Business Case: Target SQL

Context

This business case has information of 100k orders from 2016 to 2018 made at Target in Brazil. Its features allows viewing an order from multiple dimensions: from order status, price, payment and freight performance to customer location, product attributes and finally reviews written by customers.

Q1: Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset

A. Data type of columns in a table

For customers Table

Query:

```
SELECT column_name, data_type, is_nullable
FROM `target-sql-386318.target_Brazil.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name ='customers';
```

Output:

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

Analysis:

Here we can see that in customers table we have five columns i.e., customer_id,customer_unique_id,customer_zip_code_prefix,customer_city,customer_state of **Data types: string,string,integer,string,string respectively.**

For order_items Table

Query:

```
SELECT column_name, data_type, is_nullable
FROM `target-sql-386318.target_Brazil.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name ='order_items';
```

Output:

Row	column_name	data_type	is_nullable
1	order_id	STRING	YES
2	order_item_id	INT64	YES
3	product_id	STRING	YES
4	seller_id	STRING	YES
5	shipping_limit_date	TIMESTAMP	YES
6	price	FLOAT64	YES
7	freight_value	FLOAT64	YES

Analysis:

In order_items table- we have 7 columns i.e., order_id, order_item_id, product_id, seller_id,Shipping_limit_date, price, freight_value of **data types: string, integer, string, string, timestamp, float,float respectively.**

Similarly, we can find the data type of columns of remaining 6 tables.

B. Time period for which the data is given

Query:

```
select min(date(order_purchase_timestamp)) as first,
max(date(order_delivered_customer_date)) as last,
date_diff(max(date(order_delivered_customer_date)),min(date(order_purc
hase_timestamp)), day) as total_days
from `target_Brazil.orders`
```

Output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTIO

Row	first	last	total_days
1	2016-09-04	2018-10-17	773

Time-period for which the data is given **is 773 days i.e., from "2016-09-04" to "2018-10-17".**

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

Query:

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

OUTPUT:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GR

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

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?

QUERY 1:

```
select count (order_id) as total_orders,
extract(year from order_purchase_timestamp) as year
from `target_Brazil.orders`
group by year
order by total_orders desc;
```

Output:

Query results

JOB INFORMATION | RESULTS | JS

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

Analysis:

On analysing above result (table), we can say there is a tremendous rise in number of people who buy from e-commerce Brazil I.e., from **329 people in 2016 to 54011 people in 2018.**

Query 2:

```
SELECT COUNT(order_id) AS total_orders_placed,
EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
EXTRACT(MONTH FROM order_purchase_timestamp) AS month
FROM `target-sql-386318.target_Brazil.orders`
GROUP BY year, month
ORDER BY year,month,total_orders_placed desc;
```

Output:

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

Analysis:

The data reveals a clear seasonality pattern in the count of orders placed over time. There is a peak in order placements during December and January, followed by a decline in February to April. The months of May and June show a slight increase, while July to October indicate relatively stable order placement. However, September and October experience a significant drop in orders.

Recommendations:

1. Holiday Promotions: Capitalize on the high order placements in December by offering attractive holiday promotions and discounts.
2. New Year Campaigns: Launch marketing campaigns in January to leverage the increased order placements at the beginning of the year.
3. Spring Sales: Plan targeted marketing campaigns and seasonal sales events to revive order placements from February to April.
4. Summer Specials: Introduce summer-themed promotions in May and June to take advantage of the slight increase in orders.

5. Revive September and October Sales: Investigate the drop in orders during these months and launch targeted marketing campaigns and promotions to boost sales.

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

Query:

```
SELECT
  CASE
    WHEN hour BETWEEN 0 AND 6 THEN 'DAWN'
    WHEN hour BETWEEN 6 AND 12 THEN 'MORNING'
    WHEN hour BETWEEN 12 AND 18 THEN 'AFTERNOON'
    ELSE 'NIGHT'
  END AS time_period,
  COUNT(*) AS cnt
FROM (
  SELECT
    EXTRACT(HOUR FROM order_purchase_timestamp) AS hour
  FROM `target_Brazil.orders`
) AS subquery
GROUP BY time_period
ORDER BY cnt DESC
```

Output:

Row	time_period	cnt
1	AFTERNOON	38135
2	NIGHT	28331
3	MORNING	27733
4	DAWN	5242

Analysis:

Most of the Brazilians tend to buy in the Afternoon i.e., from ’12:00’ to ’18:00’ closely followed by Night and then in the Morning and the least no. of orders were placed in Dawn.

Recommendations:

1. Target Afternoon and Evening: Focus marketing efforts and promotions during these times, when most Brazilians tend to buy.
2. Enhance Customer Support: Provide efficient support during peak buying hours to enhance the shopping experience.
3. Optimize Inventory Management: Analyze sales data to align inventory levels with demand patterns throughout the day.

4. Tailor Marketing Strategies: Develop targeted campaigns highlighting benefits of buying during specific time segments.
5. Explore Delivery Options: Offer flexible delivery choices like same-day or evening delivery to accommodate customer preferences.

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

1. Get month on month orders by states

Query:

```
WITH monthly_orders AS (
  SELECT
    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS Year,
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS Month,
    p.customer_state,
    COUNT(o.order_id) AS no_of_orders
  FROM
    `target_Brazil.orders` o
  JOIN
    `target_Brazil.customers` p ON o.customer_id = p.customer_id
  GROUP BY
    Year, Month, p.customer_state
),
ordered_orders AS (
  SELECT
    Year,
    Month,
    customer_state,
    no_of_orders,
    LAG(no_of_orders) OVER (PARTITION BY customer_state ORDER BY Year, Month) AS
previous_month_orders
  FROM
    monthly_orders
)
SELECT
  Year,
  Month,
  customer_state,
  no_of_orders,
  no_of_orders - previous_month_orders AS order_difference
FROM
  ordered_orders
ORDER BY
  Year, Month, customer_state;
```

Output:

Row	Year	Month	customer_state	no_of_orders	order_difference
1	2016	9	RR	1	null
2	2016	9	RS	1	null
3	2016	9	SP	2	null
4	2016	10	AL	2	null
5	2016	10	BA	4	null
6	2016	10	CE	8	null
7	2016	10	DF	6	null
8	2016	10	ES	4	null
9	2016	10	GO	9	null
10	2016	10	MA	4	null

2. Distribution of customers across the states in Brazil

Query:

```
select customer_state, count(customer_id) as no_of_customers
from `target_Brazil.customers`
group by customer_state order by no_of_customers desc
```

Output:

Row	customer_state	no_of_customer
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

Analysis:

We can see most numbers of customers are from state ‘SP’ with 41746 customers followed by state ‘RJ’ then state ‘MG’ and the least number of customers are from state ‘RR’ with 46 customers.

Recommendations:

- Focus on State 'SP': With 41,746 customers, prioritize resources and marketing efforts in this state.
- Target State 'RJ' and 'MG': Explore growth opportunities and implement targeted campaigns.

- Address State 'RR': With only 46 customers, identify barriers and devise strategies for customer growth.

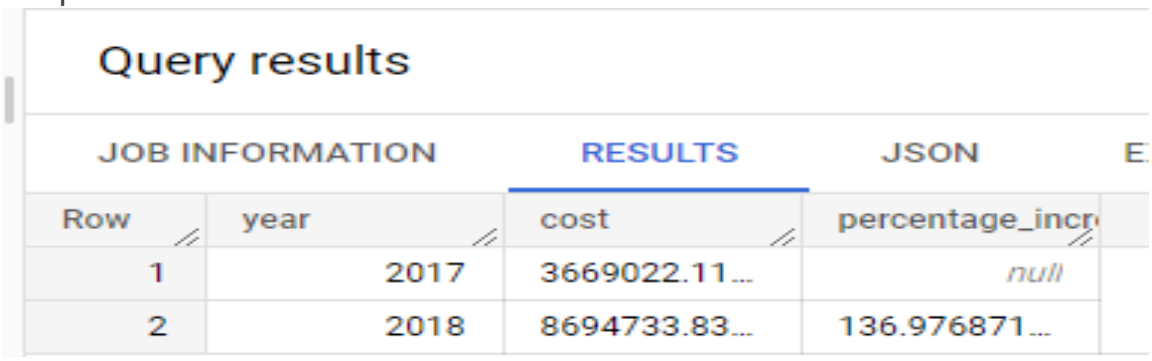
4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

1. Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use “payment_value” column in payments table

Query:

```
WITH yearly_costs AS (
  SELECT
    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
    SUM(p.payment_value) AS cost
  FROM
    `target_Brazil.orders` o
  INNER JOIN
    `target_Brazil.payments` p USING (order_id)
  WHERE
    EXTRACT(YEAR FROM o.order_purchase_timestamp) BETWEEN 2017 AND 2018
    AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
  GROUP BY
    EXTRACT(YEAR FROM o.order_purchase_timestamp)
)
SELECT
  year,
  cost,
  (cost - LAG(cost) OVER (ORDER BY year)) / LAG(cost) OVER (ORDER BY year) * 100 AS
percentage_increase
FROM
  yearly_costs;
```

Output:



Analysis:
We can see that in year 2018 there is 1.37 times increase in cost of orders compared to year 2017.

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

Query:

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

```
  SUM(oi.price) AS total_price,
  AVG(oi.freight_value) AS mean_fv,
  SUM(oi.freight_value) AS total_fv
FROM
  `target_Brazil.orders` o
INNER JOIN
  `target_Brazil.customers` c USING (customer_id) inner join `target_Brazil.order_items` oi
using(order_id)
GROUP BY
  c.customer_state;
```

Output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_state	mean_price	total_price	mean_fv	total_fv
1	MT	148.297184...	156453.529...	28.1662843...	29715.4300...
2	MA	145.204150...	119648.219...	38.2570024...	31523.7700...
3	AL	180.889211...	80314.81	35.8436711...	15914.5899...
4	SP	109.653629...	5202955.05...	15.1472753...	718723.069...
5	MG	120.748574...	1585308.02...	20.6301668...	270853.460...
6	PE	145.508322...	262788.029...	32.9178626...	59449.6599...
7	RJ	125.117818...	1824092.66...	20.9609239...	305589.310...
8	DF	125.770548...	302603.939...	21.0413549...	50625.4999...
9	RS	120.337453...	750304.020...	21.7358043...	135522.740...
10	SE	153.041168...	58920.8500...	36.6531688...	14111.4699...

Q5: Analysis on sales, freight and delivery time

1. Calculate days between purchasing, delivering and estimated delivery

Query:

```
SELECT
  order_id,
  DATE_DIFF(date(order_estimated_delivery_date), date(order_purchase_timestamp), DAY)
AS estimated_days_to_deliver,
  DATE_DIFF(date(order_delivered_customer_date), date(order_purchase_timestamp),
DAY) AS actual_days_to_deliver,
  DATE_DIFF(date(order_delivered_customer_date), date(order_estimated_delivery_date),
DAY) AS delay_or_not_by_days
FROM
  `target_Brazil.orders`
  Where
 DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) IS NOT
NULL and
 DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date, DAY) IS NOT
NULL
```

Output:

Row	order_id	estimated_days	actual_days_to_	delay_or_not_by
1	1950d777989f6a877539f5379...	18	30	12
2	2c45c33d2f9cb8ff8b1c86cc28...	60	31	-29
3	65d1e226dfaeb8cdc42f66542...	53	36	-17
4	635c894d068ac37e6e03dc54e...	33	31	-2
5	3b97562c3aee8bdedcb5c2e45...	34	33	-1
6	68f47f50f04c4cb6774570cfde...	32	30	-2
7	276e9ec344d3bf029ff83a161c...	40	44	4
8	54e1a3c2b97fb0809da548a59...	37	41	4

Analysis:

We can group above table by **delay_or_not_by_days** column and check how many orders get delivered on ---the exact promised delivery date or get delivered before the promised delivery date or after promised delivery date.

Query:

```
SELECT
  CASE
    WHEN delay_or_not_by_days = 0 THEN 'avg_ser'
    WHEN delay_or_not_by_days < 0 THEN 'fast_ser'
    ELSE 'slow_ser'
  END AS service_type,
  COUNT(delay_or_not_by_days) AS cnt
FROM
  (
    SELECT
      order_id,
      DATE_DIFF(DATE(order_estimated_delivery_date), DATE(order_purchase_timestamp),
DAY) AS estimated_days_to_deliver,
      DATE_DIFF(DATE(order_delivered_customer_date), DATE(order_purchase_timestamp),
DAY) AS actual_days_to_deliver,
      DATE_DIFF(DATE(order_delivered_customer_date),
DATE(order_estimated_delivery_date), DAY) AS delay_or_not_by_days
    FROM
      `target_Brazil.orders`
    WHERE
      DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) IS NOT
NULL
      AND DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date, DAY)
IS NOT NULL
  )
GROUP BY
  service_type;
```

Output:

Row	service_type	cnt
1	fast_ser	88649
2	slow_ser	6535
3	avg_ser	1292

Analysis:

1. Delivery Performance: The majority of orders, 88,649 in total, are delivered before the estimated delivery date. This indicates a good level of efficiency in the delivery process.
2. Late Deliveries: However, there are 6,535 orders that were delivered after the estimated delivery date. It is important to investigate the reasons behind these delays and take appropriate measures to reduce late deliveries. This could involve optimizing logistics, improving coordination with delivery partners, or addressing any operational inefficiencies.
3. On-Time Deliveries: It's encouraging to see that 1,292 orders were delivered exactly on the estimated delivery date. Maintaining a high percentage of on-time deliveries is crucial for customer satisfaction and retention.

Recommendations:

- Optimize delivery processes to reduce late deliveries.
- Improve supply chain management to ensure smooth operations.
- Implement real-time tracking and communication systems for better customer experience.
- Continuously analyze delivery performance.

2. Find time_to_delivery & diff_estimated_delivery. Formula for the same given
below:
o time_to_delivery =
order_delivered_customer_date-order_purchase_timestamp
o diff_estimated_delivery =
order_estimated_delivery_date-order_delivered_customer_date

Query:

```
SELECT order_id,
date_diff(
date(order_delivered_customer_date),date(order_purchase_timestamp),day)
as time_to_delivery,
date_diff(
date(order_estimated_delivery_date),date(order_delivered_customer_date),day)
```

```
as diff_estimated_delivery
FROM `target-sql-386318.target_Brazil.orders`
```

Output:

Row	order_id	time_to_delivery	diff_estimated_delive
1	1950d777989f6a877539f5379...	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	31	29
3	65d1e226dfaeb8cdc42f66542...	36	17
4	635c894d068ac37e6e03dc54e...	31	2
5	3b97562c3aee8bdedcb5c2e45...	33	1
6	68f47f50f04c4cb6774570cfde...	30	2
7	276e9ec344d3bf029ff83a161c...	44	-4
8	54e1a3c2b97fb0809da548a59...	41	-4
9	fd04fa4105ee8045f6a0139ca5...	37	-1
10	302bb8109d097a9fc6e9cefc5...	34	-5

3. Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

Query:

```
SELECT
  customer_state,
  AVG(oi.freight_value) AS mean_fv,
  AVG(DATE_DIFF(DATE(o.order_delivered_customer_date),
DATE(o.order_purchase_timestamp), DAY)) AS mean_time_to_delivery,
  AVG(DATE_DIFF(DATE(o.order_estimated_delivery_date),
DATE(o.order_delivered_customer_date), DAY)) AS mean_diff_est_delivery
FROM
  `target_Brazil.customers` c
  INNER JOIN `target_Brazil.orders` o ON c.customer_id = o.customer_id
  INNER JOIN `target_Brazil.order_items` oi ON o.order_id = oi.order_id
GROUP BY
  customer_state;
```

Output:

Row	customer_state	mean_fv	mean_time_to_d	mean_diff_est_d
1	MT	28.1662843...	17.9074252...	14.5718418...
2	MA	38.2570024...	21.5899999...	9.90624999...
3	AL	35.8436711...	24.4473067...	8.73536299...
4	SP	15.1472753...	8.66225265...	11.2079107...
5	MG	20.6301668...	11.9207246...	13.3426492...
6	PE	32.9178626...	18.2245131...	13.4501718...
7	RJ	20.9609239...	15.0747914...	12.0147744...
8	DF	21.0413549...	12.8938428...	12.2004246...

4. Sort the data to get the following:

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

Query: (for top 5 states with highest average freight value)

```
SELECT
  customer_state,
  AVG(oi.freight_value) AS mean_fv,
FROM
  `target_Brazil.customers` c
  INNER JOIN `target_Brazil.orders` o ON c.customer_id = o.customer_id
  INNER JOIN `target_Brazil.order_items` oi ON o.order_id = oi.order_id
GROUP BY
  customer_state
  ORDER BY mean_fv DESC LIMIT 5;
```

Output:

Row	customer_state	mean_fv
1	RR	42.9844230...
2	PB	42.7238039...
3	RO	41.0697122...
4	AC	40.0733695...
5	PI	39.1479704...

Analysis:
Top 5 states with highest mean Freight value are RR,PB,RO,AC,PI with mean freight value equal to 42.98,42.72,41.06,40.07,30.14 respectively.

Query: (for top 5 states with lowest average freight value)

```
SELECT
  customer_state,
  AVG(oi.freight_value) AS mean_fv,
FROM
  `target_Brazil.customers` c
  INNER JOIN `target_Brazil.orders` o ON c.customer_id = o.customer_id
  INNER JOIN `target_Brazil.order_items` oi ON o.order_id = oi.order_id
GROUP BY
  customer_state
ORDER BY mean_fv ASC LIMIT 5
```

Output:

Row	customer_state	mean_fv
1	SP	15.1472753...
2	PR	20.5316515...
3	MG	20.6301668...
4	RJ	20.9609239...
5	DF	21.0413549...

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

Query: (for top 5 states with highest average time to delivery)

```
SELECT
  customer_state,
  AVG(DATE_DIFF(DATE(o.order_delivered_customer_date),
DATE(o.order_purchase_timestamp), DAY)) AS avg_time_to_delivery,
FROM
  `target_Brazil.customers` c
  INNER JOIN `target_Brazil.orders` o ON c.customer_id = o.customer_id
GROUP BY
  customer_state
  ORDER BY avg_time_to_delivery DESC LIMIT 5
```

Output:

Row	customer_state	avg_time_to_del
1	RR	29.3414634...
2	AP	27.1791044...
3	AM	26.3586206...
4	AL	24.5012594...
5	PA	23.7251585...

Query: (for top 5 states with lowest average time to delivery)

```
SELECT
  customer_state,
  AVG(DATE_DIFF(DATE(o.order_delivered_customer_date),
DATE(o.order_purchase_timestamp), DAY)) AS avg_time_to_delivery,
FROM
  `target_Brazil.customers` c
  INNER JOIN `target_Brazil.orders` o ON c.customer_id = o.customer_id
GROUP BY
  customer_state
  ORDER BY avg_time_to_delivery ASC LIMIT 5
```

Output:

Row	customer_state	avg_time_to_del
1	SP	8.70053092...
2	PR	11.9380459...
3	MG	11.9465433...
4	DF	12.8990384...
5	SC	14.9075274...

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

Query: (for top 5 states where delivery is really fast compared to estimated date)

```
SELECT
  customer_state,
  AVG(DATE_DIFF(DATE(order_delivered_customer_date),
DATE(order_estimated_delivery_date), DAY)) AS avg_delay
FROM
  `target_Brazil.customers` c
  INNER JOIN `target_Brazil.orders` o USING (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	-20.724999...
2	RO	-20.102880...
3	AP	-19.686567...
4	AM	-19.565517...
5	RR	-17.292682...

Analysis:
AC state is the state where the average delay on getting orders delivered compared to estimated date is lowest i.e., an order in state AC gets delivered on average before almost 20 days from estimated date.

Query: (for top 5 states where delivery is not so fast compared to estimated date)

```
SELECT
  customer_state,
  AVG(DATE_DIFF(DATE(order_delivered_customer_date),
DATE(order_estimated_delivery_date), DAY)) AS avg_delay
FROM
  `target_Brazil.customers` c
  INNER JOIN `target_Brazil.orders` o USING (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 DESC
LIMIT 5;
```

Output:

Row	customer_state	avg_delay
1	AL	-8.7078085...
2	MA	-9.5718270...
3	SE	-10.020895...
4	ES	-10.496240...
5	BA	-10.794533...

6. Payment type analysis:

1. Month over Month count of orders for 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(order_id) AS cnt
  FROM
    `target_Brazil.orders` o
    INNER JOIN `target_Brazil.payments` p USING (order_id)
  GROUP BY
    year,
    month,
    payment_type) x
ORDER BY
  payment_type,
  year,
  month;
```

Output:

Row	year	month	payment_type	cnt	month_over_mo
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

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

Query:

```
select payment_installments,
count(order_id) as total_orders
from `target-sql-386318.target_Brazil.payments`
group by payment_installments
order by payment_installments,total_orders
```

Output:

Row	payment_installr	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:

1. Payment Preference: The majority of customers prefer to pay in a single installment (option 1). This suggests that customers prefer the convenience and simplicity of paying the full amount in less installments.
2. Popular Installment Options: Following the single installment preference, customers often choose to pay in 2, 3, or 4 installments. These installment options provide customers with the flexibility to spread out their payments over multiple periods without significant financial burden.
3. Limited Interest in High Installment Options: Installment options above 9 are rarely chosen by customers. This indicates that customers are less interested in longer-term payment plans and may prefer to pay off their purchases within a shorter timeframe.

Recommendations:

- Promote the benefits of single installment payment.
- Offer attractive installment plans for 2, 3, and 4 installments.
- Evaluate the relevance of installment options above 9.
- Provide clear payment information and educate customers about their options.