BUSINESS CASE: TARGET SQL

JULIAN KINGSLEY J

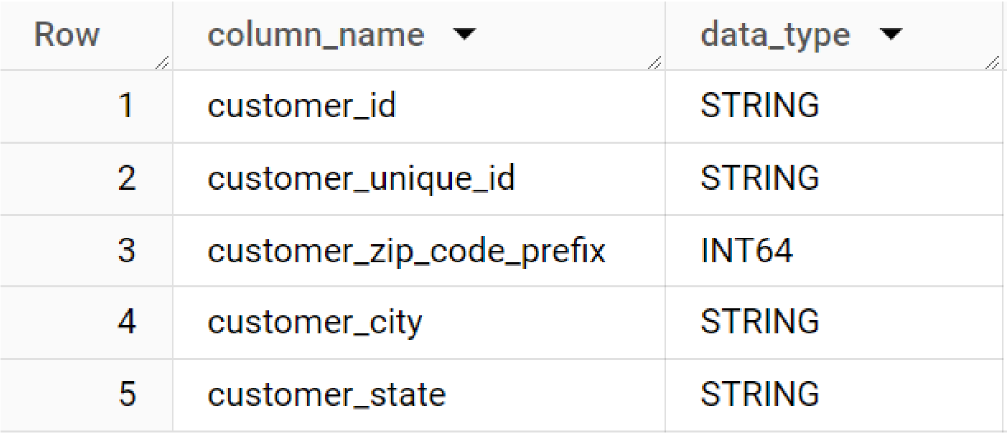
Business Case: Target SQL

I. 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:

Code:

```
SELECT column_name,data_type
FROM `Target_SQL_Business_Case.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'customers';
```



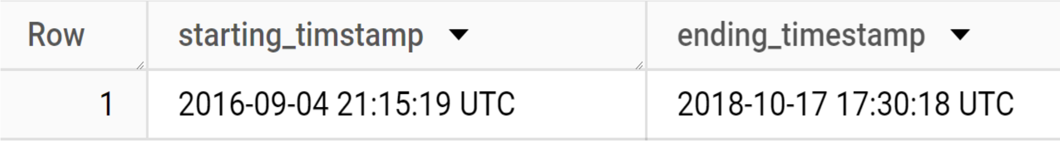
Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

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

Code:

```
SELECT

MIN (order_purchase_timestamp) AS starting_timstamp,
MAX (order_purchase_timestamp) AS ending_timestamp
FROM `Target_SQL_Business_Case.orders`;
```

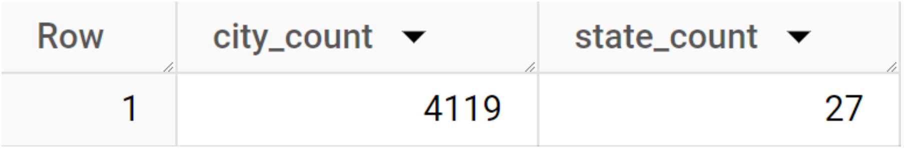


Row	starting_timstamp	ending_timestamp
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

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

Code:

```
SELECT
COUNT(DISTINCT c.customer_city) as city_count,
COUNT(DISTINCT c.customer_state) as state_count
FROM `Target_SQL_Business_Case.orders` o
JOIN `Target_SQL_Business_Case.customers` c
ON o.customer_id = c.customer_id
```



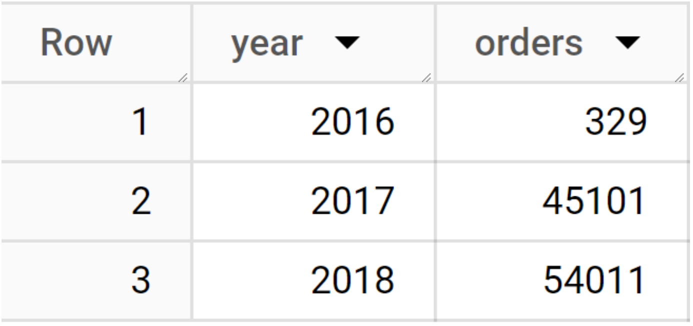
Row	city_count	state_count
1	4119	27

II. In-depth Exploration

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

Code:

```
SELECT
EXTRACT (YEAR FROM order_purchase_timestamp) as year,
COUNT(*) as orders
FROM `Target_SQL_Business_Case.orders`
GROUP BY year
ORDER BY year
```



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

Insights :-

- The number of orders from 2016 – 2018 has shown increase in growth substantially.
- In the year 2017 there was a huge surge in the number of orders placed. From **329** in 2016 to **45101** in 2017.

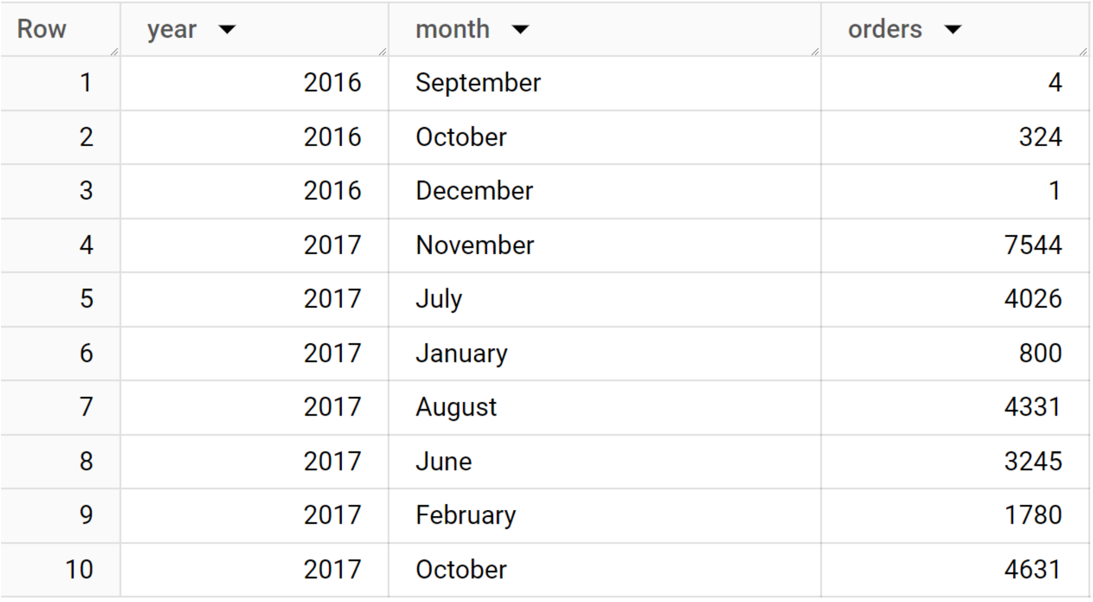
Recommendations :-

- With orders increasing every year it is best to scale inventory size, shipping logistics and customer support to meet demand.

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

Code:

```
SELECT
EXTRACT(YEAR FROM order_purchase_timestamp) as year,
FORMAT_DATE('%B',order_purchase_timestamp) as month,
COUNT(*) as orders
FROM `Target_SQL_Business_Case.orders`
GROUP BY year,month
ORDER BY year
LIMIT 10;
```



Row	year	month	orders
1	2016	September	4
2	2016	October	324
3	2016	December	1
4	2017	November	7544
5	2017	July	4026
6	2017	January	800
7	2017	August	4331
8	2017	June	3245
9	2017	February	1780
10	2017	October	4631

Insights :-

- There is a spike in number of order placed in November 2017 possibly driven by festivals.
- Despite occasional fluctuations, there is an overall growth trend in the number of orders from 2017 to 2018.

Recommendations :-

- It is best to prepare and increase demand for peak months.

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

Code:

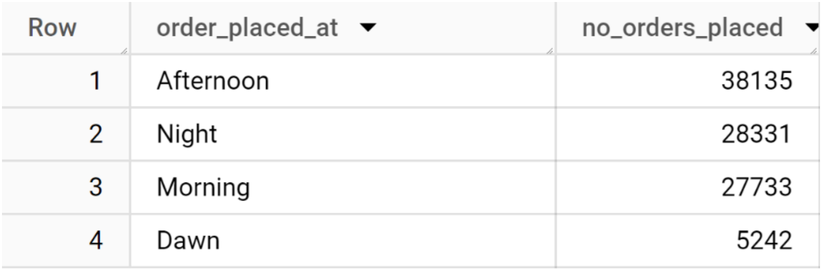
```
WITH hours AS (
  SELECT
  IFNULL(EXTRACT(HOUR FROM order_purchase_timestamp),0)
as hour

  FROM `Target_SQL_Business_Case.orders`
),
in_words AS (

  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'
  ELSE 'Night'
  END AS order_placed_at
  FROM hours
)

SELECT
order_placed_at,
COUNT(*) AS no_orders_placed

FROM in_words
GROUP BY order_placed_at
ORDER BY no_orders_placed DESC;
```



Row	order_placed_at	no_orders_placed
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

Insights :-

- Based on the analysis, most Brazilians order during afternoon period followed by night.

Recommendations :-

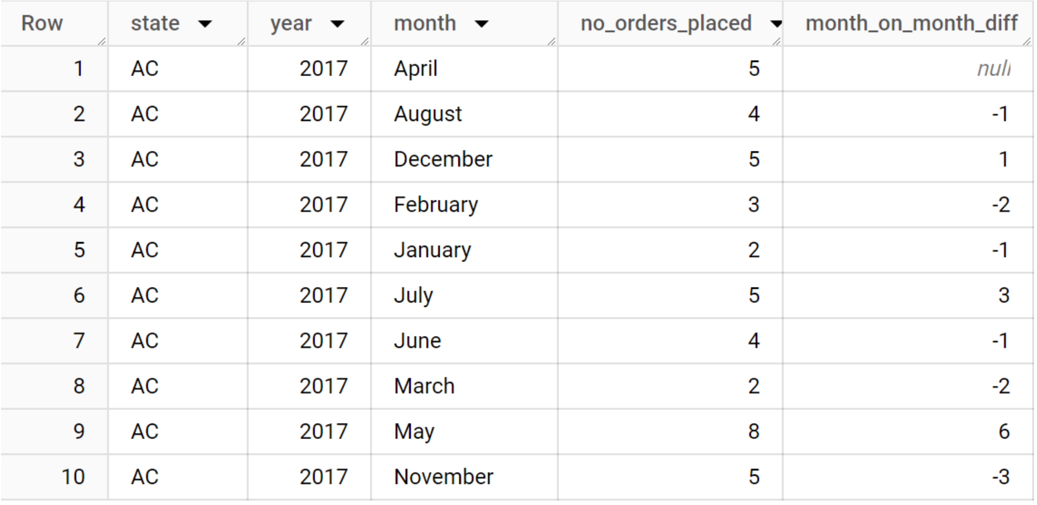
- It is best to boost promotional advertisements during afternoon as it can give the highest conversion rate.

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

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

Code:

```
WITH month_orders AS (
  SELECT
  c.customer_state AS state,
  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
  FORMAT_DATE('%B',o.order_purchase_timestamp) AS month,
  COUNT(*) AS no_orders_placed
  FROM `Target_SQL_Business_Case.orders` o
  JOIN `Target_SQL_Business_Case.customers` c
  ON o.customer_id = c.customer_id
  GROUP BY state,year,month
),
last_month_order AS(
  SELECT state,year,month,
  no_orders_placed,
  LAG(no_orders_placed) OVER (PARTITION BY state ORDER BY
year,month) AS prev_mon_order
  FROM month_orders
)
SELECT
state, year, month, no_orders_placed,
(no_orders_placed - prev_mon_order) AS month_on_month_diff
 FROM last_month_order
 ORDER BY state,year,month
 limit 10;
```



Row	state	year	month	no_orders_placed	month_on_month_diff
1	AC	2017	April	5	*null*
2	AC	2017	August	4	-1
3	AC	2017	December	5	1
4	AC	2017	February	3	-2
5	AC	2017	January	2	-1
6	AC	2017	July	5	3
7	AC	2017	June	4	-1
8	AC	2017	March	2	-2
9	AC	2017	May	8	6
10	AC	2017	November	5	-3

Insights :-

- States like AL have huge negative month-month differences in various months.

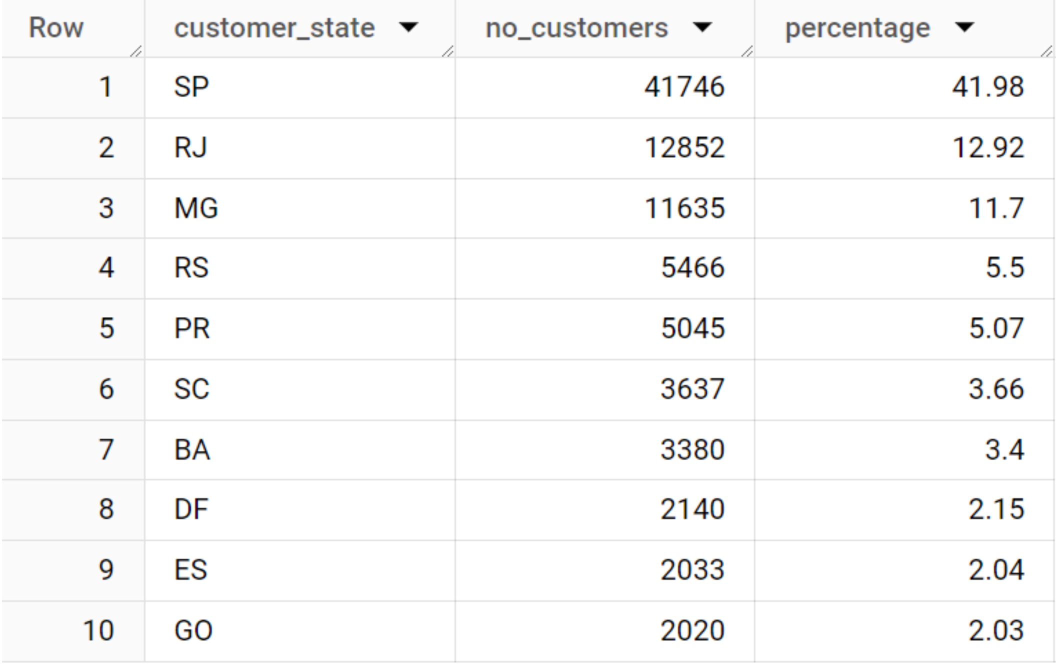
Recommendations :-

- It is best to reform and recreate the marketing strategies in such states.

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

Code:

```
SELECT
customer_state,
 COUNT(*) as no_customers,
ROUND(COUNT(*)/(SELECT COUNT(DISTINCT customer_id)
FROM `Target_SQL_Business_Case.customers`) * 100,2) as percentage
FROM `Target_SQL_Business_Case.customers`
WHERE customer_id IN (SELECT DISTINCT customer_id FROM
`Target_SQL_Business_Case.orders`)
GROUP BY customer_state
ORDER BY percentage DESC
LIMIT 10;
```



Row	customer_state	no_customers	percentage
1	SP	41746	41.98
2	RJ	12852	12.92
3	MG	11635	11.7
4	RS	5466	5.5
5	PR	5045	5.07
6	SC	3637	3.66
7	BA	3380	3.4
8	DF	2140	2.15
9	ES	2033	2.04
10	GO	2020	2.03

Insights :-

- Majority of the customers who order are from states SP, RJ, MG with SP having the highest number of orders - **41746**

Recommendations :-

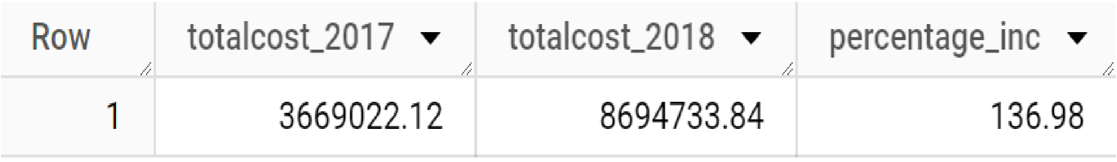
- Boost more promotional advertisements in SP.

IV. 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)

Code:

```
WITH year_2017 AS (
  SELECT
  SUM(p.payment_value) as old_value
  FROM `Target_SQL_Business_Case.orders` o
  JOIN `Target_SQL_Business_Case.payments` p
  ON o.order_id = p.order_id
  WHERE EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017 AND
       EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1
AND 8
),
year_2018 AS (
  SELECT
  SUM(p.payment_value) as new_value
  FROM `Target_SQL_Business_Case.orders` o
  JOIN `Target_SQL_Business_Case.payments` p
  ON o.order_id = p.order_id
  WHERE EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2018 AND
       EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1
AND 8
)
SELECT
ROUND(year_2017.old_value,2) AS totalcost_2017,
ROUND(year_2018.new_value,2) AS totalcost_2018,
ROUND(((year_2018.new_value -
year_2017.old_value)/year_2017.old_value)*100,2)
    AS percentage_inc
FROM year_2017,year_2018
```



Row	totalcost_2017	totalcost_2018	percentage_inc
1	3669022.12	8694733.84	136.98

Insights :-

- The total cost of orders increased significantly from 2017 to 2018, almost doubling with a percentage increase of 136.98%.
- Positive financial trend

Recommendations :-

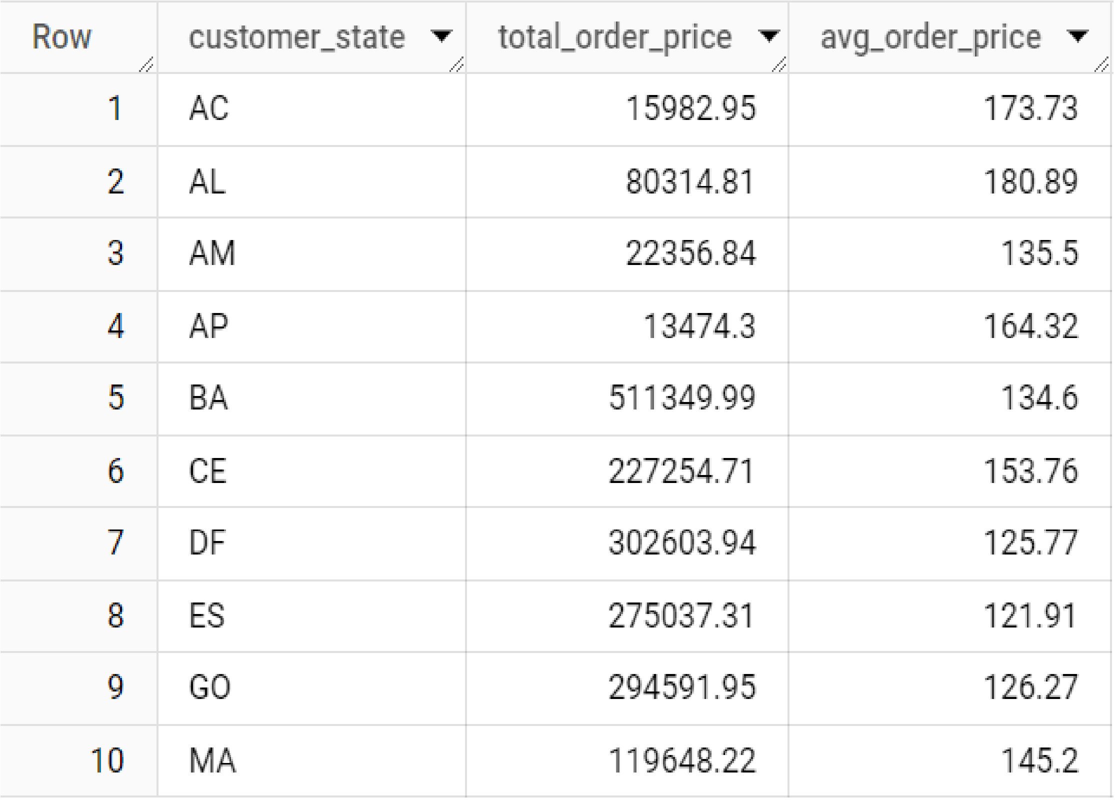
- Identify key contributing factors and based on them develop and sustainable growth strategies.

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

Code:

```
SELECT
c.customer_state,
ROUND(SUM(oi.price),2) as total_order_price,
ROUND(AVG(oi.price),2) as avg_order_price
FROM `Target_SQL_Business_Case.customers` c
```

```
    JOIN `Target_SQL_Business_Case.orders` o
      ON c.customer_id = o.customer_id
        JOIN `Target_SQL_Business_Case.order_items` oi
          ON oi.order_id = o.order_id
  GROUP BY c.customer_state
  ORDER BY c.customer_state
  LIMIT 10;
```



Row	customer_state	total_order_price	avg_order_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 :-

- There is significant variation in average order prices across different states.

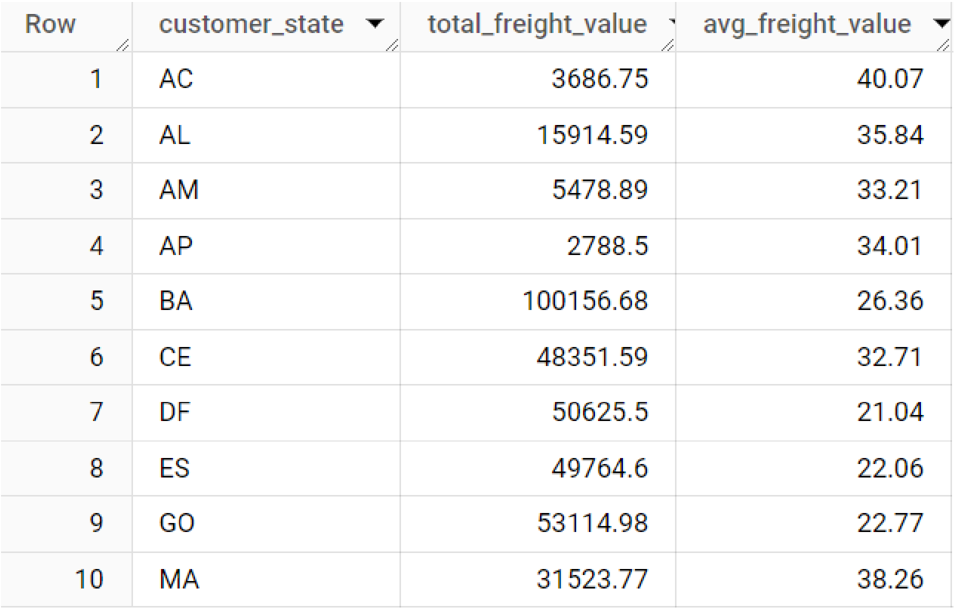
Recommendations :-

- States with lower average order prices, such as PR and RS, might present growth opportunities so region specific marketing can be done increasing more orders in these areas.

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

Code:

```
SELECT
c.customer_state,
ROUND(SUM(oi.freight_value),2) as total_freight_value,
ROUND(AVG(oi.freight_value),2) as avg_freight_value
FROM `Target_SQL_Business_Case.customers` c
  JOIN `Target_SQL_Business_Case.orders` o
    ON c.customer_id = o.customer_id
      JOIN `Target_SQL_Business_Case.order_items` oi
        ON oi.order_id = o.order_id
GROUP BY c.customer_state
ORDER BY c.customer_state
LIMIT 10;
```



Row	customer_state	total_freight_value	avg_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 :-

- There are differences in average freight values across different states. SP (São Paulo) has one of the lowest average freight value at 15.15, while RR (Roraima) has the highest at 42.98

Recommendations :-

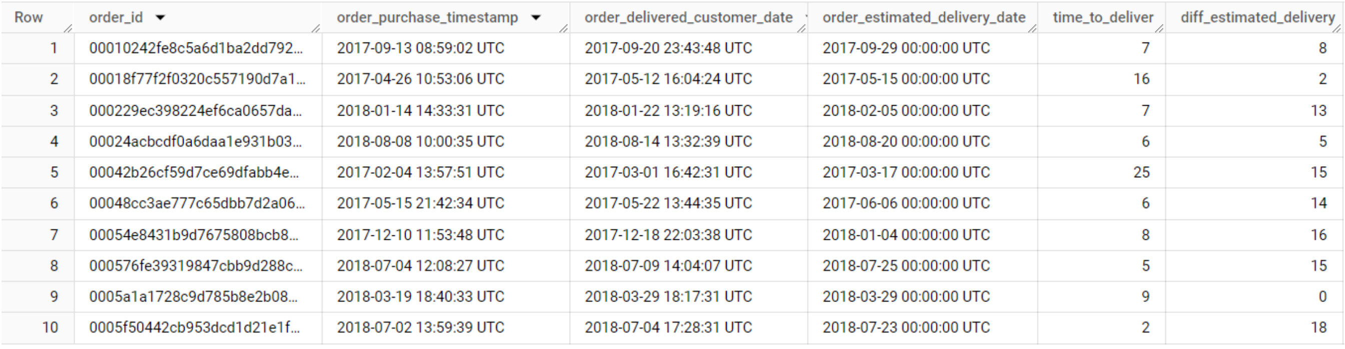
- Evaluate logistics and shipping strategies used in states with higher average freight values and implement the same in other states.

V. 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:

Code:

```
SELECT
o.order_id,
order_purchase_timestamp,
order_delivered_customer_date,
order_estimated_delivery_date,
(TIMESTAMP_DIFF(order_delivered_customer_date,
                order_purchase_timestamp,DAY))
                AS time_to_deliver,
(TIMESTAMP_DIFF(order_estimated_delivery_date,
                order_delivered_customer_date,DAY))
                AS diff_estimated_delivery,
FROM `Target_SQL_Business_Case.customers` c
  JOIN `Target_SQL_Business_Case.orders` o
    ON c.customer_id = o.customer_id
ORDER BY o.order_id
LIMIT 10;
```



Row	order_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_delivery
1	00010242fe8c5a6d1ba2dd792...	2017-09-13 08:59:02 UTC	2017-09-20 23:43:48 UTC	2017-09-29 00:00:00 UTC	7	8
2	00018f77f2f0320c557190d7a1...	2017-04-26 10:53:06 UTC	2017-05-12 16:04:24 UTC	2017-05-15 00:00:00 UTC	16	2
3	000229ec398224ef6ca0657da...	2018-01-14 14:33:31 UTC	2018-01-22 13:19:16 UTC	2018-02-05 00:00:00 UTC	7	13
4	00024acbcdf0a6daa1e931b03...	2018-08-08 10:00:35 UTC	2018-08-14 13:32:39 UTC	2018-08-20 00:00:00 UTC	6	5
5	00042b26cf59d7ce69dfabb4e...	2017-02-04 13:57:51 UTC	2017-03-01 16:42:31 UTC	2017-03-17 00:00:00 UTC	25	15
6	00048cc3ae777c65dbb7d2a06...	2017-05-15 21:42:34 UTC	2017-05-22 13:44:35 UTC	2017-06-06 00:00:00 UTC	6	14
7	00054e8431b9d7675808bcb8...	2017-12-10 11:53:48 UTC	2017-12-18 22:03:38 UTC	2018-01-04 00:00:00 UTC	8	16
8	000576fe39319847cbb9d288c...	2018-07-04 12:08:27 UTC	2018-07-09 14:04:07 UTC	2018-07-25 00:00:00 UTC	5	15
9	0005a1a1728c9d785b8e2b08...	2018-03-19 18:40:33 UTC	2018-03-29 18:17:31 UTC	2018-03-29 00:00:00 UTC	9	0
10	0005f50442cb953dcd1d21e1f...	2018-07-02 13:59:39 UTC	2018-07-04 17:28:31 UTC	2018-07-23 00:00:00 UTC	2	18

Insights :-

- There is variability in delivery times and also in difference between estimated and actual delivery dates. Some orders are delivered faster than the others.

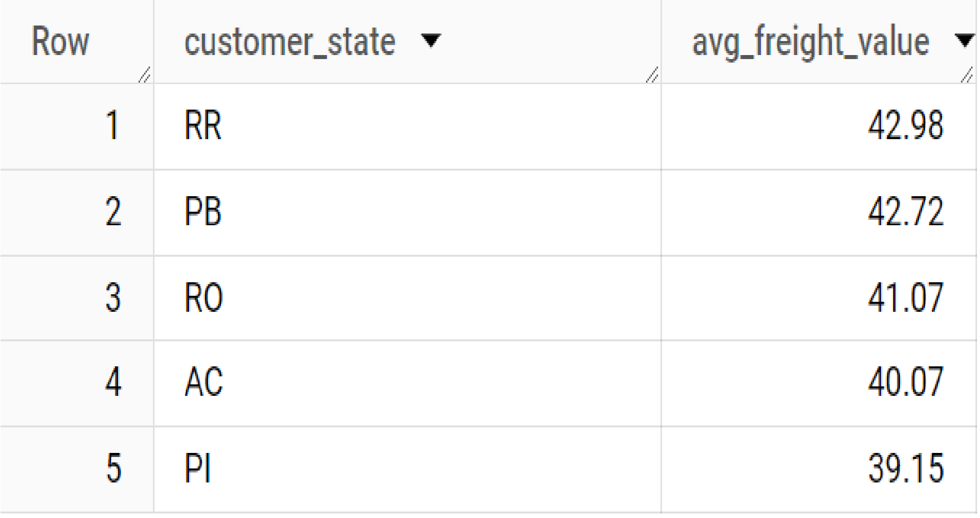
Recommendations :

- Identify the factors that leads to longer delivery times, especially for orders that took more days than others.

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

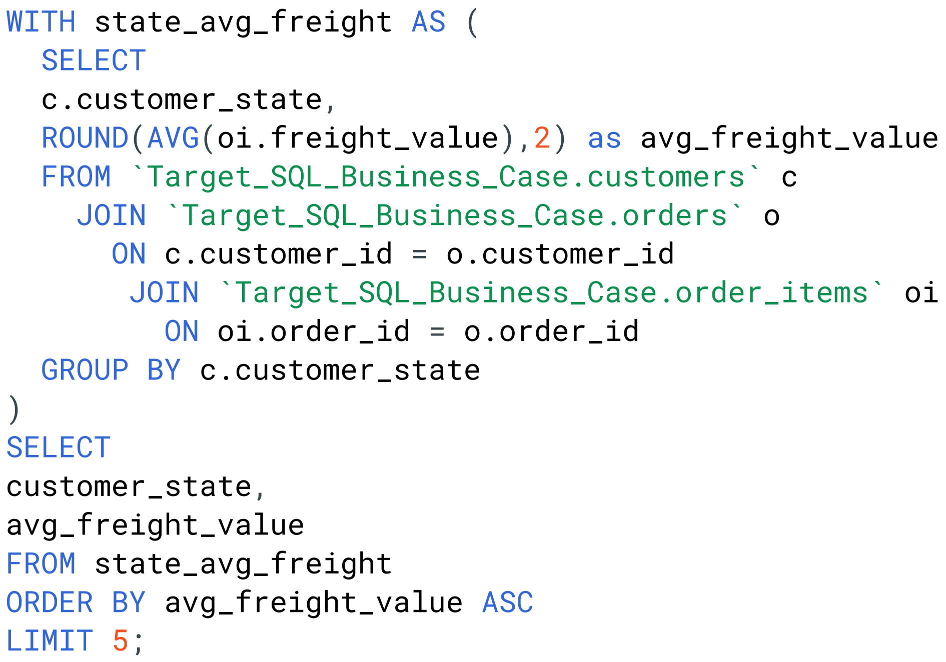
Code for states with top 5 highest average freight values:

```
WITH state_avg_freight AS (
  SELECT
  c.customer_state,
  ROUND(AVG(oi.freight_value),2) as avg_freight_value
  FROM `Target_SQL_Business_Case.customers` c
    JOIN `Target_SQL_Business_Case.orders` o
      ON c.customer_id = o.customer_id
       JOIN `Target_SQL_Business_Case.order_items` oi
         ON oi.order_id = o.order_id
  GROUP BY c.customer_state
)
SELECT
customer_state,
avg_freight_value
FROM state_avg_freight
ORDER BY avg_freight_value DESC
LIMIT 5;
```

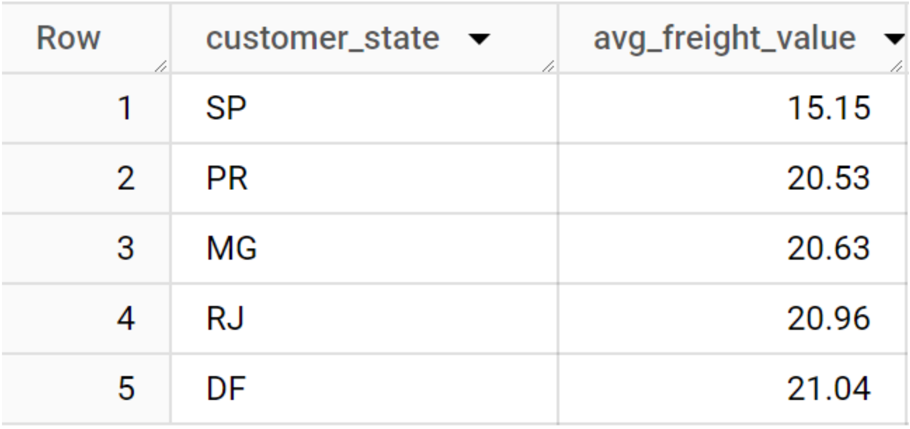


Row	customer_state	avg_freight_value
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15

Code for states with 5 lowest average freight values:



```
WITH state_avg_freight AS (
  SELECT
  c.customer_state,
  ROUND(AVG(oi.freight_value),2) as avg_freight_value
  FROM `Target_SQL_Business_Case.customers` c
    JOIN `Target_SQL_Business_Case.orders` o
      ON c.customer_id = o.customer_id
       JOIN `Target_SQL_Business_Case.order_items` oi
         ON oi.order_id = o.order_id
  GROUP BY c.customer_state
)
SELECT
customer_state,
avg_freight_value
FROM state_avg_freight
ORDER BY avg_freight_value ASC
LIMIT 5;
```



Row	customer_state	avg_freight_value
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04

Insights :-

- PR, PB, RO, AC, PI states have the top 5 highest average freight values with PR having the most - 42.98 average value.
- SP, PR, MG, RJ, DF states have the lowest average freight values.

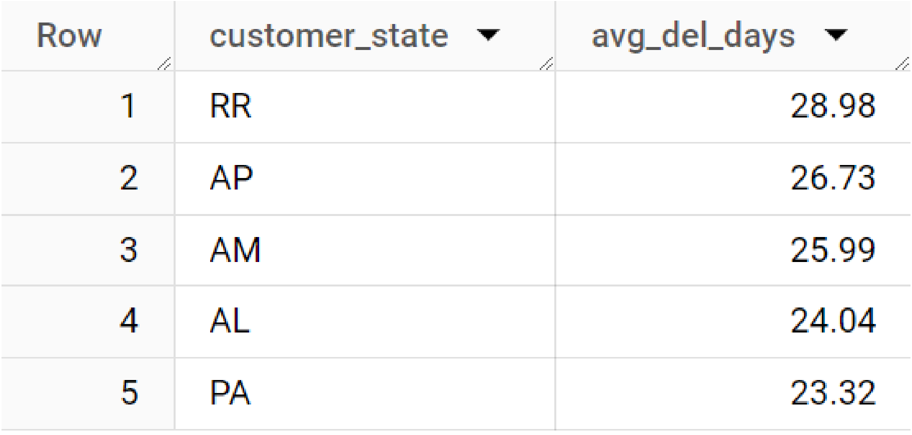
Recommendations :-

- Provide transparent pricing information to enhance customer satisfaction.

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

Code for states with 5 highest average delivery time:

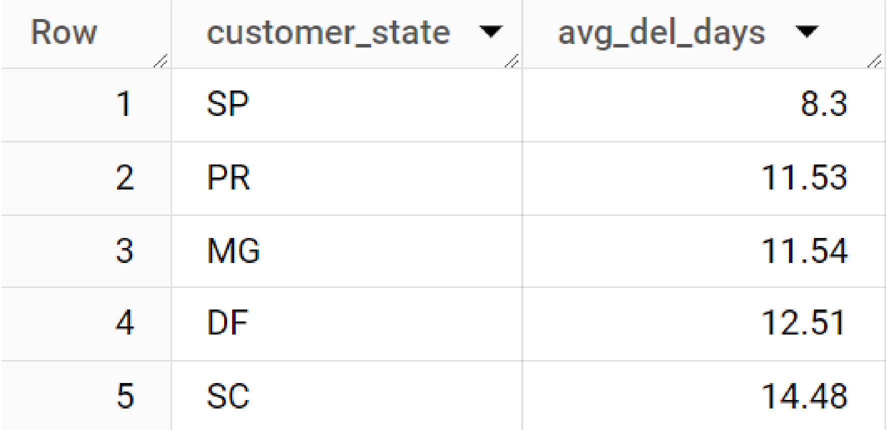
```
WITH cte AS (
  SELECT
  c.customer_state,
  ROUND(AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date,
       o.order_purchase_timestamp,DAY)),2) as avg_del_days
FROM `Target_SQL_Business_Case.customers` c
JOIN `Target_SQL_Business_Case.orders` o
ON c.customer_id = o.customer_id
GROUP BY c.customer_state
)
SELECT
customer_state,
avg_del_days
FROM cte
ORDER BY avg_del_days DESC
LIMIT 5;
```



Row	customer_state	avg_del_days
1	RR	28.98
2	AP	26.73
3	AM	25.99
4	AL	24.04
5	PA	23.32

Code for states with 5 lowest average delivery time:

```
WITH cte AS (
  SELECT
  c.customer_state,
  ROUND(AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date,
       o.order_purchase_timestamp,DAY)),2) as avg_del_days
FROM `Target_SQL_Business_Case.customers` c
JOIN `Target_SQL_Business_Case.orders` o
ON c.customer_id = o.customer_id
GROUP BY c.customer_state
)
SELECT
customer_state,
avg_del_days
FROM cte
ORDER BY avg_del_days
LIMIT 5;
```



Row	customer_state	avg_del_days
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48

Insights :-

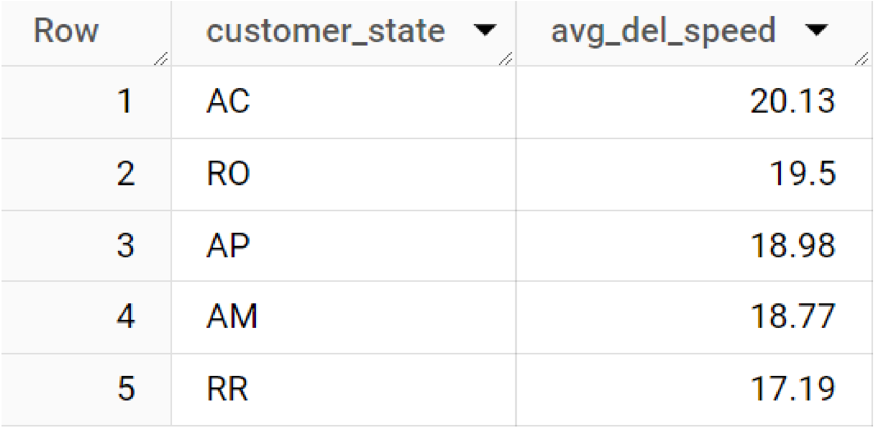
- RR, AP, AM, AL, PA states have the highest average delivery time
- SP, PR, MG, DF, SC states have the lowest average delivery time

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

Code:

```
WITH cte1 AS (
  SELECT
  c.customer_state,
  ROUND(AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date,
       o.order_purchase_timestamp,DAY)),2) as avg_del_days,
  ROUND(AVG(TIMESTAMP_DIFF(o.order_estimated_delivery_date,
       o.order_purchase_timestamp,DAY)),2) as avg_est_days
  FROM `Target_SQL_Business_Case.customers` c
   JOIN `Target_SQL_Business_Case.orders` o
    ON c.customer_id = o.customer_id
GROUP BY c.customer_state
)
```

```
SELECT
customer_state,
ROUND(AVG(avg_est_days - avg_del_days),2) AS avg_del_speed
FROM cte1
GROUP BY customer_state
ORDER BY avg_del_speed DESC
LIMIT 5;
```



Row	customer_state	avg_del_speed
1	AC	20.13
2	RO	19.5
3	AP	18.98
4	AM	18.77
5	RR	17.19

Insights :-

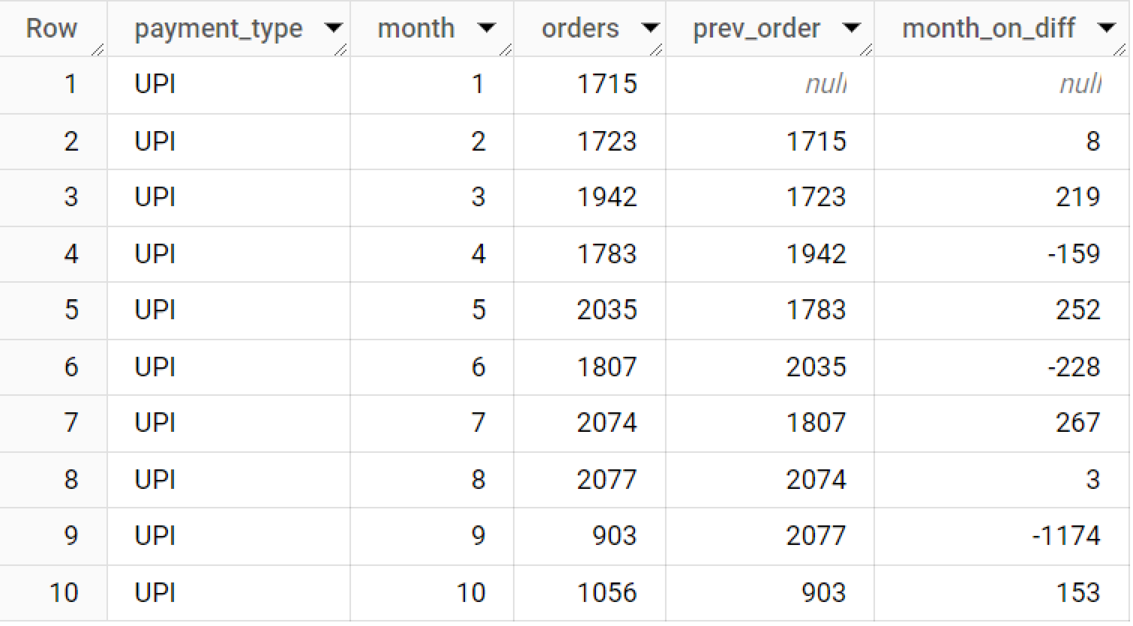
- AC, RO, AP, AM, RR states have the fastest average delivery speed

VI. Analysis based on the payments :-

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

Code:

```
WITH curr_month AS(
  SELECT
  p.payment_type,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
  COUNT(*) AS orders
  FROM `Target_SQL_Business_Case.orders` o
    JOIN `Target_SQL_Business_Case.payments` p
      ON o.order_id = p.order_id
  GROUP BY EXTRACT(MONTH FROM
order_purchase_timestamp),p.payment_type
),
prev_month AS (
  SELECT
  payment_type,
  month, orders,
  LAG(orders) OVER (PARTITION BY payment_type ORDER BY month) AS
    prev_order
  FROM curr_month
)
SELECT
payment_type,
month,
orders,
prev_order,
(orders -  prev_order) month_on_diff
FROM prev_month
ORDER BY payment_type, month
LIMIT 10;
```



Row	payment_type	month	orders	prev_order	month_on_diff
1	UPI	1	1715	*null*	*null*
2	UPI	2	1723	1715	8
3	UPI	3	1942	1723	219
4	UPI	4	1783	1942	-159
5	UPI	5	2035	1783	252
6	UPI	6	1807	2035	-228
7	UPI	7	2074	1807	267
8	UPI	8	2077	2074	3
9	UPI	9	903	2077	-1174
10	UPI	10	1056	903	153

Insights :-

- Months 9 and 12 show significant negative month-on-month differences in voucher orders (-287 and -93, respectively). This suggests a decline in voucher usage during these months compared to the previous ones.

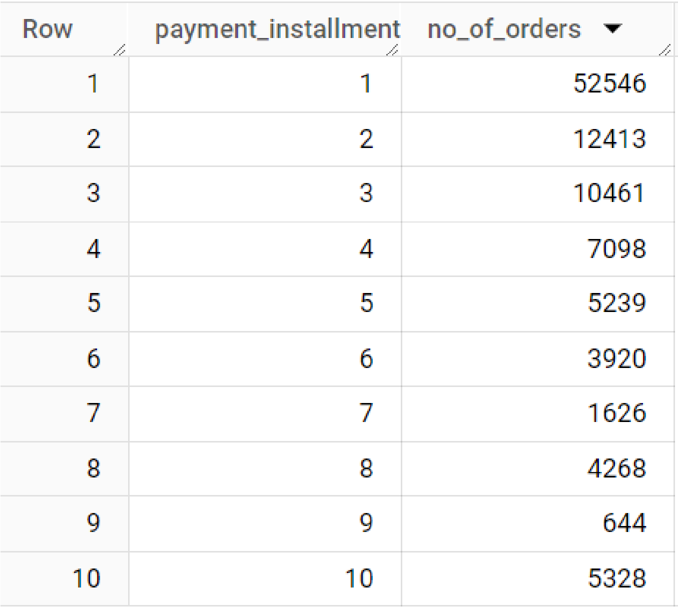
Recommendations :-

- For months with negative differences consider targeted promotions or campaigns to encourage more usage.

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

Code:

```
SELECT
p.payment_installments,
COUNT(*) as no_of_orders
FROM `Target_SQL_Business_Case.orders` o
JOIN `Target_SQL_Business_Case.payments`p
ON o.order_id = p.order_id
WHERE p.payment_installments > 0
GROUP BY p.payment_installments
ORDER BY payment_installments
LIMIT 10;
```



Row	payment_installment	no_of_orders ▼
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	5	5239
6	6	3920
7	7	1626
8	8	4268
9	9	644
10	10	5328

Insights :-

- First installment has recorded maximum number of orders indicating a lot of new orders have been placed.