_state	number_of_custome
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020
11	PE	1652
12	CE	1336

INSIGHT

- The highest number of customers are located in 'SP' and hence the probable reason for having the most number of orders placed from the same state.
- The least number of customers are from 'RR' which accounts to the reason for the least number of orders from 'RR'.

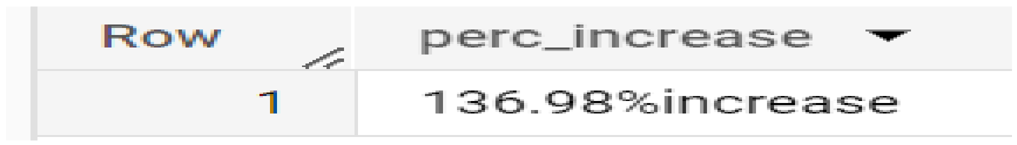
EXPLANATION

SELECT the column customer_state and use the function COUNT to count the number of customers alias number_of_customers.
Group and order the table by customer_state.

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 the “payment_value” column in the payments table.

```
WITH
year_value_2017 AS
(SELECT DISTINCT(order_year),
SUM(payment_value) OVER (PARTITION BY order_year ORDER BY order_year) AS order_sum
FROM(SELECT
EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
p.payment_value
FROM `Case_study.orders` AS o
JOIN `Case_study.payments` AS p
ON o.order_id = p.order_id
WHERE (FORMAT_DATE('%Y-%m',o.order_purchase_timestamp)) BETWEEN '2017-01' AND
'2017-08'
ORDER BY order_month,order_year) as a),

year_value_2018 AS
(SELECT DISTINCT(order_year),
SUM(payment_value) OVER (PARTITION BY order_year ORDER BY order_year) AS order_sum
FROM(SELECT
EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
p.payment_value
FROM `Case_study.orders` AS o
JOIN `Case_study.payments` AS p
ON o.order_id = p.order_id
WHERE (FORMAT_DATE('%Y-%m',o.order_purchase_timestamp)) BETWEEN '2018-01' AND
'2018-08'
ORDER BY order_month,order_year) as b )

SELECT
CONCAT(ROUND((((year_value_2018.order_sum - year_value_2017.order_sum) /
year_value_2017.order_sum) * 100),2), '%increase') AS perc_increase
FROM year_value_2017, year_value_2018
```



Row	perc_increase
1	136.98%increase

INSIGHT

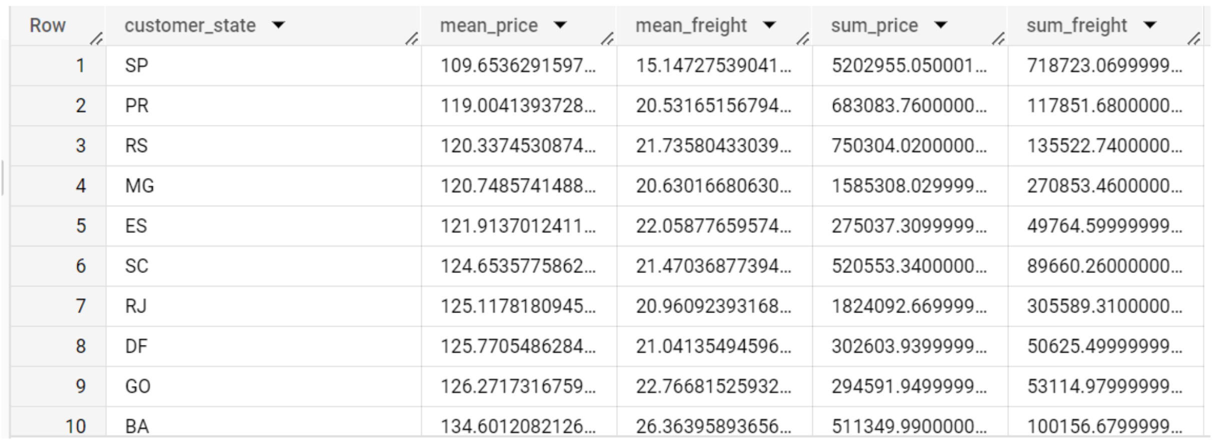
- There has been a rapid increase in the cost of orders from 2017 to 2018 (136.98 %) .
- Hence we can say that there has been a great demand for the orders and increase in the number of orders placed.

EXPLANATION

Create a query for the 2017 calendar year.
Retrieve the 2017 data from January 1 to August 30.
Find the total sum of payment_value in 2017 by using the window function.
Join the order table and payment table
Using the format function, limit the data to the months of January through August of 2017.
Create a query for the 2018 calendar year and take the data for 2018 (Jan. 01 to Aug. 30).
Find the total sum of payment_value in 2018 by using the window function.
Join the order table and payment table
Using the format function, filter the data to only include the months of January through August of 2018.
Using the above two queries, we will get the 2017 total payment value and the 2018 payment value
Use the equation to find the percentile and print the final output.

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

```
SELECT

c.customer_state,

avg(oi.price) as mean_price,

avg(oi.freight_value) as mean_freight,

sum(oi.price) as sum_price,

sum(oi.freight_value) as sum_freight

FROM `Case_study.order_items` as oi

join `Case_study.orders` as o

on oi.order_id=o.order_id

join `Case_study.customers` as c

on o.customer_id=c.customer_id

group by customer_state

order by mean_price,mean_freight,sum_price,sum_freight
```



Row	customer_state	mean_price	mean_freight	sum_price	sum_freight
1	SP	109.6536291597...	15.14727539041...	5202955.050001...	718723.0699999...
2	PR	119.0041393728...	20.53165156794...	683083.7600000...	117851.6800000...
3	RS	120.3374530874...	21.73580433039...	750304.0200000...	135522.7400000...
4	MG	120.7485741488...	20.63016680630...	1585308.029999...	270853.4600000...
5	ES	121.9137012411...	22.05877659574...	275037.3099999...	49764.59999999...
6	SC	124.6535775862...	21.47036877394...	520553.3400000...	89660.26000000...
7	RJ	125.1178180945...	20.96092393168...	1824092.669999...	305589.3100000...
8	DF	125.7705486284...	21.04135494596...	302603.9399999...	50625.49999999...
9	GO	126.2717316759...	22.76681525932...	294591.9499999...	53114.97999999...
10	BA	134.6012082126...	26.36395893656...	511349.9900000...	100156.6799999...

INSIGHT

The state ‘SP’ has the highest mean price of the products ordered at the same time it has the least mean_freight which means the additional charge taken from customers in “SP” is less.

EXPLANATION

SELECT the column customer_state

Use the SUM function on price and freight value to get the sum of price and freight and name it as sum_price,sum_freight.

Use the AVG function on price and freight value to get the mean of price and freight and name it as mean_price,mean_freight.

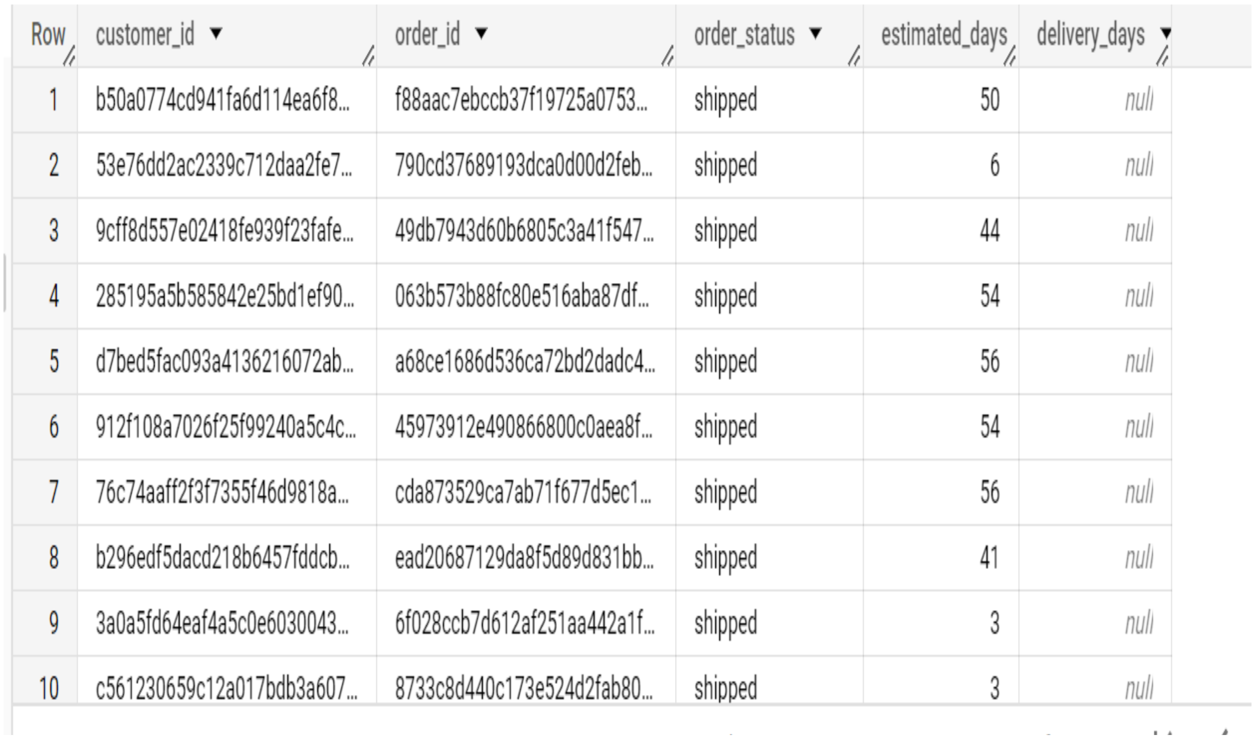
JOIN 3 tables orders,order_items and customers

GROUP the tables by customer_state and ORDER BY mean_price,mean_freight,sum_price,sum_freight

5. **Analysis on sales, freight and delivery time**

1. **Calculate days between purchasing, delivering and estimated delivery.**

```
SELECT customer_id,order_id,order_status,
DATE_DIFF(order_estimated_delivery_date,
order_purchase_timestamp, DAY) AS
estimated_days,
DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,
DAY) AS
delivery_days
FROM `Case_study.orders`
```



Row	customer_id	order_id	order_status	estimated_days	delivery_days
1	b50a0774cd941fa6d114ea6f8...	f88aac7ebccb37f19725a0753...	shipped	50	null
2	53e76dd2ac2339c712daa2fe7...	790cd37689193dca0d00d2feb...	shipped	6	null
3	9cff8d557e02418fe939f23fafe...	49db7943d60b6805c3a41f547...	shipped	44	null
4	285195a5b585842e25bd1ef90...	063b573b88fc80e516aba87df...	shipped	54	null
5	d7bed5fac093a4136216072ab...	a68ce1686d536ca72bd2dadc4...	shipped	56	null
6	912f108a7026f25f99240a5c4c...	45973912e490866800c0aea8f...	shipped	54	null
7	76c74aaff2f3f7355f46d9818a...	cda873529ca7ab71f677d5ec1...	shipped	56	null
8	b296edf5dacd218b6457fddcb...	ead20687129da8f5d89d831bb...	shipped	41	null
9	3a0a5fd64eaf4a5c0e6030043...	6f028ccb7d612af251aa442a1f...	shipped	3	null
10	c561230659c12a017bdb3a607...	8733c8d440c173e524d2fab80...	shipped	3	null

INSIGHTS

- **Those rows in delivery_days column which has ‘NULL’ values are the ones which are not yet delivered and the status is shown as “Shipped” or “Invoiced” or “Canceled”.The number of delivery_days are shows in those columns where order_status is “Delivered”**

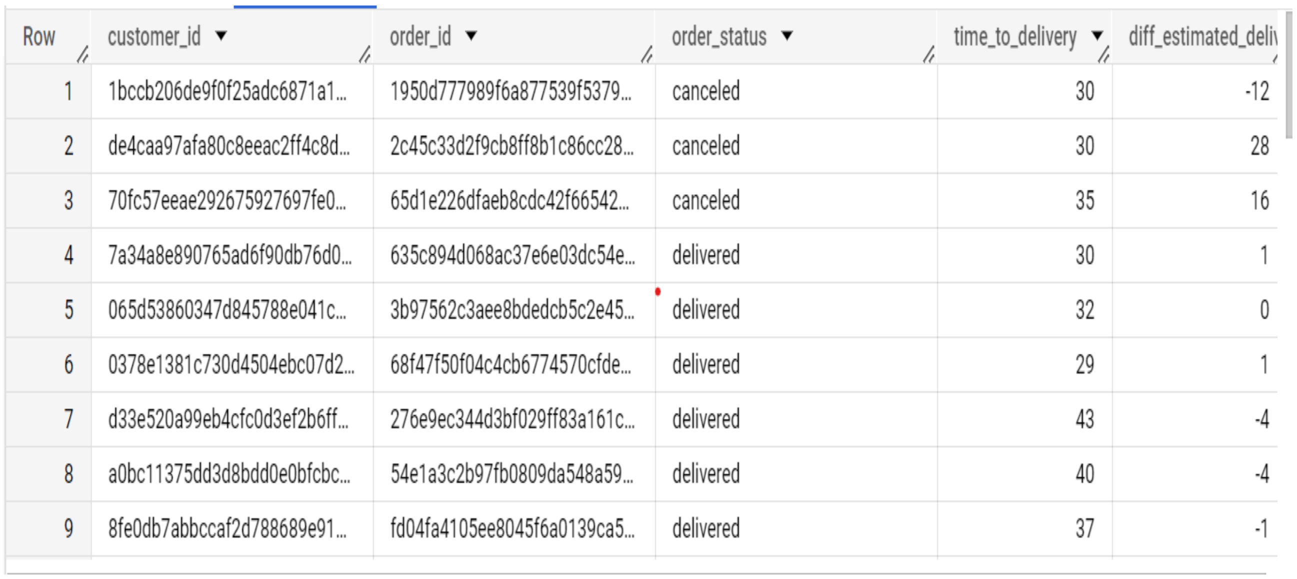
- **The estimated days shows the expected number of days it will take for the orders to get delivered .**

EXPLANATIONS

SELECT order_id,customer_id,order_status .
The days between purchasing and delivery are found using the DATE_DIFF function to calculate the difference between ordered date and delivered date (order_delivered_customer_date,order_purchase_timestamp) as delivery_days.
The days between delivery and estimated delivery is found using the DATE_DIFF function to calculate the difference between ordered date and estimated delivery date (order_estimated_delivery_date, order_purchase_timestamp,) as estimated_days.

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**

```
SELECT
customer_id,order_id,order_status,
date_diff(order_delivered_customer_date ,order_purchase_timestamp,day)
as time_to_delivery,
date_diff(order_estimated_delivery_date,order_delivered_customer_date,da
y )as diff_estimated_delivery
FROM `Case_study.orders`
```



Row	customer_id	order_id	order_status	time_to_delivery	diff_estimated_deliv
1	1bccb206de9f0f25adc6871a1...	1950d777989f6a877539f5379...	canceled	30	-12
2	de4caa97afa80c8eeac2ff4c8d...	2c45c33d2f9cb8ff8b1c86cc28...	canceled	30	28
3	70fc57eeae292675927697fe0...	65d1e226dfaeb8cdc42f66542...	canceled	35	16
4	7a34a8e890765ad6f90db76d0...	635c894d068ac37e6e03dc54e...	delivered	30	1
5	065d53860347d845788e041c...	3b97562c3aee8bdedcb5c2e45...	delivered	32	0
6	0378e1381c730d4504ebc07d2...	68f47f50f04c4cb6774570cfde...	delivered	29	1
7	d33e520a99eb4cfc0d3ef2b6ff...	276e9ec344d3bf029ff83a161c...	delivered	43	-4
8	a0bc11375dd3d8bdd0e0bfcbc...	54e1a3c2b97fb0809da548a59...	delivered	40	-4
9	8fe0db7abbccaf2d788689e91...	fd04fa4105ee8045f6a0139ca5...	delivered	37	-1

INSIGHTS

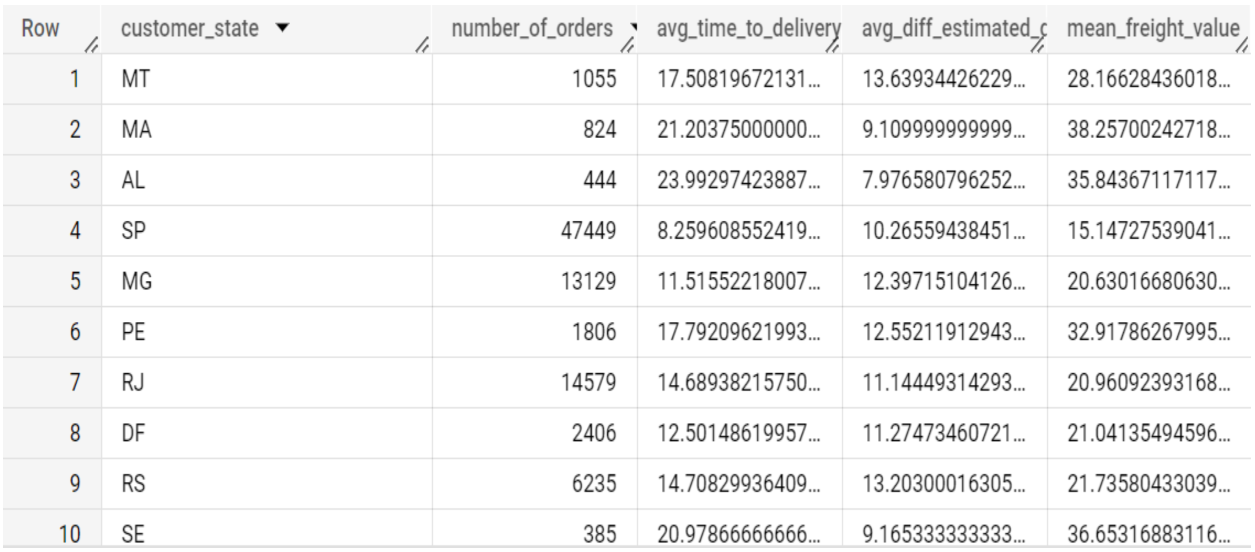
- We can see minus values in the diff_estimated_delivery column.
- It shows that those orders were all delivered after the estimated delivery date and hence there was a delay in delivery from the predicted delivery date

EXPLANATIONS

SELECT order_id,customer_id,order_status
The time_to_delivery and diff_estimated_delivery is calculated using DATE_DIFF function from the orders table
The time_to_delivery is calculated using the formula order_delivered_customer_date-order_purchase_timestamp and the 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**

```
SELECT
c.customer_state,
COUNT(o.order_id) as number_of_orders,
avg(date_diff(order_delivered_customer_date
,order_purchase_timestamp,day)) as avg_time_to_delivery,
avg(date_diff(order_estimated_delivery_date,order_delivered_customer_dat
e,day))as avg_diff_estimated_delivery,
avg(oi.freight_value) as mean_freight_value
FROM `Case_study.order_items` as oi
join `Case_study.orders` as o
on oi.order_id=o.order_id
join `Case_study.customers` as c
on o.customer_id=c.customer_id
group by customer_state
```



Row	customer_state	number_of_orders	avg_time_to_delivery	avg_diff_estimated_c	mean_freight_value
1	MT	1055	17.50819672131...	13.63934426229...	28.16628436018...
2	MA	824	21.20375000000...	9.109999999999...	38.25700242718...
3	AL	444	23.99297423887...	7.976580796252...	35.84367117117...
4	SP	47449	8.259608552419...	10.26559438451...	15.14727539041...
5	MG	13129	11.51552218007...	12.39715104126...	20.63016680630...
6	PE	1806	17.79209621993...	12.55211912943...	32.91786267995...
7	RJ	14579	14.68938215750...	11.14449314293...	20.96092393168...
8	DF	2406	12.50148619957...	11.27473460721...	21.04135494596...
9	RS	6235	14.70829936409...	13.20300016305...	21.73580433039...
10	SE	385	20.97866666666...	9.165333333333...	36.65316883116...

INSIGHTS

- Average time to deliverer in “SP” state is far less when compared to other states mean freight value is also less which indicates that the extra charges like delivery charges charged for the products is less
- This may be because of the high number of orders and customers from this state

EXPLANATIONS

Select the columns (customer_state).

Use the COUNT function to count the orders and name the column number_of_orders .

Use the AVG function to find the mean freight value and name the column mean_freight_value.

Use the AVG function to find the mean value of time_to_delivery and diff_estimated_delivery and name the column as avg_time_to_delivery and avg_diff_estimated_delivery

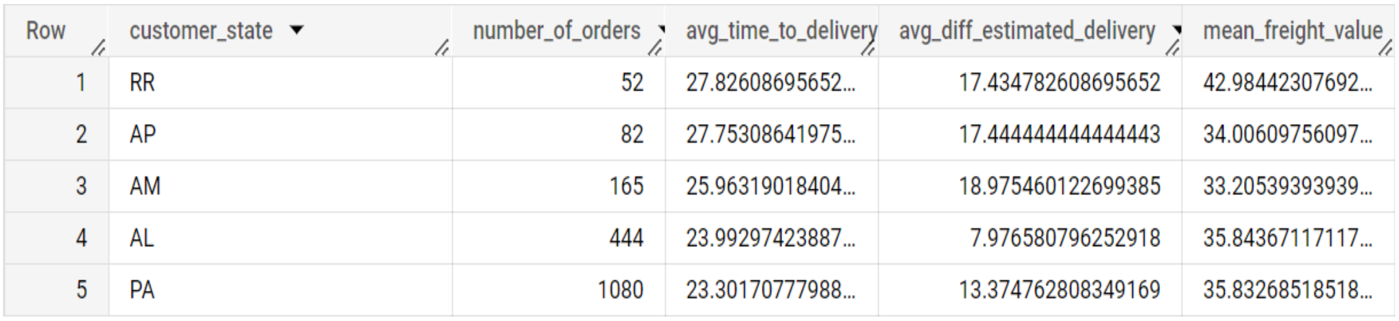
Since the state details and freight value details are in different tables, join the three tables using order_id.

Group the date accordingly with the customer_state

4.Sort the data to get the following:

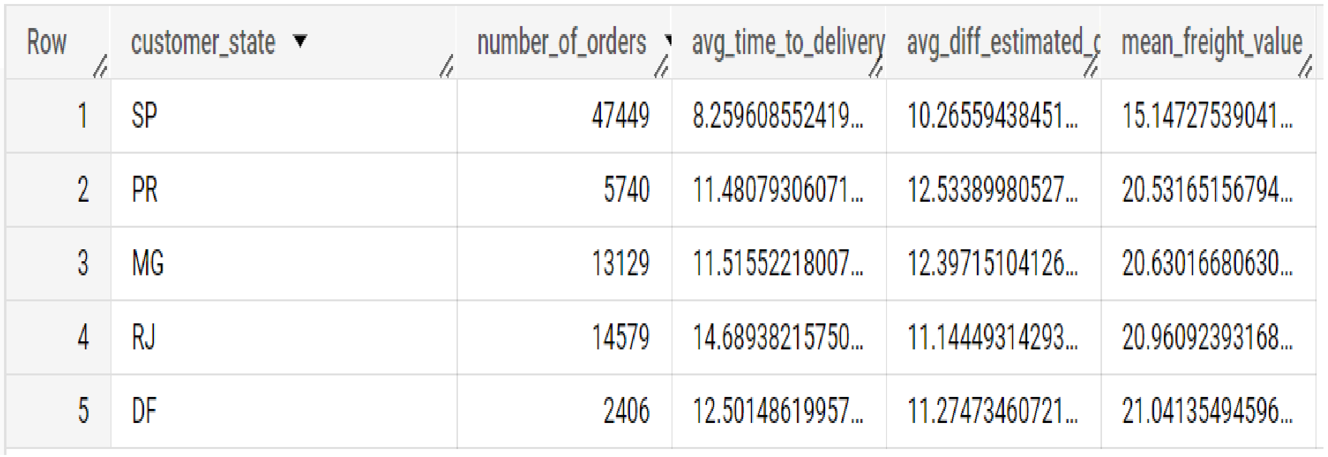
5.Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5

```
SELECT
c.customer_state,
COUNT(o.order_id) as number_of_orders,
avg(date_diff(order_delivered_customer_date ,order_purchase_timestamp,day)) as
avg_time_to_delivery,
avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day))
as avg_diff_estimated_delivery,
avg(oi.freight_value) as mean_freight_value
FROM `Case_study.order_items` as oi
join `Case_study.orders` as o
on oi.order_id=o.order_id
join `Case_study.customers` as c
on o.customer_id=c.customer_id
group by customer_state
order by mean_freight_value desc
limit 5
```



Row	customer_state	number_of_orders	avg_time_to_delivery	avg_diff_estimated_delivery	mean_freight_value
1	RR	52	27.82608695652...	17.434782608695652	42.98442307692...
2	AP	82	27.75308641975...	17.444444444444443	34.00609756097...
3	AM	165	25.96319018404...	18.975460122699385	33.20539393939...
4	AL	444	23.99297423887...	7.976580796252918	35.84367117117...
5	PA	1080	23.30170777988...	13.374762808349169	35.83268518518...

```sql
SELECT
c.customer_state,
COUNT(o.order_id) as number_of_orders,
avg(date_diff(order_delivered_customer_date ,order_purchase_timestamp,day)) as
avg_time_to_delivery,
avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day))
as avg_diff_estimated_delivery,
avg(oi.freight_value) as mean_freight_value
FROM `Case_study.order_items` as oi
join `Case_study.orders` as o
on oi.order_id=o.order_id
join `Case_study.customers` as c
on o.customer_id=c.customer_id
group by customer_state
order by mean_freight_value asc
limit 5
```



Row	customer_state	number_of_orders	avg_time_to_delivery	avg_diff_estimated_	mean_freight_value
1	SP	47449	8.259608552419...	10.26559438451...	15.14727539041...
2	PR	5740	11.48079306071...	12.53389980527...	20.53165156794...
3	MG	13129	11.51552218007...	12.39715104126...	20.63016680630...
4	RJ	14579	14.68938215750...	11.14449314293...	20.96092393168...
5	DF	2406	12.50148619957...	11.27473460721...	21.04135494596...

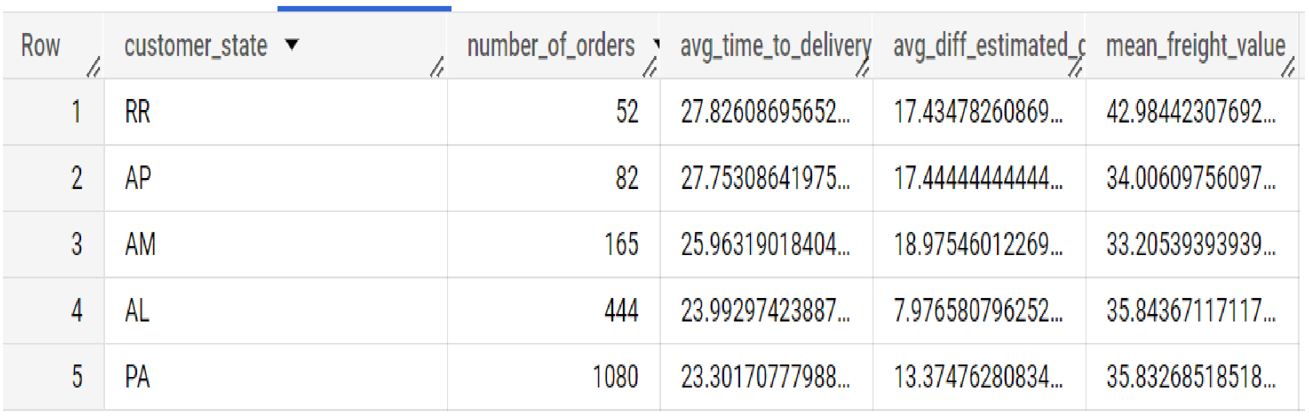
INSIGHT

- **The highest average freight value is in "RR" and lowest is in"SP" which means that the extra price charged in 'SP' is the least and highest in "RR"**
- **The reason may be because of the total number of customers ,number of orders placed .**

- **Since it is a popular state, the commutation will be much easier when compared to an interior place like “RR” .**
- **The difficulty in transportation might be a reason to charge more from “RR’**

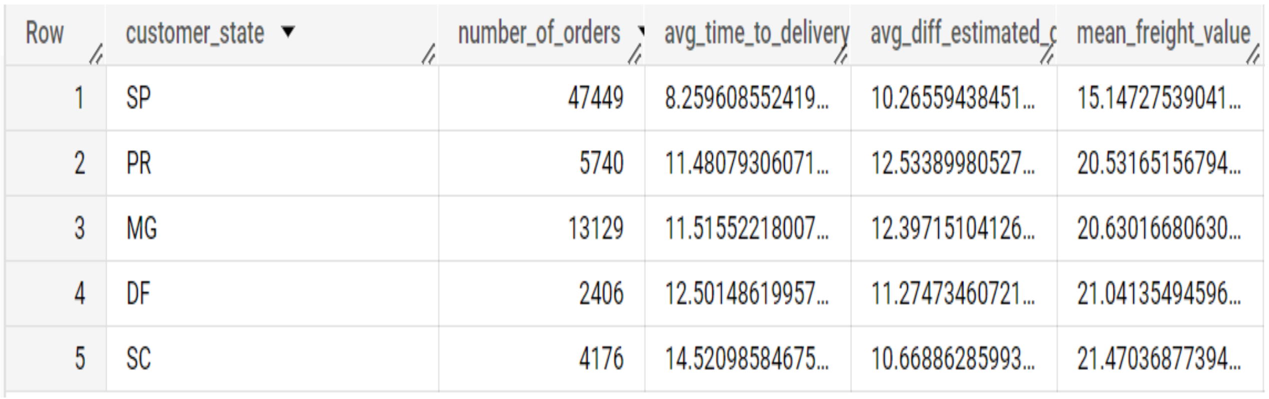
4. **Top 5 states with highest/lowest average time to delivery**

```
SELECT
c.customer_state,
COUNT(o.order_id) as number_of_orders,
avg(date_diff(order_delivered_customer_date
,order_purchase_timestamp,day)) as avg_time_to_delivery,
avg(date_diff(order_estimated_delivery_date,order_delivered_customer_dat
e,day))as avg_diff_estimated_delivery,
avg(oi.freight_value) as mean_freight_value
FROM `Case_study.order_items` as oi
join `Case_study.orders` as o
on oi.order_id=o.order_id
join `Case_study.customers` as c
on o.customer_id=c.customer_id
group by customer_state
order by avg_time_to_delivery desc
limit 5
```



Row	customer_state	number_of_orders	avg_time_to_delivery	avg_diff_estimated_	mean_freight_value
1	RR	52	27.82608695652...	17.43478260869...	42.98442307692...
2	AP	82	27.75308641975...	17.44444444444...	34.00609756097...
3	AM	165	25.96319018404...	18.97546012269...	33.20539393939...
4	AL	444	23.99297423887...	7.976580796252...	35.84367117117...
5	PA	1080	23.30170777988...	13.37476280834...	35.83268518518...

```
SELECT
c.customer_state,
COUNT(o.order_id) as number_of_orders,
avg(date_diff(order_delivered_customer_date
,order_purchase_timestamp,day)) as avg_time_to_delivery,
avg(date_diff(order_estimated_delivery_date,order_delivered_customer_dat
e,day))as avg_diff_estimated_delivery,
avg(oi.freight_value) as mean_freight_value
FROM `Case_study.order_items` as oi
join `Case_study.orders` as o
on oi.order_id=o.order_id
join `Case_study.customers` as c
on o.customer_id=c.customer_id
group by customer_state
order by avg_time_to_delivery asc
limit 5
```



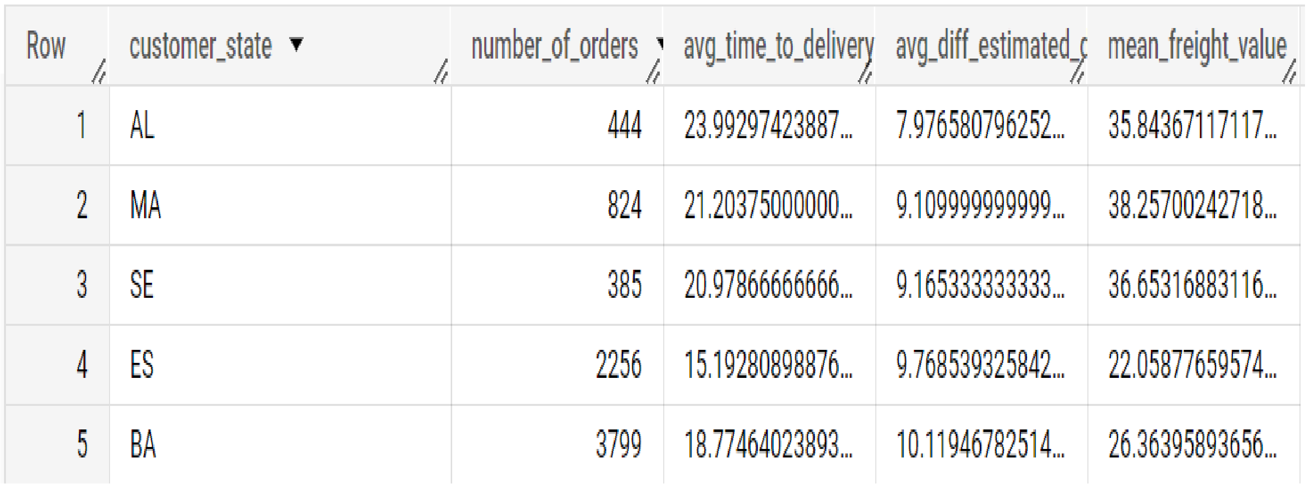
Row	customer_state	number_of_orders	avg_time_to_delivery	avg_diff_estimated_c	mean_freight_value
1	SP	47449	8.259608552419...	10.26559438451...	15.14727539041...
2	PR	5740	11.48079306071...	12.53389980527...	20.53165156794...
3	MG	13129	11.51552218007...	12.39715104126...	20.63016680630...
4	DF	2406	12.50148619957...	11.27473460721...	21.04135494596...
5	SC	4176	14.52098584675...	10.66886285993...	21.47036877394...

INSIGHT

- **Average time of delivery is the highest in “RR” and the least in “SP”**
- **Since SP is a popular state in terms of population and development , the commutation will be much easier when compared to an interior place like “RR” .**
- **The difficulty in transportation might be a reason for the highest delivery time in “RR’ when compared to"SP ".**

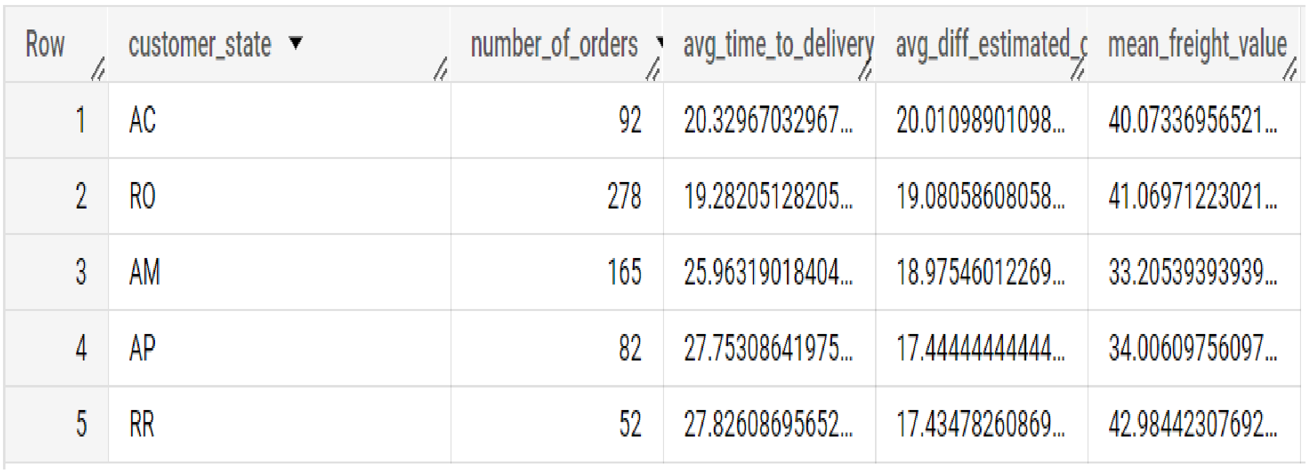
5. **Top 5 states where delivery is really fast/ not so fast compared to estimated date.**

```
SELECT
c.customer_state,
COUNT(o.order_id) as number_of_orders,
avg(date_diff(order_delivered_customer_date ,order_purchase_timestamp,day)) as
avg_time_to_delivery,
avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day))
as avg_diff_estimated_delivery,
avg(oi.freight_value) as mean_freight_value
FROM `Case_study.order_items` as oi
join `Case_study.orders` as o
on oi.order_id=o.order_id
join `Case_study.customers` as c
on o.customer_id=c.customer_id
group by customer_state
order by avg_diff_estimated_delivery asc
limit 5
```



Row	customer_state ▼	number_of_orders	avg_time_to_delivery	avg_diff_estimated_c	mean_freight_value
1	AL	444	23.99297423887...	7.976580796252...	35.84367117117...
2	MA	824	21.20375000000...	9.109999999999...	38.25700242718...
3	SE	385	20.97866666666...	9.165333333333...	36.65316883116...
4	ES	2256	15.19280898876...	9.768539325842...	22.05877659574...
5	BA	3799	18.77464023893...	10.11946782514...	26.36395893656...

```sql
SELECT
c.customer_state,
COUNT(o.order_id) as number_of_orders,
avg(date_diff(order_delivered_customer_date ,order_purchase_timestamp,day)) as
avg_time_to_delivery,
avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day))
as avg_diff_estimated_delivery,
avg(oi.freight_value) as mean_freight_value
FROM `Case_study.order_items` as oi
join `Case_study.orders` as o
on oi.order_id=o.order_id
join `Case_study.customers` as c
on o.customer_id=c.customer_id
group by customer_state
order by avg_diff_estimated_delivery des
limit 5
```



Row	customer_state ▼	number_of_orders	avg_time_to_delivery	avg_diff_estimated_c	mean_freight_value
1	AC	92	20.32967032967...	20.01098901098...	40.07336956521...
2	RO	278	19.28205128205...	19.08058608058...	41.06971223021...
3	AM	165	25.96319018404...	18.97546012269...	33.20539393939...
4	AP	82	27.75308641975...	17.44444444444...	34.00609756097...
5	RR	52	27.82608695652...	17.43478260869...	42.98442307692...

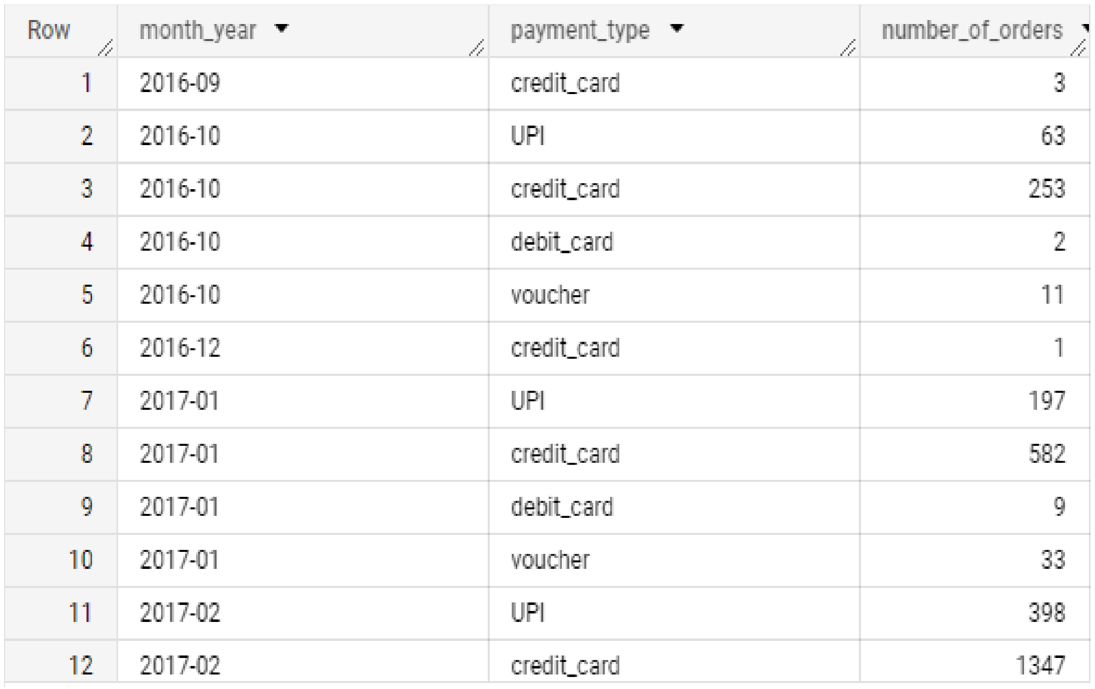
INSIGHT

The difference between the estimated delivery date and the order delivered date is the least in “AL” which means that the delivery was very fast and the difference between the estimated delivery date and the order delivered date is highest in “AC” which means it took longer time for products to get delivered in “AC” state.The reason could be because of less number of orders placed by people in “AC”

6.Payment type analysis:

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

```
SELECT
FORMAT_DATE('%Y-%m', o.order_purchase_timestamp) AS month_year,
p.payment_type,
COUNT(distinct o.order_id) AS number_of_orders
FROM `Case_study.orders` as o
join `Case_study.payments` as p
ON o.order_id=p.order_id
GROUP BY month_year,payment_type
ORDER BY month_year,payment_type,number_of_orders
```



Row	month_year	payment_type	number_of_orders
1	2016-09	credit_card	3
2	2016-10	UPI	63
3	2016-10	credit_card	253
4	2016-10	debit_card	2
5	2016-10	voucher	11
6	2016-12	credit_card	1
7	2017-01	UPI	197
8	2017-01	credit_card	582
9	2017-01	debit_card	9
10	2017-01	voucher	33
11	2017-02	UPI	398
12	2017-02	credit_card	1347

INSIGHTS

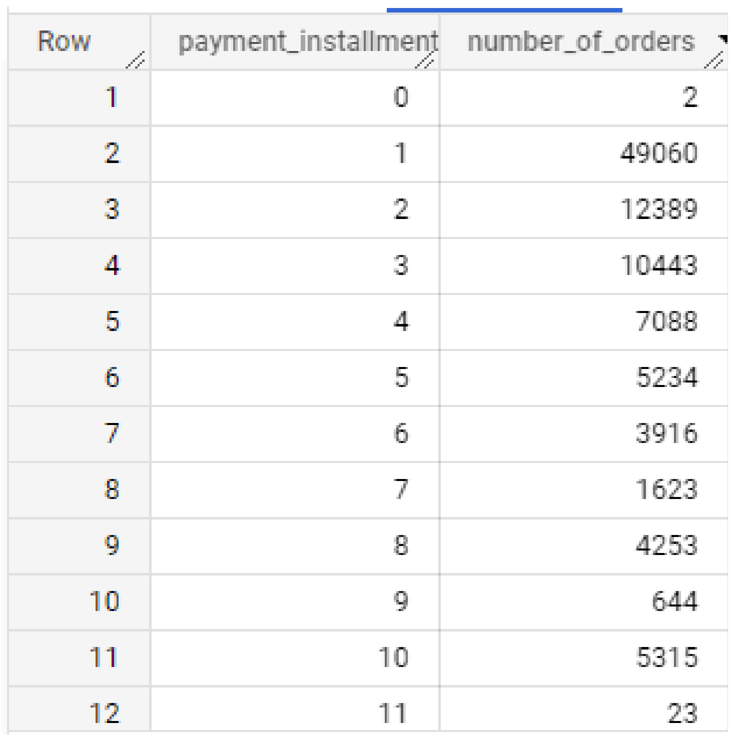
- The different types of payment used by customers were credit card,debit card, UPI, voucher
- Most number of orders have been made using credit card (2018-3 :‘5674’), (2018-4 :‘5441’), (2016-10 :‘253’),
- (2017-8 :‘3272’) ,(2017-9 :‘3274’)
- There were also 2 orders placed using a payment method which is “unknown”.
- Surprisingly none of the customers have opted for COD (Cash on Delivery)

<u>EXPLANATION</u>

Select the column the payment_type and use the function FORMAT DATE to fetch the details of the year and month from the order_purchase_timestamp and also name it as month_year.
Use the COUNT function to count the distinct orders and name it as number_of_orders.Since the payment details are available from a different table we have to join orders and payments table on order_id.
Group and order the data by month and payment type.

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

```
SELECT
p.payment_installments,
COUNT(distinct o.order_id) AS number_of_orders
FROM `Case_study.orders` as o
join `Case_study.payments` as p
ON o.order_id=p.order_id
GROUP BY payment_installments
ORDER BY payment_installments,number_of_orders
```



Row	payment_installment	number_of_orders
1	0	2
2	1	49060
3	2	12389
4	3	10443
5	4	7088
6	5	5234
7	6	3916
8	7	1623
9	8	4253
10	9	644
11	10	5315
12	11	23

INSIGHTS

- The highest number of orders were placed by customers through payment_installment of 1 month (49060 orders) hence making it the most preferred installment period.
- The least preferred were 22 and 23 months.

EXPLANATION

Select the column the payment_installments.
Use the COUNT function to count the distinct orders and name it as number_of_orders.Since the payment details are available from

```
a different table we have to join orders and payments table on
order_id.
Group data by payment installments and order the data by payment
installments and number_of_orders
```

RECOMMENDATIONS

- **The company is performing pretty well as there has been a significant growth from 2016-2018 in the number of orders placed by customers**

- **The state “SP” has high potential customers as it has the highest number of customers and orders placed throughout the years**

- **The state “RR” has less customers so the company should focus more on states like “RR” by giving them more offers to attract customers**

- **Most of the customers are preferring payment through credit cards and installments of 2 months**

- **In overall the company is doing great**