Business Case

TARGET SQL

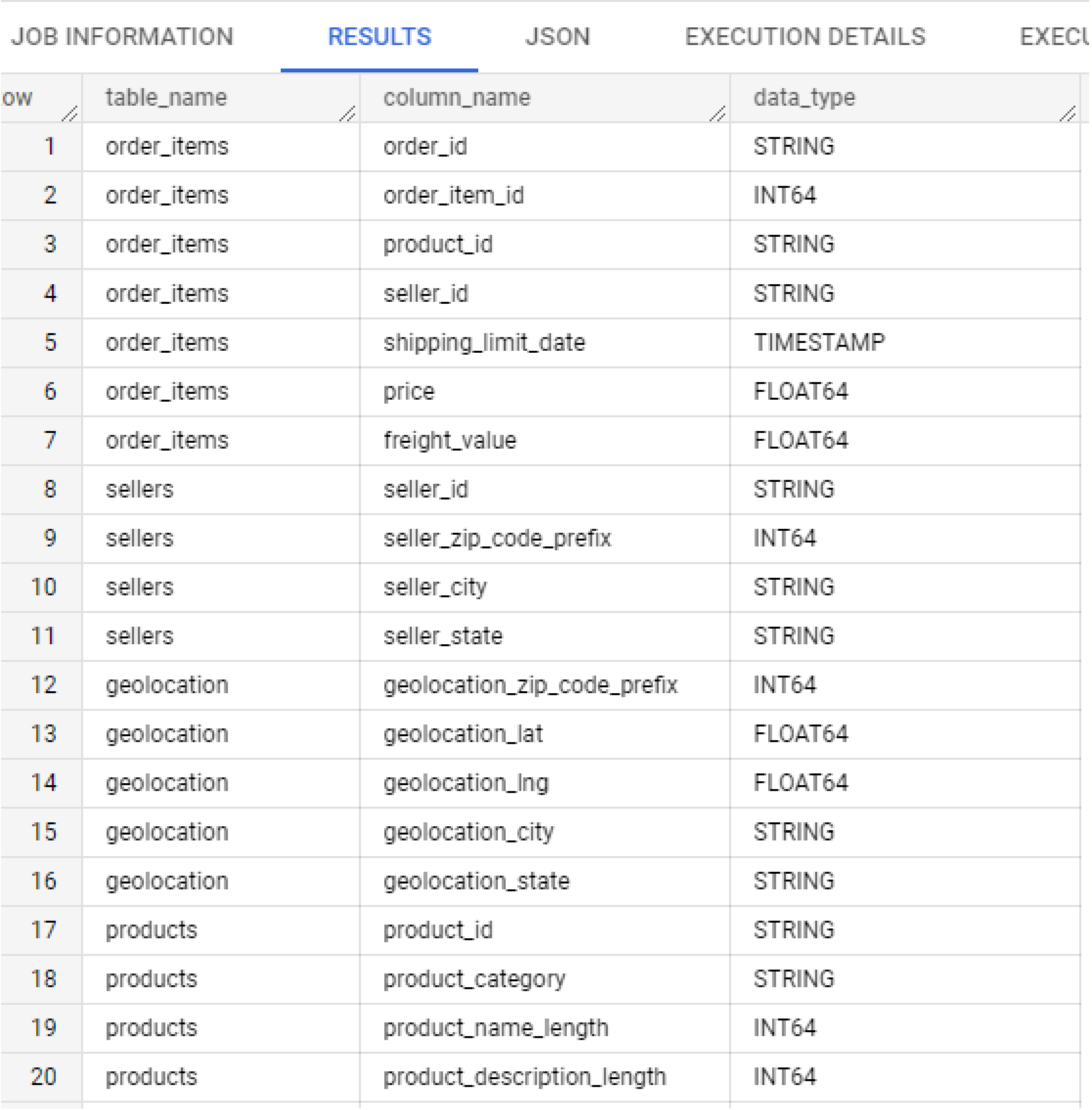
Name: **Iraban Dutta**
Email: **irabandutta.2020@gmail.com**
Github: **https://github.com/iraban-dutta**

1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset
   1.1. Data type of columns in a table

SQL Query: *Show all columns in each table along with their datatype in schema*

```
SELECT table_name, column_name, data_type
FROM `target.INFORMATION_SCHEMA.COLUMNS`
```

Result:



ow	table_name	column_name	data_type
1	order_items	order_id	STRING
2	order_items	order_item_id	INT64
3	order_items	product_id	STRING
4	order_items	seller_id	STRING
5	order_items	shipping_limit_date	TIMESTAMP
6	order_items	price	FLOAT64
7	order_items	freight_value	FLOAT64
8	sellers	seller_id	STRING
9	sellers	seller_zip_code_prefix	INT64
10	sellers	seller_city	STRING
11	sellers	seller_state	STRING
12	geolocation	geolocation_zip_code_prefix	INT64
13	geolocation	geolocation_lat	FLOAT64
14	geolocation	geolocation_lng	FLOAT64
15	geolocation	geolocation_city	STRING
16	geolocation	geolocation_state	STRING
17	products	product_id	STRING
18	products	product_category	STRING
19	products	product_name_length	INT64
20	products	product_description_length	INT64

(Note: Only first 20 rows snapshot is shown)

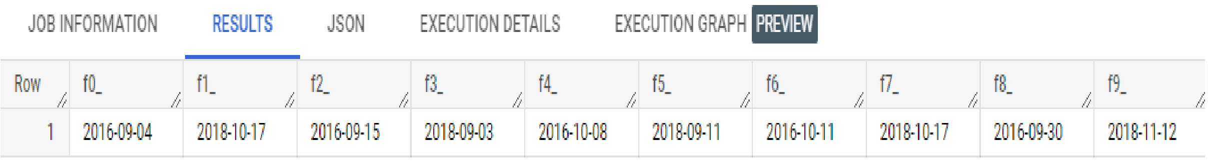
1.2. Time period for which data is given

To get a sense of the timeframe of the data in the dataset, we inspect those attributes whose data type is TIMESTAMP. Here is a list of all such attributes from the orders table.

<u>SQL Query:</u> *Getting min and max dates for particular columns from orders table*

```
SELECT
  MIN(DATE(order_purchase_timestamp)), MAX(DATE(order_purchase_timestamp)),
  MIN(DATE(order_approved_at)), MAX(DATE(order_approved_at)),
  MIN(DATE(order_delivered_carrier_date)),
MAX(DATE(order_delivered_carrier_date)),
  MIN(DATE(order_delivered_customer_date)),
MAX(DATE(order_delivered_customer_date)),
  MIN(DATE(order_estimated_delivery_date)),
MAX(DATE(order_estimated_delivery_date))
FROM `target.orders`;
```

<u>Result:</u>



Row	f0_	f1_	f2_	f3_	f4_	f5_	f6_	f7_	f8_	f9_
1	2016-09-04	2018-10-17	2016-09-15	2018-09-03	2016-10-08	2018-09-11	2016-10-11	2018-10-17	2016-09-30	2018-11-12

Table name	**Column name**	**Time-frame**	
		Start	**End**
orders	order_purchase_timestamp	2016-09-04	2018-10-17
	order_approved_at	2016-09-15	2018-09-03
	order_delivered_carrier_date	2016-10-08	2018-09-11
	order_delivered_customer_date	2016-10-11	2018-10-17
	order_estimated_delivery_date	2016-09-30	2018-11-12

1.3. Cities and states covered in dataset

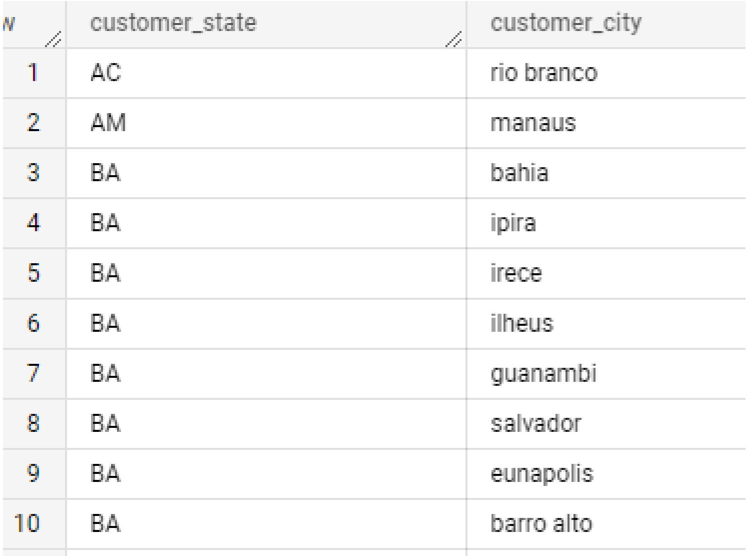
To get a sense of the different cities and states present in the dataset, we try to find all unique pairs of states and cities from the following tables:

- customers
- sellers

SQL Query: *Getting the names of unique state-city pairs from customers and sellers table*

```
SELECT
  DISTINCT customer_state, customer_city
FROM `target.customers`
UNION DISTINCT (
  SELECT
    DISTINCT seller_state, seller_city
  FROM `target.sellers`
);
```

Result:



w	customer_state	customer_city
1	AC	rio branco
2	AM	manaus
3	BA	bahia
4	BA	ipira
5	BA	irece
6	BA	ilheus
7	BA	guanambi
8	BA	salvador
9	BA	eunapolis
10	BA	barro alto

2. In-depth Exploration:

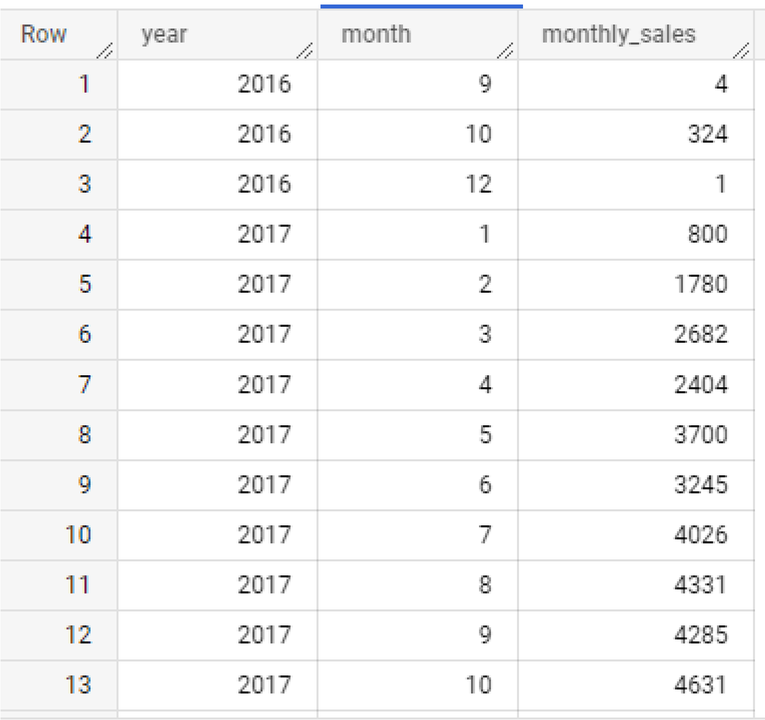
2.1. Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?

We want to find the total number of orders placed month by month over the given timeframe of the data.

SQL Query:

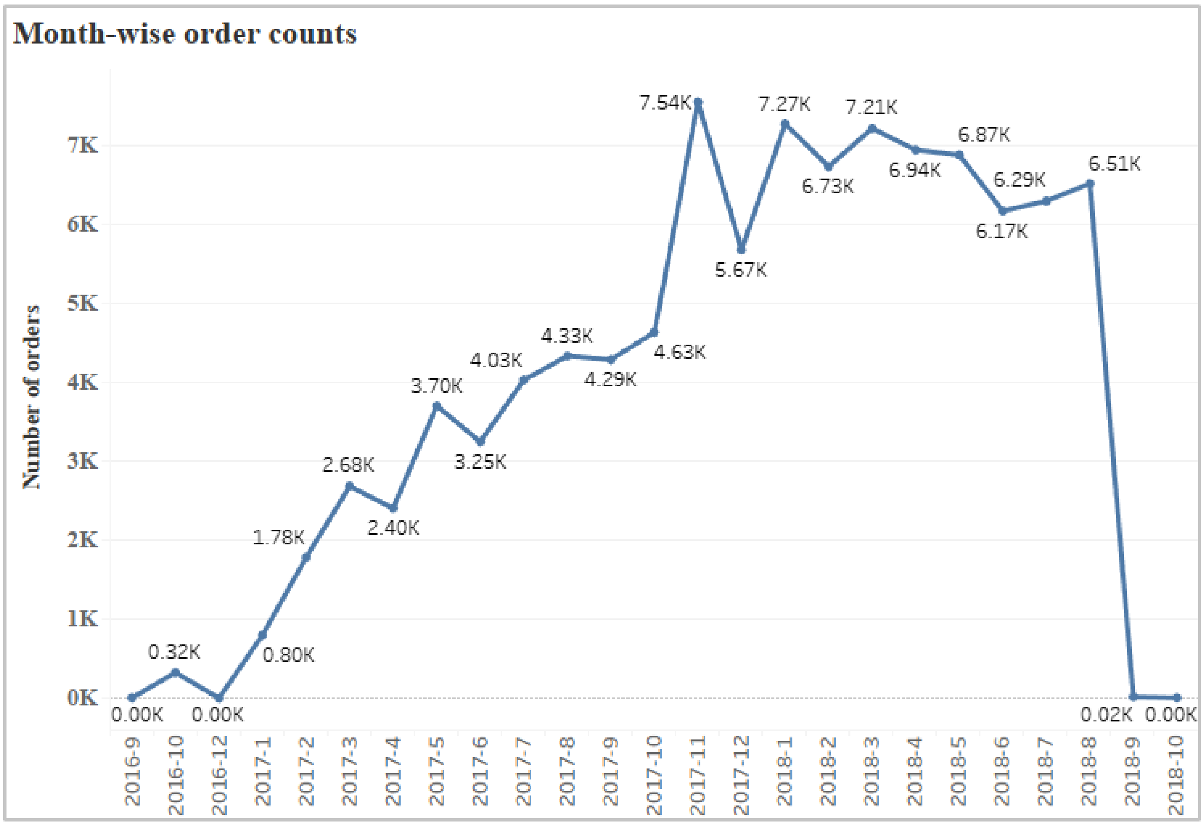
```
WITH base1 AS (
  SELECT
    *, EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
    EXTRACT(MONTH FROM order_purchase_timestamp) AS month
  FROM `target.orders`
)
SELECT year, month, COUNT(*) AS monthly_sales
FROM base1
GROUP BY 1, 2 ORDER BY 1, 2;
```

Result:



Row	year	month	monthly_sales
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
11	2017	8	4331
12	2017	9	4285
13	2017	10	4631

Is there growing trend on e-commerece in Brazil? Any seasonal impact on orders?



From the above figure, we can make the following observations:

- **Trends**
  - The number of orders increases steadily throughout 2017
  - The growth in order count remians somewhat stagnant in 2018
    - In fact we see a slow decline across Q2 of 2018 (Apr, May, June)
    - This followed by a slow recovery over the next quarter of 2018 (July, August)
  - The last 2 months of 2018 (Sep, Oct) in the above plot can be ignored since there is an abnormal decline in the values and we should not infer from such anomalies without knowing the proper cause for the same.
- **Seasonality**
  - November seems to be the peak season as we can cleary observe that the number of orders sharply increases in November. This may be due to the festivities in relation to Thanksgiving Day

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

We divide the entire 24 hours in a day into 4 equal slots (as shown below) and find the count of orders that were ordered per slot over the entire time frame. We also find what percentage of the total orders were ordered per slot.

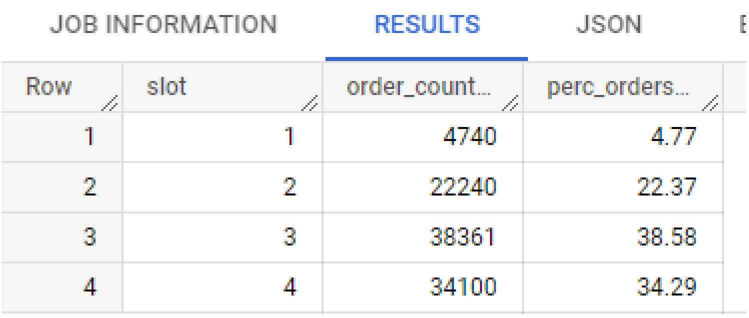
Slots:

- Dawn : 12:00AM to 5:59AM
- Morning: 6:00AM to 11:59AM
- Afternoon: 12:00PM to 5:59PM
- Night: 6:00PM to 11:59PM

SQL Query:

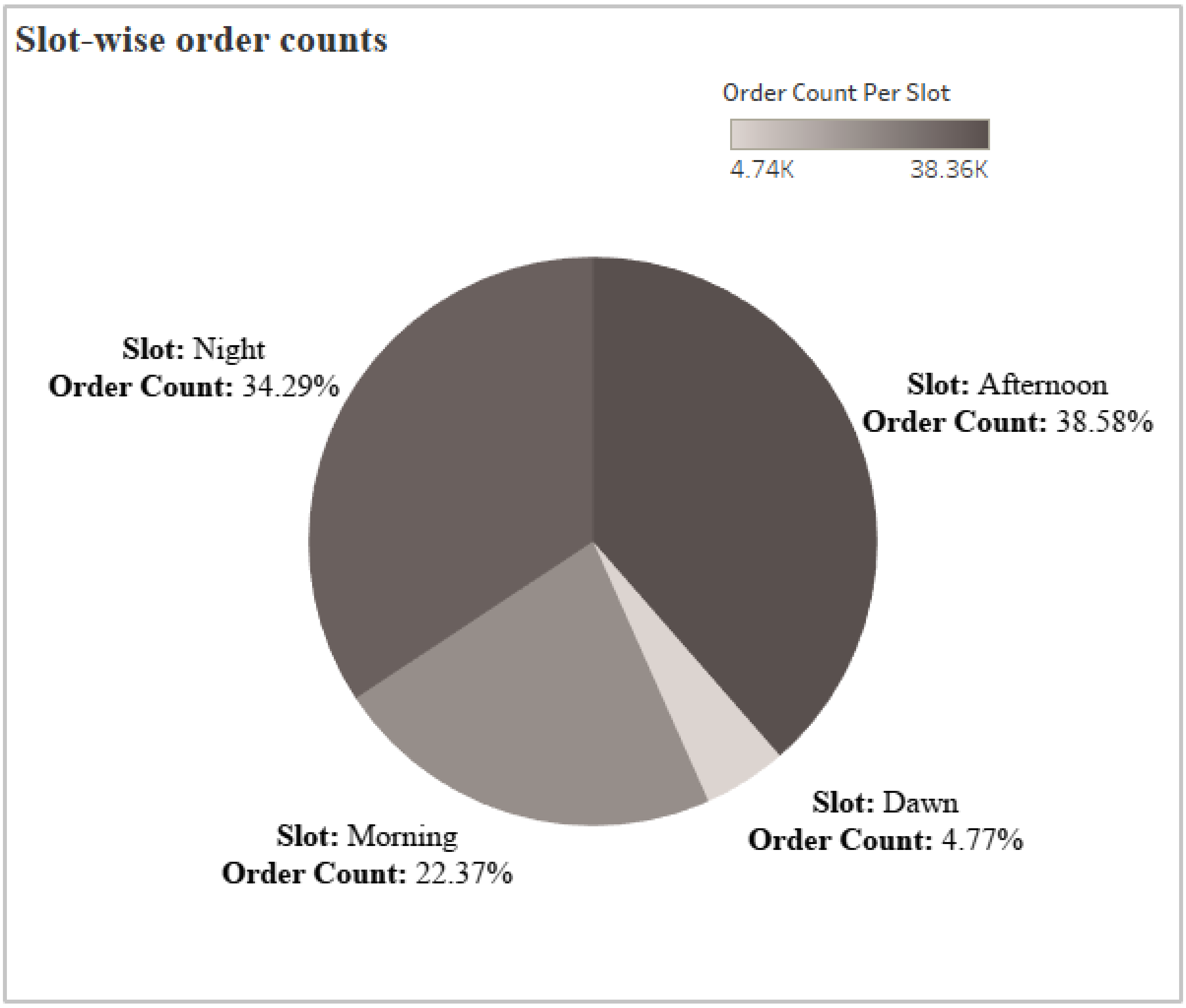
```
WITH base1 AS (
  SELECT
    *, EXTRACT(HOUR FROM order_purchase_timestamp) AS hour
  FROM `target.orders`
),
base2 AS (
  SELECT
    *,
    CASE
      WHEN hour BETWEEN 0 AND 5 THEN 1
      WHEN hour BETWEEN 6 AND 11 THEN 2
      WHEN hour BETWEEN 12 AND 17 THEN 3
      WHEN hour BETWEEN 18 AND 23 THEN 4
      ELSE 100
    END AS slot
  FROM base1
)
SELECT
  slot, COUNT(*) AS order_count_per_slot,
  ROUND(100 * (COUNT(*) / 99441), 2) AS perc_orders_per_slot
FROM base2
GROUP BY 1 ORDER BY 1;
```

Result:



Row	slot	order_count...	perc_orders...
1	1	4740	4.77
2	2	22240	22.37
3	3	38361	38.58
4	4	34100	34.29

Observations regarding time-slot where Brazilians tend to purchase the most:



From the above figure, we can make the following observations:

- The sales during Afternoon (12:00PM to 5:59PM) is the highest
- The sales during Dawn (12:00AM to 5:59AM) is the lowest

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

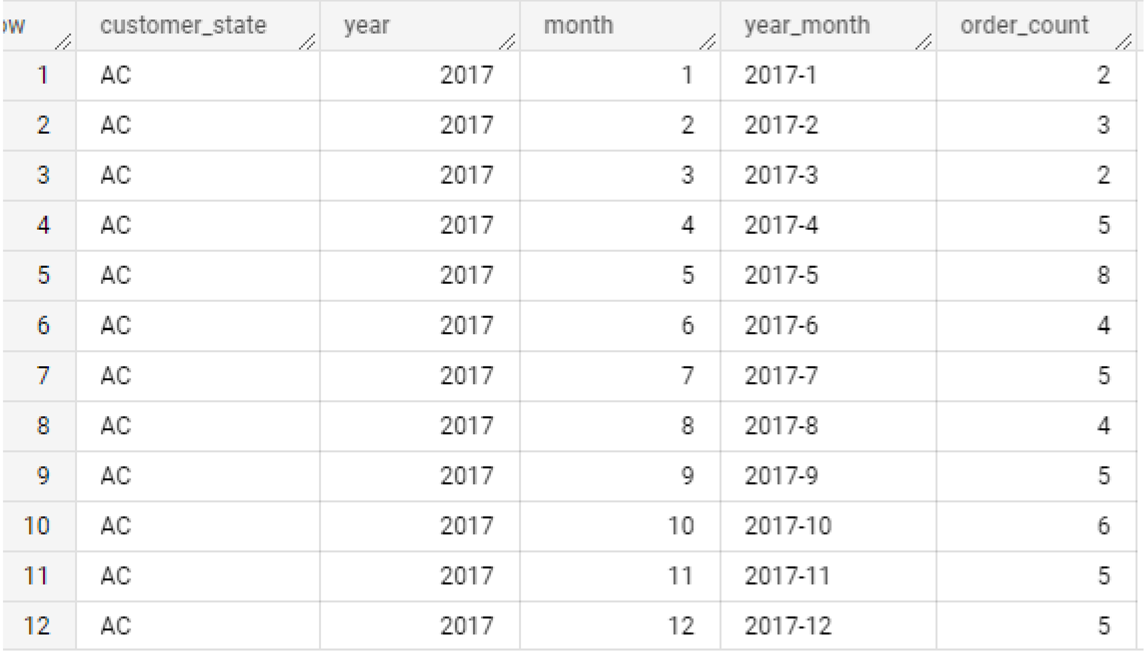
3.1. Get month on month orders by region, states

We will restrict the timeframe of the data from Jan-2017 to Aug-2018. Our aim is to find out the total number of orders placed over time (month by month) across different states in Brazil.

SQL Query:

```
WITH base1 AS (
  SELECT o.order_id, o.customer_id, o.order_purchase_timestamp,
    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
    c.customer_unique_id, c.customer_state, c.customer_city
  FROM `target.orders` o JOIN `target.customers`c ON o.customer_id = c.customer_id
)
SELECT
  customer_state, year, month, CONCAT(year, '-', month) AS year_month,
  COUNT(*) AS order_count
FROM base1
WHERE
  (year = 2017 AND month BETWEEN 1 AND 12) OR
  (year = 2018 AND month BETWEEN 1 AND 8)
GROUP BY 1, 2, 3 ORDER BY customer_state, year, month
```

Results:



ow	customer_state	year	month	year_month	order_count
1	AC	2017	1	2017-1	2
2	AC	2017	2	2017-2	3
3	AC	2017	3	2017-3	2
4	AC	2017	4	2017-4	5
5	AC	2017	5	2017-5	8
6	AC	2017	6	2017-6	4
7	AC	2017	7	2017-7	5
8	AC	2017	8	2017-8	4
9	AC	2017	9	2017-9	5
10	AC	2017	10	2017-10	6
11	AC	2017	11	2017-11	5
12	AC	2017	12	2017-12	5

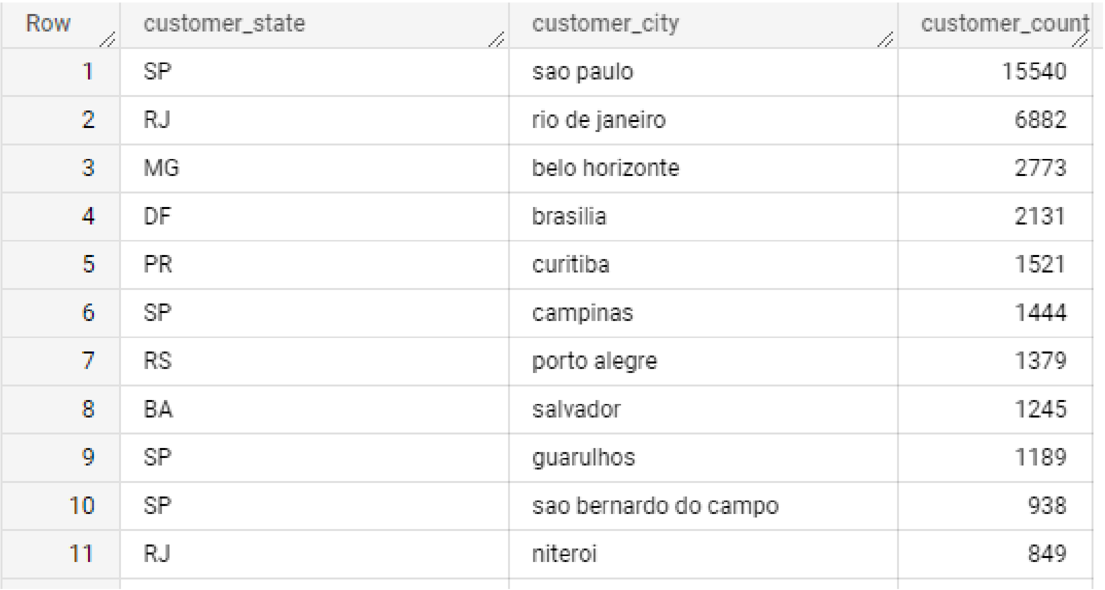
3.2. How are customers distributed in Brazil

Here we want to show the count of customers in every unique pair of states and cities in Brazil. We have ordered the table in descending order of customer count.

SQL Query:

```
SELECT
  customer_state, customer_city,COUNT(*) AS customer_count
FROM `target.customers`
GROUP BY 1, 2 ORDER BY COUNT(*) DESC
```

Result:



Row	customer_state	customer_city	customer_count
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
11	RJ	niteroi	849

As evident from the above result, the leading 5 cities in terms of customer counts in Brazil are as follows:

1. São Paulo, SP
2. Rio de Janeiro, RJ
3. Belo Horizonte, MG
4. Brasilia, DF
5. Curitiba, PR

4. Impact on Economy: Analyze the money moved by e-commerce by looking at order prices, freight and others.
    4.1. Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only)

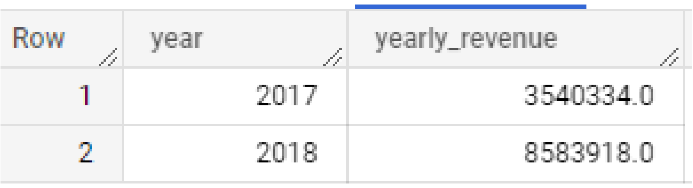
We will find the % increase in the cost of orders from 2017 to 2018 (Jan-Aug) over:

1. Yearly basis
2. Monthly basis

__SQL Query:__ *Yearly basis*

```
WITH base1 AS (
  SELECT order_id, ROUND(SUM(payment_value), 2) AS total_order_price
  FROM `target.payments`
  WHERE payment_value > 0 GROUP BY 1
),
base2 AS (
  SELECT * FROM `target.orders`
  WHERE order_status IN ('invoiced', 'shipped', 'delivered')
),
base3 AS (
  SELECT b1.*, b2.order_purchase_timestamp,
    EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
    EXTRACT(MONTH FROM order_purchase_timestamp) AS month
  FROM base1 b1 JOIN base2 b2 ON b1.order_id = b2.order_id
)
SELECT year, ROUND(SUM(total_order_price), 0) AS yearly_revenue
FROM base3
WHERE
    (year = 2017 AND month BETWEEN 1 AND 8) OR
    (year = 2018 AND month BETWEEN 1 AND 8)
GROUP BY 1 ORDER BY year
```

__Result:__



Row	year	yearly_revenue
1	2017	3540334.0
2	2018	8583918.0

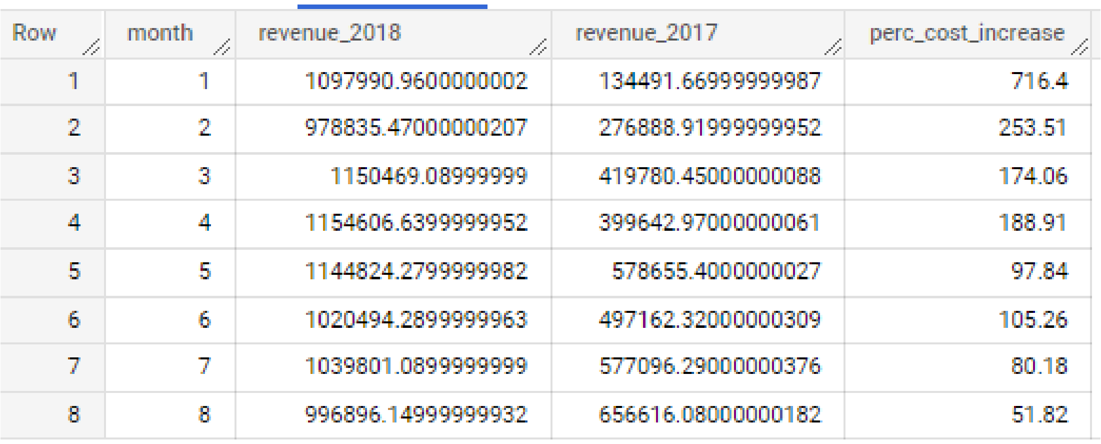
<u>From the above data, we can make the following comment:</u>

- The total cost of orders increased from about 3.5M in 2017 to 8.6M in 2018
- Thus % increase in total cost of orders from 2017 to 2018: **+142.46%**

SQL Query: *Monthly basis*

```
WITH base1 AS (
  SELECT order_id, ROUND(SUM(payment_value), 2) AS total_order_price
  FROM `target.payments`
  WHERE payment_value > 0 GROUP BY 1
),
base2 AS (
  SELECT * FROM `target.orders`
  WHERE order_status IN ('invoiced', 'shipped', 'delivered')
),
base3 AS (
  SELECT b1.*, b2.order_purchase_timestamp,
    EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
    EXTRACT(MONTH FROM order_purchase_timestamp) AS month
  FROM base1 b1 JOIN base2 b2 ON b1.order_id = b2.order_id
),
base4 AS (
  SELECT month, year, SUM(total_order_price) AS mon_revenue
  FROM base3
  WHERE
    (year = 2017 AND month BETWEEN 1 AND 8) OR
    (year = 2018 AND month BETWEEN 1 AND 8)
  GROUP BY 1, 2 ORDER BY month, year
),
base5 AS (
  SELECT *,
    LAG(mon_revenue) OVER(PARTITION BY month ORDER BY year) AS prev_revenue
  FROM base4 ORDER BY month
)
SELECT month, mon_revenue AS revenue_2018, prev_revenue AS revenue_2017,
  ROUND(100 * ((mon_revenue - prev_revenue) / prev_revenue), 2) AS
  perc_cost_increase
FROM base5 WHERE year = 2018
```

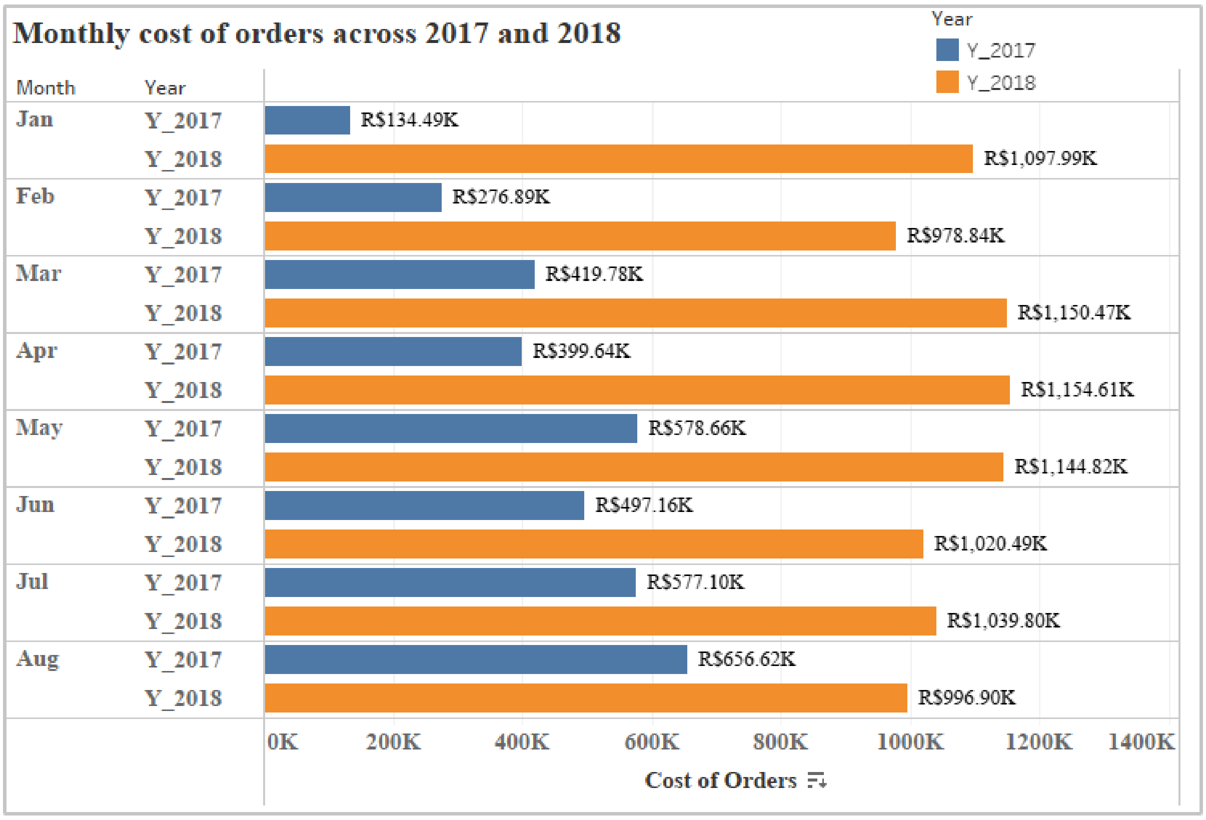
Result:

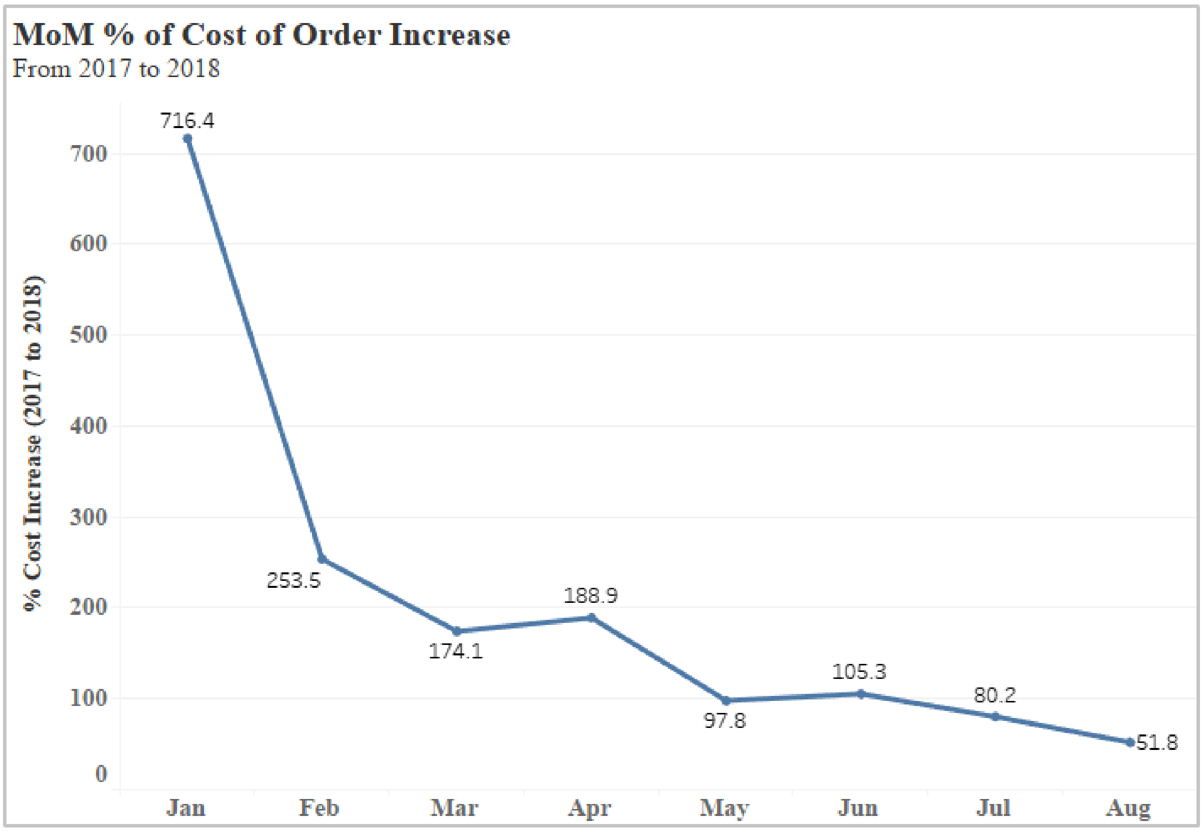


Row	month	revenue_2018	revenue_2017	perc_cost_increase
1	1	1097990.9600000002	134491.66999999987	716.4
2	2	978835.47000000207	276888.91999999952	253.51
3	3	1150469.08999999	419780.45000000088	174.06
4	4	1154606.6399999952	399642.97000000061	188.91
5	5	1144824.2799999982	578655.4000000027	97.84
6	6	1020494.2899999963	497162.32000000309	105.26
7	7	1039801.0899999999	577096.29000000376	80.18
8	8	996896.14999999932	656616.08000000182	51.82

Following observations can be drawn from the above data:

- The % increase in the month-wise cost of orders from 2017 to 2018 reduces sharply from January to August


Monthly cost of orders across 2017 and 2018
Year
Y_2017
Y_2018
Month
Year
Jan
Y_2017
R$134.49K
Y_2018
R$1,097.99K
Feb
Y_2017
R$276.89K
Y_2018
R$978.84K
Mar
Y_2017
R$419.78K
Y_2018
R$1,150.47K
Apr
Y_2017
R$399.64K
Y_2018
R$1,154.61K
May
Y_2017
R$578.66K
Y_2018
R$1,144.82K
Jun
Y_2017
R$497.16K
Y_2018
R$1,020.49K
Jul
Y_2017
R$577.10K
Y_2018
R$1,039.80K
Aug
Y_2017
R$656.62K
Y_2018
R$996.90K
0K
200K
400K
600K
800K
1000K
1200K
1400K
Cost of Orders

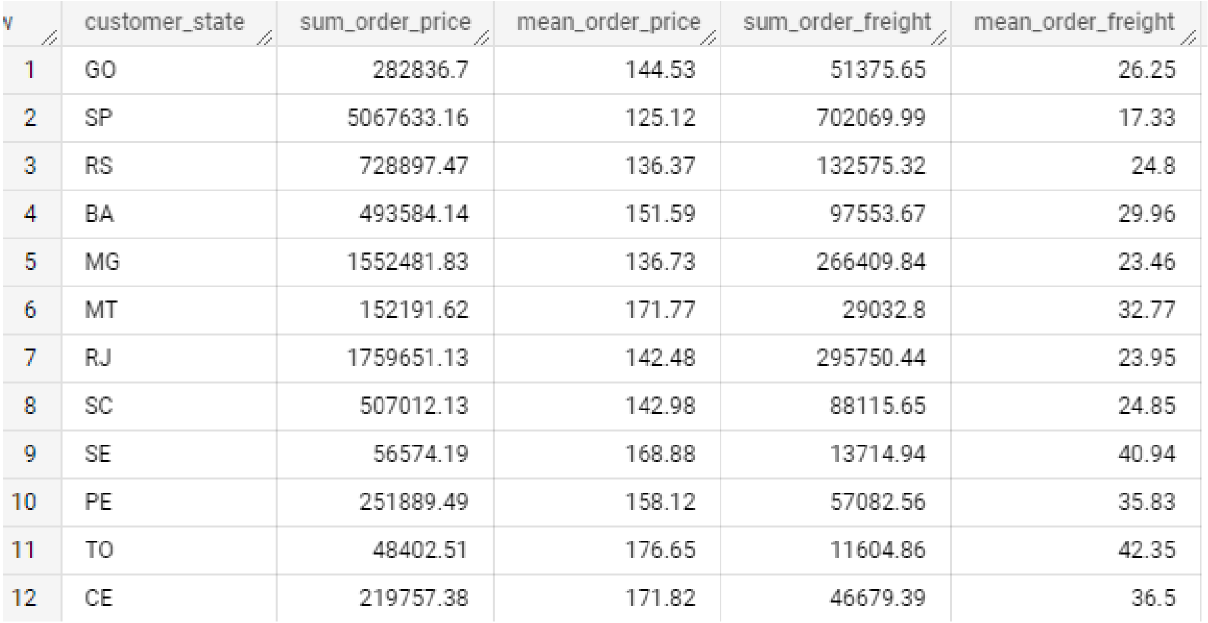

MoM % of Cost of Order Increase
From 2017 to 2018
% Cost Increase (2017 to 2018)
0
100
200
300
400
500
600
700
716.4
253.5
174.1
188.9
97.8
105.3
80.2
51.8
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug

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

SQL Query:

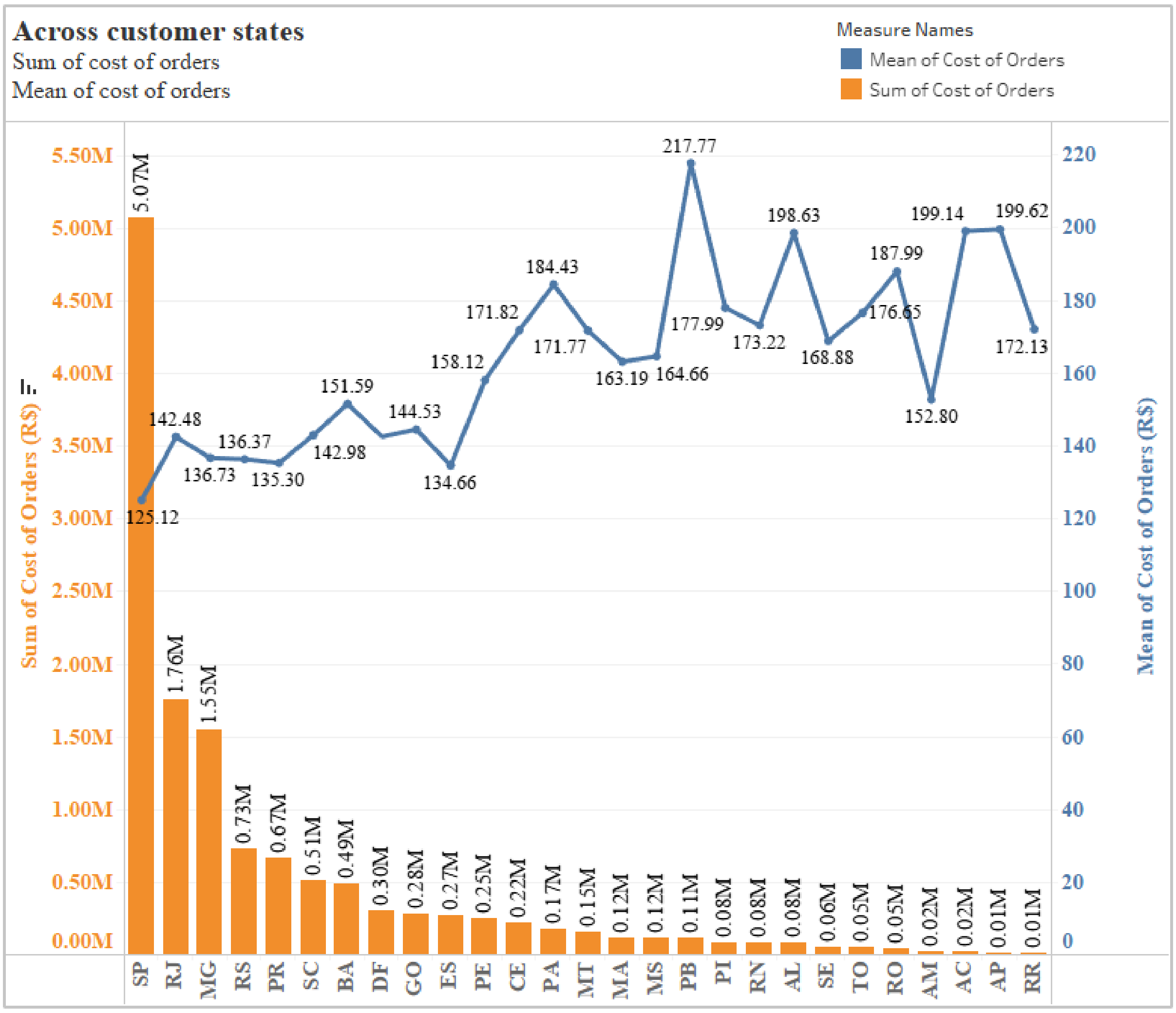
```
WITH base1 AS (
  SELECT
    order_id, SUM(price) AS order_price,
    SUM(freight_value) AS order_freight_value
  FROM `target.order_items` GROUP BY 1
),
base2 AS (
  SELECT * FROM `target.orders` WHERE order_status = 'delivered'
)
SELECT
  c.customer_state,
  ROUND(SUM(order_price), 2) AS sum_order_price,
  ROUND(AVG(order_price), 2) AS mean_order_price,
  ROUND(SUM(order_freight_value), 2) AS sum_order_freight,
  ROUND(AVG(order_freight_value), 2) AS mean_order_freight,
FROM
  base1 b1 JOIN base2 b2 ON b1.order_id = b2.order_id
  JOIN `target.customers`c ON b2.customer_id = c.customer_id
GROUP BY 1
```

Result:



v	customer_state	sum_order_price	mean_order_price	sum_order_freight	mean_order_freight
1	GO	282836.7	144.53	51375.65	26.25
2	SP	5067633.16	125.12	702069.99	17.33
3	RS	728897.47	136.37	132575.32	24.8
4	BA	493584.14	151.59	97553.67	29.96
5	MG	1552481.83	136.73	266409.84	23.46
6	MT	152191.62	171.77	29032.8	32.77
7	RJ	1759651.13	142.48	295750.44	23.95
8	SC	507012.13	142.98	88115.65	24.85
9	SE	56574.19	168.88	13714.94	40.94
10	PE	251889.49	158.12	57082.56	35.83
11	TO	48402.51	176.65	11604.86	42.35
12	CE	219757.38	171.82	46679.39	36.5

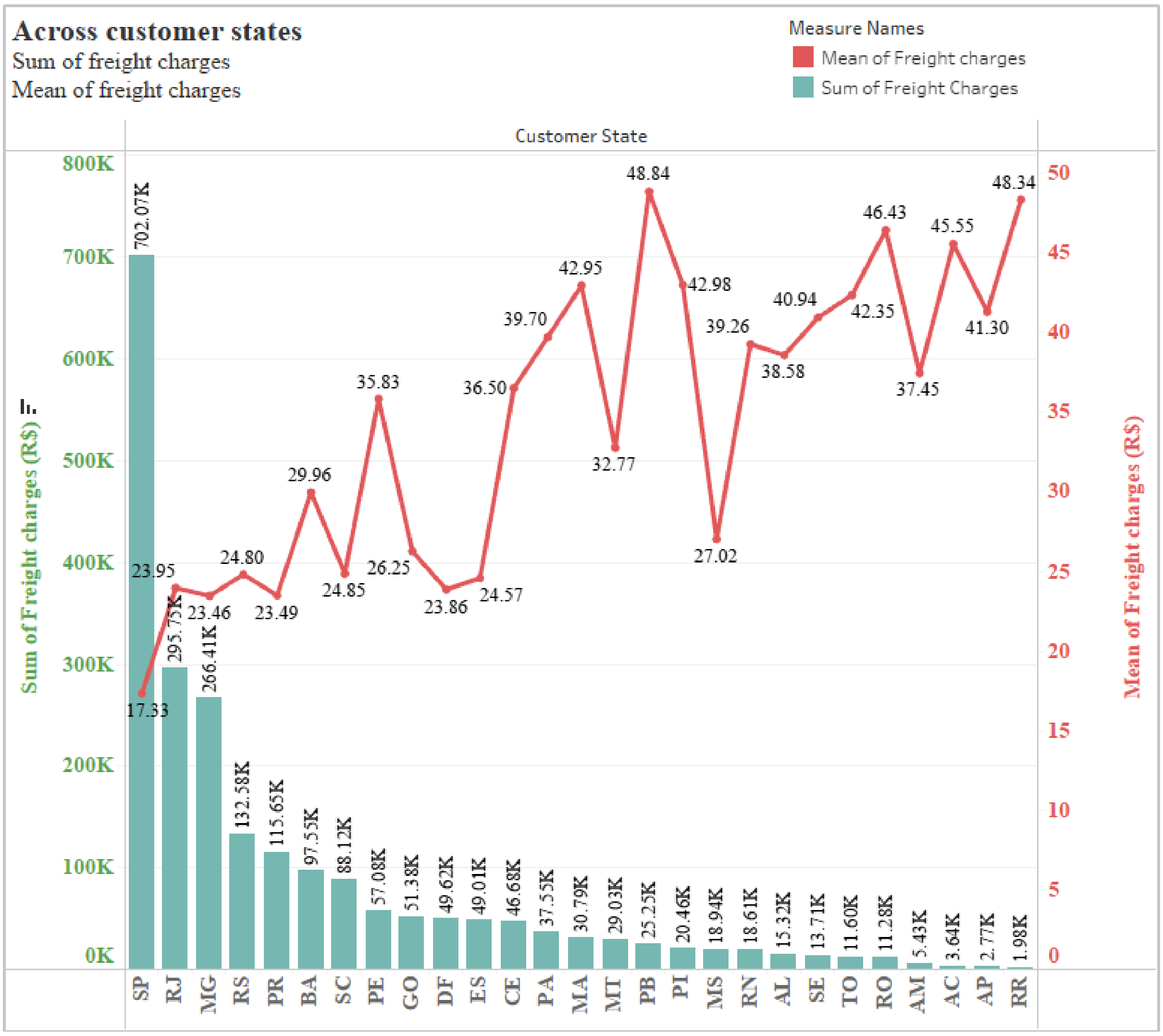
Sum and Mean of cost of oders across customer states:



<u>Following observations can be made from the above plot:</u>

- SP is the state which is the primary contributor to the revenue of Target
    - Interestingly it has the lowest mean out of all the states
    - This opens up to a possiblity of increasing the gross revenue by a huge factor by creating a small positive increment to the mean of cost of orders from SP

Sum and Mean of order-wise freight price across customer states:



<u>Following observations can be made from the above plot:</u>

- SP is the state with the highest total of order-wise freight values
  - Also it has the lowest mean out of all the states

5. Analysis on sales, freight and delivery time

5.1. Calculate days between purchasing, delivering and estimated delivery

5.2. Create columns:

- time_to_delivery = order_purchase_timestamp - order_delivered_customer_date
- diff_estimated_delivery = order_estimated_delivery_date - order_delivered_customer_date

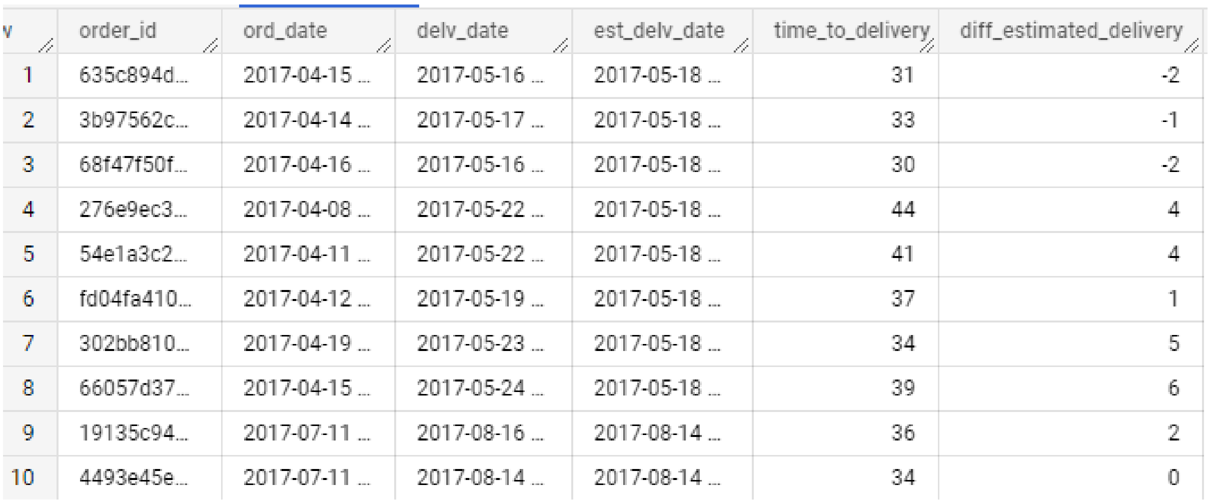
We will define the new columns as follows:

- **time_to_delivery:**
  - *order_delivered_customer_date* - *order_purchase_timestamp*
  - difference between the date of order delivery and that of order purchase date
  - higher the value, the longer it has taken from the order purchase date to reach the customer
- **diff_estimated_delivery:**
  - *order_delivered_customer_date* - *order_estimated_delivery_date*
  - difference between the date of order delivery and date of estimated delivery
  - Value of 0 means that order was delivered on the estimated delivery date
  - Negative value means order has arrived before estimate date and positive value means order arrives after delivery date

SQL Query:

```
WITH base1 AS
  (SELECT order_id, customer_id, order_purchase_timestamp AS ord_date,
    order_delivered_customer_date AS delv_date,
    order_estimated_delivery_date AS est_delv_date
  FROM `target.orders`
  WHERE order_status = 'delivered' AND order_delivered_customer_date IS NOT NULL)
SELECT order_id, ord_date, delv_date, est_delv_date,
  DATE_DIFF(DATE(delv_date), DATE(ord_date), DAY) AS time_to_delivery,
  DATE_DIFF(DATE(delv_date), DATE(est_delv_date), DAY) AS diff_estimated_delivery
FROM base1
```

Result:



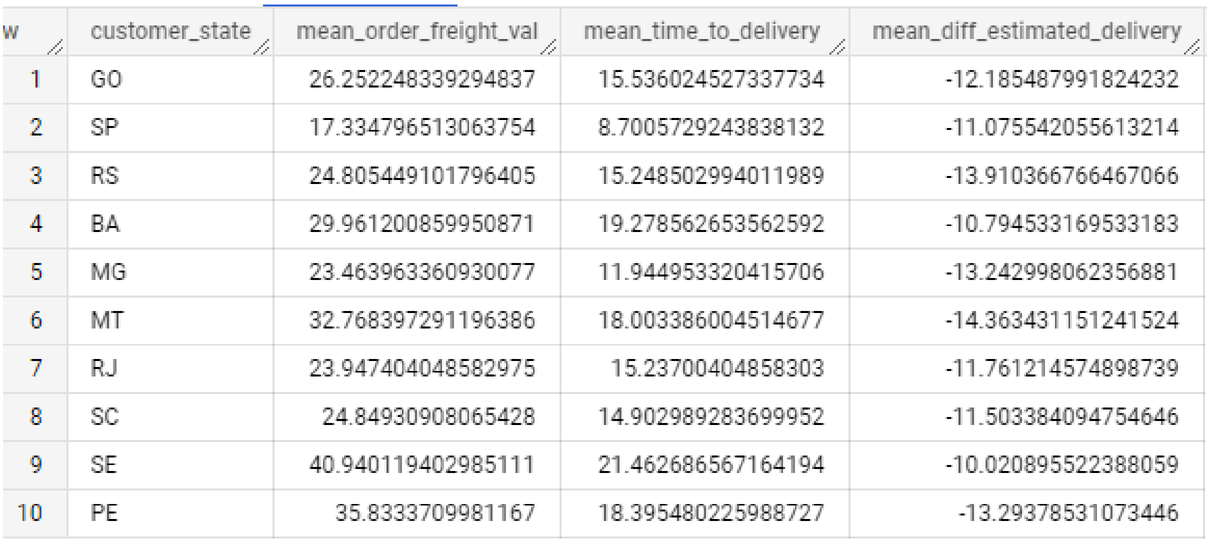
v	order_id	ord_date	delv_date	est_delv_date	time_to_delivery	diff_estimated_delivery
1	635c894d...	2017-04-15 ...	2017-05-16 ...	2017-05-18 ...	31	-2
2	3b97562c...	2017-04-14 ...	2017-05-17 ...	2017-05-18 ...	33	-1
3	68f47f50f...	2017-04-16 ...	2017-05-16 ...	2017-05-18 ...	30	-2
4	276e9ec3...	2017-04-08 ...	2017-05-22 ...	2017-05-18 ...	44	4
5	54e1a3c2...	2017-04-11 ...	2017-05-22 ...	2017-05-18 ...	41	4
6	fd04fa410...	2017-04-12 ...	2017-05-19 ...	2017-05-18 ...	37	1
7	302bb810...	2017-04-19 ...	2017-05-23 ...	2017-05-18 ...	34	5
8	66057d37...	2017-04-15 ...	2017-05-24 ...	2017-05-18 ...	39	6
9	19135c94...	2017-07-11 ...	2017-08-16 ...	2017-08-14 ...	36	2
10	4493e45e...	2017-07-11 ...	2017-08-14 ...	2017-08-14 ...	34	0

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

SQL Query:

```
WITH base1 AS (
  SELECT order_id, customer_id, order_purchase_timestamp AS ord_date,
    order_delivered_customer_date AS delv_date,
    order_estimated_delivery_date AS est_delv_date
  FROM `target.orders`
  WHERE order_status = 'delivered' AND order_delivered_customer_date IS NOT NULL
),
base2 AS (
  SELECT *, DATE_DIFF(DATE(delv_date), DATE(ord_date), DAY) AS time_to_delivery,
    DATE_DIFF(DATE(delv_date), DATE(est_delv_date), DAY) AS diff_estimated_delivery
  FROM base1
),
base3 AS (
  SELECT order_id, ROUND(SUM(freight_value), 2) AS order_freight_val
  FROM `target.order_items` GROUP BY 1
)
SELECT customer_state, AVG(order_freight_val) AS mean_order_freight_val,
  AVG(time_to_delivery) AS mean_time_to_delivery,
  AVG(diff_estimated_delivery) AS mean_diff_estimated_delivery
FROM
  base2 b2 JOIN base3 b3 ON b2.order_id = b3.order_id
  JOIN `target.customers` c ON b2.customer_id = c.customer_id
GROUP BY 1
```

Result:



w	customer_state	mean_order_freight_val	mean_time_to_delivery	mean_diff_estimated_delivery
1	GO	26.252248339294837	15.536024527337734	-12.185487991824232
2	SP	17.334796513063754	8.7005729243838132	-11.075542055613214
3	RS	24.805449101796405	15.248502994011989	-13.910366766467066
4	BA	29.961200859950871	19.278562653562592	-10.794533169533183
5	MG	23.463963360930077	11.944953320415706	-13.242998062356881
6	MT	32.768397291196386	18.003386004514677	-14.363431151241524
7	RJ	23.947404048582975	15.23700404858303	-11.761214574898739
8	SC	24.84930908065428	14.902989283699952	-11.503384094754646
9	SE	40.940119402985111	21.462686567164194	-10.020895522388059
10	PE	35.8333709981167	18.395480225988727	-13.29378531073446

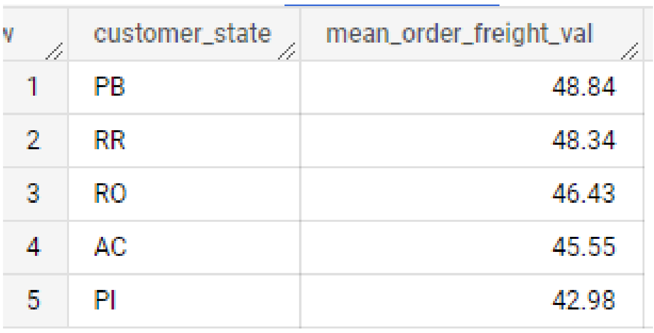
5.4. Sort the data to get the following:

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

SQL Query: *Top 5 states with highest mean freight_value*

```
WITH base1 AS (
  SELECT order_id, customer_id, order_purchase_timestamp AS ord_date,
    order_delivered_customer_date AS delv_date,
    order_estimated_delivery_date AS est_delv_date
  FROM `target.orders`
  WHERE order_status = 'delivered' AND order_delivered_customer_date IS NOT NULL
),
base2 AS (
  SELECT *, DATE_DIFF(DATE(delv_date), DATE(ord_date), DAY) AS time_to_delivery,
    DATE_DIFF(DATE(delv_date), DATE(est_delv_date), DAY) AS diff_estimated_delivery
  FROM base1
),
base3 AS (
  SELECT order_id, ROUND(SUM(freight_value), 2) AS order_freight_val
  FROM `target.order_items` GROUP BY 1
)
SELECT customer_state, ROUND(AVG(order_freight_val), 2) AS mean_order_freight_val
FROM
  base2 b2 JOIN base3 b3 ON b2.order_id = b3.order_id
  JOIN `target.customers` c ON b2.customer_id = c.customer_id
GROUP BY 1 ORDER BY mean_order_freight_val DESC LIMIT 5
```

Result:

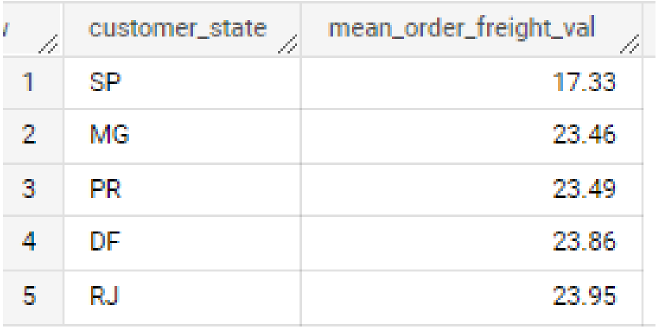


v	customer_state	mean_order_freight_val
1	PB	48.84
2	RR	48.34
3	RO	46.43
4	AC	45.55
5	PI	42.98

SQL Query: *Top 5 states with lowest mean freight_value*

```
WITH base1 AS (
  SELECT order_id, customer_id, order_purchase_timestamp AS ord_date,
    order_delivered_customer_date AS delv_date,
    order_estimated_delivery_date AS est_delv_date
  FROM `target.orders`
  WHERE order_status = 'delivered' AND order_delivered_customer_date IS NOT NULL
),
base2 AS (
  SELECT *, DATE_DIFF(DATE(delv_date), DATE(ord_date), DAY) AS time_to_delivery,
    DATE_DIFF(DATE(delv_date), DATE(est_delv_date), DAY) AS diff_estimated_delivery
  FROM base1
),
base3 AS (
  SELECT order_id, ROUND(SUM(freight_value), 2) AS order_freight_val
  FROM `target.order_items` GROUP BY 1
)
SELECT customer_state, ROUND(AVG(order_freight_val), 2) AS mean_order_freight_val
FROM
  base2 b2 JOIN base3 b3 ON b2.order_id = b3.order_id
  JOIN `target.customers` c ON b2.customer_id = c.customer_id
GROUP BY 1 ORDER BY mean_order_freight_val LIMIT 5
```

Result:



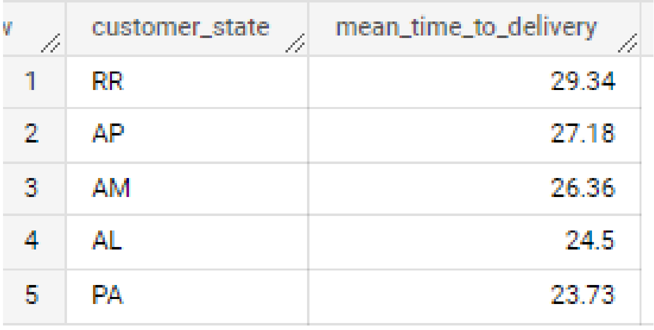
	customer_state	mean_order_freight_val
1	SP	17.33
2	MG	23.46
3	PR	23.49
4	DF	23.86
5	RJ	23.95

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

SQL Query: *Top 5 states with highest mean time_to_delivery (Longer delivery time)*

```
WITH base1 AS (
  SELECT order_id, customer_id, order_purchase_timestamp AS ord_date,
    order_delivered_customer_date AS delv_date,
    order_estimated_delivery_date AS est_delv_date
  FROM `target.orders`
  WHERE order_status = 'delivered' AND order_delivered_customer_date IS NOT NULL
),
base2 AS (
  SELECT *, DATE_DIFF(DATE(delv_date), DATE(ord_date), DAY) AS time_to_delivery,
    DATE_DIFF(DATE(delv_date), DATE(est_delv_date), DAY) AS diff_estimated_delivery
  FROM base1
),
base3 AS (
  SELECT order_id, ROUND(SUM(freight_value), 2) AS order_freight_val
  FROM `target.order_items` GROUP BY 1
)
SELECT customer_state, ROUND(AVG(time_to_delivery), 2) AS mean_time_to_delivery
FROM
  base2 b2 JOIN base3 b3 ON b2.order_id = b3.order_id
  JOIN `target.customers` c ON b2.customer_id = c.customer_id
GROUP BY 1 ORDER BY mean_time_to_delivery DESC LIMIT 5
```

Result:

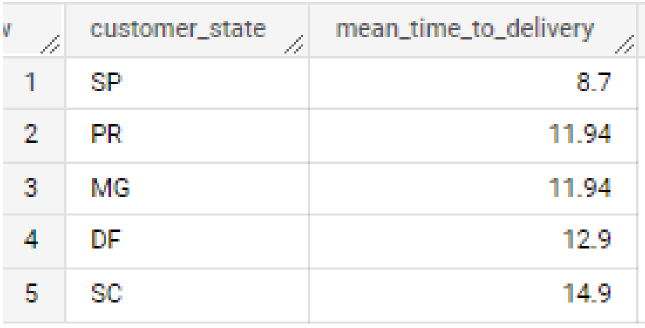


v	customer_state	mean_time_to_delivery
1	RR	29.34
2	AP	27.18
3	AM	26.36
4	AL	24.5
5	PA	23.73

<u>SQL Query</u>: *Top 5 states with lowest mean time_to_delivery (Shortest delivery time)*

```
WITH base1 AS (
  SELECT order_id, customer_id, order_purchase_timestamp AS ord_date,
    order_delivered_customer_date AS delv_date,
    order_estimated_delivery_date AS est_delv_date
  FROM `target.orders`
  WHERE order_status = 'delivered' AND order_delivered_customer_date IS NOT NULL
),
base2 AS (
  SELECT *, DATE_DIFF(DATE(delv_date), DATE(ord_date), DAY) AS time_to_delivery,
    DATE_DIFF(DATE(delv_date), DATE(est_delv_date), DAY) AS diff_estimated_delivery
  FROM base1
),
base3 AS (
  SELECT order_id, ROUND(SUM(freight_value), 2) AS order_freight_val
  FROM `target.order_items` GROUP BY 1
)
SELECT customer_state, ROUND(AVG(time_to_delivery), 2) AS mean_time_to_delivery
FROM
  base2 b2 JOIN base3 b3 ON b2.order_id = b3.order_id
  JOIN `target.customers` c ON b2.customer_id = c.customer_id
GROUP BY 1 ORDER BY mean_time_to_delivery LIMIT 5
```

<u>Result:</u>



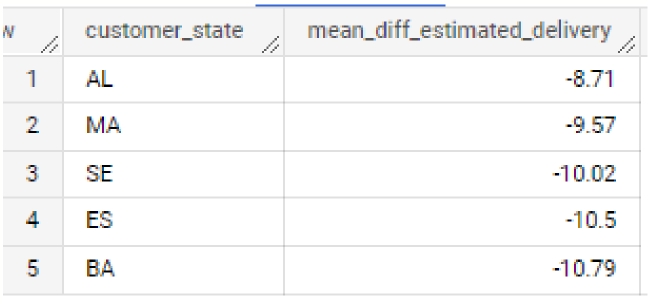
	customer_state	mean_time_to_delivery
1	SP	8.7
2	PR	11.94
3	MG	11.94
4	DF	12.9
5	SC	14.9

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

SQL Query: *Top 5 states with highest mean diff_estimated_delivery (Late delivery!!)*

```
WITH base1 AS (
  SELECT order_id, customer_id, order_purchase_timestamp AS ord_date,
    order_delivered_customer_date AS delv_date,
    order_estimated_delivery_date AS est_delv_date
  FROM `target.orders`
  WHERE order_status = 'delivered' AND order_delivered_customer_date IS NOT NULL
),
base2 AS (
  SELECT *, DATE_DIFF(DATE(delv_date), DATE(ord_date), DAY) AS time_to_delivery,
    DATE_DIFF(DATE(delv_date), DATE(est_delv_date), DAY) AS diff_estimated_delivery
  FROM base1
),
base3 AS (
  SELECT order_id, ROUND(SUM(freight_value), 2) AS order_freight_val
  FROM `target.order_items` GROUP BY 1
)
SELECT customer_state,
  ROUND(AVG(diff_estimated_delivery), 2) AS mean_diff_estimated_delivery
FROM
  base2 b2 JOIN base3 b3 ON b2.order_id = b3.order_id
  JOIN `target.customers` c ON b2.customer_id = c.customer_id
GROUP BY 1 ORDER BY mean_diff_estimated_delivery DESC LIMIT 5
```

Result:

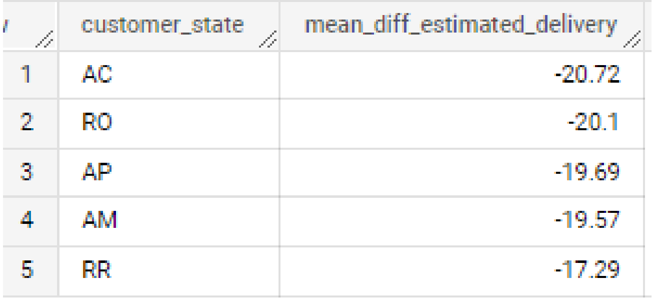


w	customer_state	mean_diff_estimated_delivery
1	AL	-8.71
2	MA	-9.57
3	SE	-10.02
4	ES	-10.5
5	BA	-10.79

SQL Query: *Top 5 states with lowest mean diff_estimated_delivery (Early delivery!!)*

```
WITH base1 AS (
  SELECT order_id, customer_id, order_purchase_timestamp AS ord_date,
    order_delivered_customer_date AS delv_date,
    order_estimated_delivery_date AS est_delv_date
  FROM `target.orders`
  WHERE order_status = 'delivered' AND order_delivered_customer_date IS NOT NULL
),
base2 AS (
  SELECT *, DATE_DIFF(DATE(delv_date), DATE(ord_date), DAY) AS time_to_delivery,
    DATE_DIFF(DATE(delv_date), DATE(est_delv_date), DAY) AS diff_estimated_delivery
  FROM base1
),
base3 AS (
  SELECT order_id, ROUND(SUM(freight_value), 2) AS order_freight_val
  FROM `target.order_items` GROUP BY 1
)
SELECT customer_state,
  ROUND(AVG(diff_estimated_delivery), 2) AS mean_diff_estimated_delivery
FROM
  base2 b2 JOIN base3 b3 ON b2.order_id = b3.order_id
  JOIN `target.customers` c ON b2.customer_id = c.customer_id
GROUP BY 1 ORDER BY mean_diff_estimated_delivery LIMIT 5
```

Result:



/	customer_state	mean_diff_estimated_delivery
1	AC	-20.72
2	RO	-20.1
3	AP	-19.69
4	AM	-19.57
5	RR	-17.29

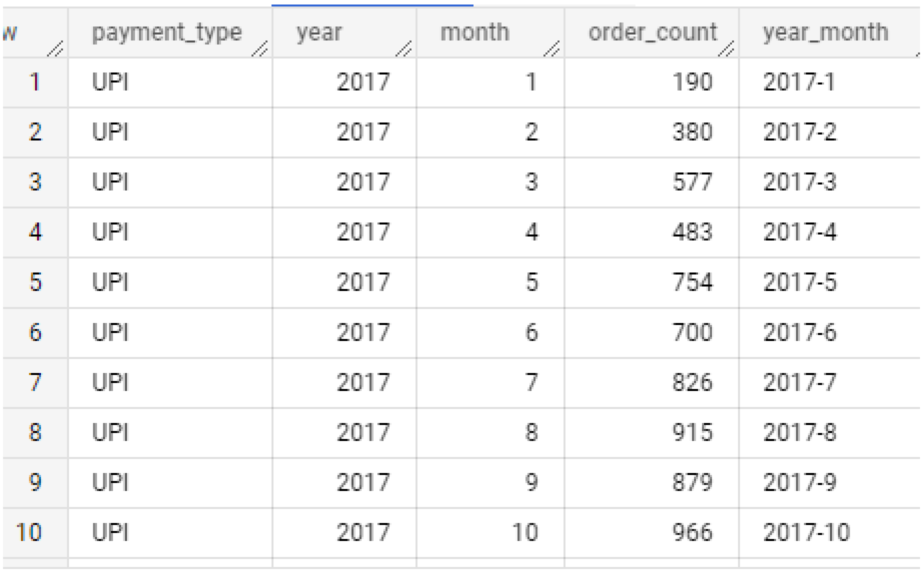
6. Payment type analysis:

6.1. Month over Month count of orders for different payment types

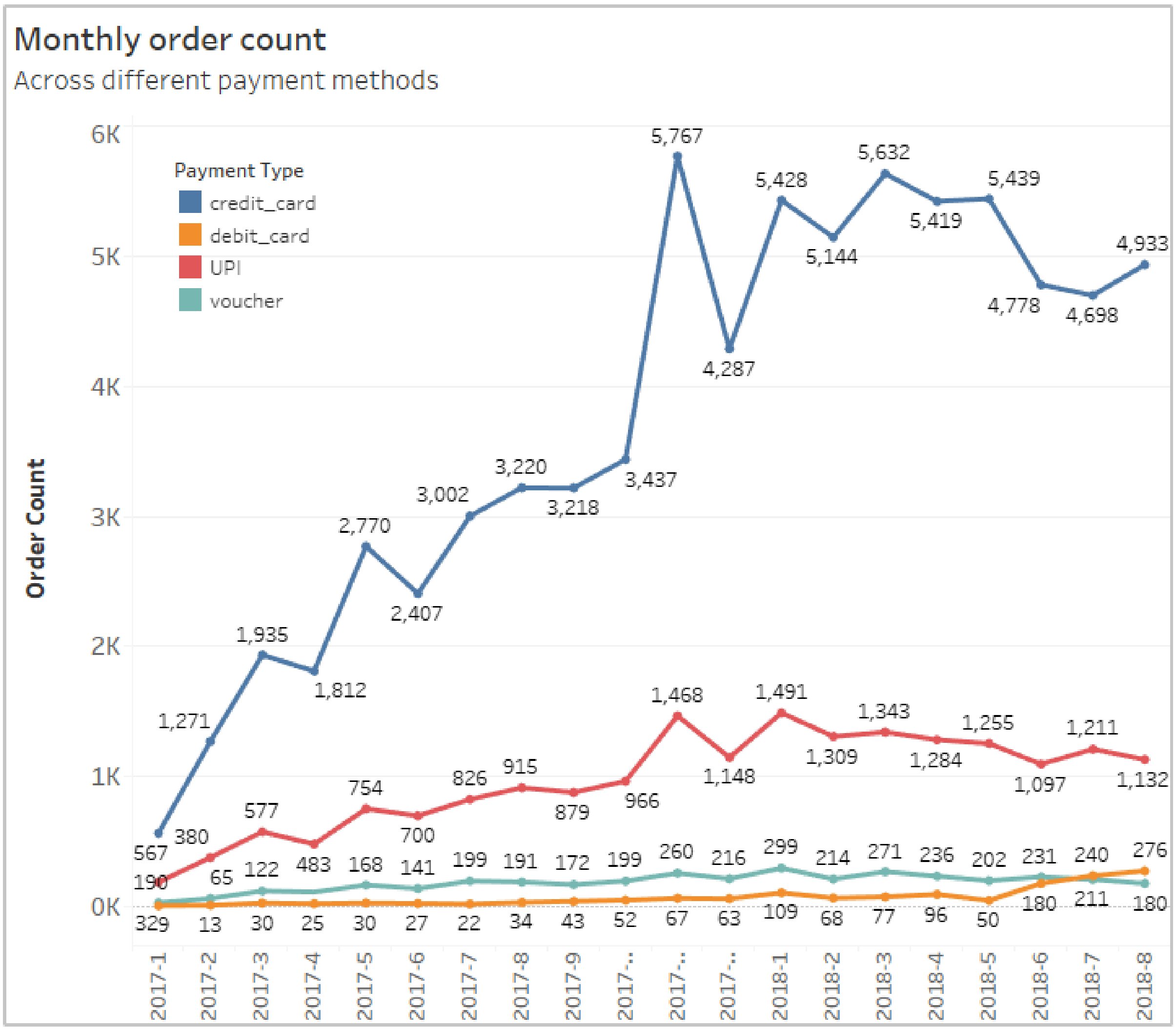
SQL Query:

```
WITH base1 AS(
SELECT DISTINCT order_id, payment_type
FROM ( SELECT * FROM `target.payments`
  WHERE payment_type NOT IN ('not_defined') AND payment_value > 0)
  ),
base2 AS (
  SELECT * FROM `target.orders`
  WHERE order_status IN ('shipped','invoiced','delivered')
),
base3 AS (
  SELECT payment_type, EXTRACT(YEAR FROM b2.order_purchase_timestamp) AS year,
    EXTRACT(MONTH FROM b2.order_purchase_timestamp) AS month, COUNT(*) AS
order_count
  FROM base1 b1 JOIN base2 b2 ON b1.order_id = b2.order_id
  GROUP BY 1, 2, 3 ORDER BY 1, 2, 3
)
SELECT *, CONCAT(year, '-', month) AS year_month FROM base3
WHERE (year = 2017 AND month BETWEEN 1 AND 12) OR
  (year = 2018 AND month BETWEEN 1 AND 8)
```

Result:



w	payment_type	year	month	order_count	year_month
1	UPI	2017	1	190	2017-1
2	UPI	2017	2	380	2017-2
3	UPI	2017	3	577	2017-3
4	UPI	2017	4	483	2017-4
5	UPI	2017	5	754	2017-5
6	UPI	2017	6	700	2017-6
7	UPI	2017	7	826	2017-7
8	UPI	2017	8	915	2017-8
9	UPI	2017	9	879	2017-9
10	UPI	2017	10	966	2017-10



<u>Following comments can be made from the above plot:</u>

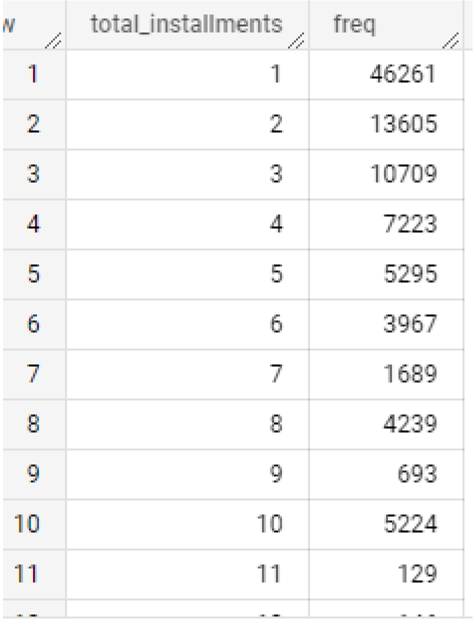
- Credit card by far the most popular choice of payment
- Debit cards along with vouchers are the least preferred choice of payment.
  - Vouchers slightly outperform debit cards over the majority of the timeframe but the observation reverses at the last few months of the above timeframe

6.2. Distribution of payment installments and count of orders

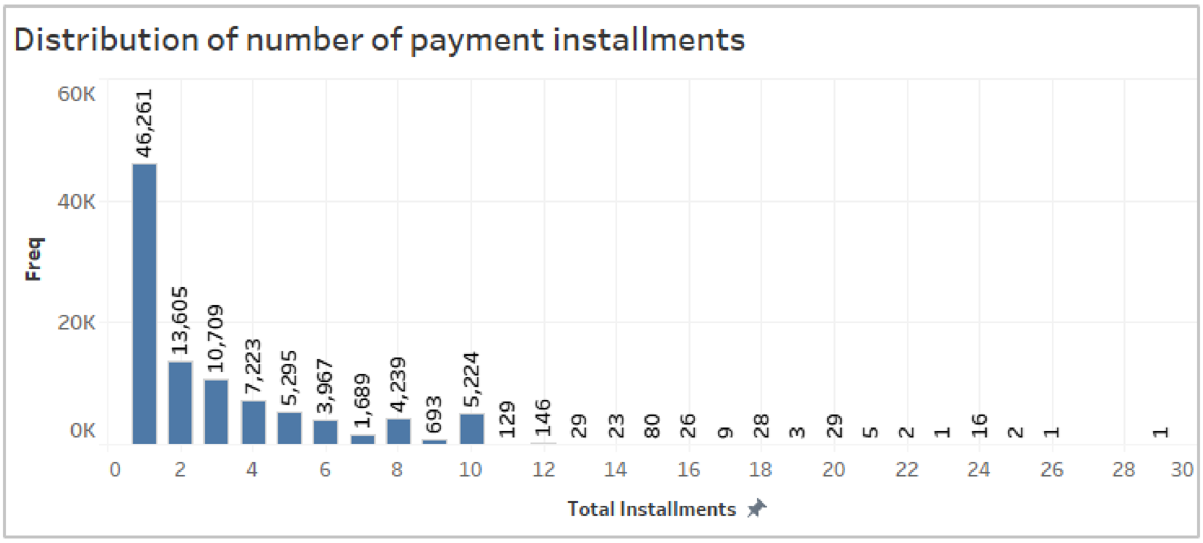
SQL Query:

```
WITH base1 AS (
  SELECT order_id, SUM(payment_installments) AS total_installments
  FROM `target.payments`
  WHERE payment_type NOT IN ('not_defined') AND payment_installments NOT IN (0)
  GROUP BY 1
)
SELECT total_installments, COUNT(*) AS freq
FROM base1
GROUP BY 1 ORDER BY total_installments
```

Result:



w	total_installments	freq
1	1	46261
2	2	13605
3	3	10709
4	4	7223
5	5	5295
6	6	3967
7	7	1689
8	8	4239
9	9	693
10	10	5224
11	11	129

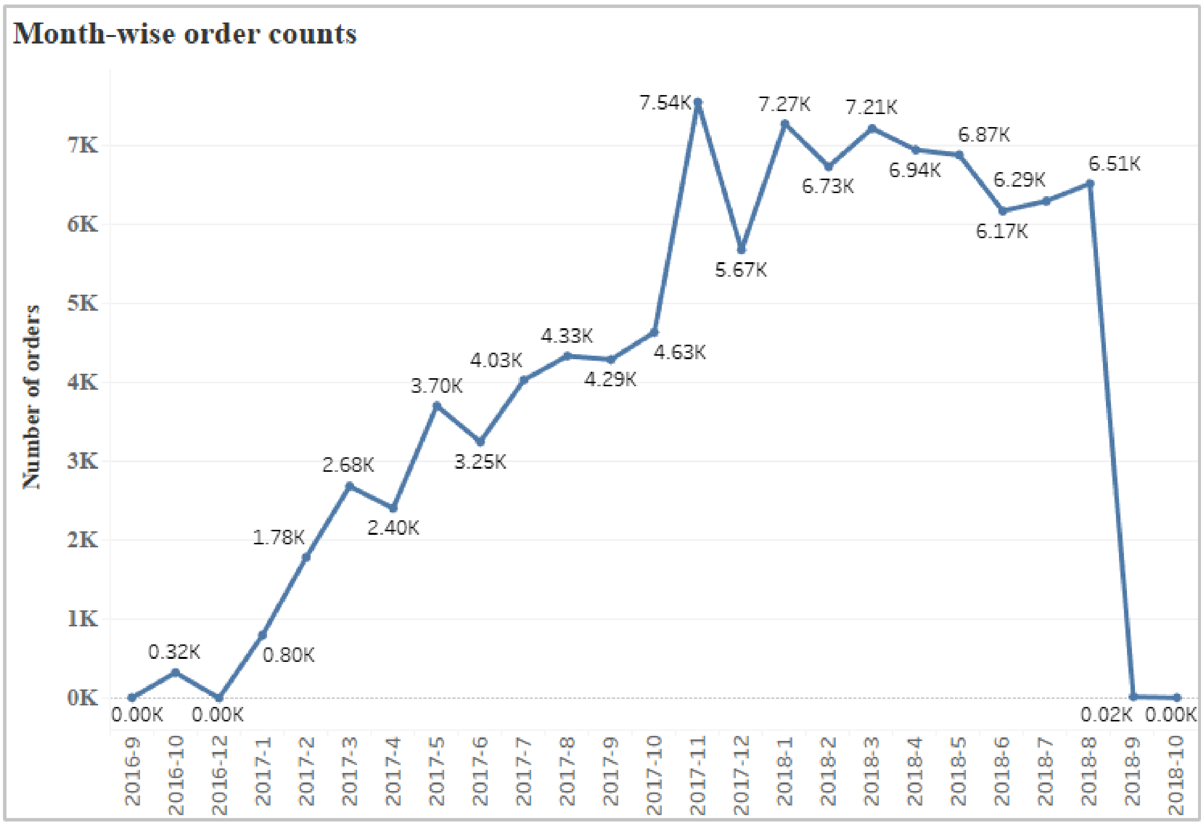


Actionable Insights and Recommendations:

1. *Stagnation of growth in number of orders in 2018*

Insight:

- There is a visible stagnation in the growth of number of orders placed in 2018 with respect to that in 2017
- In fact we see a slow decline across Q2 of 2018 (Apr, May, June)
- This followed by a slow recovery over the next quarter of 2018 (July, August)



Recommendations:

- Root cause why the order count could not grow like in 2017
- Possible reasons:
  - Saturation in the top revenue generating states. This creates the need of improving the performance of the low revenue generating states
  - Rise of competition in the e-commerce market
  - Quality of the products delivered and customer satisfaction with the entire user experience (starting from website for browsing products, ease of transaction, delivery experience etc)

2. *Peak season*

Insight:

- November is the month where the peak sales happen

Recommendations:

- Stocking of products before the peak season comes so that the demand is met
- New products can be launched around this time of the year for better reach to a wide spectrum of customers

3. *Peak slot during the day for placing orders*

Insight:

- Afternoon (12:00PM to 5:59PM) and Night (6:00PM to 11:59PM) are the two leading slots where the bulk of the purchasing happens
- Lowest orders are placed duribg Dawn (12:00AM to 5:59AM)

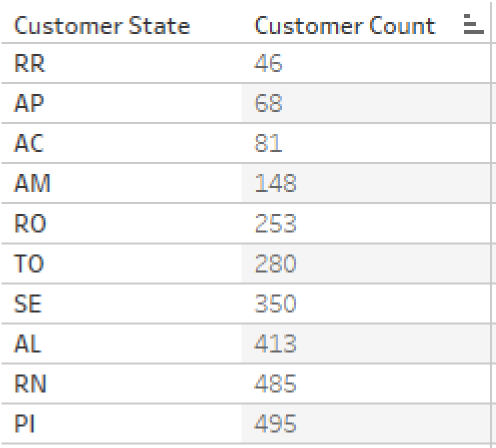
Recommendations:

- Promotions and ad campaigns can be run during afternoon and night
- Website maintainance and other routine checks can be run during dawn to ensure proper functioning of the platform throughout the day

4. *On-boarding new customers*

Insights:

- We have identified the top 10 states with the lowest customer count.



Customer State	Customer Count
RR	46
AP	68
AC	81
AM	148
RO	253
TO	280
SE	350
AL	413
RN	485
PI	495

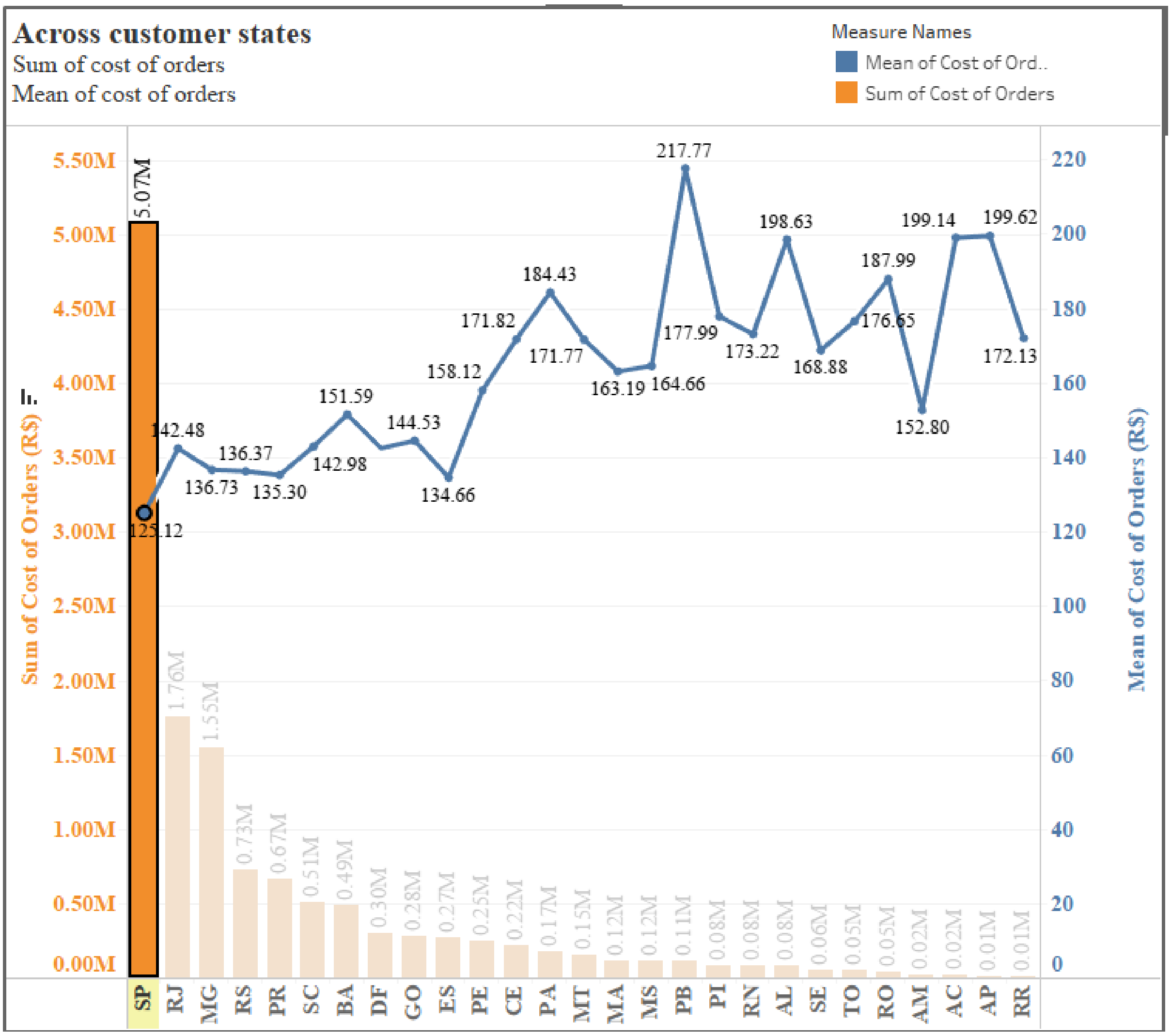
Recommendations:

- Targeted ad campaigns can be run in these states to encourage new customers to purchase items from target
- To on-board new customers attractive discounts and offers can be rolled out

5. Increase mean price of orders in SP

Insights:

- SP state generates the maximum revenue for Target and yet has the lowest mean for the cost of orders
- If the mean cost of orders in SP state can be incremented, the boost to the revenue is going to be massive since SP is the most dominant state



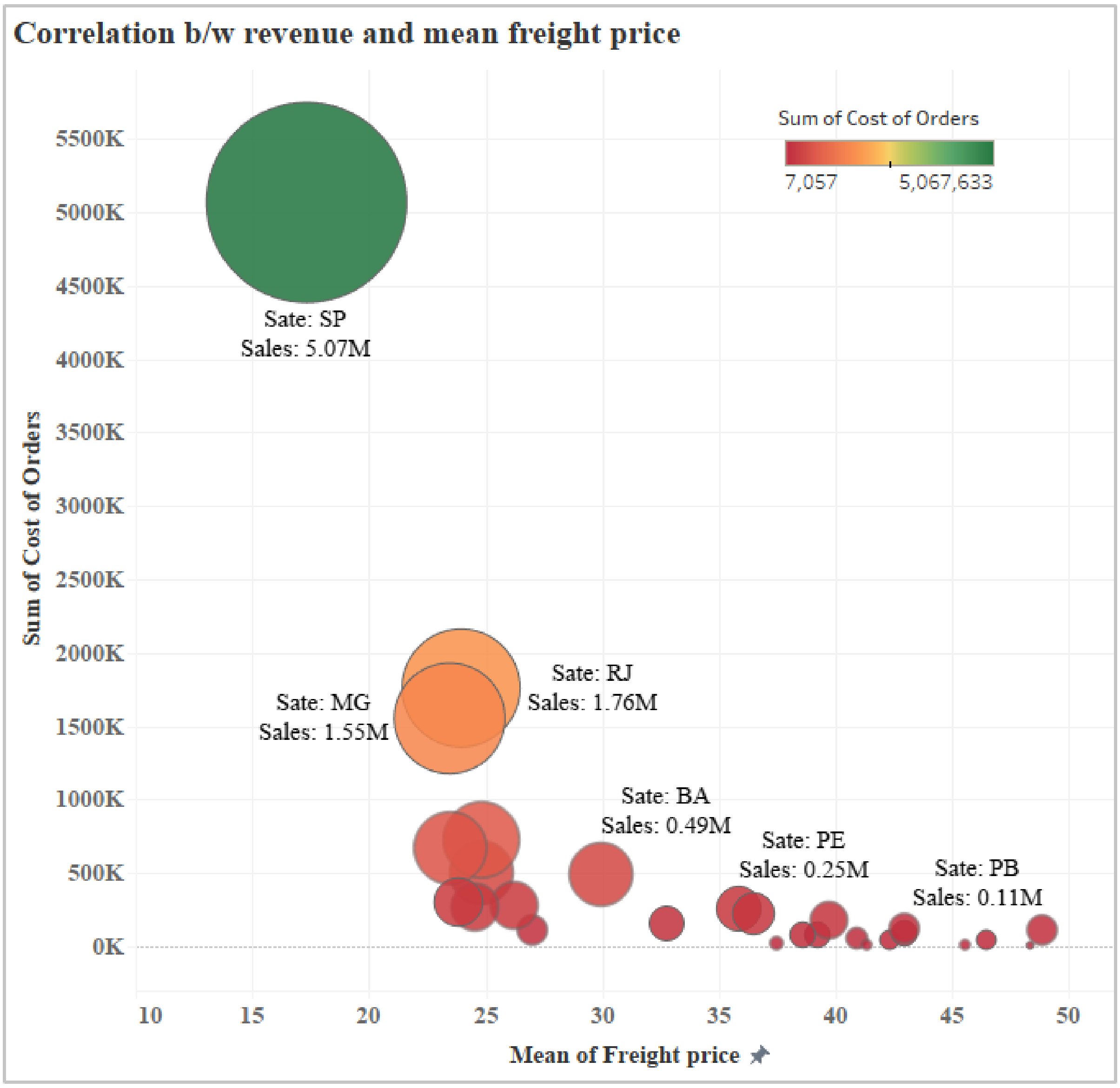
Recommendations:

- To increase the mean of cost of orders the customers in SP can be shown related products to the products that they are buying at the time of checkout
- For example:
  - If a customer is buying a mobile phone, accessories related to the phone can be shown to him as a suggestion. This will increase the chance of the customer adding these suggested items to his cart thereby increasing his order amount

6. *Negative correlation between revenue generated by state and the mean freight price in that state*

Insights:

- It has been observed that the revenue generated by a state is inversely proportional to the mean freight price in that state
- For example, the below chart shows that states which have grossed higher revenues tend to have lower average freight price
- Often the customer may be discouraged to order an item online just because he feels that the shipping charges are unreasonable



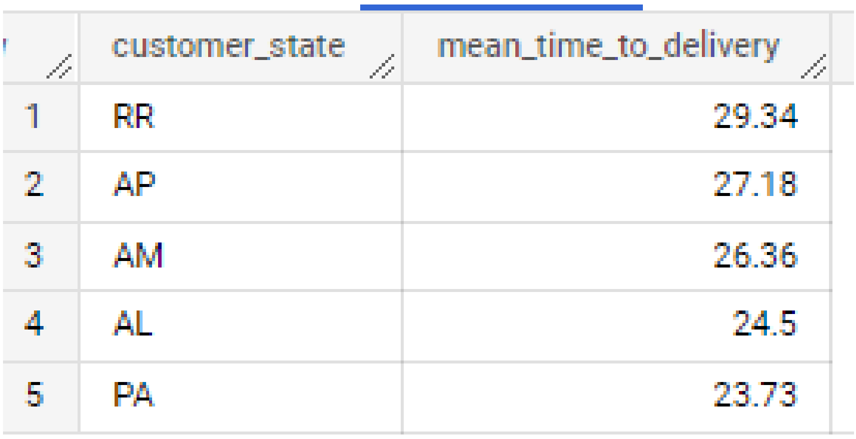
Recommendations:

- The states having high average freight price should try to reduce this parameter
- Warehousing of products can be done to lower the average shipping charges of concerned states

7. *Improving delivery time of those states where delivery takes the longest*

Insights:

- The top 5 states which have the longest delivery time are shown below
- The state RR has a mean delivery time of almost a month.
    - It is no wonder that any customer from these states will not opt for ordering via Target, especially if the item they are ordering is locally availbale



	customer_state	mean_time_to_delivery
1	RR	29.34
2	AP	27.18
3	AM	26.36
4	AL	24.5
5	PA	23.73

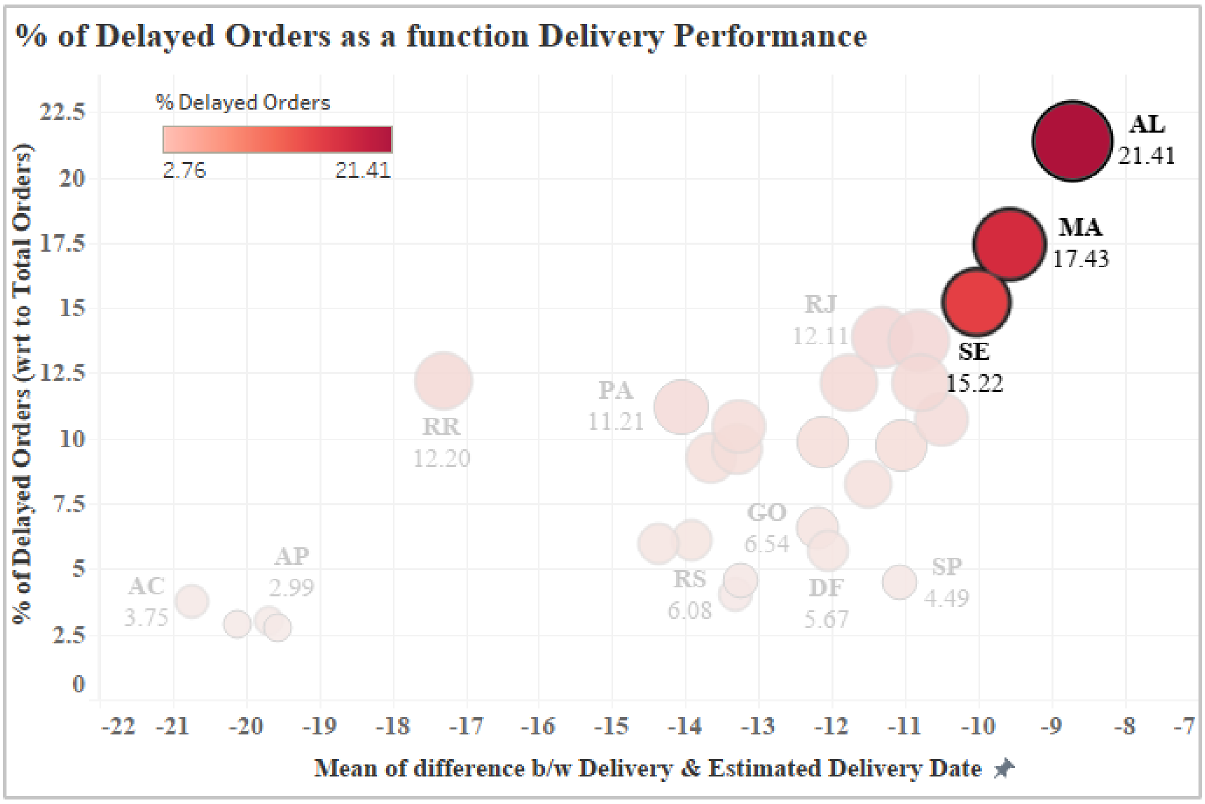
Recommendations:

- The concerned states should take an action to reduce the mean delivery time
- Increasing the number of delivery partners can be a possible solution
- Owing to the geographical location of some states, shipping time may be high. If such is the case with the concerned states, warehousing of products can be an option

8. Relooking at the model which estimates order delivery date for specific states

Insights:

- Across customer states, we try to find a correlation between
  - y = % of delayed order
    - This means what % of the total orders in a state were delivered after the estimated delivery date
  - x = mean of difference between order delivery date and estimated delivery date
    - This variable is a measure of the delivery performance
    - The more negative it is, the better which in turn means orders are being delivered long before the estimated delivery date
- We identify the top 3 states having the highest % of delayed orders (> 15%)
  - They have been highlighted in the below figure
- Also, as expected the delivery performance of these states are the worst among all the states



Recommendations:

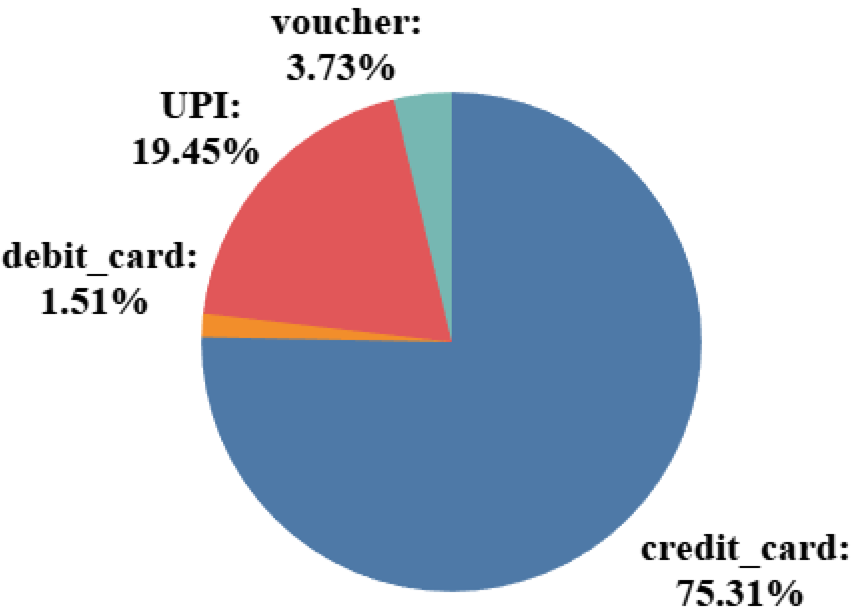
- The primary objective for above identified states would be to look into what is causing these delayed deliveries
- Along with resolving the bottlenecks that are causing delivey delays, one can also look into the model which predicts the estimated delivery date for these states
- If the model is giving a too optimistic date for estimated delivery, it needs to be fine tuned to suit the logistical and operational shortcomings present in these states
- Delayed deliveries hurt the customer sentiment and hence should be addressed with utmost importance

9. Payment options: How to increase orders and sales from the payment data?

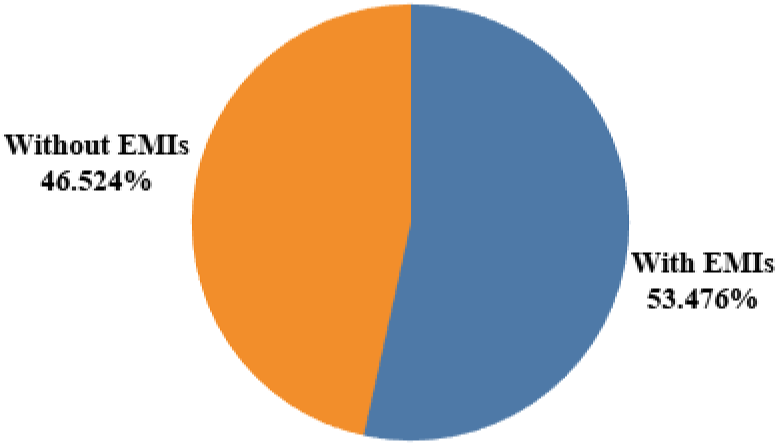
Insights:

- FigA: Credit card is by far the most used form of payment
- FigB: Majority of customers tend to pay without EMIs. However almost 47% of customers pay with EMIs (more than one payment towards the total order price is considered as an EMI)

FigA: Relative % of payment methods



FigB: Relative % of customers
Without EMI vs With EMIs



Recommendations:

- Ease of payment while ordering a product is one of the key factors that adds to the user experience.
  - Having multiple options to make payments are an added advantage.
  - Add to that discount deals, coupon codes are all key marketing strategies that can be incorporated to make the customer experience good
- **Credit cards:**
  - Bring more payment firms which offer credit card services in Target for making payments more accessible to customers
  - Discounts and attractive offers can be rolled out on credit card payments which will lure more customer to place orders
- **EMIs:**
  - Many people find owing a product by paying via EMIs affordable. We can see that almost 47% of the customers are choosing EMIs to order
  - More affordable EMIs options can be offered including No Cost EMIs so that customers find it easier to afford the items that they want to buy. This will surely have an positive impact on increasing the number of orders placed