TARGET BUSINESS CASE STUDY

The data we are going to analyse is from American Retail Company Target Corporation. It was founded on July 1902 and its e-retail stores are very active in American Subcontinent. In our business case, we will focus on whole e-commerce data of Target Brazil from 2016 to 2018. In the dataset it comprises of total 8 tables. The table names are orders, **order_items, order_reviews, customer, geolocation, payments, products, sellers**. The datatypes present in our dataset are STRING, INTEGER, FLOAT and TIMESTAMP.

Below are some insights from our case study.

1. Total no. of order placed is **99441**.

2. There are total **99441 customers** in our dataset.

3. Highest payment for a particular order is **13664.08 real**.

4. There are **3095 total sellers** who are associated with Target Brazil, and 1849 from Sao Paolo which is highest is any particular state.

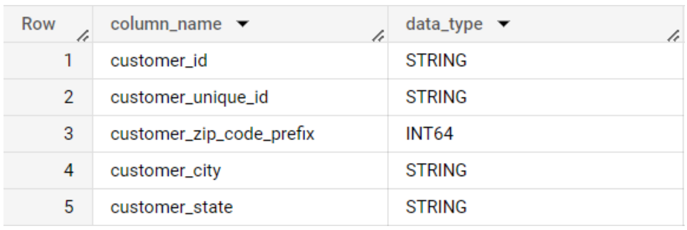
5. RR (Roraima), AL (Alagoas), TO (Tocantins) are three states with **0 sellers**.

6. There are 73 different product category.

7. Highest number of orders placed in a single day is 24 November 2017 with whooping amount of 1176 orders. The reason behind is **black Friday sale**.

Q.1.1 Data type of all columns in the "customers" table.

```
SELECT column_name,data_type
FROM `target_sql`.INFORMATION_SCHEMA.COLUMNS
WHERE table_name='customers';
```

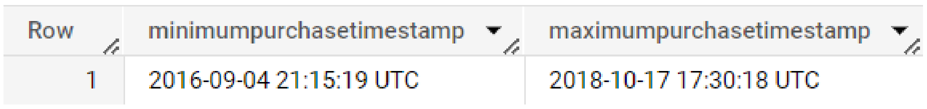


Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

There are total 5 columns in a “customers” table with 2 different datatypes which are STRING and FLOAT.

Q.1.2 Get the time range between which the orders were placed.

```
SELECT MIN( order_purchase_timestamp) minimumpurchasetimestamp,MAX(
order_purchase_timestamp) maximumpurchasetimestamp
FROM `target_sql.orders`;
```

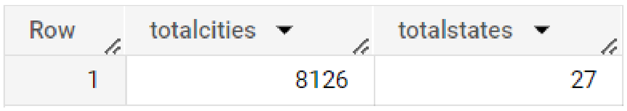


Row	minimumpurchasetimestamp	maximumpurchasetimestamp
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

The earliest order in our dataset is **2016-09-04** and latest is **2018-10-17**.

Q.1.3 Count the number of Cities and States in our dataset.

```
WITH cte AS(
  SELECT DISTINCT geolocation_city AS totalcities,
  geolocation_state AS totalstates
  FROM `target_sql.geolocation`
  UNION ALL
  SELECT  DISTINCT customer_city AS totalcities,
   customer_state AS totalstates
  FROM `target_sql.customers`
  UNION ALL
  SELECT DISTINCT seller_city AS totalcities,
   seller_state AS totalstates
  FROM `target_sql.sellers`
)
SELECT COUNT( DISTINCT cte.totalcities) as totalcities,
COUNT( DISTINCT cte.totalstates) AS totalstates
FROM cte;
```

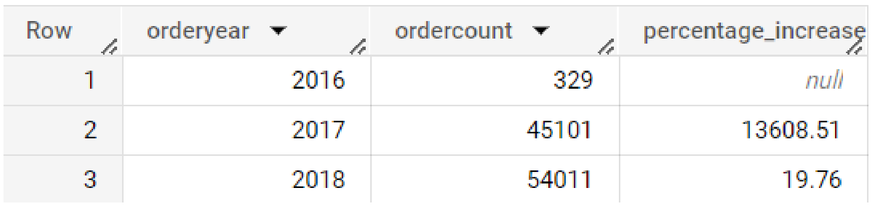


Row	totalcities	totalstates
1	8126	27

There are total **8126 cities and 27 states** of Brazil in our entire dataset.

Q.2.1 Is there a growing trend in the no. of orders placed over the past years?

```
WITH yearwiseorder AS(
SELECT *,LAG(ordercount)OVER(ORDER BY orderyear) as previous_order_count
FROM
(SELECT
EXTRACT(year FROM order_purchase_timestamp) AS orderyear,
COUNT(order_id) AS ordercount
FROM `target_sql.orders`
GROUP BY orderyear
ORDER BY orderyear)tbl)
SELECT orderyear,yearwiseorder.ordercount,
ROUND((ordercount-previous_order_count)*100/previous_order_count,2) AS percentage_increase
FROM yearwiseorder
ORDER BY orderyear;
```

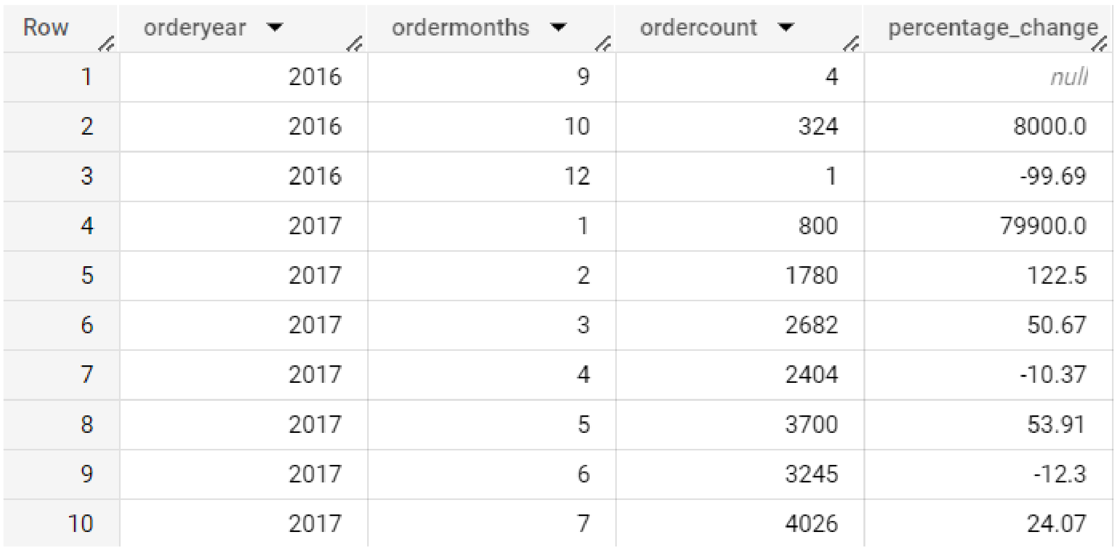


Row	orderyear	ordercount	percentage_increase
1	2016	329	*null*
2	2017	45101	13608.51
3	2018	54011	19.76

We can see sudden jump in order counts from 2016 to 2017(around 13609 %) and little more on 2018(around 20%). As in our dataset, only 4 months data is there for 2016. We may suspect that abnormal difference if the whole year data would have been provided.

Q.2.2 Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

```
WITH order_growth_month_lead AS
(
SELECT
*,
LAG(ordercount) OVER(ORDER BY orderyear, ordermonths) AS previous_month_order_count,
FROM
(
SELECT
EXTRACT(year FROM order_purchase_timestamp) AS orderyear,
EXTRACT(month FROM order_purchase_timestamp) AS ordermonths,
COUNT(order_id) AS ordercount,
FROM `target_sql.orders`
GROUP BY ordermonths, orderyear
ORDER BY 1,2
) AS order_growth_month
ORDER BY 1,2
)
SELECT orderyear,ordermonths,ordercount,
ROUND((ordercount-previous_month_order_count)*100/previous_month_order_count,2) AS
percentage_change
FROM order_growth_month_lead
ORDER BY orderyear,ordermonths;
```



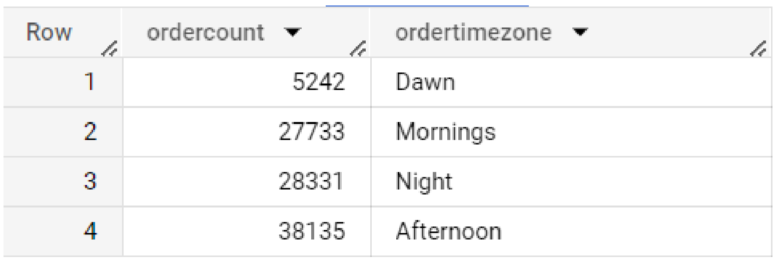
Row	orderyear	ordermonths	ordercount	percentage_change
1	2016	9	4	*null*
2	2016	10	324	8000.0
3	2016	12	1	-99.69
4	2017	1	800	79900.0
5	2017	2	1780	122.5
6	2017	3	2682	50.67
7	2017	4	2404	-10.37
8	2017	5	3700	53.91
9	2017	6	3245	-12.3
10	2017	7	4026	24.07

We can observe that during winter season (from May to September) the order counts are high because it their vacation period and also around October and November because of Black Friday sale.

Q.2.3 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

```
WITH otime AS(SELECT order_id,
EXTRACT(time FROM order_purchase_timestamp) AS ordertime
FROM `target_sql.orders`)
SELECT COUNT(order_id) AS ordercount,
CASE
WHEN ordertime BETWEEN '00:00:00' AND '06:59:59'
THEN 'Dawn'
WHEN ordertime BETWEEN '07:00:00' AND '12:59:59'
THEN 'Mornings'
WHEN ordertime BETWEEN '13:00:00' AND '18:59:59'
THEN 'Afternoon'
ELSE 'Night'
END AS ordertimezone
FROM otime
GROUP BY ordertimezone
ORDER BY ordercount;
```



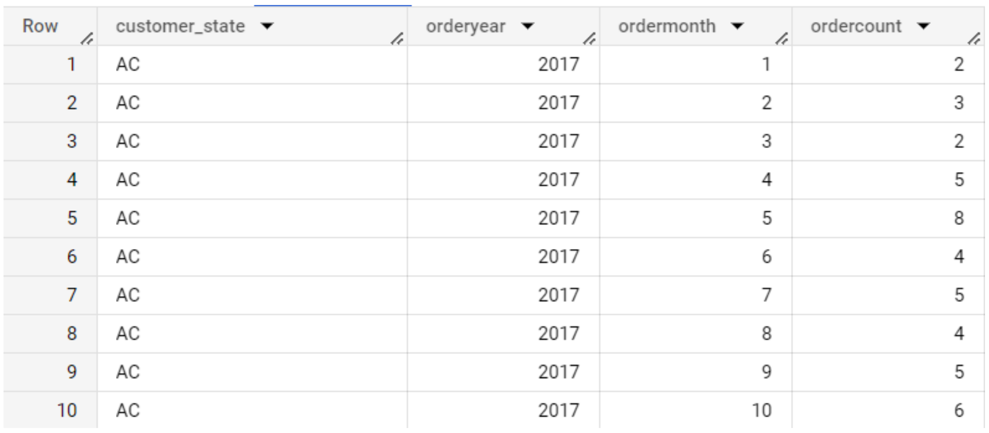
Row	ordercount	ordertimezone
1	5242	Dawn
2	27733	Mornings
3	28331	Night
4	38135	Afternoon

The normal **working hours for Brazil people is 8:00 to 18:00**. Therefore we can study that during night and afternoon order count is high because normally people get free after 18:00 evening.

Q.3.1 Get the month on month no. of orders placed in each state.

```
SELECT c.customer_state,
EXTRACT(year FROM order_purchase_timestamp) AS orderyear,
EXTRACT(month FROM order_purchase_timestamp) AS ordermonth,
COUNT(o.order_id) AS ordercount
FROM `target_sql.orders` AS o
INNER JOIN `target_sql.customers` AS c
ON o.customer_id = c.customer_id
GROUP BY ordermonth,c.customer_state,orderyear
```

```
ORDER BY c.customer_state,orderyear,ordermonth;
```

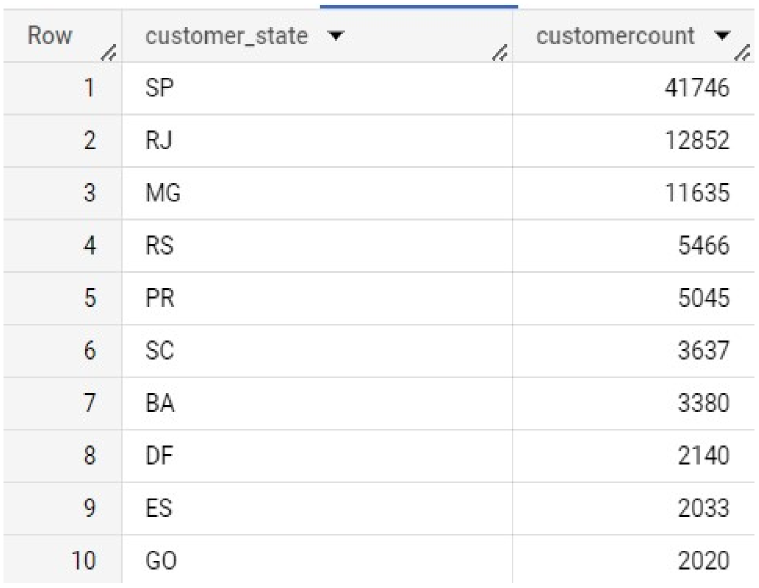


Row	customer_state	orderyear	ordermonth	ordercount
1	AC	2017	1	2
2	AC	2017	2	3
3	AC	2017	3	2
4	AC	2017	4	5
5	AC	2017	5	8
6	AC	2017	6	4
7	AC	2017	7	5
8	AC	2017	8	4
9	AC	2017	9	5
10	AC	2017	10	6

Higher number of orders placed during winter and vacation time in Brazil. For example. May, Jun, July and August.

Q.3.2 How are the customers distributed across all the states?

```
SELECT customer_state,COUNT(customer_unique_id) AS customercount
FROM `target_sql.customers`
GROUP BY customer_state
ORDER BY customercount DESC;
```

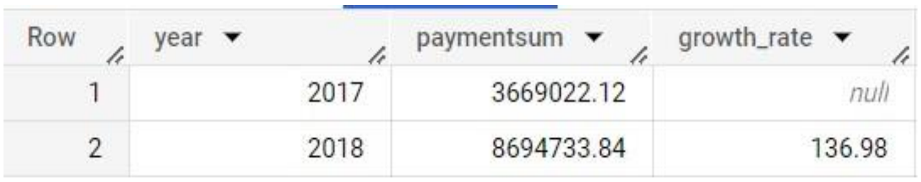


Row	customer_state	customercount
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020

Sao Paulo, Rio De Jeneiro, Minas Gerais are **most populated states** of Brazil, therefore they large amount of customers from our dataset.

Q.4.1 Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only). You can use the "payment_value" column in the payments table to get the cost of orders.

```
WITH cte AS(
  SELECT EXTRACT(year from o.order_purchase_timestamp) AS year,
  ROUND(SUM(p.payment_value),2) AS paymentsum
  FROM `target_sql.orders` AS o
  INNER JOIN `target_sql.payments` AS p
  ON o.order_id = p.order_id
  WHERE EXTRACT(year from o.order_purchase_timestamp) IN (2017,2018)
  AND EXTRACT(month from o.order_purchase_timestamp) BETWEEN 1 and 8
  GROUP BY year
  ORDER BY year
)
SELECT tb.year,tb.paymentsum,
ROUND(((tb.paymentsum-tb.lag_value)*100/tb.lag_value),2) as growth_rate
FROM
(SELECT *,
LAG(paymentsum) over ( order by cte.year asc) as lag_value
FROM cte
ORDER BY cte.year asc
) as tb;
```

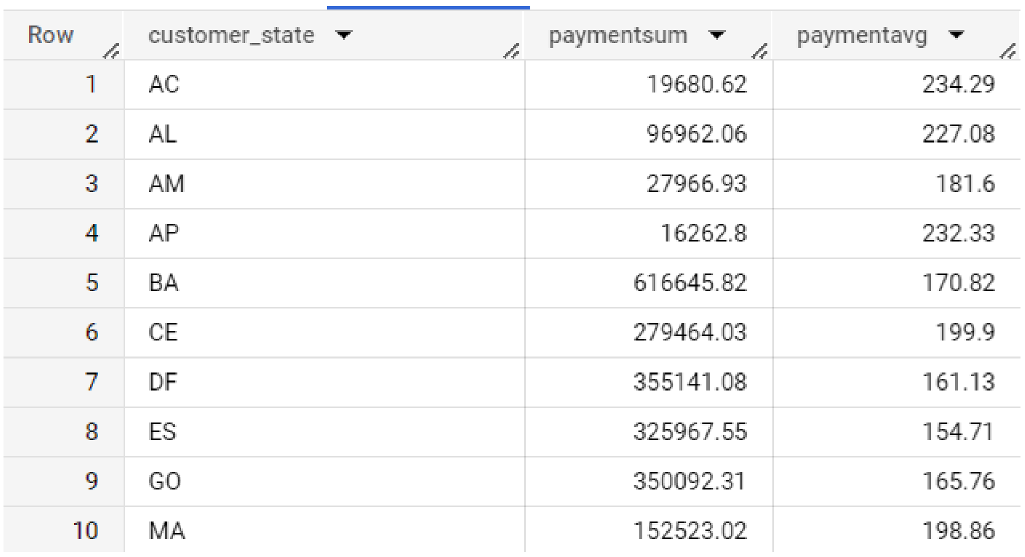


Row	year	paymentsum	growth_rate
1	2017	3669022.12	null
2	2018	8694733.84	136.98

We can see a significant growth in cost of orders from 2017 to 2018 which is around 137%.

Q.4.2 Calculate the Total & Average value of order price for each state.

```
SELECT c.customer_state ,ROUND(SUM(p.payment_value),2) AS
paymentsum,ROUND(AVG(p.payment_value),2) AS paymentavg
FROM `target_sql.payments` AS p
INNER JOIN `target_sql.orders` AS o
ON p.order_id=o.order_id
INNER JOIN `target_sql.customers` AS c
ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY c.customer_state;
```

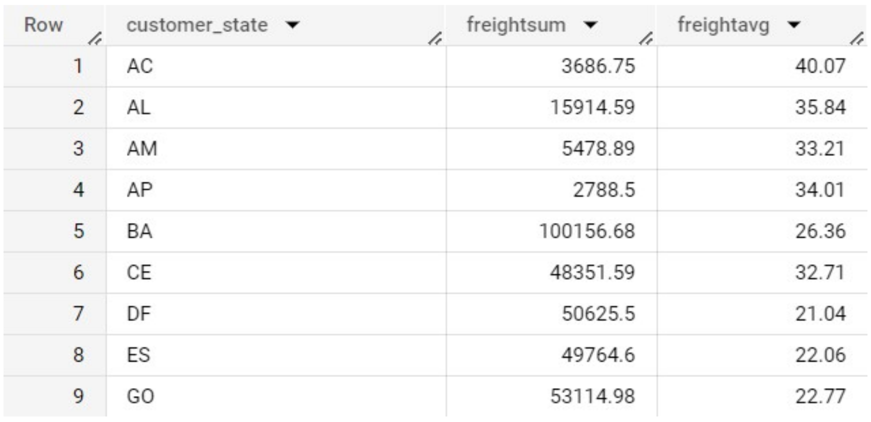


Row	customer_state	paymentsum	paymentavg
1	AC	19680.62	234.29
2	AL	96962.06	227.08
3	AM	27966.93	181.6
4	AP	16262.8	232.33
5	BA	616645.82	170.82
6	CE	279464.03	199.9
7	DF	355141.08	161.13
8	ES	325967.55	154.71
9	GO	350092.31	165.76
10	MA	152523.02	198.86

We can observe that payment sum is at least 15000 real and payment average is at least 150 real for all states.

Q.4.3 Calculate the Total & Average value of order freight for each state.

```
SELECT c.customer_state, ROUND(SUM(oi.freight_value),2) AS
freightsum,ROUND(AVG(oi.freight_value),2) AS freightavg
FROM `target_sql.order_items` AS oi
INNER JOIN `target_sql.orders` AS o
ON oi.order_id = o.order_id
INNER JOIN `target_sql.customers` AS c
ON o.customer_id=c.customer_id
GROUP BY c.customer_state
ORDER BY c.customer_state;
```



Row	customer_state	freightsum	freightavg
1	AC	3686.75	40.07
2	AL	15914.59	35.84
3	AM	5478.89	33.21
4	AP	2788.5	34.01
5	BA	100156.68	26.36
6	CE	48351.59	32.71
7	DF	50625.5	21.04
8	ES	49764.6	22.06
9	GO	53114.98	22.77

We can observe that freight sum is at least **2700 real** and freight average is at least **21 real** for all states.

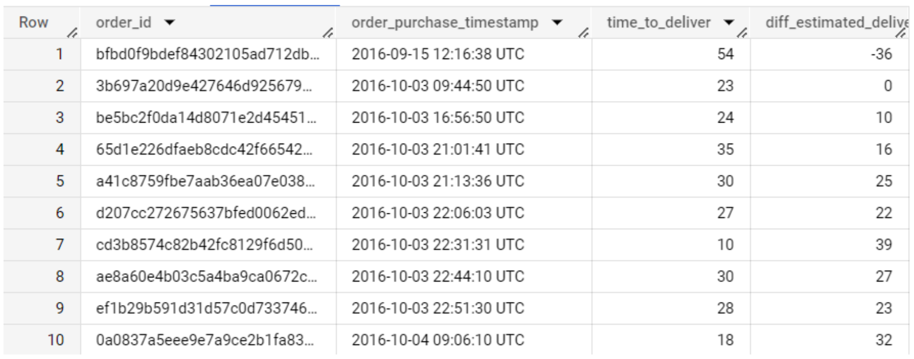
Q.5.1 Find the no. of days taken to deliver each order from the order's purchase date as delivery time.

Also, calculate the difference (in days) between the estimated & actual delivery date of an order. Do this in a single query.

You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

- time_to_deliver = order_delivered_customer_date - order_purchase_timestamp
- diff_estimated_delivery = order_estimated_delivery_date - order_delivered_customer_date

```
SELECT *
FROM(
SELECT order_id,order_purchase_timestamp,
DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,day) AS time_to_deliver,
DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,day) AS
diff_estimated_delivery
FROM `target_sql.orders`
ORDER BY order_purchase_timestamp
)
WHERE time_to_deliver IS NOT NULL
AND diff_estimated_delivery IS NOT NULL;
```



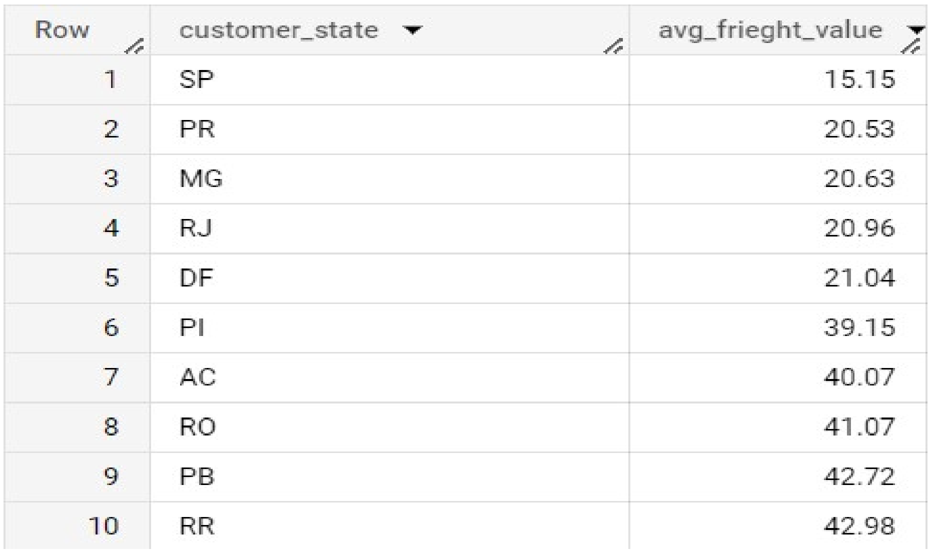
Row	order_id	order_purchase_timestamp	time_to_deliver	diff_estimated_delive
1	bfbd0f9bdef84302105ad712db...	2016-09-15 12:16:38 UTC	54	-36
2	3b697a20d9e427646d925679...	2016-10-03 09:44:50 UTC	23	0
3	be5bc2f0da14d8071e2d45451...	2016-10-03 16:56:50 UTC	24	10
4	65d1e226dfaeb8cdc42f66542...	2016-10-03 21:01:41 UTC	35	16
5	a41c8759fbe7aab36ea07e038...	2016-10-03 21:13:36 UTC	30	25
6	d207cc272675637bfed0062ed...	2016-10-03 22:06:03 UTC	27	22
7	cd3b8574c82b42fc8129f6d50...	2016-10-03 22:31:31 UTC	10	39
8	ae8a60e4b03c5a4ba9ca0672c...	2016-10-03 22:44:10 UTC	30	27
9	ef1b29b591d31d57c0d733746...	2016-10-03 22:51:30 UTC	28	23
10	0a0837a5eee9e7a9ce2b1fa83...	2016-10-04 09:06:10 UTC	18	32

In the above operation we remove those orders which are having null values in order delivery date and estimated delivery date.

Q.5.2 Find out the top 5 states with the highest & lowest average freight value.

```
WITH cte AS(SELECT *,
DENSE_RANK()over(ORDER BY avg_frieght_value) AS frieghtrank
FROM
(SELECT c.customer_state,ROUND(AVG(oi.freight_value),2) AS avg_frieght_value
FROM `target_sql.orders` AS o
INNER JOIN `target_sql.order_items` AS oi
ON o.order_id = oi.order_id
INNER JOIN `target_sql.customers` AS c
ON o.customer_id=c.customer_id
GROUP BY c.customer_state
ORDER BY avg_frieght_value DESC)tbl
)
SELECT customer_state,cte.avg_frieght_value FROM
cte
WHERE frieghtrank BETWEEN 1 AND 5
OR frieghtrank BETWEEN  23 AND 27
```

```
ORDER BY avg_frieght_value;
```

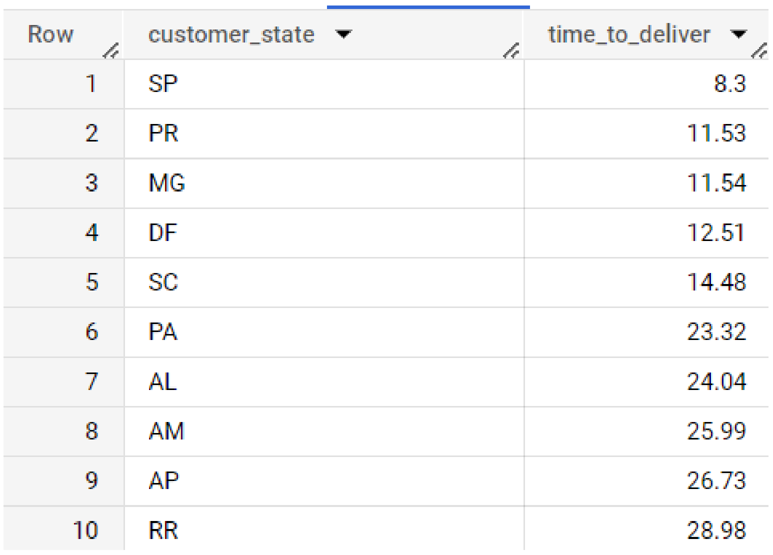


Row	customer_state	avg_frieght_value
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04
6	PI	39.15
7	AC	40.07
8	RO	41.07
9	PB	42.72
10	RR	42.98

In our table the above 5 rows have **lowest 5 avg_freight_value** and below 5 rows have **top 5 highest one**. States like Sao Paulo, Minas Gerais have high demands of orders and distance between seller and customer is less. Which results low freight value. States like RO (Rondonia) and RR (Roraima) are least populated and have less demands. Therefore sellers are very limited. That's why freight value is higher.

Q.5.3 Find out the top 5 states with the highest & lowest average delivery time.

```
WITH cte1 AS(SELECT*,
ROW_NUMBER()OVER(ORDER BY time_to_deliver) AS timerank
FROM(
SELECT
ROUND(AVG(DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp,day)),2) AS
time_to_deliver,
c.customer_state
FROM `target_sql.orders` AS o
INNER JOIN `target_sql.customers` AS c
ON o.customer_id=c.customer_id
GROUP BY c.customer_state
ORDER BY time_to_deliver DESC
)tbl
)
SELECT customer_state,time_to_deliver FROM
cte1
WHERE timerank BETWEEN 1 AND 5
OR timerank BETWEEN 23 AND 27
ORDER BY time_to_deliver;
```

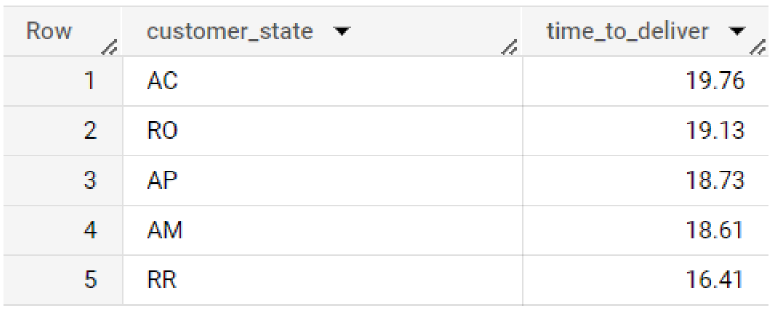


Row	customer_state	time_to_deliver
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48
6	PA	23.32
7	AL	24.04
8	AM	25.99
9	AP	26.73
10	RR	28.98

States like (SP) Sao Paulo, (MG) Minas Gerais are **economically powerful** and have **better transportation system**. Therefore it takes less time to reach the customer. On the other hand state like RR (Roraima) are hilly areas and have rough transportation system. Which in result takes more time.

Q.5.4 Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery. You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

```
SELECT
c.customer_state,ROUND(AVG(DATE_DIFF(order_estimated_delivery_date,order_delivered_customer
_date,day)),2) AS time_to_deliver
FROM `target_sql.orders` AS o
INNER JOIN `target_sql.customers` AS c
ON o.customer_id=c.customer_id
GROUP BY c.customer_state
ORDER BY time_to_deliver DESC
LIMIT 5;
```

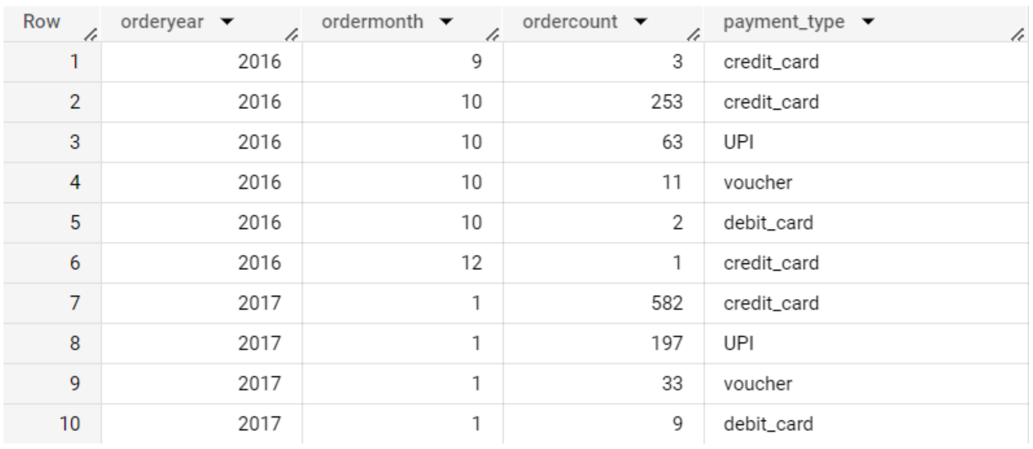


Row	customer_state	time_to_deliver
1	AC	19.76
2	RO	19.13
3	AP	18.73
4	AM	18.61
5	RR	16.41

The orders are delivered in these states is less than 20 days.

Q.6.1 Find the month on month no. of orders placed using different payment types.

```
SELECT EXTRACT(year FROM o.order_purchase_timestamp) AS orderyear,
EXTRACT(month FROM o.order_purchase_timestamp) AS ordermonth,COUNT(DISTINCT o.order_id) AS
ordercount,p.payment_type
FROM `target_sql.orders` AS o
INNER JOIN `target_sql.payments` AS p
ON o.order_id = p.order_id
GROUP BY orderyear,ordermonth,p.payment_type
ORDER BY orderyear,ordermonth;
```

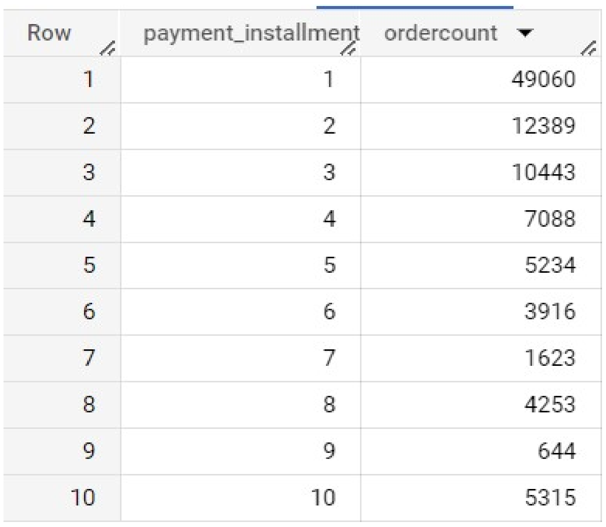


Row	orderyear	ordermonth	ordercount	payment_type
1	2016	9	3	credit_card
2	2016	10	253	credit_card
3	2016	10	63	UPI
4	2016	10	11	voucher
5	2016	10	2	debit_card
6	2016	12	1	credit_card
7	2017	1	582	credit_card
8	2017	1	197	UPI
9	2017	1	33	voucher
10	2017	1	9	debit_card

We can inspect that there is not repetition of usage of particular payment type over the months and years. Customers are using all different type of payments which are available.

Q.6.2 Find the no. of orders placed on the basis of the payment installments that have been paid.

```
SELECT payment_installments,COUNT(DISTINCT order_id) AS ordercount
FROM `target_sql.payments`
WHERE payment_installments!=0
GROUP BY payment_installments
ORDER BY payment_installments;
```



Row	payment_installment	ordercount
1	1	49060
2	2	12389
3	3	10443
4	4	7088
5	5	5234
6	6	3916
7	7	1623
8	8	4253
9	9	644
10	10	5315

We can conclude by the above table that in our dataset majority of the orders have 1, 2 or 3 payment instalment completed.

RECOMMENDATIONS

After studying the dataset, we can say that most of the orders, customers, sellers are coming from the major states like Sao Paulo, Rio De Jeneiro or Minas Gerias. Company needs some attention smaller states and improve their services as well. These states also have potential customers, which in results Company may be able to increase their sales and revenue.