Business Case: Target SQL

This business case is having data which contains the information of 99441 customers and orders made from 2016 to 2018 in Brazil. With the help of this data the trends in customers and market can be observed and the customer service can be rated.

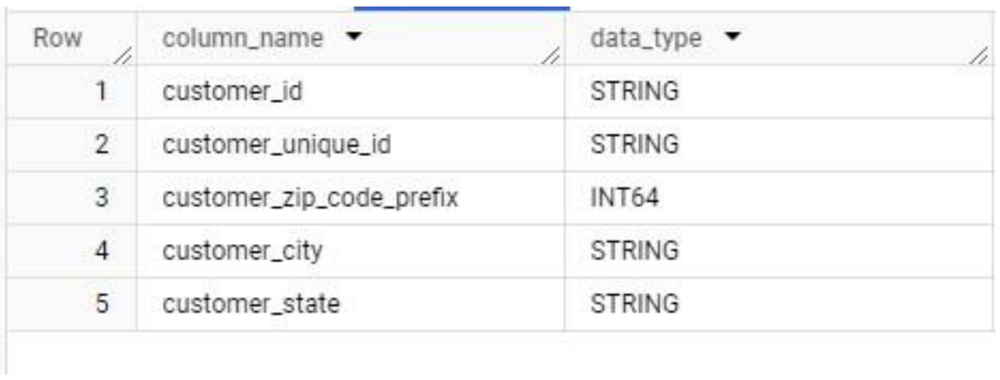
Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:

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

Query:

```
SELECT
column_name, data_type
FROM 'target-sql-400618.target_brazil.INFORMATION_SCHEMA.COLUMNS'
WHERE table_name = 'customers'
```

Output:



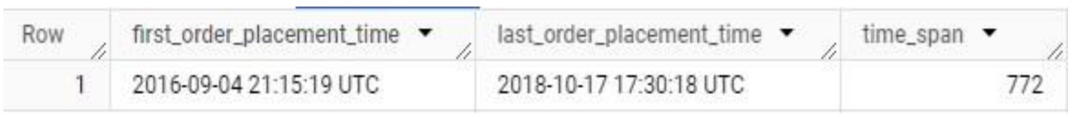
Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

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

Query:

```
SELECT
MIN(order_purchase_timestamp) AS first_order_placement_time,
MAX(order_purchase_timestamp) AS last_order_placement_time,
DATE_DIFF(MAX(order_purchase_timestamp),
MIN(order_purchase_timestamp), DAY) AS time_span
FROM 'target_brazil.orders'
```

Output:



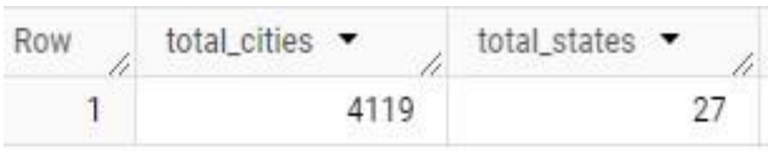
Row	first_order_placement_time	last_order_placement_time	time_span
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC	772

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

Query:

```
SELECT
COUNT(DISTINCT customer_city) AS total_cities,
COUNT(DISTINCT customer_state) AS total_states
FROM 'target_brazil.customers'
```

Output:



Row	total_cities	total_states
1	4119	27

Analysis:

The business has expanded to total 4119 cities of 27 states of Brazil in around 25 months of its operations in Brazil.

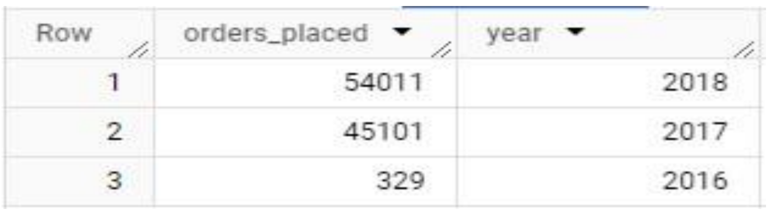
In-depth Exploration:

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

Query:

```
SELECT
COUNT(order_id) AS orders_placed,
EXTRACT(YEAR FROM order_purchase_timestamp) AS year
FROM 'target_brazil.orders'
GROUP BY year
ORDER BY orders_placed DESC
```

Output:



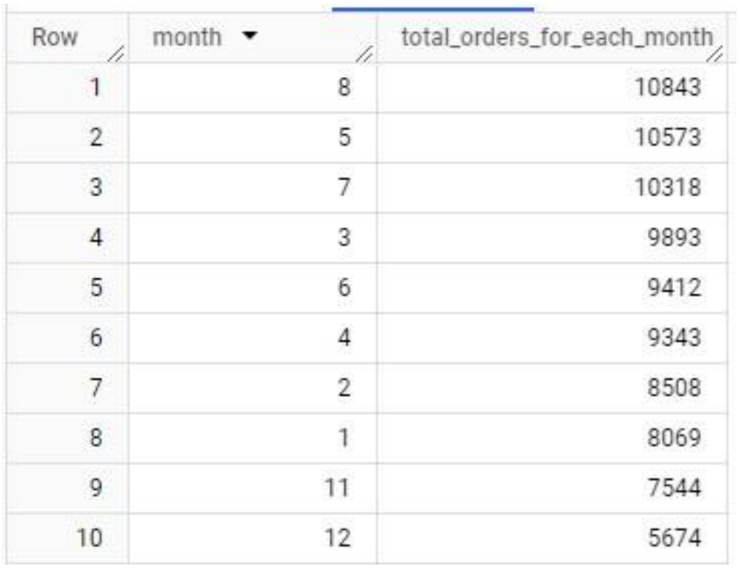
Row	orders_placed	year
1	54011	2018
2	45101	2017
3	329	2016

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

Query:

```
SELECT
COUNT(order_id) AS orders_placed,
EXTRACT(YEAR FROM order_purchase_timestamp) AS year
FROM 'target_brazil.orders'
GROUP BY year
ORDER BY orders_place DESC
```

Output:



Row	month	total_orders_for_each_month
1	8	10843
2	5	10573
3	7	10318
4	3	9893
5	6	9412
6	4	9343
7	2	8508
8	1	8069
9	11	7544
10	12	5674

3. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

Query:

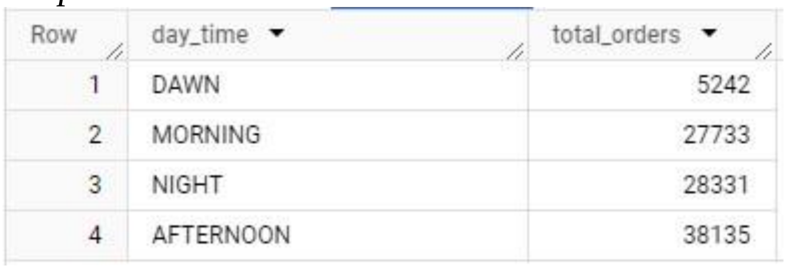
```
SELECT

CASE

WHEN hour BETWEEN 0 AND 6

THEN "DAWN"
WHEN hour BETWEEN 7 AND 12
THEN "MORNING"
WHEN hour BETWEEN 13 AND 18
THEN "AFTERNOON"
WHEN hour BETWEEN 19 AND 23
THEN "NIGHT"
END AS day_time,
COUNT (*) AS total_orders
FROM(
SELECT
EXTRACT (HOUR FROM order_purchase_timestamp) AS hour
FROM `target_brazil.orders`) AS hr_tble
GROUP BY day_time
ORDER BY total_orders
```

Output:



Row	day_time	total_orders
1	DAWN	5242
2	MORNING	27733
3	NIGHT	28331
4	AFTERNOON	38135

Analysis:

The data shows that the Brazilians are interested in buying from this commercial venture as the sales have increased from 329 in 2016 to 54011 in 2018. The number of orders placed are highest in the month of August and lowest in the month of December and highest in the afternoon and lowest in the dawn. The company should take measures to handle the traffic during peak hours and offer timely discounts in the trough time.

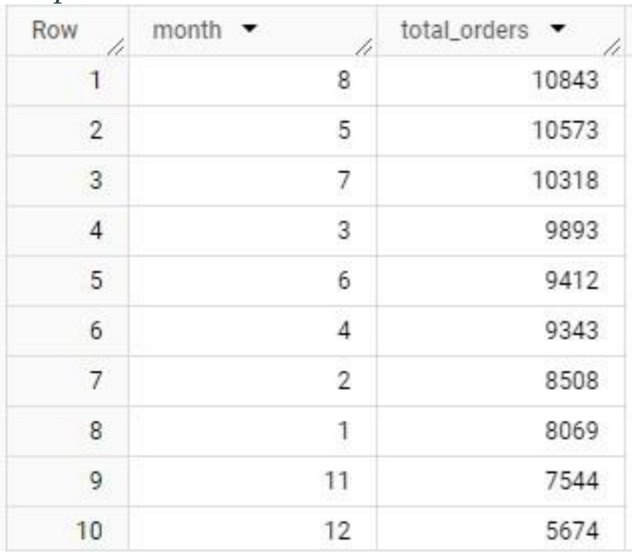
Evolution of E-commerce orders in the Brazil region:

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

Query:

```
SELECT
EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
COUNT(*) AS total_orders
FROM `target_brazil.orders`
GROUP BY month
ORDER BY total_orders DESC
```

Output:



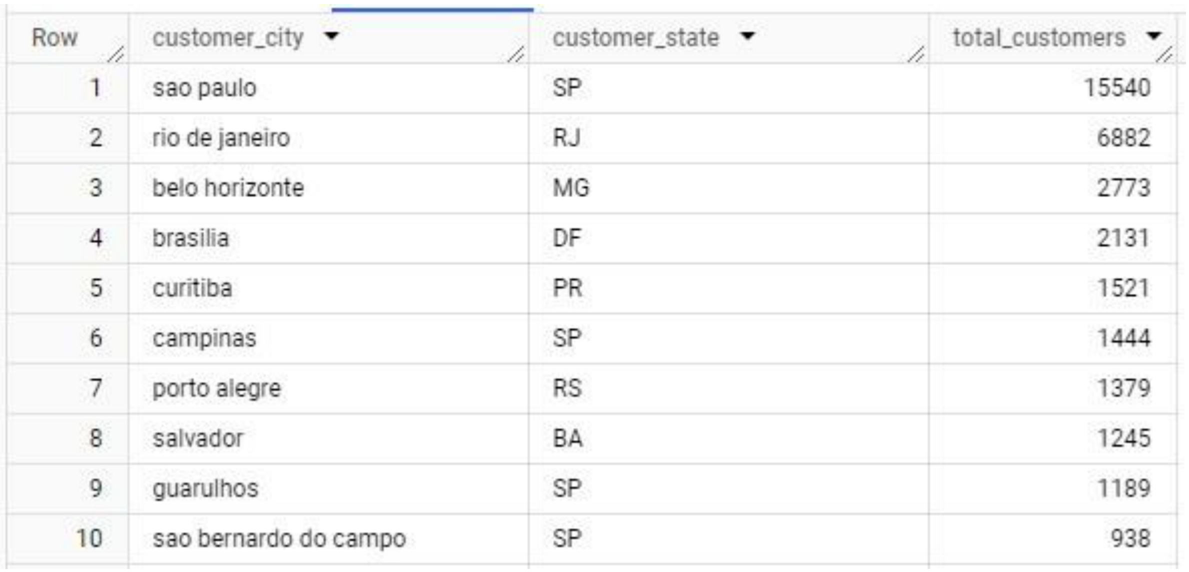
Row	month	total_orders
1	8	10843
2	5	10573
3	7	10318
4	3	9893
5	6	9412
6	4	9343
7	2	8508
8	1	8069
9	11	7544
10	12	5674

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

Query:

```
SELECT
customer_city,
customer_state,
COUNT(*) AS total_customers
FROM `target_brazil.customers`
GROUP BY customer_city, customer_state
ORDER BY total_customers DESC
```

Output:



Row	customer_city	customer_state	total_customers
1	sao paulo	SP	15540
2	rio de janeiro	RJ	6882
3	belo horizonte	MG	2773
4	brasilia	DF	2131
5	curitiba	PR	1521
6	campinas	SP	1444
7	porto alegre	RS	1379
8	salvador	BA	1245
9	guarulhos	SP	1189
10	sao bernardo do campo	SP	938

Analysis:
The data shows the city named as ‘sao Paulo of state marked as ‘SP’ has the highest number of customers and the city named as’ sao bernardo do campo’ of the same state has lowest number of customers. The business should improve it customer service so that it can attract more customers from other cities as well.

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).

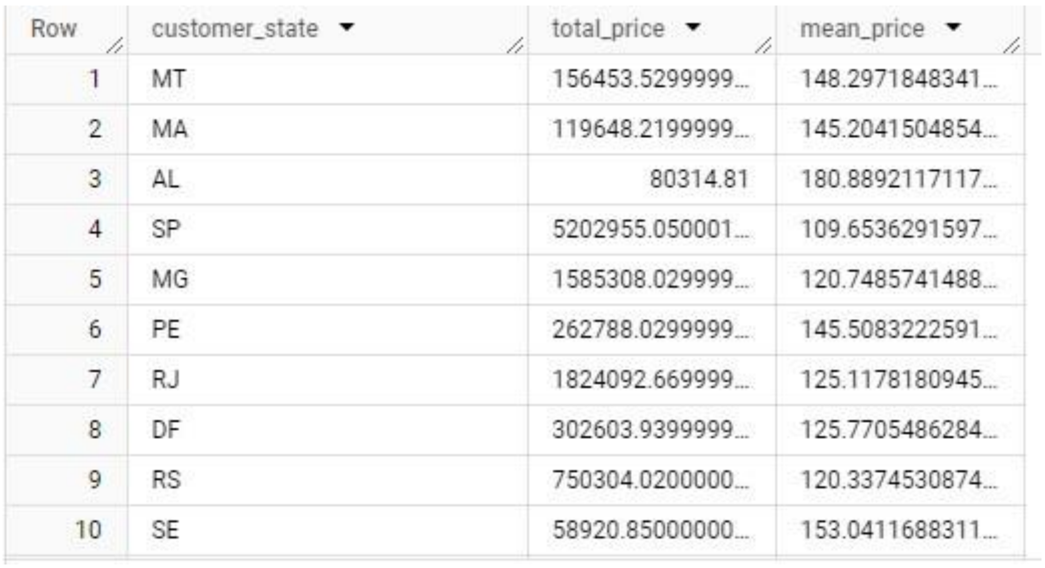
2. Calculate the Total & Average value of order price for each state.

Query:

```
SELECT
c.customer_state,
SUM(oi.price) AS total_price,
AVG(oi.price) AS mean_price
```

```
FROM 'target_brazil.customers' AS c
INNER JOIN 'target_brazil.orders' AS od
ON c.customer_id = oi.customer_id
INNER JOIN 'target_brazil.order_items' AS oi
ON od.order_id = oi.order_id
GROUP BY c.customer_state
```

Output:



Row	customer_state	total_price	mean_price
1	MT	156453.5299999...	148.2971848341...
2	MA	119648.2199999...	145.2041504854...
3	AL	80314.81	180.8892117117...
4	SP	5202955.050001...	109.6536291597...
5	MG	1585308.029999...	120.7485741488...
6	PE	262788.0299999...	145.5083222591...
7	RJ	1824092.669999...	125.1178180945...
8	DF	302603.9399999...	125.7705486284...
9	RS	750304.0200000...	120.3374530874...
10	SE	58920.85000000...	153.0411688311...

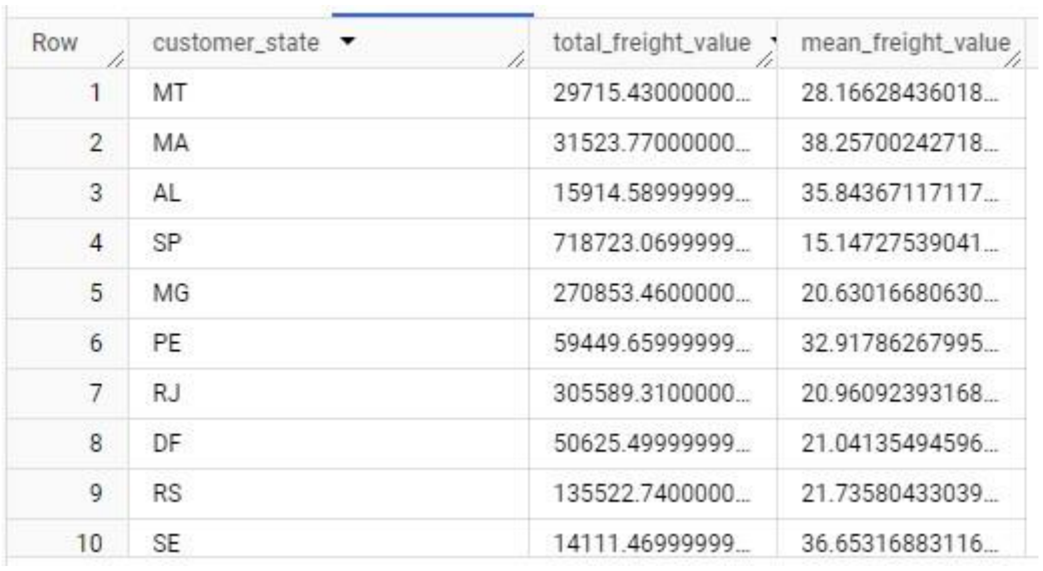
3. Calculate the Total & Average value of order freight for each state

```
SELECT

c.customer_state,

SUM(oi.freight_value) AS total_freight_value,
AVG(oi. freight_value) AS mean.freight_value
FROM 'target_brazil.customers' AS c
INNER JOIN 'target_brazil.orders' AS od
ON c.customer_id = oi.customer_id
INNER JOIN 'target_brazil.order_items' AS oi
ON od.order_id = oi.order_id
GROUP BY c.customer_state
```

Output:



Row	customer_state	total_freight_value	mean_freight_value
1	MT	29715.43000000...	28.16628436018...
2	MA	31523.77000000...	38.25700242718...
3	AL	15914.58999999...	35.84367117117...
4	SP	718723.0699999...	15.14727539041...
5	MG	270853.4600000...	20.63016680630...
6	PE	59449.65999999...	32.91786267995...
7	RJ	305589.3100000...	20.96092393168...
8	DF	50625.49999999...	21.04135494596...
9	RS	135522.7400000...	21.73580433039...
10	SE	14111.46999999...	36.65316883116...

Analysis:

This data shows the total freight value which is mentioned in the column 'total_freight_value' and the average freight value which is mentioned in the column 'mean_freight_value for each customer state.

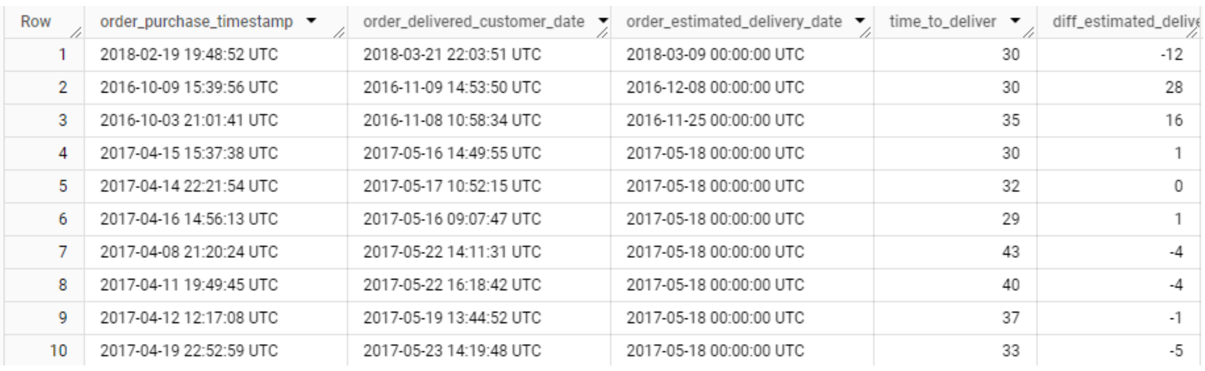
Analysis based on sales, freight and delivery time.

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.

Query:

```
SELECT
order_purchase_timestamp,
order_delivered_customer_date,
order_estimated_delivery_date,
DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,
DAY) AS time_to_deliver,
DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_dat
e, DAY) AS diff_estimated_delivery
FROM 'target_brazil_orders'
```

Output:



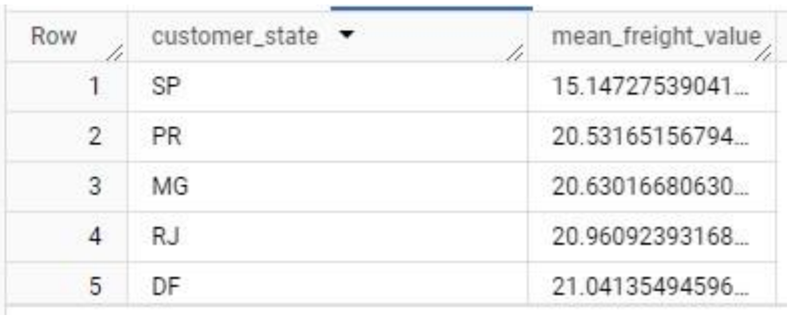
Row	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_delive
1	2018-02-19 19:48:52 UTC	2018-03-21 22:03:51 UTC	2018-03-09 00:00:00 UTC	30	-12
2	2016-10-09 15:39:56 UTC	2016-11-09 14:53:50 UTC	2016-12-08 00:00:00 UTC	30	28
3	2016-10-03 21:01:41 UTC	2016-11-08 10:58:34 UTC	2016-11-25 00:00:00 UTC	35	16
4	2017-04-15 15:37:38 UTC	2017-05-16 14:49:55 UTC	2017-05-18 00:00:00 UTC	30	1
5	2017-04-14 22:21:54 UTC	2017-05-17 10:52:15 UTC	2017-05-18 00:00:00 UTC	32	0
6	2017-04-16 14:56:13 UTC	2017-05-16 09:07:47 UTC	2017-05-18 00:00:00 UTC	29	1
7	2017-04-08 21:20:24 UTC	2017-05-22 14:11:31 UTC	2017-05-18 00:00:00 UTC	43	-4
8	2017-04-11 19:49:45 UTC	2017-05-22 16:18:42 UTC	2017-05-18 00:00:00 UTC	40	-4
9	2017-04-12 12:17:08 UTC	2017-05-19 13:44:52 UTC	2017-05-18 00:00:00 UTC	37	-1
10	2017-04-19 22:52:59 UTC	2017-05-23 14:19:48 UTC	2017-05-18 00:00:00 UTC	33	-5

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

Query: For Highest

```
SELECT
 c.customer_state,
AVG(oi.freight_value) AS mean_freight_value
FROM 'target_brazil.customers' AS c
INNER JOIN 'target_brazil.orders' AS od
ON c.customer_id = od.customer_id
INNER JOIN 'target_brazil.order_items' AS oi
ON od.order_id = oi.order_id
GROUP BY c.customer_state
ORDER BY mean_freight_value DESC
LIMIT 5
```

Output:



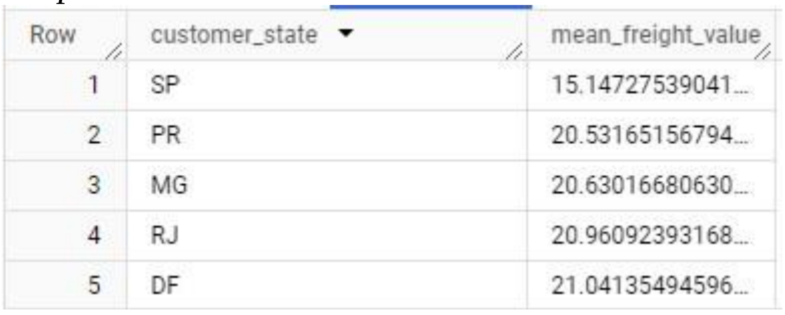
Row	customer_state	mean_freight_value
1	SP	15.14727539041...
2	PR	20.53165156794...
3	MG	20.63016680630...
4	RJ	20.96092393168...
5	DF	21.04135494596...

Query: For Lowest

```
SELECT
 c.customer_state,
AVG(oi.freight_value) AS mean_freight_value
FROM 'target_brazil.customers' AS c
INNER JOIN 'target_brazil.orders' AS od
ON c.customer_id = od.customer_id
INNER JOIN 'target_brazil.order_items' AS oi
ON od.order_id = oi.order_id
GROUP BY c.customer_state
```

```
ORDER BY mean_freight_value
LIMIT 5
```

Output:



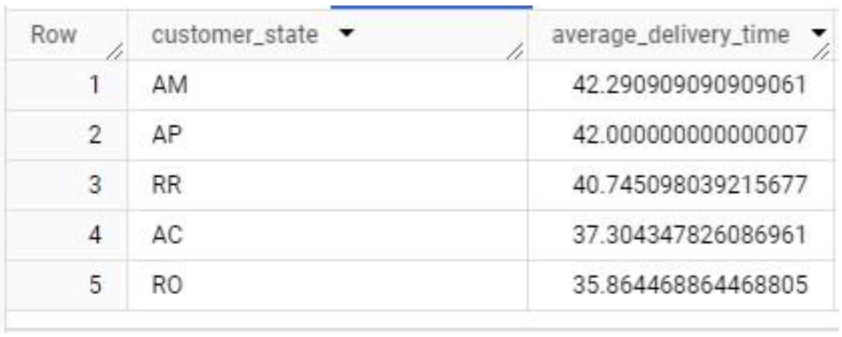
Row	customer_state	mean_freight_value
1	SP	15.14727539041...
2	PR	20.53165156794...
3	MG	20.63016680630...
4	RJ	20.96092393168...
5	DF	21.04135494596...

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

Query: For Highest

```
SELECT
c.customer_state,
AVG(DATE_DIFF(od.order_estimated_delivery_date,
od.order_delivered_carrier_date, DAY)) AS average_delivery_time
FROM `target_brazil.customers` AS c
INNER JOIN `target_brazil.orders` AS od
ON c.customer_id = od.customer_id
INNER JOIN `target_brazil.order_items` AS oi
ON od.order_id = oi.order_id
GROUP BY c.customer_state
ORDER BY average_delivery_time DESC
LIMIT 5
```

Output:

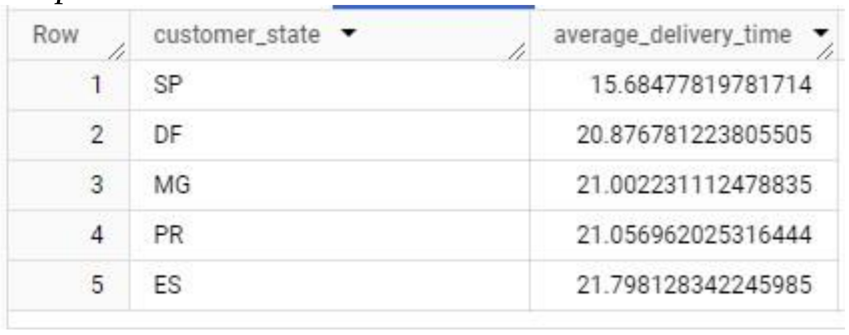


Row	customer_state	average_delivery_time
1	AM	42.290909090909061
2	AP	42.000000000000007
3	RR	40.745098039215677
4	AC	37.304347826086961
5	RO	35.864468864468805

Query: For Lowest

```
SELECT
 c.customer_state,
AVG(DATE_DIFF(od.order_estimated_delivery_date,
od.order_delivered_carrier_date, DAY)) AS average_delivery_time
FROM `target_brazil.customers` AS c
INNER JOIN `target_brazil.orders` AS od
ON c.customer_id = od.customer_id
INNER JOIN `target_brazil.order_items` AS oi
ON od.order_id = oi.order_id
GROUP BY c.customer_state
ORDER BY average_delivery_time ASC
LIMIT 5
```

Output:



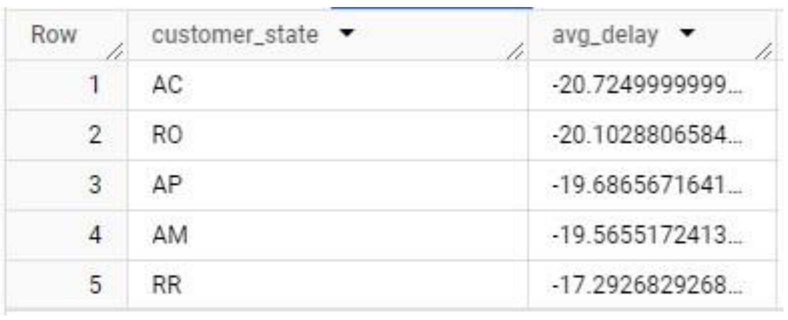
Row	customer_state	average_delivery_time
1	SP	15.68477819781714
2	DF	20.876781223805505
3	MG	21.002231112478835
4	PR	21.056962025316444
5	ES	21.798128342245985

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

Query:

```
SELECT
 customer_state,
AVG(DATE_DIFF(DATE(order_delivered_customer_date),
DATE(order_estimated_delivery_date), DAY)) AS avg_delay
FROM `target_brazil.customers` AS c
INNER JOIN `target_brazil.orders` AS od
ON c.customer_id = od.customer_id
WHERE DATE_DIFF(DATE(order_delivered_customer_date),
DATE(order_estimated_delivery_date), DAY) IS NOT NULL
GROUP BY customer_state
ORDER BY avg_delay ASC
LIMIT 5
```

Output:



Row	customer_state	avg_delay
1	AC	-20.7249999999...
2	RO	-20.1028806584...
3	AP	-19.6865671641...
4	AM	-19.5655172413...
5	RR	-17.2926829268...

Analysis:

According to the data there is more than 25 days of difference In average time to deliver the order to the customer and some cities are getting order 20 days prior to the estimated date. The company should pay more attention to the delayed deliveries to retain the trust of customers in the company

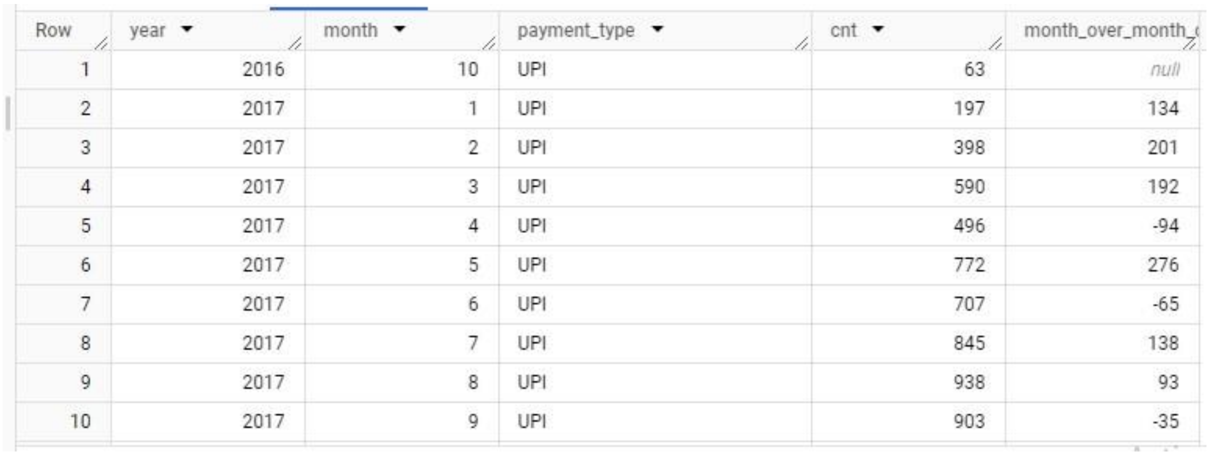
Analysis based on the payments:

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

Query:

```
SELECT
year,
month,
payment_type,
cnt,
cnt-LAG(cnt) OVER (PARTITION BY payment_type ORDER BY year, month)
 AS month_over_month_count
 FROM
 (SELECT
   EXTRACT(YEAR FROM order_purchase_timestamp)AS year,
   EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
   payment_type,
   COUNT(o.order_id) AS cnt
   FROM `target_brazil.orders` AS o
   INNER JOIN `target_brazil.payments` AS p
   ON o.order_id = p.order_id
   GROUP BY year, month, payment_type) AS tbl
   ORDER BY payment_type, year, month
```

Output:



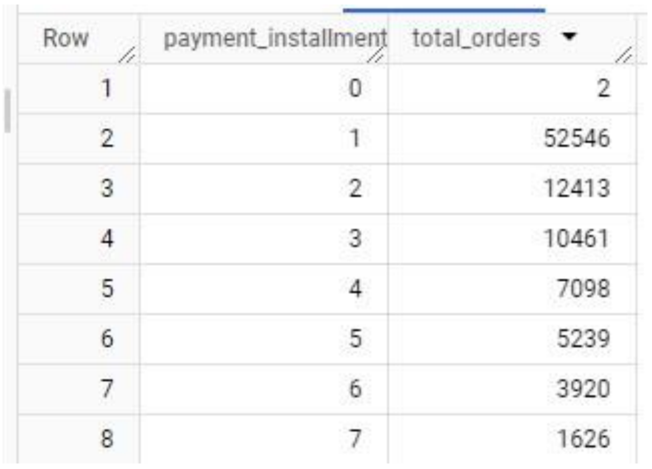
Row	year	month	payment_type	cnt	month_over_month_(
1	2016	10	UPI	63	null
2	2017	1	UPI	197	134
3	2017	2	UPI	398	201
4	2017	3	UPI	590	192
5	2017	4	UPI	496	-94
6	2017	5	UPI	772	276
7	2017	6	UPI	707	-65
8	2017	7	UPI	845	138
9	2017	8	UPI	938	93
10	2017	9	UPI	903	-35

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

Query:

```
SELECT
payment_installments,
COUNT(order_id) AS total_orders
FROM `target_brazil.payments`
GROUP BY payment_installments
ORDER BY payment_installments, total_orders DESC
```

Output:



Row	payment_installment	total_orders
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626

Analysis:

Majority of customers prefer to pay in one installment. Very less number of customers prefer to pay in higher instalments i.e in more than 10 installments. UPI is the payment option preferred by most of the customers.

Recommendations:

- More advertisements in states having less customers.
- Limitation of delivery delays.
- Providing offers in trough season.
- Enable fast delivery option.