Target Business Case Study

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

1. **Data type of columns in a table -**

Answer-

```
Select

column_name,

data_type
from `scaler-dsml-sql-381505`.Target_SQL.INFORMATION_SCHEMA.COLUMNS
  WHERE table_name = 'customers';
```

Query results

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

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. **Time period for which the data is given-**

Answer –

```
SELECT TIMESTAMP_DIFF(MAX(order_purchase_timestamp), MIN(order_purchase_tim
estamp), DAY) AS Time_period

FROM `Target_SQL.orders`;
```

Query results

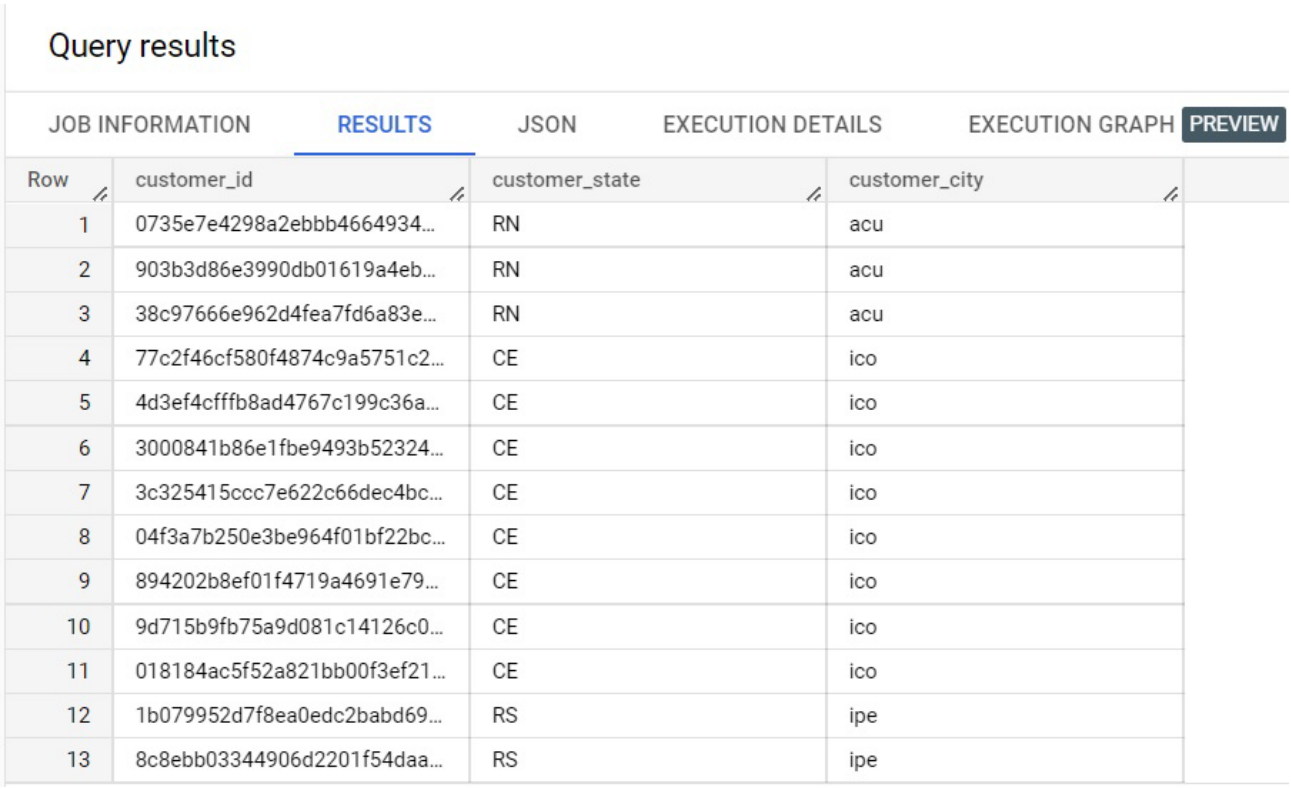
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	Time_period
1	772

3. **Cities and States of customers ordered during the given period-**

Answer-

```
SELECT
DISTINCT c.customer_id,
c.customer_state,
c.customer_city
from `Target_SQL.customers` as c
join `Target_SQL.orders` as o
on c.customer_id = o.customer_id
where o.order_purchase_timestamp between (SELECT MIN(order_purchase_timestamp) FR
OM `Target_SQL.orders`) AND (SELECT MAX(order_purchase_timestamp) FROM `Targe
t_SQL.orders`);
```



Query results

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

Row	customer_id	customer_state	customer_city
1	0735e7e4298a2ebbb4664934...	RN	acu
2	903b3d86e3990db01619a4eb...	RN	acu
3	38c97666e962d4fea7fd6a83e...	RN	acu
4	77c2f46cf580f4874c9a5751c2...	CE	ico
5	4d3ef4cfffb8ad4767c199c36a...	CE	ico
6	3000841b86e1fbe9493b52324...	CE	ico
7	3c325415ccc7e622c66dec4bc...	CE	ico
8	04f3a7b250e3be964f01bf22bc...	CE	ico
9	894202b8ef01f4719a4691e79...	CE	ico
10	9d715b9fb75a9d081c14126c0...	CE	ico
11	018184ac5f52a821bb00f3ef21...	CE	ico
12	1b079952d7f8ea0edc2babd69...	RS	ipe
13	8c8ebb03344906d2201f54daa...	RS	ipe

Q2. 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?**

Answer-

```
SELECT
COUNT(order_id) AS orders_count,
EXTRACT(year FROM order_purchase_timestamp) AS Year,
EXTRACT(month FROM order_purchase_timestamp) AS Month
FROM `Target_SQL.orders`
GROUP BY Year, Month
ORDER BY Year, Month;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	total_orders_count	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
9	3245	2017	6
10	4026	2017	7
11	4331	2017	8
12	4285	2017	9
13	4631	2017	10
14	7544	2017	11

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

Answer-

```
SELECT COUNT(customer_id) AS total_customer_count,
CASE
WHEN EXTRACT(hour FROM order_purchase_timestamp) BETWEEN 2 AND 6 THEN
'Dawn'
WHEN EXTRACT(hour FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN '
Morning'
WHEN EXTRACT(hour FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN
 'Afternoon'
ELSE 'Night'
END AS Purchase_time
FROM `Target_SQL.orders`
GROUP BY Purchase_time
ORDER BY Purchase_time
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	total_customer_	Purchase_time
1	38135	Afternoon
2	1678	Dawn
3	27733	Morning
4	31895	Night

Q3. Evolution of E-commerce orders in the Brazil region:

1. **Get month on month orders by states :-**

Answer –

```
SELECT *,
ROUND((m.order_id_count - (LAG(m.order_id_count,1)
OVER(PARTITION BY m.customer_state ORDER BY m.customer_state, Year, Month)))*1
00/LAG(m.order_id_count,1)
OVER(PARTITION BY m.customer_state ORDER BY m.customer_state, Year, Month),2)
AS  over_month
FROM
(SELECT DISTINCT c.customer_state,COUNT(o.order_id) AS order_id_count,
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.customers` AS c
USING(customer_id)
GROUP BY c.customer_state, Year, Month
ORDER BY c.customer_state, Year, Month) AS m
ORDER BY m.customer_state, Year, Month
```

Query results

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

Row	customer_state	order_id_count	Year	Month	over_month
1	AC	2	2017	1	null
2	AC	3	2017	2	50.0
3	AC	2	2017	3	-33.33
4	AC	5	2017	4	150.0
5	AC	8	2017	5	60.0
6	AC	4	2017	6	-50.0
7	AC	5	2017	7	25.0
8	AC	4	2017	8	-20.0
9	AC	5	2017	9	25.0
10	AC	6	2017	10	20.0
11	AC	5	2017	11	-16.67
12	AC	5	2017	12	0.0
13	AC	6	2018	1	20.0

2. **Distribution of customers across the states in Brazil:-**

Answer –

```
SELECT CS.customer_state,
ROUND(CS.c_count*100/SUM(CS.c_count) OVER(),2) AS count_per100 FROM(
SELECT DISTINCT COUNT(customer_id) AS c_count, customer_state
FROM `Target_SQL.customers`
GROUP BY customer_state) AS CS
ORDER BY count_per100 DESC
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	count_per100
1	SP	41.98
2	RJ	12.92
3	MG	11.7
4	RS	5.5
5	PR	5.07
6	SC	3.66
7	BA	3.4
8	DF	2.15
9	ES	2.04
10	GO	2.03
11	PE	1.66
12	CE	1.34

Q4. 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**

Answer –

```
SELECT *,
CONCAT(ROUND((y.total_payment - (LAG(y.total_payment,1) OVER(ORDER BY Year,
Month)))*100/LAG(y.total_payment,1) OVER(ORDER BY Year, Month),2),"%") AS m_o_
m_per100
FROM
(select
extract(YEAR from order_purchase_timestamp) as Year,
extract (MONTH from order_purchase_timestamp) as Month ,
round(SUM(p.payment_value),2) as total_payment
from `Target_SQL.orders` as o
join `Target_SQL.payments` as p
on  o.order_id = p.order_id
group by Year, Month
order by Year, Month) AS y
where y.Year BETWEEN 2017 and 2018 and y.Month between 1 and 8
order by Year, Month;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION

Row	Year	Month	total_payment	m_o_m_per100
1	2017	1	138488.04	*null*
2	2017	2	291908.01	110.78%
3	2017	3	449863.6	54.11%
4	2017	4	417788.03	-7.13%
5	2017	5	592918.82	41.92%
6	2017	6	511276.38	-13.77%
7	2017	7	592382.92	15.86%
8	2017	8	674396.32	13.84%
9	2018	1	1115004.18	65.33%
10	2018	2	992463.34	-10.99%
11	2018	3	1159652.12	16.85%
12	2018	4	1160785.48	0.1%
13	2018	5	1153982.15	-0.59%
14	2018	6	1023880.5	-11.27%

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

Answer -

```
select
sum (price) as Sum_price,
sum (freight_value) as Sum_fv,
avg (price) as Mean_price,
avg (freight_value) as Mean_fv,
customer_state
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;
```

Query results

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

Row	Sum_price	Sum_fv	Mean_price	Mean_fv	customer_state
1	15982.9499...	3686.74999...	173.727717...	40.0733695...	AC
2	80314.81	15914.5899...	180.889211...	35.8436711...	AL
3	22356.8400...	5478.88999...	135.495999...	33.2053939...	AM
4	13474.2999...	2788.50000...	164.320731...	34.0060975...	AP
5	511349.990...	100156.679...	134.601208...	26.3639589...	BA
6	227254.709...	48351.5899...	153.758261...	32.7142016...	CE
7	302603.939...	50625.4999...	125.770548...	21.0413549...	DF
8	275037.309...	49764.5999...	121.913701...	22.0587765...	ES
9	294591.949...	53114.9799...	126.271731...	22.7668152...	GO
10	119648.219...	31523.7700...	145.204150...	38.2570024...	MA
11	1585308.02...	270853.460...	120.748574...	20.6301668...	MG
12	116812.639...	19144.0300...	142.628376...	23.3748840...	MS
13	156453.529...	29715.4300...	148.297184...	28.1662843...	MT

Q5. Analysis on sales, freight and delivery time:-

1. **Calculate days between purchasing, delivering and estimated delivery**

Answer –

```
SELECT order_id,
TIMESTAMP_DIFF(order_estimated_delivery_date,order_purchase_timestamp, Day) as Da
ys_purchase_est_delivery,
TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, Day) as D
ays_purchase_delivery,
TIMESTAMP_DIFF(order_estimated_delivery_date, order_delivered_customer_date, Day) a
s Days_delivery
FROM `Target_SQL.orders`
```

Query results

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

Row	order_id	Days_purchase_est_delivery	Days_purchase_delivery	Days_delivery
1	f88aac7ebccb37f19725a0753...	50	null	null
2	790cd37689193dca0d00d2feb...	6	null	null
3	49db7943d60b6805c3a41f547...	44	null	null
4	063b573b88fc80e516aba87df...	54	null	null
5	a68ce1686d536ca72bd2dadc4...	56	null	null
6	45973912e490866800c0aea8f...	54	null	null
7	cda873529ca7ab71f677d5ec1...	56	null	null
8	ead20687129da8f5d89d831bb...	41	null	null
9	6f028ccb7d612af251aa442a1f...	3	null	null
10	8733c8d440c173e524d2fab80...	3	null	null
11	986dfd5411cb5a65f3fe024bdb...	47	null	null
12	34d981c2cff2bb39afd6bb3f42...	44	null	null
13	369d4391cc475b184da61af43...	43	null	null

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

Answer –

```
select
order_id,
timestamp_diff (order_delivered_customer_date ,order_purchase_timestamp, Day ) as time_t
o_delivery,
timestamp_diff (order_delivered_customer_date,order_estimated_delivery_date ,Day )as diff
_estimated_delivery
```

```
from `Target_SQL.orders` ;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_id	time_to_delivery	diff_estimated_c
1	1950d777989f6a877539f5379...	30	12
2	2c45c33d2f9cb8ff8b1c86cc28...	30	-28
3	65d1e226dfaeb8cdc42f66542...	35	-16
4	635c894d068ac37e6e03dc54e...	30	-1
5	3b97562c3aee8bdedcb5c2e45...	32	0
6	68f47f50f04c4cb6774570cfde...	29	-1
7	276e9ec344d3bf029ff83a161c...	43	4
8	54e1a3c2b97fb0809da548a59...	40	4
9	fd04fa4105ee8045f6a0139ca5...	37	1
10	302bb8109d097a9fc6e9cefc5...	33	5
11	66057d37308e787052a32828...	38	6
12	19135c945c554eebfd7576c73...	36	2
13	4493e45e7ca1084efcd38ddeb...	34	0

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

Answer –

```
SELECT
c.customer_state,
avg (oi.freight_value) as Mean_fv,
ROUND(AVG (timestamp_diff (o.order_delivered_customer_date ,o.order_purchase_timesta
mp, Day )),2) as time_to_delivery,
ROUND (AVG (timestamp_diff (o.order_delivered_customer_date,o.order_estimated_delive
ry_date ,Day )),2) as diff_estimated_delivery
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 ;

Query results

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

Row	customer_state	Mean_fv	time_to_delivery	diff_estimated_delivery
1	RN	35.6523629...	18.87	-13.06
2	CE	32.7142016...	20.54	-10.26
3	RS	21.7358043...	14.71	-13.2
4	SC	21.4703687...	14.52	-10.67
5	SP	15.1472753...	8.26	-10.27
6	MG	20.6301668...	11.52	-12.4
7	BA	26.3639589...	18.77	-10.12
8	RJ	20.9609239...	14.69	-11.14
9	GO	22.7668152...	14.95	-11.37
10	MA	38.2570024...	21.2	-9.11
11	PE	32.9178626...	17.79	-12.55
12	PB	42.7238039...	20.12	-12.15
13	ES	22.0587765...	15.19	-9.77

4. **Sort the data to get the following:**

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

Answer –

```
SELECT * FROM(
SELECT *, RANK() OVER(ORDER BY Highest_freight_value) AS r2 FROM(SELECT c.c
ustomer_state, ROUND(AVG(oi.freight_value),2) AS Highest_freight_value
FROM `Target_SQL.orders` AS o
JOIN `Target_SQL.order_items` as oi
USING(order_id)
JOIN `Target_SQL.customers` AS c
ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY Highest_freight_value
LIMIT 5)) AS q
JOIN(
SELECT *, RANK() OVER(ORDER BY Highest_freight_value1) AS r1 FROM(
```

```
SELECT c.customer_state AS customer_state1, ROUND(AVG(oi.freight_value),2) AS High
est_freight_value1,
FROM `Target_SQL.orders` AS o
JOIN `Target_SQL.order_items` as oi
USING(order_id)
JOIN `Target_SQL.customers` AS c
ON c.customer_id = o.customer_id
GROUP BY customer_state1
ORDER BY Highest_freight_value1 DESC
LIMIT 5)) AS b
ON b.r1 = q.r2
ORDER BY r1
```

Query results

SAVE RESU

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

Row	customer_state	Highest_freight	r2	customer_state1	Highest_freight	r1
1	SP	15.15	1	PI	39.15	1
2	PR	20.53	2	AC	40.07	2
3	MG	20.63	3	RO	41.07	3
4	RJ	20.96	4	PB	42.72	4
5	DF	21.04	5	RR	42.98	5

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

Answer –

```
SELECT * FROM(
SELECT *,
RANK() OVER(ORDER BY Highest_Avg_Time_To_Delivery) AS r2 FROM(SELECT c.cu
stomer_state,
ROUND(AVG(TIMESTAMP_DIFF(order_delivered_customer_date,order_purchase_timest
amp, Day)),2) AS Highest_Avg_Time_To_Delivery,
FROM `Target_SQL.orders` AS o
JOIN `Target_SQL.order_items` AS oi
USING(order_id)
JOIN `Target_SQL.customers` AS c
USING(customer_id)
GROUP BY c.customer_state
ORDER BY Highest_Avg_Time_To_Delivery DESC
```

```
LIMIT 5)) AS q
JOIN(
SELECT *, RANK() OVER(ORDER BY Lowest_Avg_Time_To_Delivery) AS r1 FROM(
SELECT c.customer_state,
ROUND(AVG(TIMESTAMP_DIFF(order_delivered_customer_date,order_purchase_timest
amp, Day)),2) AS Lowest_Avg_Time_To_Delivery,
FROM `Target_SQL.orders` AS o
JOIN `Target_SQL.order_items` AS oi
USING(order_id)
JOIN `Target_SQL.customers` AS c
USING(customer_id)
GROUP BY c.customer_state
ORDER BY Lowest_Avg_Time_To_Delivery
LIMIT 5)) AS b
ON b.r1 = q.r2
ORDER BY r1
```

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	customer_state	Highest_Avg_Time_To_Delivery	r2	customer_state_1	Lowest_Avg_Time_To_Delivery	r1
1	PA	23.3	1	SP	8.26	1
2	AL	23.99	2	PR	11.48	2
3	AM	25.96	3	MG	11.52	3
4	AP	27.75	4	DF	12.5	4
5	RR	27.83	5	SC	14.52	5

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

Answer –

```
SELECT DISTINCT * FROM(
SELECT *, DENSE_RANK() OVER(ORDER BY Fast_delivery) AS r2 FROM(SELECT DI
STINCT c.customer_state, TIMESTAMP_DIFF(order_delivered_customer_date, order_esti
mated_delivery_date, day) AS Fast_delivery
FROM `Target_SQL.orders` AS o
JOIN `Target_SQL.order_items` AS oi
USING(order_id)
JOIN `Target_SQL.customers` AS c
ON c.customer_id = o.customer_id
WHERE order_delivered_customer_date IS NOT NULL
ORDER BY Fast_delivery DESC
LIMIT 5)) AS q
JOIN(
```

```
SELECT *, DENSE_RANK() OVER(ORDER BY Not_So_Fast) AS r1 FROM(
SELECT DISTINCT c.customer_state, TIMESTAMP_DIFF(order_delivered_customer_date,
 order_estimated_delivery_date, day) AS Not_So_Fast
FROM `Target_SQL.orders` AS o
JOIN `Target_SQL.order_items` AS oi
USING(order_id)
JOIN `Target_SQL.customers` AS c
ON c.customer_id = o.customer_id
WHERE order_delivered_customer_date IS NOT NULL
ORDER BY Not_So_Fast
LIMIT 5)) AS b
ON b.r1 = q.r2
WHERE Not_So_Fast IS NOT NULL AND Fast_delivery IS NOT NULL
ORDER BY r1
```

Query results

SAVE RESULTS

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

Row	customer_state	Fast_delivery	r2	customer_state_1	Not_So_Fast	r1
1	SE	166	1	SP	-146	1
2	SP	167	2	MA	-139	2
3	SP	175	3	RS	-134	3
4	ES	181	4	SP	-123	4
5	RJ	188	5	RJ	-108	5

Q6. Payment type analysis:

1. **Month over Month count of orders for different payment types**

Answer –

```
SELECT *,
ROUND((t.total_orders - (LAG(t.total_orders,1) OVER(PARTITION BY t.payment_type O
RDER BY Year, Month)))*100/LAG(t.total_orders,1) OVER(PARTITION BY t.payment_ty
pe ORDER BY Year, Month),2) AS m_o_m_per100
FROM
(select
```

```
count(oi.order_id) as total_orders,
p.payment_type,
extract (year from o.order_purchase_timestamp) as Year,
extract (month from o.order_purchase_timestamp) as Month
from
`Target_SQL.payments` as p
join `Target_SQL.order_items` as oi
on p.order_id = oi.order_id
join `Target_SQL.orders` as o
on oi.order_id = o.order_id
group by p.payment_type, Year, Month
order by Year, Month ) as t
ORDER BY t.payment_type,Year, Month;
```

Query results

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

Row	total_orders	payment_type	Year	Month	m_o_m_per100
1	71	UPI	2016	10	null
2	232	UPI	2017	1	226.76
3	438	UPI	2017	2	88.79
4	667	UPI	2017	3	52.28
5	567	UPI	2017	4	-14.99
6	870	UPI	2017	5	53.44
7	810	UPI	2017	6	-6.9
8	970	UPI	2017	7	19.75
9	1067	UPI	2017	8	10.0
10	1041	UPI	2017	9	-2.44
11	1169	UPI	2017	10	12.3
12	1771	UPI	2017	11	51.5

2. **Count of orders based on the no. of payment installments:-**

Answer –

```
select
count (order_id) as Total_count,
payment_installments
from
`Target_SQL.payments`
group  by payment_installments;
```

Query results

JOB INFORMATION | RESULTS | JSON

Row	Total_count	payment_installments
1	2	0
2	52546	1
3	12413	2
4	10461	3
5	7098	4
6	5239	5
7	3920	6
8	1626	7
9	4268	8
10	644	9
11	5328	10
12	23	11
13	133	12
14	16	13

INSIGHTS

1. Here the Data types are String and Integers.
2. There is a growing trend in orders in starting time 2016 and then later 2018 we have seen a sharp dip in orders. Initially we see in 2017 March, we get to see a peak. Also, we can find peaks during 2017 Nov and 2018 Jan, March. With given data we find peak in March repeated, indicating seasonality in the region.
3. Customers prefer afternoons and nights to carry out purchases.
4. When we do see rise in payment value between 2016 to 2017 only considering Jan to Aug month on month even with oscillating growth percentages.
5. We see customers prefer credit card payments more than UPI and debit card.
6. From the result obtained, it is found that the maximum customer distribution is from the state named “SP” in Brazil.
7. For those customers who pay by installments, we see high number for less than 10 installments and few for more than 10 installments.

ASSUMPTIONS

1. We have not used the local time zone of Brazil for classifying the time of the day the orders were made. Also we have taken custom bins for categorizing the time as afternoon, evening, night and dawn.
2. We have taken the customer delivery date and not the carrier delivery date as default delivery date.