Target SQL Business Case

Description:

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 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 analysing 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.

Problem Statement:

Assuming you are a data analyst/ scientist at Target, you have been assigned the task of analysing the given dataset to extract valuable insights and provide actionable recommendations.

What does 'good' look like?

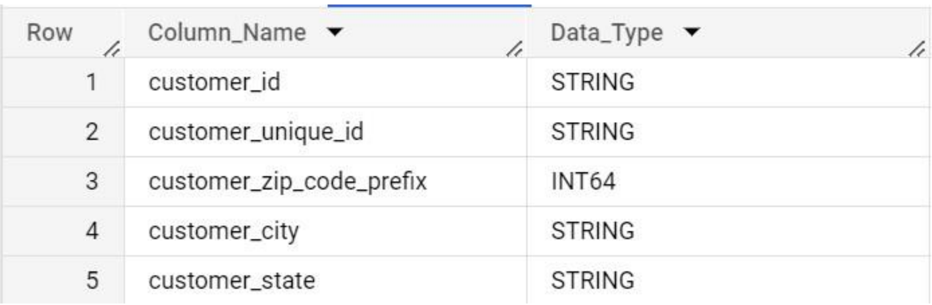
1. **Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:**

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

       Query:

       ```
       SELECT
         column_name AS Column_Name,
         data_type AS Data_Type
       FROM
         `Target_SQL.INFORMATION_SCHEMA.COLUMNS`
       WHERE
         table_name = 'customers'
       ```

Results:



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

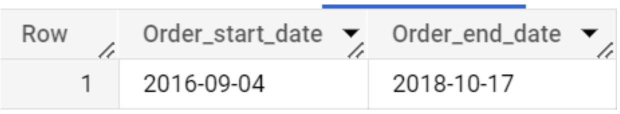
Note: In BigQuery we can get the above information using "schema" tab itselfwithout writing any query by just selecting the respective table under dataset.

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

Query:

```
SELECT
  DATE(MIN(order_purchase_timestamp)) AS Order_start_date,
  DATE(MAX(order_purchase_timestamp)) AS Order_end_date
FROM
  `Target_SQL.orders`
```

Results:

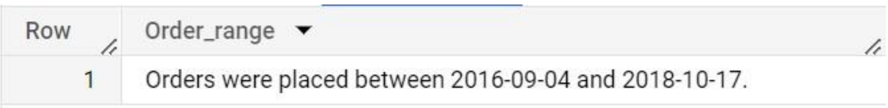


Row	Order_start_date	Order_end_date
1	2016-09-04	2018-10-17

Another Method: Instead of providing the dates in 2 different columns the output shown in statement.

```
SELECT
  CONCAT("Orders were placed between ", Order_start_date, " and ",
Order_end_date, ".") AS Order_range
FROM (
  SELECT
    DATE(MIN(order_purchase_timestamp)) AS Order_start_date,
    DATE(MAX(order_purchase_timestamp)) AS Order_end_date
  FROM
    `Target_SQL.orders`) AS X
```

Results:



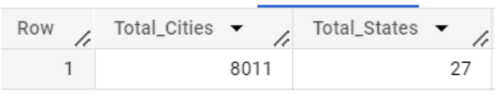
Row	Order_range
1	Orders were placed between 2016-09-04 and 2018-10-17.

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

Query:

```
SELECT
  COUNT(DISTINCT geolocation_city) AS Total_Cities,
  COUNT(DISTINCT geolocation_state) AS Total_States
FROM
  `Target_SQL.geolocation`
```

Results:



Row	Total_Cities	Total_States
1	8011	27

Insights:

- It's surprised that there were no orders placed by the customers who are living from the cities 3892 (i.e Order missing from cities = Total cities from geolocation table = 8011 - Total cities from customers table = 4119)

2. In-depth Exploration:

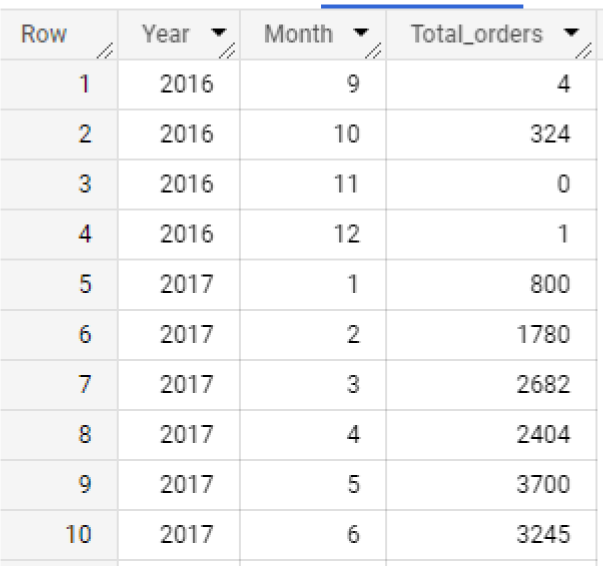
1. Is there a growing trend in the no. of orders placed over the past years?

Query:

```
SELECT
  Year, Month,
  COUNT(order_id) AS Total_orders
FROM (
  SELECT
    DISTINCT order_id,
    EXTRACT(year FROM order_purchase_timestamp) AS Year, EXTRACT(month FROM
order_purchase_timestamp) AS Month
  FROM
  `Target_SQL.orders`) AS X
GROUP BY Year,Month

Union DISTINCT
(select 2016 as Year,
11 as Month,
 0 as Total_orders
from `Target_SQL.orders`)
ORDER BY Year,Month
```

Note: Screenshot shows only 1st 10 rows as we no need to paste entire results.

Results:



Row	Year	Month	Total_orders
1	2016	9	4
2	2016	10	324
3	2016	11	0
4	2016	12	1
5	2017	1	800
6	2017	2	1780
7	2017	3	2682
8	2017	4	2404
9	2017	5	3700
10	2017	6	3245

Insights:

- Yes, the dataset provided clearly illustrates that there is a growing trend over the years from 2016 to 2018 however, if we look at the data month on month then, the trend is inconsistent as there are ups and downs.

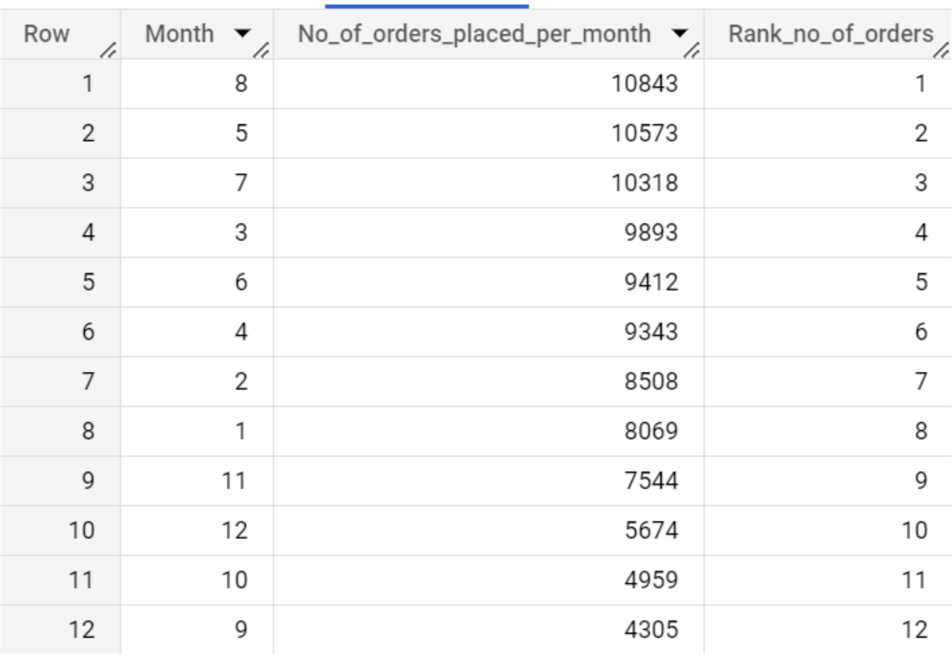
2. Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

Query:

```
SELECT
  EXTRACT(month FROM order_purchase_timestamp) AS Month,
  COUNT(order_id) AS No_of_orders_placed_per_month,
  DENSE_RANK() OVER(ORDER BY COUNT(order_id) DESC) AS
  Rank_no_of_orders
FROM
  `Target_SQL.orders`
GROUP BY
```

```
  Month
ORDER BY
  Rank_no_of_orders
```

Results:



Row	Month	No_of_orders_placed_per_month	Rank_no_of_orders
1	8	10843	1
2	5	10573	2
3	7	10318	3
4	3	9893	4
5	6	9412	5
6	4	9343	6
7	2	8508	7
8	1	8069	8
9	11	7544	9
10	12	5674	10
11	10	4959	11
12	9	4305	12

Insights:

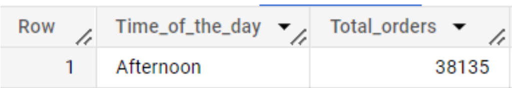
- Above results clearly shows that the maximum no. of orders were placed in the *month of August* and there is no doubt in that because, August is one of the best times to visit Brazil due to the *comfortable temperatures and lack of rain*.
- August brings *big festivals* (like Bumba Meu Boi in São Luís), ideal conditions for wildlife viewing in the Pantanal, and sunshine to the coastal cities.
- In addition, *Dia dos Pais, the Brazilian observation of Father's Day*, is celebrated every second Sunday in August
- Also, we can conclude that the least no. of orders were placed in the month of *September*

3. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)
    1. 0-6 hrs : Dawn
    2. 7-12 hrs : Mornings
    3. 13-18 hrs : Afternoon
    4. 19-23 hrs : Night

Query:

```
WITH Extracting_hrs as
  (
    SELECT
      EXTRACT(hour FROM order_purchase_timestamp) AS hours,
      order_id
    FROM
      `Target_SQL.orders`
  ),
  Total_orders_placed as
  (
    SELECT
      CASE
        WHEN hours BETWEEN 0 and 6 THEN "Dawn"
        WHEN hours BETWEEN 7 and 12 THEN "Mornings"
        WHEN hours BETWEEN 13 and 18 THEN "Afternoon"
      ELSE
        "Night"
      END as Time_of_the_day,
      COUNT(order_id) AS Total_orders
    FROM
        Extracting_hrs
    GROUP BY
      Time_of_the_day
  )
SELECT
  Time_of_the_day,
  Total_orders
FROM
  Total_orders_placed
ORDER BY
  Total_orders DESC
LIMIT
  1
```

Results:



Row	Time_of_the_day	Total_orders
1	Afternoon	38135

Insights:

- With the above results, we can conclude that most of the customers areplacing the orders in the Afternoon.

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

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

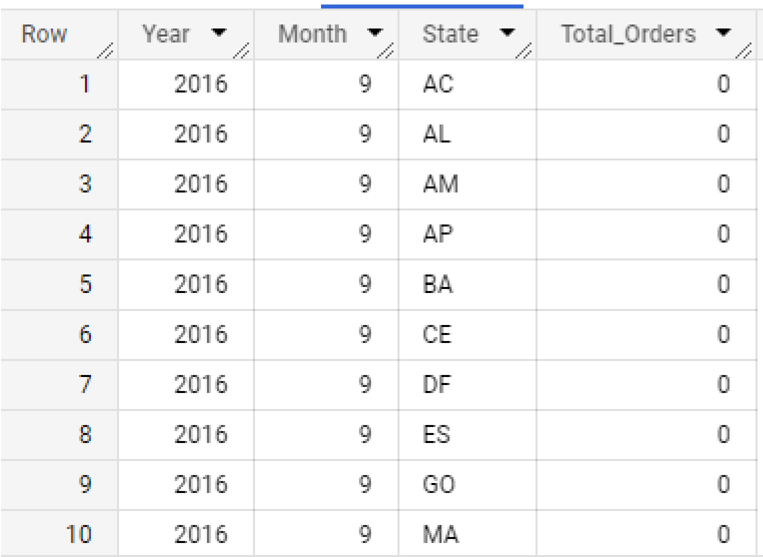
Query:

```
WITH Month_On_Month_Orders_Statewise as
(
SELECT Year,Month,State, COUNT(DISTINCT order_id) as Total_orders
FROM
  (
  SELECT
    EXTRACT(year FROM o.order_purchase_timestamp) AS Year,
    EXTRACT(month FROM o.order_purchase_timestamp) AS Month,
    o.order_id,o.customer_id,c.customer_state as State
  FROM
    `Target_SQL.customers` as c
  JOIN
   `Target_SQL.orders` as o
  ON o.customer_id = c.customer_id
  ) as X
GROUP BY Year,Month,State
),
Creating_Table_For_Missing_Month AS
(
  SELECT
    X.Year,X.Month,Y.State,0 as Total_orders
  FROM
    (SELECT 2016 as Year,11 as Month,
     FROM `Target_SQL.orders`
     GROUP BY Year,Month) AS X
  CROSS JOIN
    Month_On_Month_Orders_Statewise as Y
  GROUP BY X.Year,X.Month,Y.State
),
Creating_Table_For_Missing_States as
(
  SELECT
  A.Year,A.Month,B.customer_state as State,0 as Total_orders
  FROM
    Month_On_Month_Orders_Statewise as A
  CROSS JOIN
    `Target_SQL.customers` as B
  GROUP BY A.Year,A.Month,State
)

SELECT
  Year,Month,State, SUM(Total_Orders) as Total_Orders
FROM
  (SELECT
    DISTINCT X.Year, X.Month, X.State,
    (X.Total_orders + Y.Total_orders) AS Total_Orders
  FROM Month_On_Month_Orders_Statewise as X
  JOIN Creating_Table_For_Missing_States as Y
  ON X.Year = Y.Year and X.Month = Y.Month

       UNION DISTINCT

  SELECT
  Year, Month, State,Total_Orders
  FROM Creating_Table_For_Missing_Month
```

```
      UNION DISTINCT

 SELECT
 Year, Month, State,Total_Orders
 FROM Creating_Table_For_Missing_States
 ) as C
GROUP BY Year,Month,State
ORDER BY Year,Month,State
```

Note: Screenshot shows only 1st 10 rows as we no need to paste entire results.

Results:



Row	Year	Month	State	Total_Orders
1	2016	9	AC	0
2	2016	9	AL	0
3	2016	9	AM	0
4	2016	9	AP	0
5	2016	9	BA	0
6	2016	9	CE	0
7	2016	9	DF	0
8	2016	9	ES	0
9	2016	9	GO	0
10	2016	9	MA	0

Insights:

The most no. of orders were placed by the customers who are from state **SP** (~ 42% of orders) followed by **RJ** (~13% of orders) and **MG** (~12% of orders).

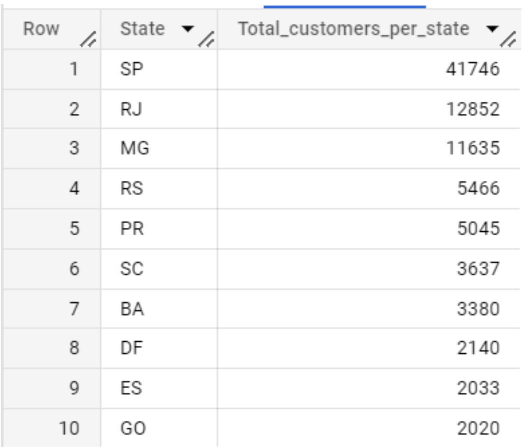
2. How are the customers distributed across all the states?

Query:

```
SELECT
  customer_state AS State,
  COUNT(DISTINCT customer_id) AS Total_customers_per_state
FROM
  `Target_SQL.customers`
GROUP BY
  State
ORDER BY
  Total_customers_per_state DESC
```

Note: Screenshot shows only 1st 10 rows as we no need to paste entire results.

Results:



Row	State	Total_customers_per_state
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

Insights:

The most no. of customers were from state **SP** (~ 42%) followed by **RJ** (~13%) and **MG** (~12%).

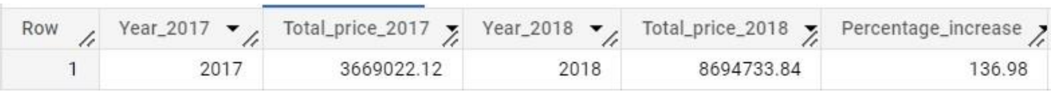
4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

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.

Query:

```
with Total_price_2017 as
(
  select
    Year,
    round(sum(payment_value),2) as Total_price,
    dense_rank() over(order by year) as rnk
  from
  (
  select
    p.payment_value,
    EXTRACT(month FROM o.order_purchase_timestamp) as Month,
    EXTRACT(year FROM o.order_purchase_timestamp) as Year
  from `Target_SQL.orders` as o
  join `Target_SQL.payments` as p ON o.order_id = p.order_id
  ) as X
  where Year = 2017 and (Month between 1 and 8)
  Group by Year
),
Total_price_2018 as
(
  select
    Year,
    round(sum(payment_value),2) as Total_price,
    dense_rank() over(order by year) as rnk
  from
  (
  select
    p.payment_value,
    EXTRACT(month FROM o.order_purchase_timestamp) as Month,
    EXTRACT(year FROM o.order_purchase_timestamp) as Year
  from `Target_SQL.orders` as o
  join `Target_SQL.payments` as p ON o.order_id = p.order_id
  ) as Y
  where Year = 2018 and (Month between 1 and 8)
  Group by Year
)
select
  t1.Year as Year_2017,
  t1.Total_price as Total_price_2017,
  t2.Year as Year_2018,
  t2.Total_price as Total_price_2018,
  round((((t2.Total_price - t1.Total_price) / t1.Total_price) *
  100),2) as Percentage_increase
from Total_price_2017 as t1
join Total_price_2018 as t2 ON t1.rnk = t2.rnk
```

Results:



Row	Year_2017	Total_price_2017	Year_2018	Total_price_2018	Percentage_increase
1	2017	3669022.12	2018	8694733.84	136.98

Insights:

- With the results above, we can conclude that there is a tremendous percentage increase (~137%) in the cost of orders from year 2017 to 2018.

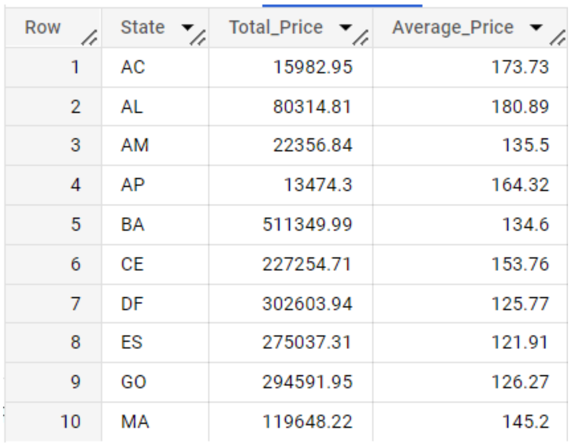
2. Calculate the Total & Average value of order price for each state.

Query:

```
SELECT
  c.customer_state AS State,
  ROUND(SUM(oi.price),2) AS Total_Price,
  ROUND(AVG(oi.price),2) AS Average_Price,
FROM `Target_SQL.customers` AS c
JOIN `Target_SQL.orders` AS o
ON
  c.customer_id = o.customer_id
JOIN
  `Target_SQL.order_items` AS oi
ON
  o.order_id = oi.order_id
GROUP BY
  c.customer_state
ORDER BY
  c.customer_state
```

Note: Screenshot shows only 1st 10 rows as we no need to paste entire results.

Results:



Row	State	Total_Price	Average_Price
1	AC	15982.95	173.73
2	AL	80314.81	180.89
3	AM	22356.84	135.5
4	AP	13474.3	164.32
5	BA	511349.99	134.6
6	CE	227254.71	153.76
7	DF	302603.94	125.77
8	ES	275037.31	121.91
9	GO	294591.95	126.27
10	MA	119648.22	145.2

Insights:

- Even though the total price of states SP, RJ and MG are more compare to states PB, AL and AC the Average price is less.
- It might be the items purchased from states SP, RJ and MG seems less price hence their averages are in the lower end and it's vice versa for states PB, AL and AC

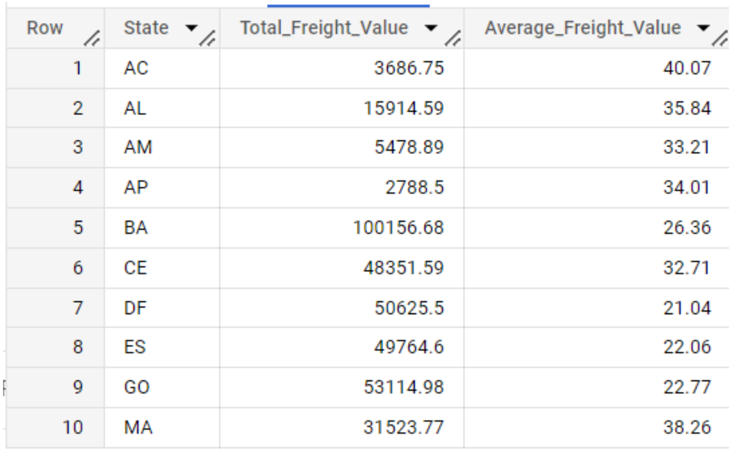
3. Calculate the Total & Average value of order freight for each state.

Query:

```
SELECT
  c.customer_state AS State,
  ROUND(SUM(oi.freight_value),2) AS Total_Freight_Value,
  ROUND(AVG(oi.freight_value),2) AS Average_Freight_Value,
FROM
  `Target_SQL.customers` AS c
JOIN
  `Target_SQL.orders` AS o
ON
  c.customer_id = o.customer_id
JOIN
  `Target_SQL.order_items` AS oi
ON
  o.order_id = oi.order_id
GROUP BY
  c.customer_state
ORDER BY
  c.customer_state
```

Note: Screenshot shows only 1st 10 rows as we no need to paste entire results.

Results:



Row	State	Total_Freight_Value	Average_Freight_Value
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
10	MA	31523.77	38.26

Insights:

- Even though the total freight value of states SP, RJ and MG are more compare to states RR, PB and RO the Average freight value is less.
- It might be due to the states SP, RJ and MG located near to the shipping location hence their freight averages are in the lower end and it's vice versa for states RR, PB and RO

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

1. Find the no. of days taken to deliver each order from the order's purchase date as delivery time.
   Also, calculate the difference (in days) between the estimated & actual delivery date of an order.
   Do this in a single query.

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

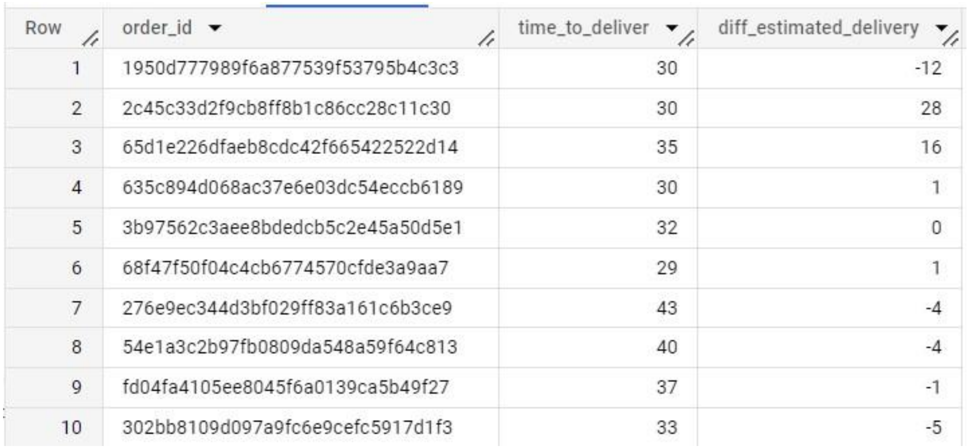
   1. **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
   2. **diff_estimated_delivery** = order_estimated_delivery_date - order_delivered_customer_date

Query:

```
SELECT
  order_id,
  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`
```

Note: Screenshot shows only 1st 10 rows as we no need to paste entire results.

Results:



Row	order_id	time_to_deliver	diff_estimated_delivery
1	1950d777989f6a877539f53795b4c3c3	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28c11c30	30	28
3	65d1e226dfaeb8cdc42f665422522d14	35	16
4	635c894d068ac37e6e03dc54eccb6189	30	1
5	3b97562c3aee8bdedcb5c2e45a50d5e1	32	0
6	68f47f50f04c4cb6774570cfde3a9aa7	29	1
7	276e9ec344d3bf029ff83a161c6b3ce9	43	-4
8	54e1a3c2b97fb0809da548a59f64c813	40	-4
9	fd04fa4105ee8045f6a0139ca5b49f27	37	-1
10	302bb8109d097a9fc6e9cefc5917d1f3	33	-5

Insights:

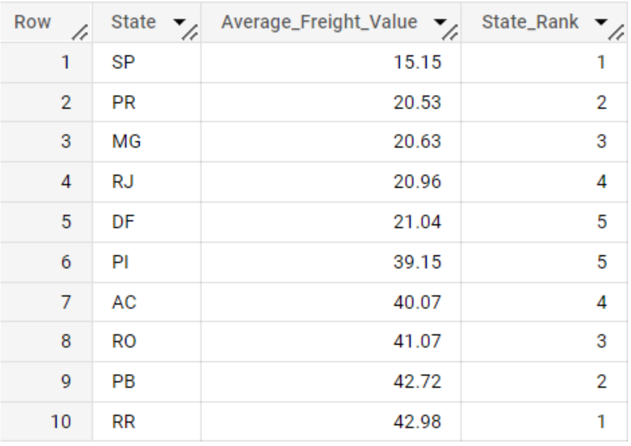
- Only ~6.6% of orders got delivered within the estimated delivery date and it shows the poor on tome delivery percentage.
- Please work the carriers and identify the root cause for the delay and try to improve it.

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

Query:

```
WITH Top_5_State_Average_Freight_Value as

(
  SELECT
      State,
      Average_Freight_Value,
      DENSE_RANK() OVER(ORDER BY Average_Freight_Value DESC) AS
      State_Rank
  FROM
     (SELECT
      c.customer_state AS State,
      ROUND(AVG(oi.freight_value),2) AS Average_Freight_Value
    FROM
      `Target_SQL.customers` AS c

    JOIN
      `Target_SQL.orders` AS o
    ON
      c.customer_id = o.customer_id
    JOIN
      `Target_SQL.order_items` AS oi
    ON
      o.order_id = oi.order_id
    GROUP BY
      c.customer_state
    ORDER BY
      c.customer_state) as X
),
Bottom_5_State_Average_Freight_Value as
(
  SELECT
      State,
      Average_Freight_Value,
      DENSE_RANK() OVER(ORDER BY Average_Freight_Value) AS State_Rank
  FROM
    (SELECT
      c.customer_state AS State,
      ROUND(AVG(oi.freight_value),2) AS Average_Freight_Value
    FROM
      `Target_SQL.customers` AS c
    JOIN
      `Target_SQL.orders` AS o
    ON
      c.customer_id = o.customer_id
    JOIN
      `Target_SQL.order_items` AS oi
    ON
      o.order_id = oi.order_id
    GROUP BY
      c.customer_state
    ORDER BY
      c.customer_state) as Y
)

SELECT
  *
FROM
  Top_5_State_Average_Freight_Value
WHERE
  State_Rank <= 5
    UNION ALL
SELECT
  *
FROM
  Bottom_5_State_Average_Freight_Value
WHERE
  State_Rank <= 5

ORDER BY Average_Freight_Value
```

Results:



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

Insights:

- In general, the freight value is directly proportional to the distance between delivery locations/states and the shipping location.

However,there are many factors that determines the freight value.

- The common determining factors of freight rates are: mode of transportation (truck, ship, train, aircraft), weight, size, distance, points ofpickup and delivery, and the actual goods being shipped

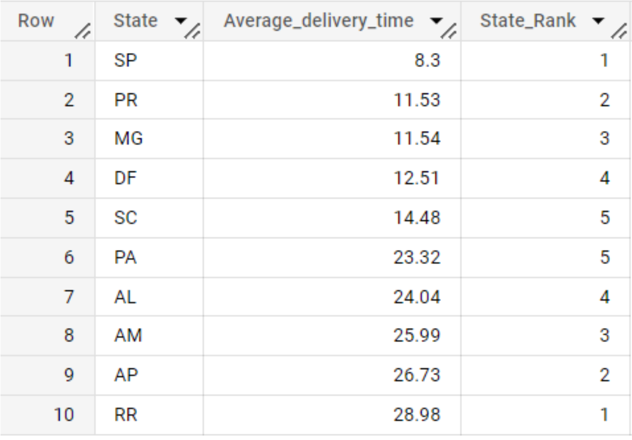
3. Find out the top 5 states with the highest & lowest average delivery time.

Query:

```
WITH Top_5_State_Average_Delivery_Time as
(
  SELECT
    State,
    ROUND(AVG(time_to_deliver),2) as Average_delivery_time,
    DENSE_RANK() OVER(ORDER BY ROUND(AVG(time_to_deliver),2) DESC) as
    State_Rank
  FROM
  (
    SELECT
      c.customer_state as State,
      o.order_id,
      DATE_DIFF(order_delivered_customer_date,
                order_purchase_timestamp, day) AS time_to_deliver
    FROM
      `Target_SQL.orders` as o
    JOIN
      `Target_SQL.customers` as c
    ON
      o.customer_id = c.customer_id
  ) as X
  GROUP BY State
),
Bottom_5_State_Average_Delivery_Time as(
  SELECT
    State,
    ROUND(AVG(time_to_deliver),2) as Average_delivery_time,
    DENSE_RANK() OVER(ORDER BY ROUND(AVG(time_to_deliver),2)) as
    State_Rank
  FROM
  (
    SELECT
      c.customer_state as State,
      o.order_id,
      DATE_DIFF(order_delivered_customer_date,
                order_purchase_timestamp, day) AS time_to_deliver
    FROM
      `Target_SQL.orders` as o
    JOIN
      `Target_SQL.customers` as c
    ON
      o.customer_id = c.customer_id
  ) as Y
  GROUP BY State
)

SELECT
  *
FROM
  Top_5_State_Average_Delivery_Time
```

```
WHERE
  State_Rank <= 5
    UNION ALL
SELECT
  *
FROM
  Bottom_5_State_Average_Delivery_Time
WHERE
  State_Rank <=5
ORDER BY
  Average_delivery_time
```

Results:



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

Insights:

- Delivery time can vary based on many factors like mode of transportation (truck, ship, train, aircraft), points of pickup and delivery, and the actual goods being shipped.

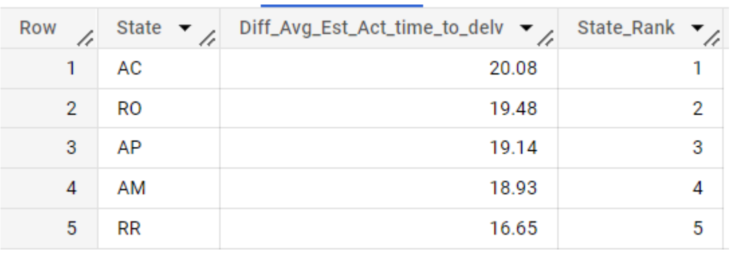
4. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.
   You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

Query:

```
WITH State_Average_Delivery_Time as
(
  SELECT
    State,
    ROUND(AVG(Act_time_to_delv),2) as Act_Avg_delv_time,
    ROUND(AVG(Est_time_to_delv),2) as Est_Avg_dev_time
  FROM
  (
    SELECT
      c.customer_state as State,
      o.order_id,
      DATE_DIFF(order_delivered_customer_date,
               order_purchase_timestamp, day) AS Act_time_to_delv,
      DATE_DIFF(order_estimated_delivery_date,
                order_purchase_timestamp, day) AS Est_time_to_delv
    FROM
```

```
      `Target_SQL.orders` as o
    JOIN
      `Target_SQL.customers` as c
    ON
      o.customer_id = c.customer_id
    WHERE o.order_status = "delivered"
  ) as X
  GROUP BY State
),
Tof_5_States_Average_Delivery_Time as
(
  SELECT
    State,
    ROUND(AVG(Diff_Est_Act_time_to_delv),2) AS
    Diff_Avg_Est_Act_time_to_delv,
    DENSE_RANK() OVER(ORDER BY ROUND(AVG(Diff_Est_Act_time_to_delv),2)
    DESC) as State_Rank
  FROM
    (
    SELECT
      State,
      (Est_Avg_dev_time-Act_Avg_delv_time) AS
       Diff_Est_Act_time_to_delv,
    FROM
      State_Average_Delivery_Time
      ) as Y
  GROUP BY State
)
SELECT
  *
FROM
  Tof_5_States_Average_Delivery_Time
WHERE
  State_Rank <= 5
ORDER BY State_Rank
```

Results:



Row	State	Diff_Avg_Est_Act_time_to_delv	State_Rank
1	AC	20.08	1
2	RO	19.48	2
3	AP	19.14	3
4	AM	18.93	4
5	RR	16.65	5

Insights:

- Above resulted 5 states are the ones which are takes less time (fast) for delivery than the estimated time so, identify the carriers who handles these lanes and see if you can onboard them to the other states which are taking more time to delivery.

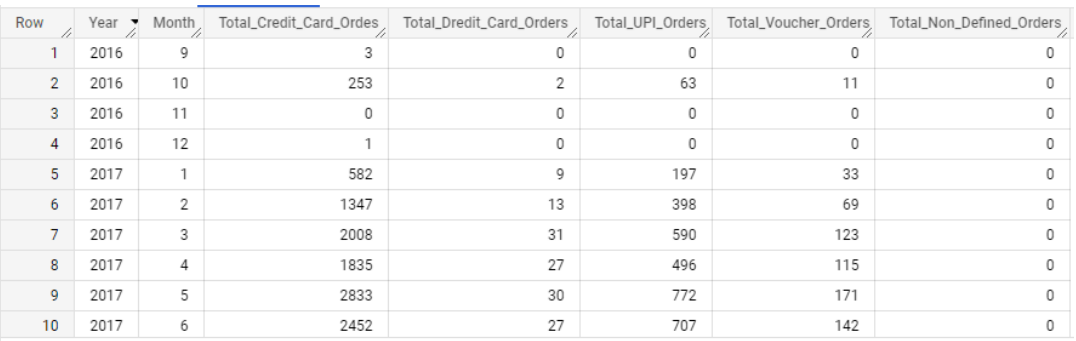
6. Analysis based on the payments:

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

Query:

```
With CTE1 as
(
  SELECT
    DISTINCT o.order_id,p.payment_type,
    EXTRACT(year FROM order_purchase_timestamp) AS Year,
    EXTRACT(month FROM order_purchase_timestamp) AS Month
  FROM
    `Target_SQL.orders` as o
  JOIN
    `Target_SQL.payments` as p
  ON
  o.order_id = p.order_id
),
Creating_Table_For_Missing_Month AS
(
  SELECT
    X.Year,X.Month,
    0 as Total_Credit_Card_Ordes,
    0 as Total_Dredit_Card_Orders,
    0 as Total_UPI_Orders,
    0 as Total_Voucher_Orders,
    0 as Total_Non_Defined_Orders
  FROM
    (SELECT 2016 as Year,11 as Month,
     FROM `Target_SQL.orders`
     GROUP BY Year,Month) AS X
  CROSS JOIN CTE1 as Y
  GROUP BY X.Year,X.Month
),
Categorize_payment_type as
(
  SELECT
  Year,
  Month,
  SUM(CASE WHEN payment_type = "credit_card" THEN 1 ELSE 0 END) AS
Total_Credit_Card_Ordes,
  SUM(CASE WHEN payment_type = "debit_card" THEN 1 ELSE 0 END) AS
Total_Dredit_Card_Orders,
  SUM(CASE WHEN payment_type = "UPI" THEN 1 ELSE 0 END) AS Total_UPI_Orders,
  SUM(CASE WHEN payment_type = "voucher" THEN 1 ELSE 0 END) AS
Total_Voucher_Orders,
  SUM(CASE WHEN payment_type = "not_defined" THEN 1 ELSE 0 END) AS
Total_Non_Defined_Orders
FROM CTE1
GROUP BY Year, Month
)

SELECT * FROM Categorize_payment_type
  UNION DISTINCT
SELECT * FROM Creating_Table_For_Missing_Month

ORDER BY Year, Month
```

Results:



Row	Year	Month	Total_Credit_Card_Ordes	Total_Dredit_Card_Orders	Total_UPI_Orders	Total_Voucher_Orders	Total_Non_Defined_Orders
1	2016	9	3	0	0	0	0
2	2016	10	253	2	63	11	0
3	2016	11	0	0	0	0	0
4	2016	12	1	0	0	0	0
5	2017	1	582	9	197	33	0
6	2017	2	1347	13	398	69	0
7	2017	3	2008	31	590	123	0
8	2017	4	1835	27	496	115	0
9	2017	5	2833	30	772	171	0
10	2017	6	2452	27	707	142	0

Insights:

- The above results illustrates that most of the customers are using theircredit cards for EMI/payments compare to other payment methods.
- It’s weird that there is no payment details for the order_id “bfbd0f9bdef84302105ad712db648a6c” which shows as delivered status in Orders table.
- Thera are 3 cancelled orders recorded in payments table with payment type as “not defined”

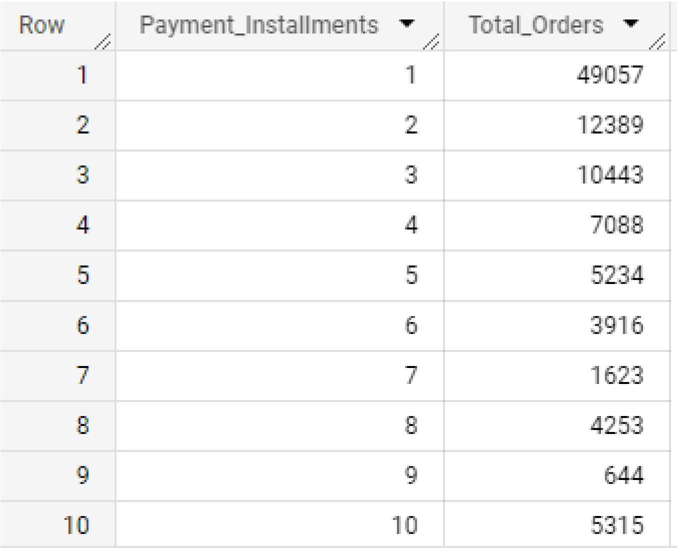
2. Find the no. of orders placed on the basis of the payment installments that have been paid.

Query:

```
SELECT
  p.payment_installments AS Payment_Installments,
  COUNT(DISTINCT o.order_id) as Total_Orders
FROM
  `Target_SQL.orders` AS o
JOIN
  `Target_SQL.payments` AS p
ON
  o.order_id = p.order_id
WHERE
  p.payment_value >0
  AND p.payment_installments> 0
GROUP BY
Payment_Installments
ORDER BY
Payment_Installments
```

Note: Screenshot shows only 1st 10 rows as we no need to paste entire results.

Results:



Row	Payment_Installments	Total_Orders
1	1	49057
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

Insights:

- The results above shows that ~49% payments were cleared by customers with single installment, ~12% and ~10% of payments were cleared within two and three installments respectively.
- Note that this conclusion has been made based on the assumption that there were no EMI's pending as we don't have a separate column which shows the balance amount in payments table.

Recommendations:

- Please rollout the discounts and rewards in the day of time (Dawn, Mornings and Nights) to impress the consumers/customers.
- Work with the data engineers or the ones who maintains the data base to include the payment information for the order_id "bfbd0f9bdef84302105ad712db648a6c" which shows as delivered but no payment record.
- Thera are 3 cancelled orders recorded in payments table with payment type as "not defined" so, these can be removed from the payments table as they are showing as cancelled In orders table.
- It's surprise that all the customers were purchased only once over a period of more than 2 years so, request the data engineers to investigate and see if the data maintained accurately? If it's true then, you must improve your customer service and build a good rapport with the customers.
- Kindly focus on the states which has placed least number of orders and rollout some coupons/discounts/rewards to increase the number of orders.

- Focus on the carriers who are unable to deliver the load on time (OTIF) and see if you can replace with the best performing carriers or understand the challenges and see if you can offer any additional benefits/allowances that help them to make the on time delivery.

- I have noticed that there were no orders placed in the month of Nov’2016 and it might be missing form the data base so, work with the concern team to get them added.

- Last but not least, kindly focus on the cities (3892) where not even single order placed by the customers (i.e Total cities from geolocation table = 8011 and Total cities from customers table = 4119) I would recommend to provide the BOGO – Buy One Get One offer and see if the customers are interested and that will help to expand the business and increases the customers as well.