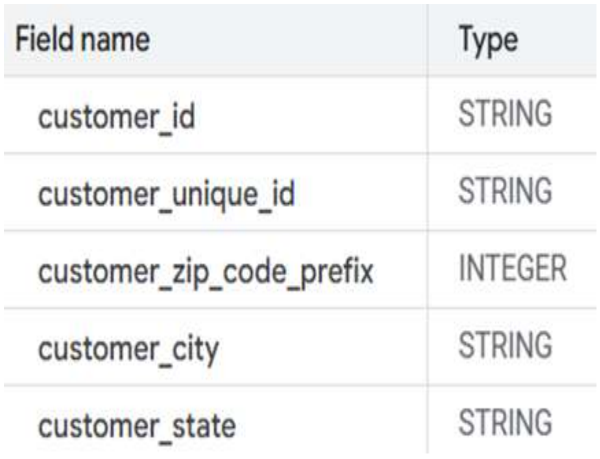
BUSINESS CASE STUDY – TARGET

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.**

      Check the schema tab for the customers table of the Target dataset.
      Or
      Run the below query to display the data type of all columns of the customers table
      Query:

```
SELECT column_name, data_type
FROM `scaler-rdbms.Target.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'customers';
```

Result:



Field name	Type
customer_id	STRING
customer_unique_id	STRING
customer_zip_code_prefix	INTEGER
customer_city	STRING
customer_state	STRING

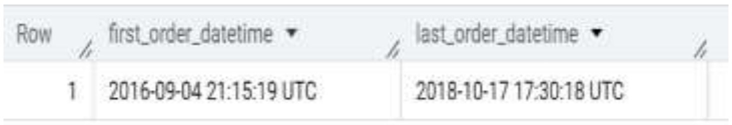
Insights: **1.** There are a total of 5 attributes in the customers table.
2. Four attributes are of string data type and one attribute is integer data type.
3. The table contains details of the customer id and the location details i.e., the city, state and zip code of each customer.

2. Time range between which the orders were placed.

Query:

```
SELECT MIN(order_purchase_timestamp) AS first_order_datetime,
       MAX(order_purchase_timestamp) AS last_order_datetime
FROM `Target.orders`
```

Result:



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

Insights: 1. The first order has been placed on September 4th 2016 at around 9:15 PM UTC.

2. The orders data until 17th October 2018 5:30 PM UTC is available.

3. The orders data for close to 2 years is available.

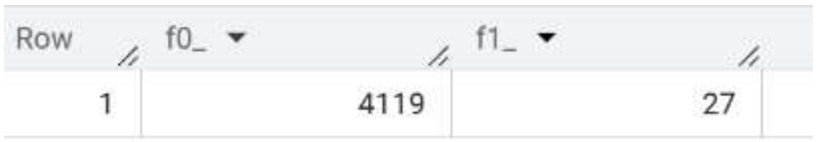
3. Count of the Cities & States of customers who ordered during the given period.

(Assuming: Given period to be the period as asked in the previous question)

Query:

```
SELECT COUNT(DISTINCT c.customer_city),
      COUNT(DISTINCT c.customer_state)
FROM `Target.customers` c JOIN `Target.orders` o
ON c.customer_id = o.customer_id
WHERE o.order_purchase_timestamp BETWEEN
                                  (SELECT MIN(order_purchase_timestamp) FROM `Target.orders`)
                              AND (SELECT MAX(order_purchase_timestamp) FROM `Target.orders`)
```

Result:



Row	f0_	f1_
1	4119	27

Insights: 1. A total of 27 states have placed ordered in around 2 years' time. (Sep 2016 to Oct 2018)

2. 4119 cities from 27 states have placed orders.

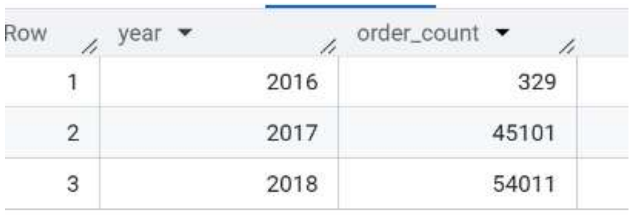
2. Growth trend Exploration:

1. Growth trend in the no. of orders placed over the past years.

Query:

```
SELECT  EXTRACT(year FROM order_purchase_timestamp) AS year,
        COUNT(*) as order_count
FROM `Target.orders`
GROUP BY  EXTRACT(year FROM order_purchase_timestamp)
ORDER BY year
```

Result:



Row	year	order_count
1	2016	329
2	2017	45101
3	2018	54011

Insights: 1. In 2016, 329 orders have been placed.

2. In 2017, 45101 orders have been placed, a growth in the number of orders by around 44800.

3. Similarly, 2018 also shows a growth trends in the number of orders by close to 9000 orders.

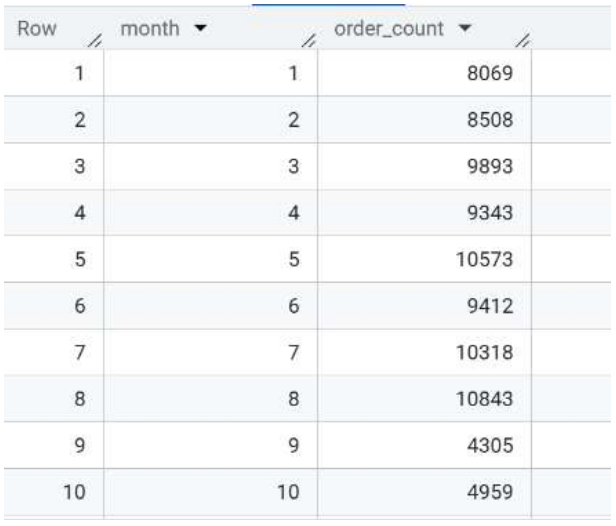
2. Monthly seasonality in terms of the no. of orders being placed.

Query to check only the monthly seasonality irrespective of the year.

Query1:

```
SELECT  EXTRACT(month FROM order_purchase_timestamp) AS month,
        COUNT(*) as order_count
FROM `Target.orders`
GROUP BY  EXTRACT(month FROM order_purchase_timestamp)
ORDER BY month
```

Result1:



Row	month	order_count
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318
8	8	10843
9	9	4305
10	10	4959

Insights: 1. Maximum orders have been placed in the month of May, July and August.

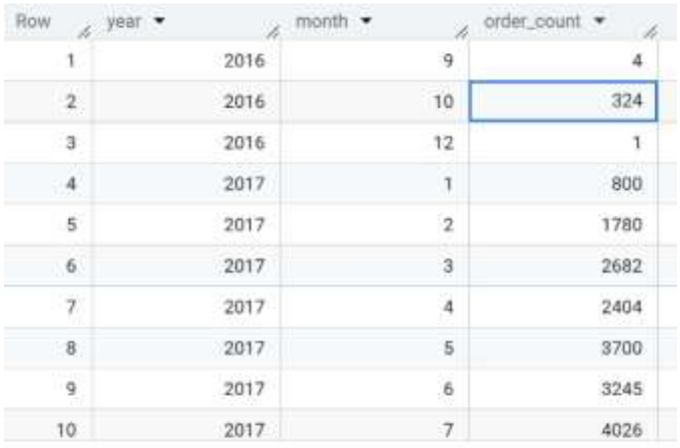
2. September has the minimum orders placed. Most orders are in the the year 2017.

Query to check the monthly seasonality by the year as orders have been placed since 2016 Sep until 2018 Oct.

Query2:

```
SELECT  EXTRACT(year FROM order_purchase_timestamp) year,
        EXTRACT(month FROM order_purchase_timestamp) AS month,
        COUNT(*) as order_count
FROM `Target.orders`
GROUP BY  EXTRACT(year FROM order_purchase_timestamp),
          EXTRACT(month FROM order_purchase_timestamp)
ORDER BY year, month
```

Result2:



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

Insights: 1. There is a growing trend in the number of orders month on month every year.

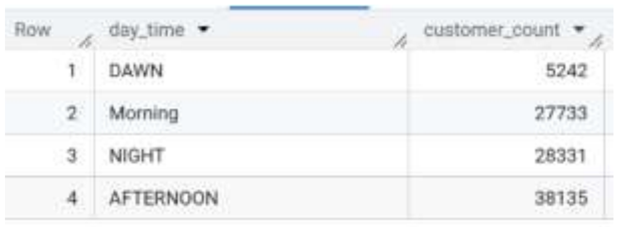
2. November of 2017, has the most orders placed.
3. The orders count has been declined towards the end of 2018 from September.
4. There is reduction in the number of orders in December of every year(2016, 2017) when compared to the rising pattern in the other months.

3. **Time of the day, 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:

```
SELECT day_time, COUNT(*) AS order_count
FROM (
SELECT  *,
        CASE WHEN FORMAT_DATE("%H",order_purchase_timestamp) BETWEEN "00" AND "06" THEN "DAWN"
             WHEN FORMAT_DATE("%H",order_purchase_timestamp) BETWEEN "07" AND "12" THEN "Morning"
             WHEN FORMAT_DATE("%H",order_purchase_timestamp) BETWEEN "13" AND "18" THEN "AFTERNOON"
             ELSE "NIGHT"
        END AS day_time
FROM `Target.orders`
)
GROUP BY day_time
ORDER BY order_count
```

Result:



Row	day_time	customer_count
1	DAWN	5242
2	Morning	27733
3	NIGHT	28331
4	AFTERNOON	38135

Insights: 1. Most orders are placed during the afternoon i.e., between 1 pm to 6 pm

2. Very few orders have been placed during the early hours of the day i.e., from midnight till 6 in the morning.
3. The orders placed in the morning (7 to 1pm) and night (6pm to midnight) are almost the same.

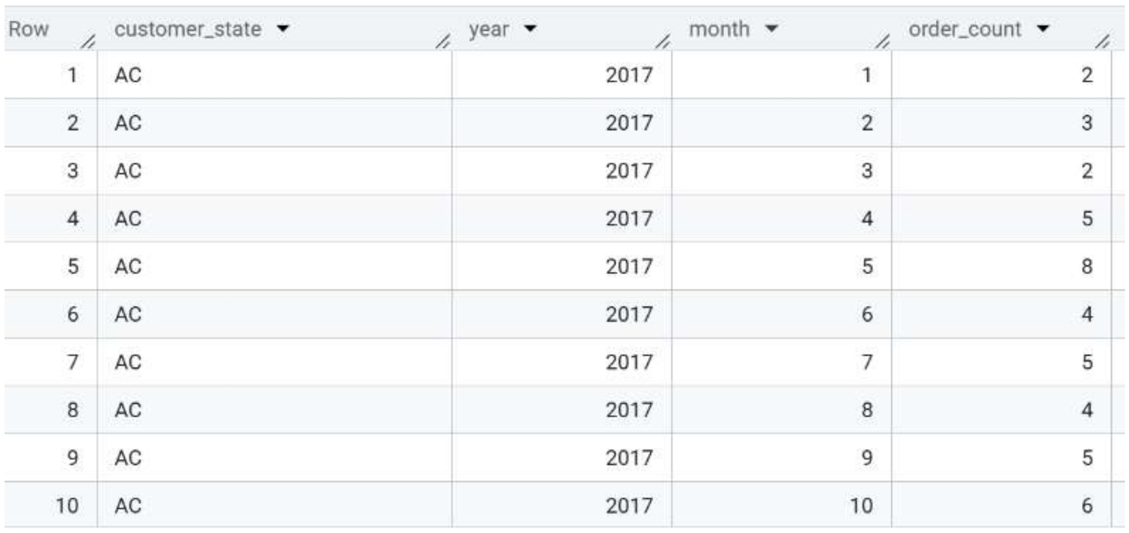
3. Evolution of E-commerce orders in the Brazil region:

1. Month on month no. of orders placed in each state.

Query:

```
SELECT  c.customer_state,
       EXTRACT(year FROM o.order_purchase_timestamp) year,
        EXTRACT(month FROM o.order_purchase_timestamp) AS month,
        COUNT(*) as order_count
FROM `Target.customers`c JOIN `Target.orders` o
ON c.customer_id = o.customer_id
GROUP BY    c.customer_state,
            EXTRACT(year FROM o.order_purchase_timestamp),
            EXTRACT(month FROM o.order_purchase_timestamp)
ORDER BY c.customer_state, year, month
```

Result:



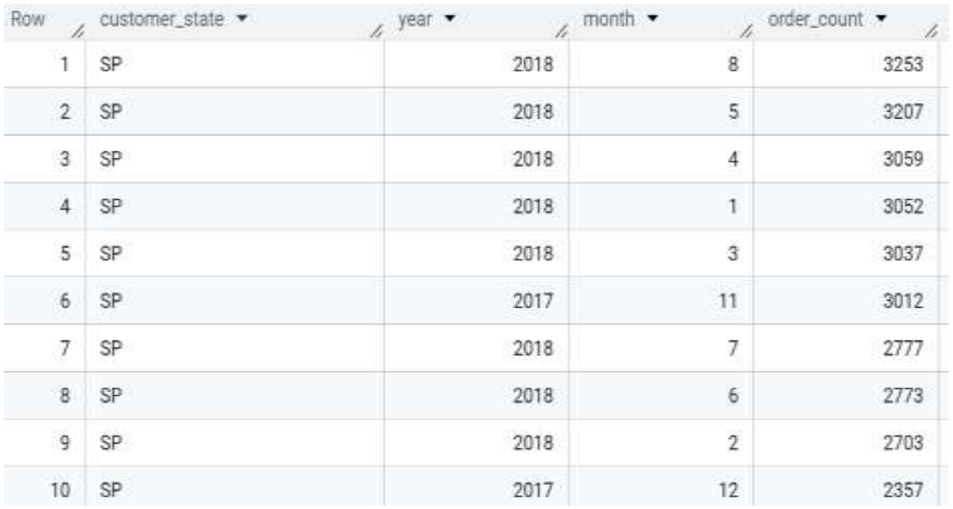
Row	customer_state	year	month	order_count
1	AC	2017	1	2
2	AC	2017	2	3
3	AC	2017	3	2
4	AC	2017	4	5
5	AC	2017	5	8
6	AC	2017	6	4
7	AC	2017	7	5
8	AC	2017	8	4
9	AC	2017	9	5
10	AC	2017	10	6

(Ordered the same query by the count of the orders)

Query:

```
SELECT  c.customer_state, EXTRACT(year FROM o.order_purchase_timestamp) year,
        EXTRACT(month FROM o.order_purchase_timestamp) AS month,
        COUNT(*) as order_count
FROM `Target.customers`c JOIN `Target.orders` o
ON c.customer_id = o.customer_id
GROUP BY    c.customer_state,EXTRACT(year FROM o.order_purchase_timestamp),
            EXTRACT(month FROM o.order_purchase_timestamp)
ORDER BY order_count desc
```

Result:



Row	customer_state	year	month	order_count
1	SP	2018	8	3253
2	SP	2018	5	3207
3	SP	2018	4	3059
4	SP	2018	1	3052
5	SP	2018	3	3037
6	SP	2017	11	3012
7	SP	2018	7	2777
8	SP	2018	6	2773
9	SP	2018	2	2703
10	SP	2017	12	2357

Insights: 1. Most orders have been placed from the state Sao Paulo – SP.

2. Very few orders have been placed from the state Acre – AC.

3. Some states have somewhat constant orders in every month while some states have shown an increase in the number of orders over the months.

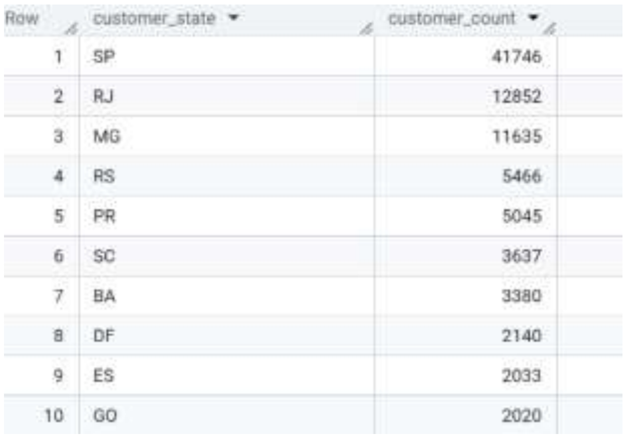
4. Some states like Sao Paulo - SP, Minas Gerias - MG and Rio de Janeiro - RJ have the most orders.

2. Customers distribution across all the states.

Query:

```
SELECT  customer_state,
        COUNT(*) as customer_count
FROM `Target.customers`
GROUP BY customer_state
ORDER BY customer_count DESC
```

Result:



Row	customer_state	customer_count
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: 1. Most customers are from the state Sao Paulo – SP.

2. There are good number of customers from the states Minas Gerias - MG and Rio de Janeiro – RJ.

3. Very few customers are from Roraima - RR, Amapa - AP, Acre - AC

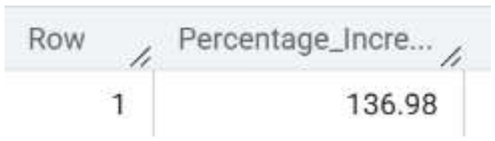
4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

1. % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

Query:

```
WITH ORDERS_PAYMENT AS
(
  SELECT  EXTRACT(YEAR FROM order_purchase_timestamp) year ,
          SUM(payment_value) total_amount
  FROM (
          SELECT  o.order_id, o.order_purchase_timestamp,
                  p.payment_value
          FROM `Target.orders` o JOIN `Target.payments` p
                ON o.order_id = p.order_id
          WHERE EXTRACT(MONTH FROM order_purchase_timestamp) BETWEEN 01 AND 08
        )
  WHERE EXTRACT(YEAR FROM order_purchase_timestamp) IN (2017,2018)
  GROUP BY EXTRACT(YEAR FROM order_purchase_timestamp)
)
SELECT ROUND((P2018.total_amount - P2017.total_amount)*100/P2017.total_amount,2) AS Percentage_Increase
FROM ORDERS_PAYMENT P2017 JOIN ORDERS_PAYMENT P2018
ON P2017.year = 2017 AND P2018.year = 2018
```

Result:



Row	Percentage_Incre...
1	136.98

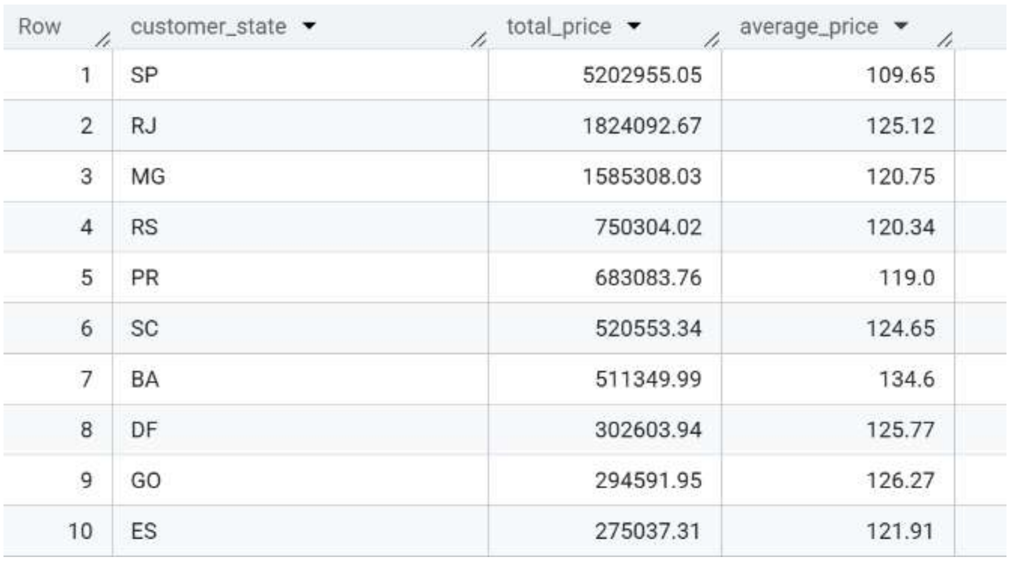
Insights: 1. There is around 137% increase in the order value from 2017 to 2018 for the orders taken between the months January to August. (Including both the months)

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

Query:

```
SELECT  c.customer_state,
        ROUND(SUM(oi.price),2) AS total_price,
        ROUND(AVG(oi.price),2) AS average_price
FROM `Target.order_items` oi
JOIN `Target.orders` o    ON oi.order_id = o.order_id
JOIN `Target.customers` c ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY total_price DESC
```

Result:



Row	customer_state	total_price	average_price
1	SP	5202955.05	109.65
2	RJ	1824092.67	125.12
3	MG	1585308.03	120.75
4	RS	750304.02	120.34
5	PR	683083.76	119.0
6	SC	520553.34	124.65
7	BA	511349.99	134.6
8	DF	302603.94	125.77
9	GO	294591.95	126.27
10	ES	275037.31	121.91

Insights: 1. Sao Paulo – SP has the maximum total order price but has the lowest average order price.

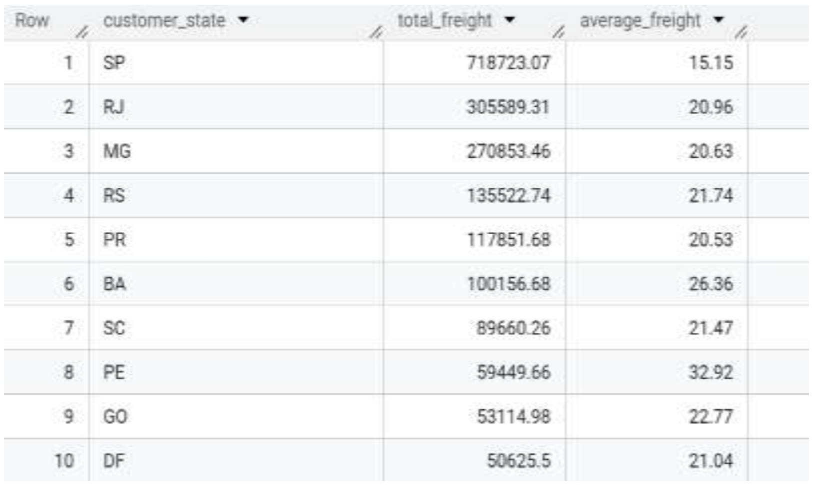
2. Paraiba – PB has the maximum average order price of 191.48. (Can be seen when ordered by average_price desc)

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

Query:

```
SELECT  c.customer_state,
        ROUND(SUM(oi.freight_value),2) AS total_freight,
        ROUND(AVG(oi.freight_value),2) AS average_freight
FROM `Target.order_items` oi
JOIN `Target.orders` o    ON oi.order_id = o.order_id
JOIN `Target.customers` c ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY total_freight desc
```

Result:



Row	customer_state	total_freight	average_freight
1	SP	718723.07	15.15
2	RJ	305589.31	20.96
3	MG	270853.46	20.63
4	RS	135522.74	21.74
5	PR	117851.68	20.53
6	BA	100156.68	26.36
7	SC	89660.26	21.47
8	PE	59449.66	32.92
9	GO	53114.98	22.77
10	DF	50625.5	21.04

Insights: 1. Though Sao Paulo – SP has the highest total freight; it has the lowest average freight.

2. Roraima - RR has the highest average freight and lowest total freight.

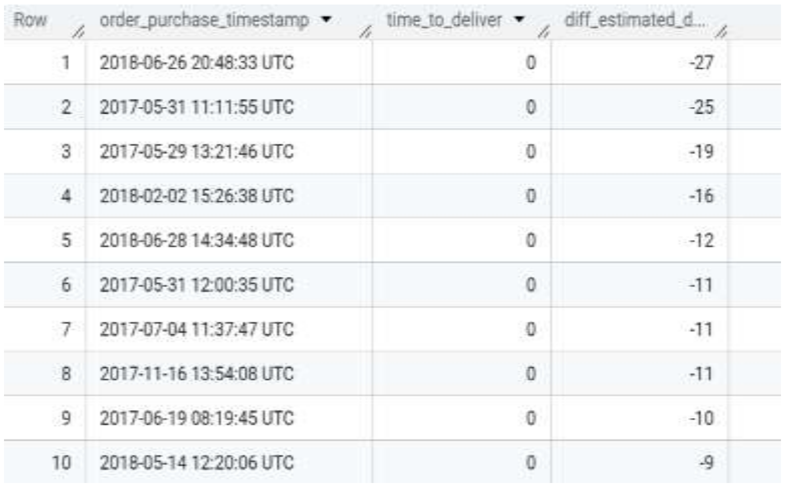
5. Analysis based on sales, freight and delivery time.

1. **The no. of days taken to deliver each order from the order's purchase date as delivery time.**

Query:

```
SELECT  order_purchase_timestamp,
        DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) AS time_to_deliver,
        DATE_DIFF(order_delivered_customer_date,order_estimated_delivery_date, DAY) AS
diff_estimated_delivery
FROM `Target.orders`
WHERE order_delivered_customer_date IS NOT  NULL
ORDER BY time_to_deliver , diff_estimated_delivery
```

Result:



Row	order_purchase_timestamp	time_to_deliver	diff_estimated_d...
1	2018-06-26 20:48:33 UTC	0	-27
2	2017-05-31 11:11:55 UTC	0	-25
3	2017-05-29 13:21:46 UTC	0	-19
4	2018-02-02 15:26:38 UTC	0	-16
5	2018-06-28 14:34:48 UTC	0	-12
6	2017-05-31 12:00:35 UTC	0	-11
7	2017-07-04 11:37:47 UTC	0	-11
8	2017-11-16 13:54:08 UTC	0	-11
9	2017-06-19 08:19:45 UTC	0	-10
10	2018-05-14 12:20:06 UTC	0	-9

Insights: 1. There are six records which show that the order_status is ‘canceled’, but have taken these records also into account as it had delivered customer date.

2. Many orders have been delivered much before the estimated delivery time.

3. However, some orders have exceeded the estimated delivery time.

4. Some orders have been delivered close to 7 months after purchase.

5. Also, few orders have been delivered around 5 - 6 months after the estimated delivery date.

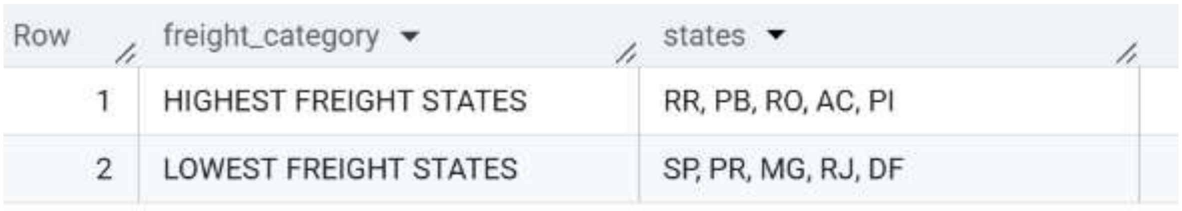
2. The top 5 states with the highest & lowest average freight value.

(Query to show only the top 5 states (comma separated) with the highest and lowest average freight)

Query:

```
WITH AVG_FREIGHT AS
(
        SELECT  c.customer_state,
        ROUND(AVG(oi.freight_value),2) AS average_freight
FROM `Target.order_items` oi
JOIN `Target.orders` o    ON oi.order_id = o.order_id
JOIN `Target.customers` c ON o.customer_id = c.customer_id
GROUP BY c.customer_state
)
SELECT "HIGHEST FREIGHT STATES" AS freight_category,
        STRING_AGG(customer_state,', ') AS states
FROM
(
SELECT *
FROM AVG_FREIGHT
ORDER BY average_freight DESC
LIMIT  5
)
UNION ALL
SELECT "LOWEST FREIGHT STATES" AS freight_category,
        STRING_AGG(customer_state,', ') AS states
FROM
(
SELECT *
FROM AVG_FREIGHT
ORDER BY average_freight
LIMIT  5
)
```

Result:



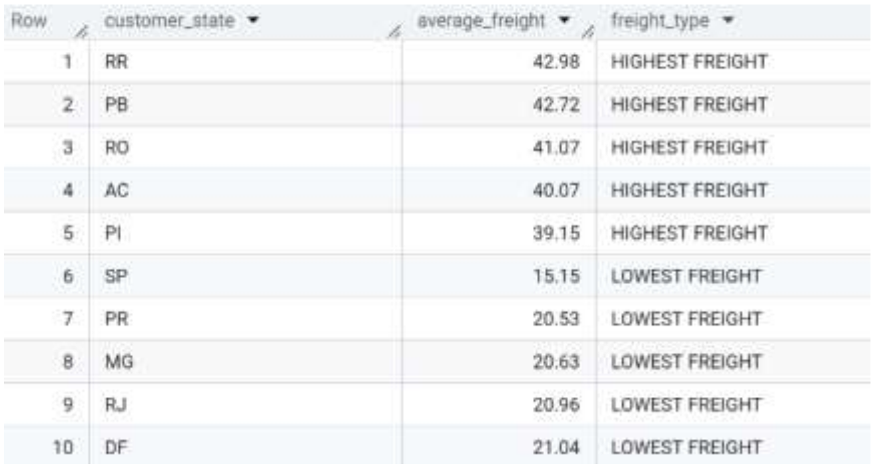
Row	freight_category	states
1	HIGHEST FREIGHT STATES	RR, PB, RO, AC, PI
2	LOWEST FREIGHT STATES	SP, PR, MG, RJ, DF

(Query to show the states and the freight value of the top 5 states with the highest and lowest average freight)

Query:

```
WITH AVG_FREIGHT AS
(
        SELECT  c.customer_state,
        ROUND(AVG(oi.freight_value),2) AS average_freight
FROM `Target.order_items` oi
JOIN `Target.orders` o    ON oi.order_id = o.order_id
JOIN `Target.customers` c ON o.customer_id = c.customer_id
GROUP BY c.customer_state
)
(
SELECT *, "HIGHEST FREIGHT" AS freight_type
FROM AVG_FREIGHT
ORDER BY average_freight DESC
LIMIT 5
)
UNION ALL
(
SELECT *, "LOWEST FREIGHT" AS freight_type
FROM AVG_FREIGHT
ORDER BY average_freight
LIMIT 5
)
```

Result:



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

Insights: **1.** The top five states with the highest average freight are RR, PB, RO, AC and PI

2. The top five states with the lowest average freight are SP, PR, MG, RJ and DF

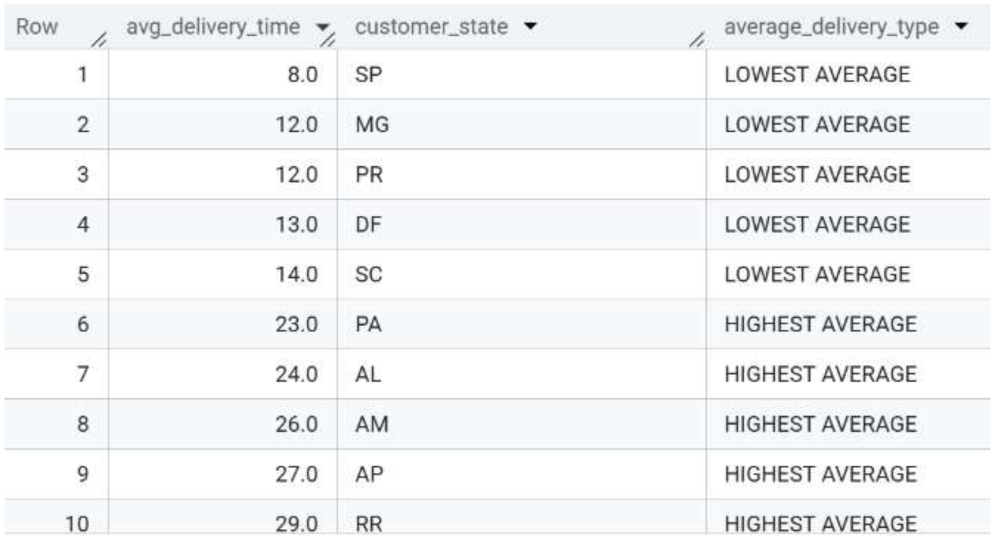
3. The average freight ranges from 15.15 to 42.98.

3. Top 5 states with the highest & lowest average delivery time.

Query:

```
WITH AVG_DELIVERY_TIME AS
(
 SELECT
        ROUND(AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY))) AS
        avg_delivery_time,
        c.customer_state
 FROM `Target.orders` o JOIN `Target.customers` c
 ON o.customer_id = c.customer_id
 WHERE order_delivered_customer_date IS NOT  NULL
 GROUP BY c.customer_state
)
(
SELECT *, "HIGHEST AVERAGE" AS average_delivery_type
FROM AVG_DELIVERY_TIME
ORDER BY avg_delivery_time DESC
LIMIT 5
)
UNION ALL
(
SELECT *, "LOWEST AVERAGE" AS average_delivery_type
FROM AVG_DELIVERY_TIME
ORDER BY avg_delivery_time
LIMIT 5
)
ORDER BY avg_delivery_time
```

Result:



Row	avg_delivery_time	customer_state	average_delivery_type
1	8.0	SP	LOWEST AVERAGE
2	12.0	MG	LOWEST AVERAGE
3	12.0	PR	LOWEST AVERAGE
4	13.0	DF	LOWEST AVERAGE
5	14.0	SC	LOWEST AVERAGE
6	23.0	PA	HIGHEST AVERAGE
7	24.0	AL	HIGHEST AVERAGE
8	26.0	AM	HIGHEST AVERAGE
9	27.0	AP	HIGHEST AVERAGE
10	29.0	RR	HIGHEST AVERAGE

Insights: **1.** The states SP, PR, MG, DF and SC have the lowest delivery average.

2. While the states PA, AL, AM, AP and RR have the highest delivery average.

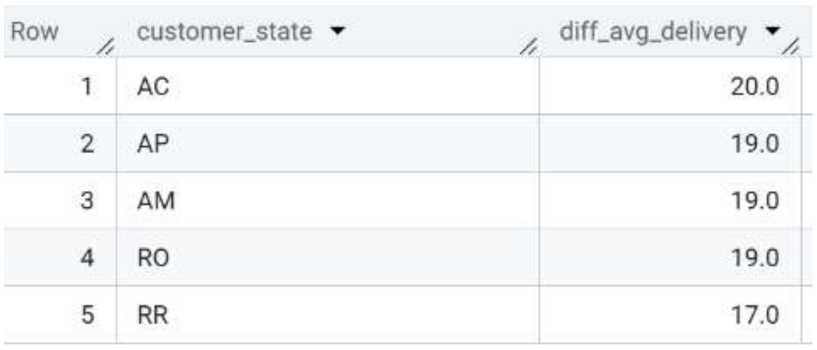
3. The delivery average ranges from 8 to 29 days.

4. Top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

Query:

```
SELECT  customer_state,
        ROUND((estimated_delivery_days - actual_delivery_days)) AS diff_avg_delivery
FROM (
      SELECT  c.customer_state,
              AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY)) AS
              actual_delivery_days,
              AVG(DATE_DIFF(o.order_estimated_delivery_date,o.order_purchase_timestamp, DAY)) AS
              estimated_delivery_days
      FROM `Target.orders` o JOIN `Target.customers` c
      ON o.customer_id = c.customer_id
      WHERE order_delivered_customer_date IS NOT  NULL
      GROUP BY  c.customer_state
      )
ORDER BY diff_avg_delivery DESC
LIMIT 5
```

Result:



Row	customer_state	diff_avg_delivery
1	AC	20.0
2	AP	19.0
3	AM	19.0
4	RO	19.0
5	RR	17.0

Insights: 1. Acre – AC has the fastest delivery with respect to its estimated delivery with 20 days before its estimated delivery time.

2. The top five states with fastest delivery when comapred to their estimated delivery are AC, AP, AM, RO and RR.

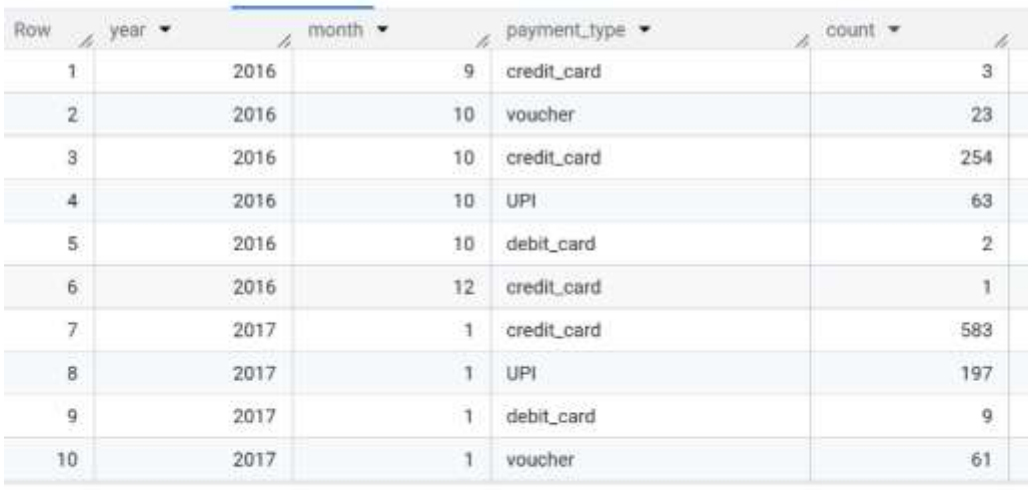
6. Analysis based on the payments:

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

Query:

```
SELECT  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
        EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
        p.payment_type,
        COUNT(*) AS count
FROM `Target.orders` o JOIN  `Target.payments` p ON o.order_id = p.order_id
GROUP BY EXTRACT(YEAR FROM o.order_purchase_timestamp), EXTRACT(MONTH FROM
o.order_purchase_timestamp), p.payment_type
ORDER BY year, month
```

Result:



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

Insights: 1. The different payment modes available are credit card, voucher, UPI, debit card etc.

2. If ordered by the count in descending order, we can see that the payment mode most used is credit card followed by UPI.

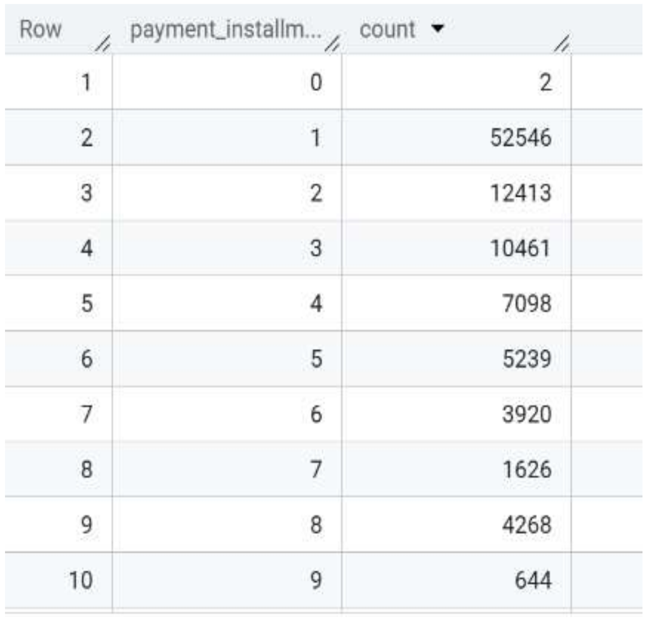
3. There are also instances when the payment type is not defined. We can see this when we order by the count in ascending order.

2. Orders placed on the basis of the payment installments that have been paid.

Query:

```
SELECT payment_installments, COUNT(*) AS count
FROM `Target.payments`
GROUP BY payment_installments
```

Result:



Row	payment_installm...	count
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626
9	8	4268
10	9	644

Insights: 1. Customers have taken from 0 up to 24 installments

2. The most popular installment options are 1,2,3,4 and 10 installments.

3. Very few customers have taken 0, 22 and 23 installments.