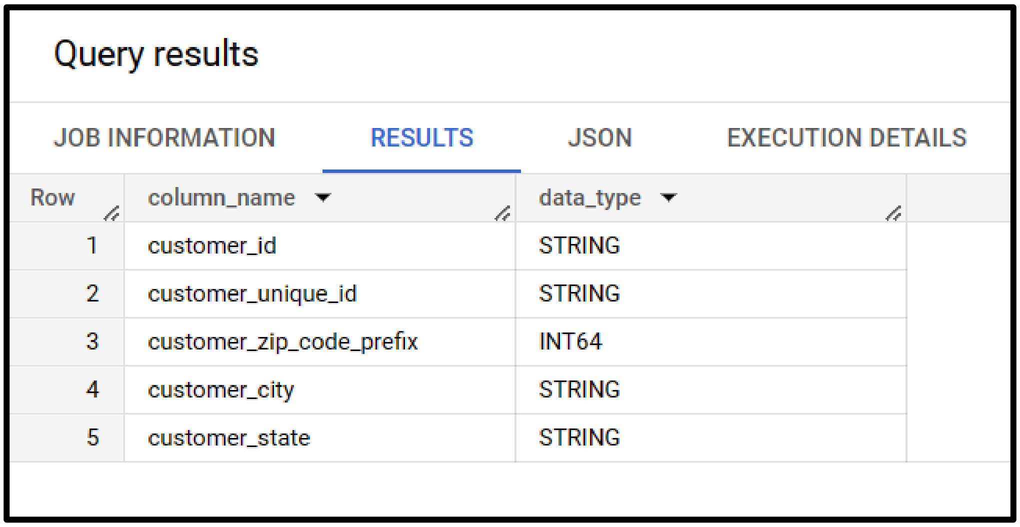
Target SQL Business Case Study

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

A. Data type of all columns in the “customers” table.

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

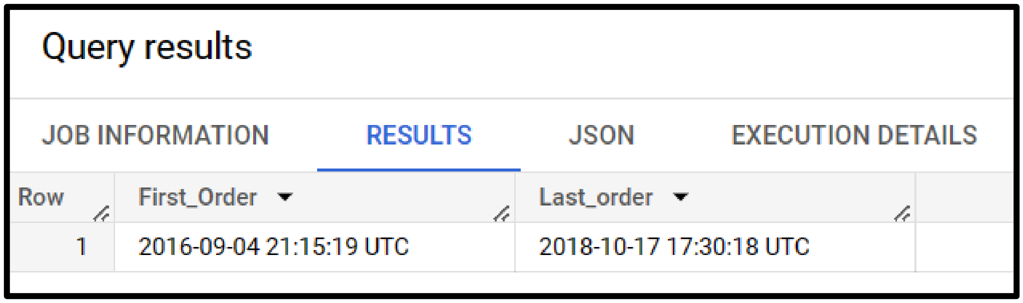


Query results

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

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

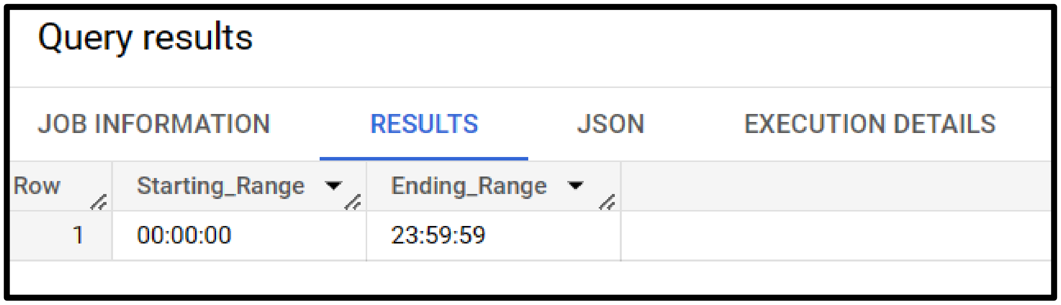
```
SELECT
  FIRST_VALUE(order_purchase_timestamp) OVER(order by order_purchase_timestamp )
as First_Order,
  FIRST_VALUE(order_purchase_timestamp) OVER(order by order_purchase_timestamp
desc ) as Last_order
FROM
  `SQL_Casestudy.orders`
LIMIT 1;
```



Query results

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

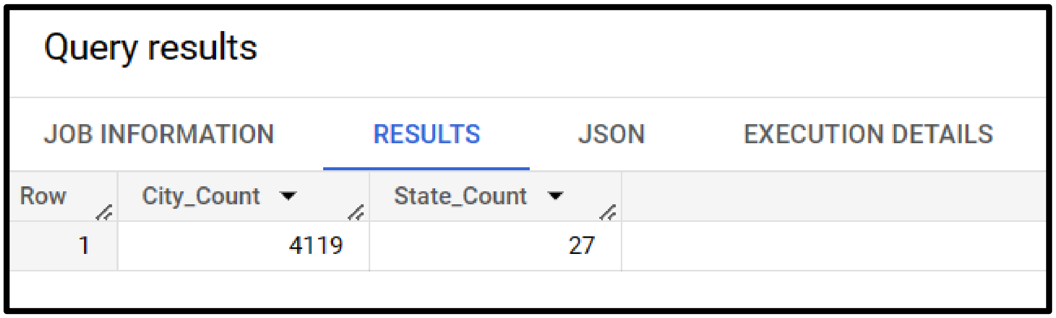
```
# Time Range
SELECT
  MIN(TIME(order_purchase_timestamp)) AS Starting_Range,
  MAX(TIME(order_purchase_timestamp)) AS Ending_Range
FROM
  `SQL_Casestudy.orders`;
```



Row	Starting_Range	Ending_Range
1	00:00:00	23:59:59

C. Count the number of Cities and States in our dataset.

```
SELECT
  COUNT(DISTINCT customer_city) AS City_Count,
  COUNT(DISTINCT customer_state) AS State_Count
FROM
  `SQL_Casestudy.customers`;
```

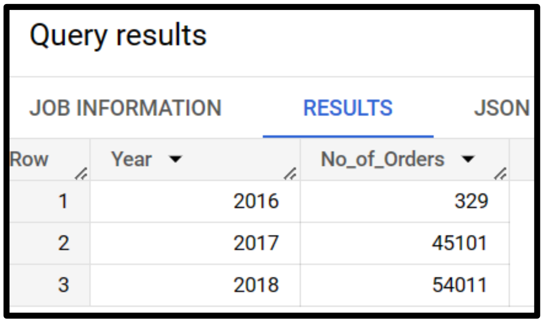


Row	City_Count	State_Count
1	4119	27

II. In-depth Exploration:

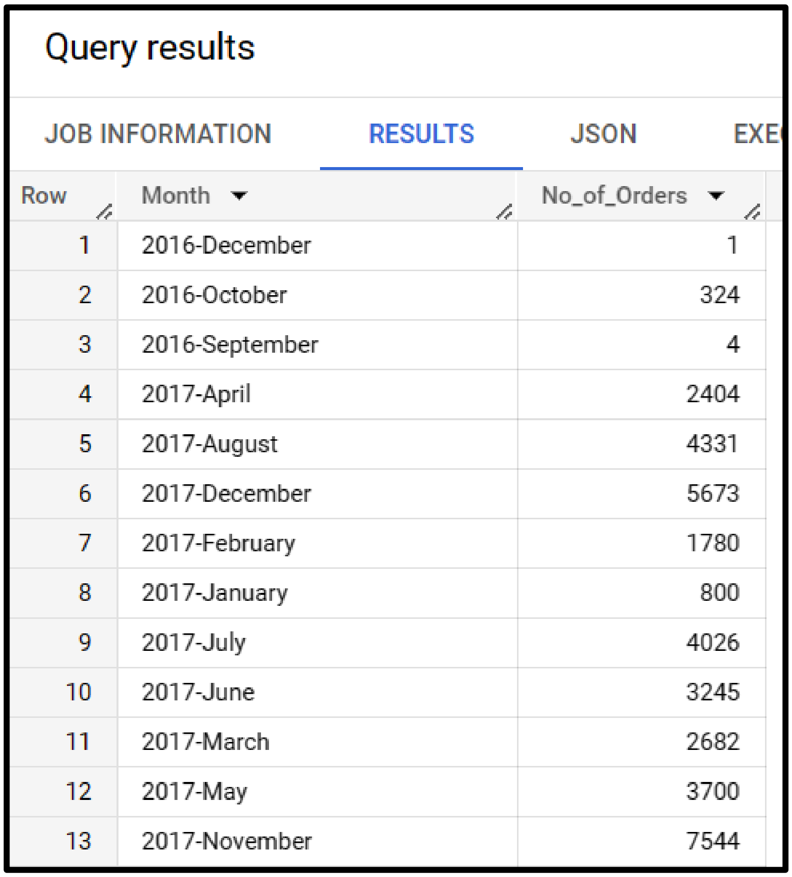
A. Is there a growing trend in the no. of orders placed over the past years?

```
-- Year_wise
SELECT
  COUNT(order_id) as No_of_Orders,
  extract(year from order_purchase_timestamp) as Year
FROM `SQL_Casestudy.orders`
  GROUP BY Year
  ORDER BY Year;
```



Row	Year	No_of_Orders
1	2016	329
2	2017	45101
3	2018	54011

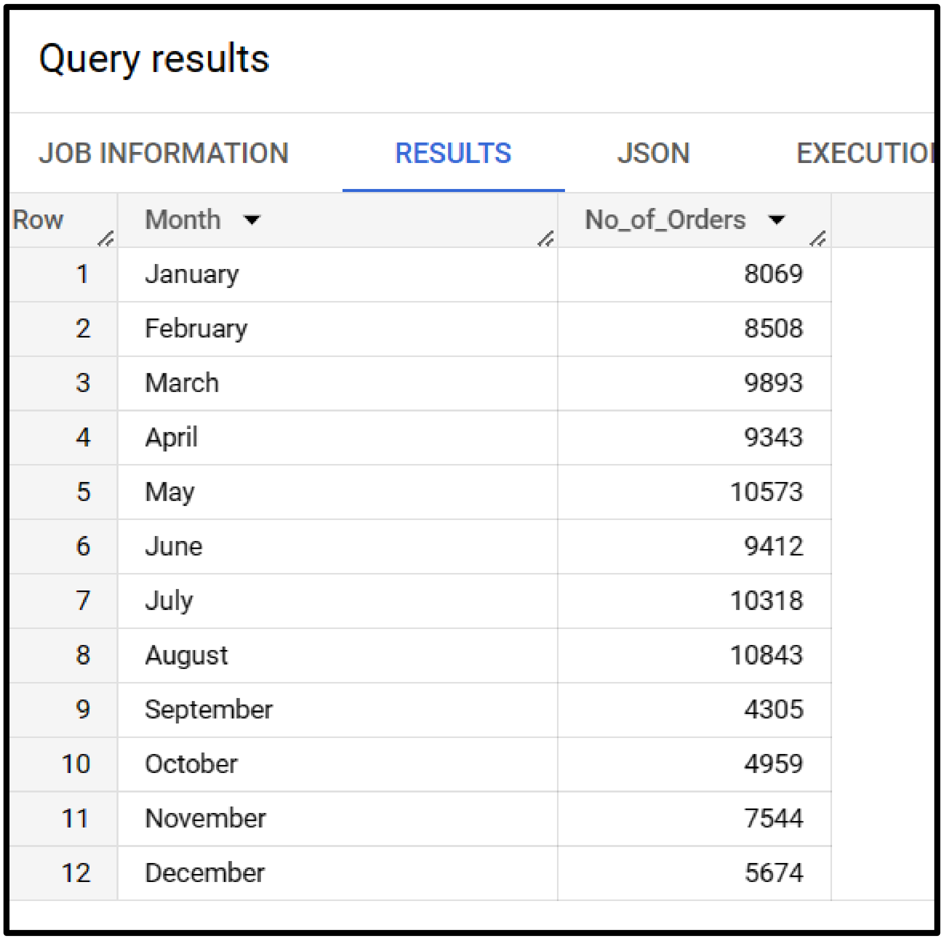
```
-- Month_Wise
       WITH CTE AS
       (
       SELECT
         format_timestamp("%Y-%B",order_purchase_timestamp) as Month,
         COUNT(order_id) as No_of_Orders,
       FROM `SQL_Casestudy.orders`
         GROUP BY Month
         ORDER BY Month
       )
       SELECT
         Month ,No_of_Orders
       FROM CTE
```



Row	Month	No_of_Orders
1	2016-December	1
2	2016-October	324
3	2016-September	4
4	2017-April	2404
5	2017-August	4331
6	2017-December	5673
7	2017-February	1780
8	2017-January	800
9	2017-July	4026
10	2017-June	3245
11	2017-March	2682
12	2017-May	3700
13	2017-November	7544

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

```
WITH CTE AS
(
  SELECT
    format_timestamp('%B', order_purchase_timestamp) As Month,
    COUNT(order_id) as No_of_Orders,
    extract (month from order_purchase_timestamp) as MN
  FROM `SQL_Casestudy.orders`
  GROUP BY Month,MN
  ORDER BY MN
)
SELECT
  CTE.Month,CTE.No_of_Orders
FROM CTE
```

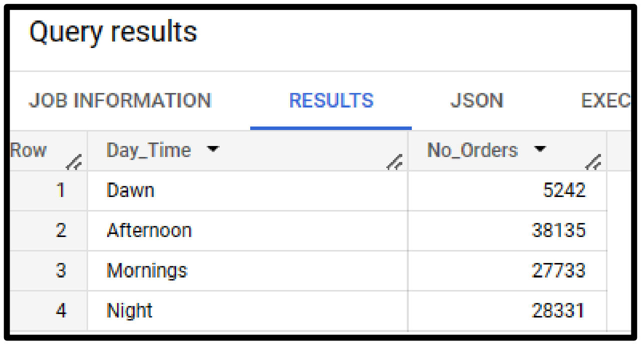


Row	Month	No_of_Orders
1	January	8069
2	February	8508
3	March	9893
4	April	9343
5	May	10573
6	June	9412
7	July	10318
8	August	10843
9	September	4305
10	October	4959
11	November	7544
12	December	5674

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

- 0-6 hrs : Dawn
- 7-12 hrs : Mornings
- 13-18 hrs : Afternoon
- 19-23 hrs : Night

```
WITH CTE AS
(
  SELECT
  Day_Time,
  No_Orders,
  CASE
    WHEN Day_Time = 'Dawn' THEN 1
    WHEN Day_Time = 'Morning' THEN 2
    WHEN Day_Time = 'Afternoon' THEN 3
    ELSE 4
  END AS Day_order
FROM
(
  SELECT
  count(order_id) as No_Orders ,
  CASE
    WHEN hours BETWEEN 0 AND 6 THEN 'Dawn'
    WHEN hours BETWEEN 7 AND 12 THEN 'Mornings'
    WHEN hours BETWEEN 13 AND 18 THEN 'Afternoon'
  ELSE 'Night'
  END AS Day_Time
  FROM
    (SELECT
        order_id,order_purchase_timestamp,
        EXTRACT(hour from order_purchase_timestamp) as hours,
      FROM `SQL_Casestudy.orders`
      GROUP BY order_id,order_purchase_timestamp,hours
    )
  GROUP BY Day_Time
) as Semi
ORDER By Day_order
)
SELECT Day_Time,No_Orders
FROM CTE
```

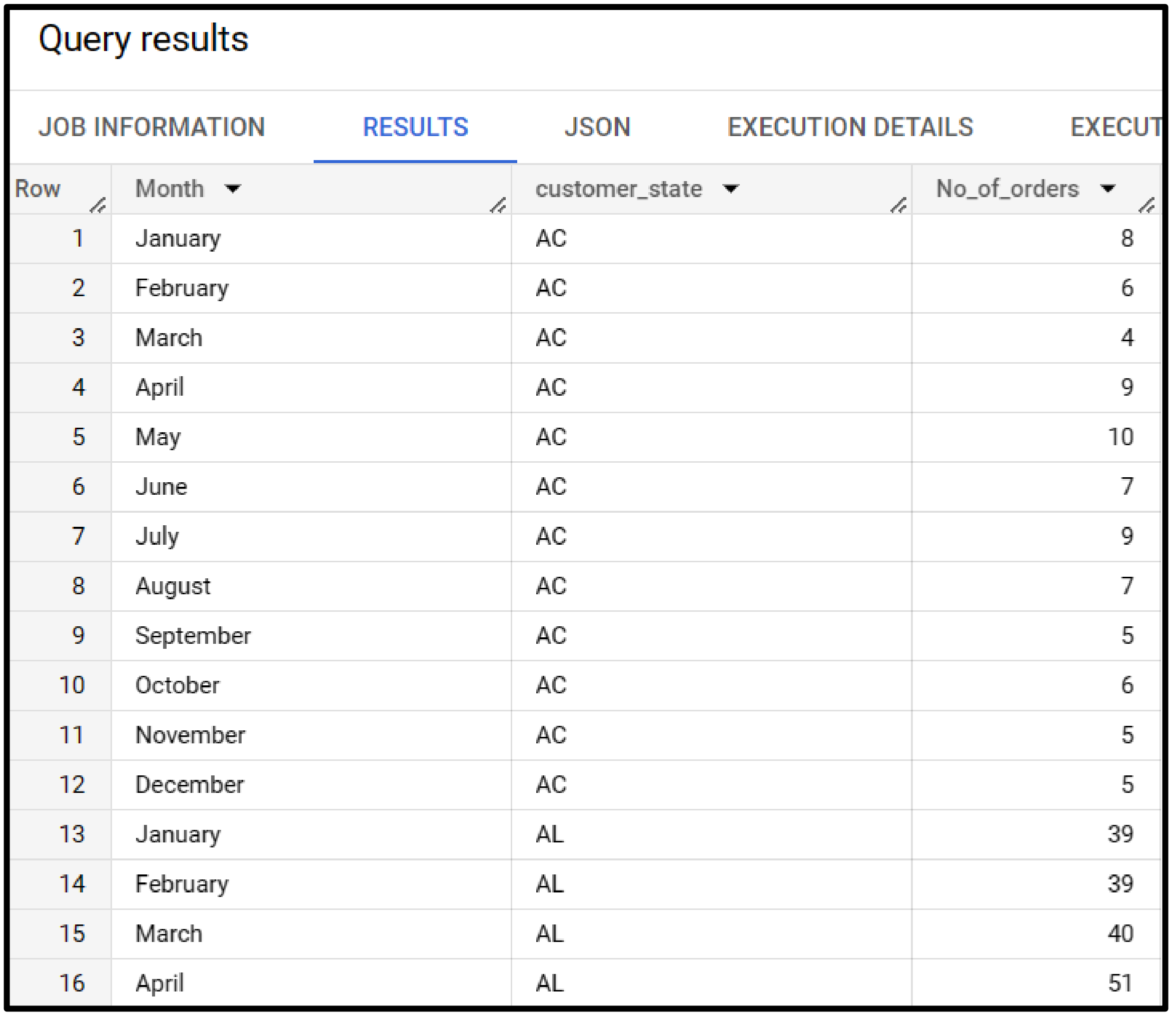


Row	Day_Time	No_Orders
1	Dawn	5242
2	Afternoon	38135
3	Mornings	27733
4	Night	28331

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

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

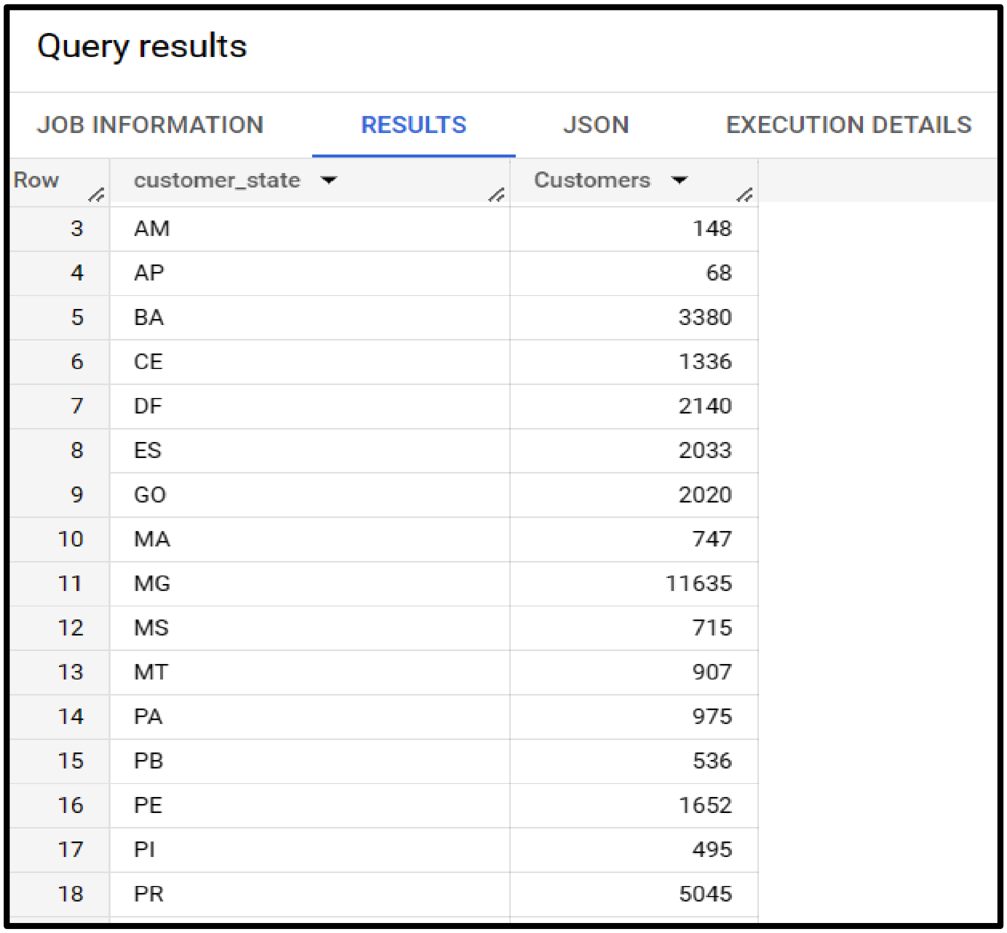
```
WITH CTE AS(
SELECT
    COUNT(order_id) as No_of_orders,
    format_timestamp('%B', order_purchase_timestamp) As
Month,customer_state,
    extract (month from order_purchase_timestamp) as MN
  FROM `SQL_Casestudy.orders` o INNER JOIN `SQL_Casestudy.customers` c
  ON o.customer_id = c.customer_id
  GROUP BY Month,customer_state,MN
  ORDER BY customer_state,MN)
SELECT
  Month,customer_state,No_of_orders
FROM CTE
```



Row	Month	customer_state	No_of_orders
1	January	AC	8
2	February	AC	6
3	March	AC	4
4	April	AC	9
5	May	AC	10
6	June	AC	7
7	July	AC	9
8	August	AC	7
9	September	AC	5
10	October	AC	6
11	November	AC	5
12	December	AC	5
13	January	AL	39
14	February	AL	39
15	March	AL	40
16	April	AL	51

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

```
SELECT
    customer_state,count(customer_unique_id) as Customers,
FROM `SQL_Casestudy.customers`
GROUP BY customer_state
ORDER BY customer_state;
```



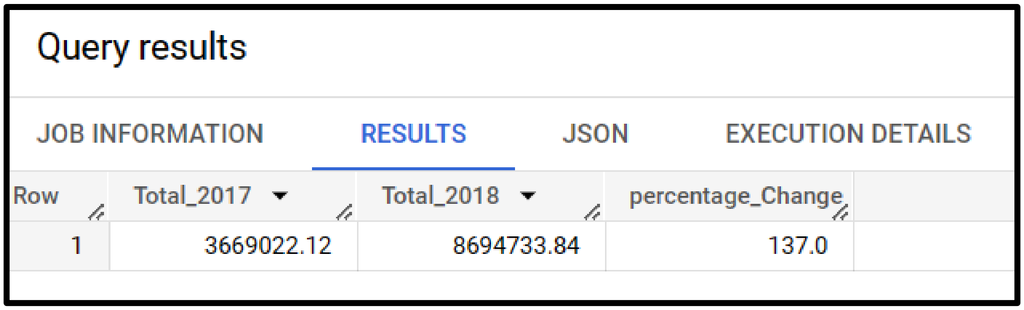
Query results

Row	customer_state	Customers
3	AM	148
4	AP	68
5	BA	3380
6	CE	1336
7	DF	2140
8	ES	2033
9	GO	2020
10	MA	747
11	MG	11635
12	MS	715
13	MT	907
14	PA	975
15	PB	536
16	PE	1652
17	PI	495
18	PR	5045

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

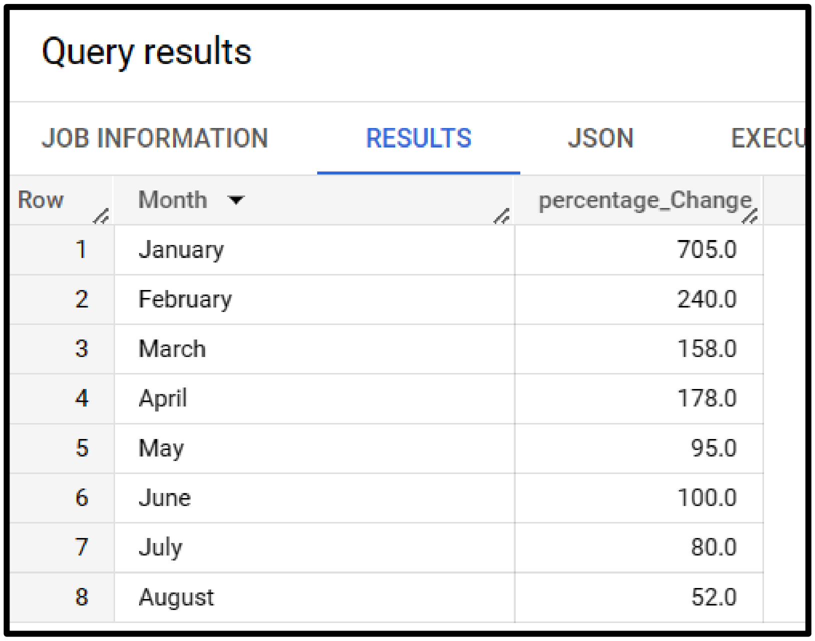
A. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

```
-- Year Wise
WITH Month_Total AS
(
SELECT
  format_timestamp('%Y',o.order_purchase_timestamp) as Year,
  sum(p.payment_value) as month_total
FROM `SQL_Casestudy.orders` o INNER JOIN `SQL_Casestudy.payments` p ON
o.order_id  = p.order_id
WHERE (extract(year from order_purchase_timestamp) BETWEEN 2017 AND
2018)  AND (extract(month from order_purchase_timestamp) BETWEEN 1 AND 8)
GROUP BY Year
ORDER BY Year
)

SELECT
  round(a.month_total,2) as Total_2017,
  round(b.month_total,2) as Total_2018,
  round((((b.month_total - a.month_total)/a.month_total)*100)) as
percentage_Change
FROM Month_Total a JOIN Month_Total b ON a.Year < b.year
```



Row	Total_2017	Total_2018	percentage_Change
1	3669022.12	8694733.84	137.0

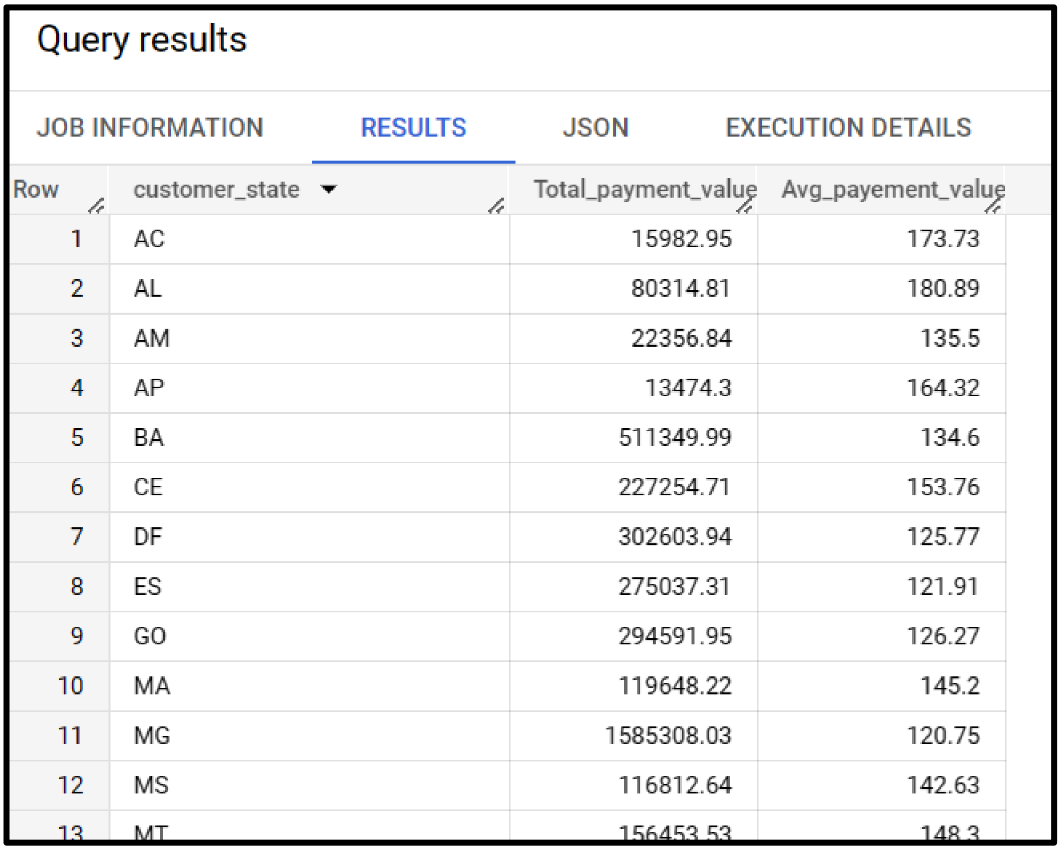
```
-- Month Wise
WITH Month_Total AS
(
SELECT
  format_timestamp('%Y',o.order_purchase_timestamp) as Year,
  format_timestamp('%B',o.order_purchase_timestamp) as Month,
  extract(month from order_purchase_timestamp) as MN,
  sum(p.payment_value) as month_total
FROM `SQL_Casestudy.orders` o INNER JOIN `SQL_Casestudy.payments` p ON
o.order_id  = p.order_id
WHERE (extract(year from order_purchase_timestamp) BETWEEN 2017 AND
2018)  AND (extract(month from order_purchase_timestamp) BETWEEN 1 AND 8)
GROUP BY Year,Month,MN
ORDER BY Year,MN
)

SELECT
  a.Month,
  round((((b.month_total - a.month_total)/a.month_total)*100)) as
percentage_Change
FROM Month_Total a JOIN Month_Total b ON a.Year < b.year and a.MN = b.MN
ORDER BY a.MN;
```



Row	Month	percentage_Change
1	January	705.0
2	February	240.0
3	March	158.0
4	April	178.0
5	May	95.0
6	June	100.0
7	July	80.0
8	August	52.0

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

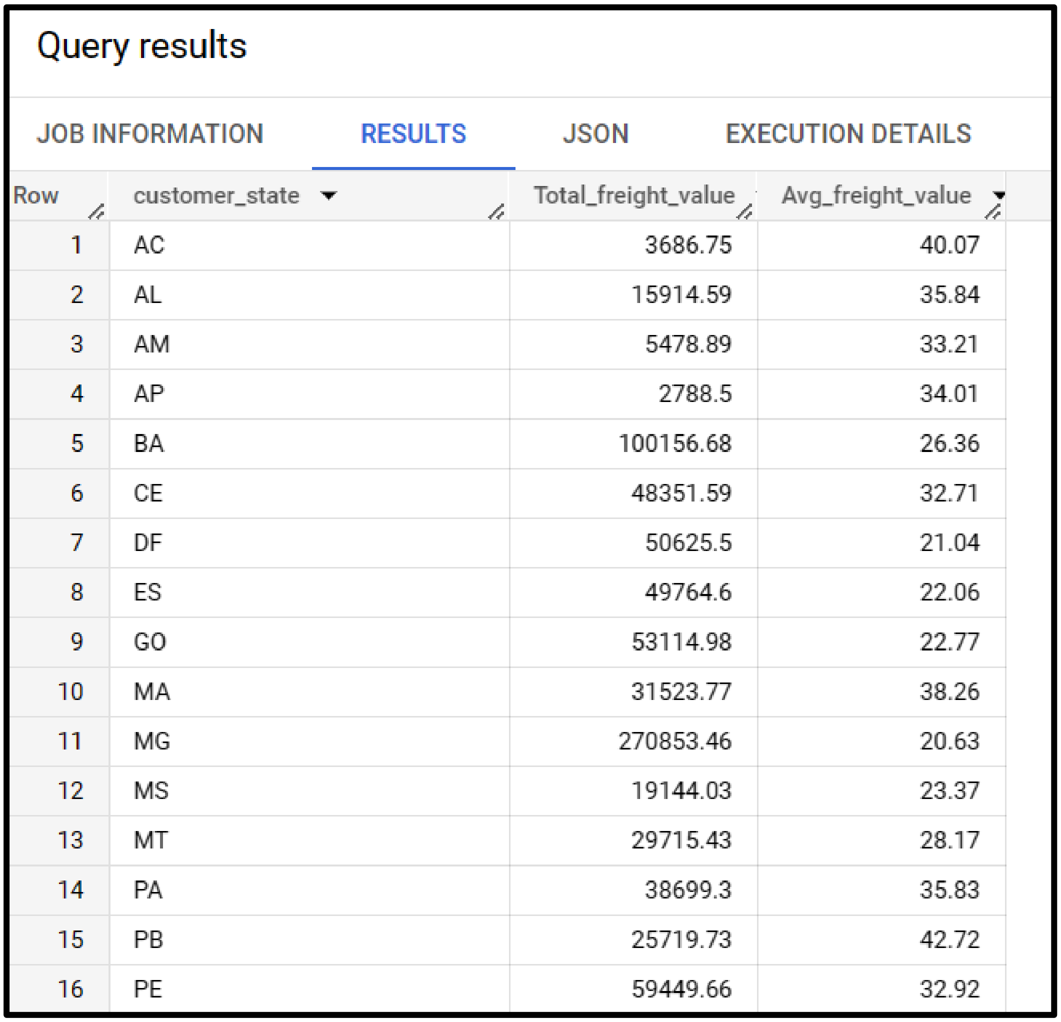
```
SELECT
  c.customer_state,
  round(sum(oi.price),2) as Total_payment_value,
  round(AVG(oi.price),2) as Avg_payement_value
FROM `SQL_Casestudy.orders` o
  INNER JOIN `SQL_Casestudy.order_items` oi ON o.order_id =
oi.order_id
  INNER JOIN `SQL_Casestudy.customers` c ON c.customer_id =
o.customer_id
GROUP BY customer_state
ORDER BY customer_state;
```



Row	customer_state	Total_payment_value	Avg_payement_value
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
11	MG	1585308.03	120.75
12	MS	116812.64	142.63
13	MT	156453.53	148.3

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

```
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 `SQL_Casestudy.orders` o
  INNER JOIN `SQL_Casestudy.order_items` oi ON o.order_id = oi.order_id
  INNER JOIN `SQL_Casestudy.customers` c ON c.customer_id = o.customer_id
GROUP BY customer_state
ORDER BY customer_state;
```

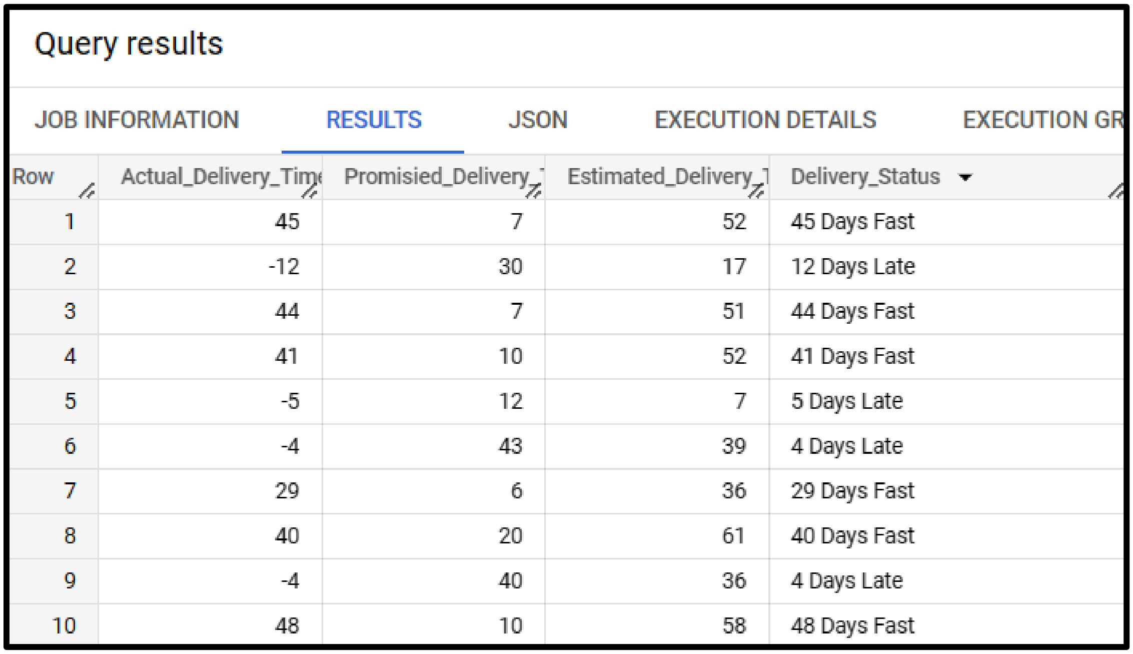


Row	customer_state	Total_freight_value	Avg_freight_value
1	AC	3686.75	40.07
2	AL	15914.59	35.84
3	AM	5478.89	33.21
4	AP	2788.5	34.01
5	BA	100156.68	26.36
6	CE	48351.59	32.71
7	DF	50625.5	21.04
8	ES	49764.6	22.06
9	GO	53114.98	22.77
10	MA	31523.77	38.26
11	MG	270853.46	20.63
12	MS	19144.03	23.37
13	MT	29715.43	28.17
14	PA	38699.3	35.83
15	PB	25719.73	42.72
16	PE	59449.66	32.92

V. Analysis based on sales, freight and delivery time.

A. 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. Do this in a single query.

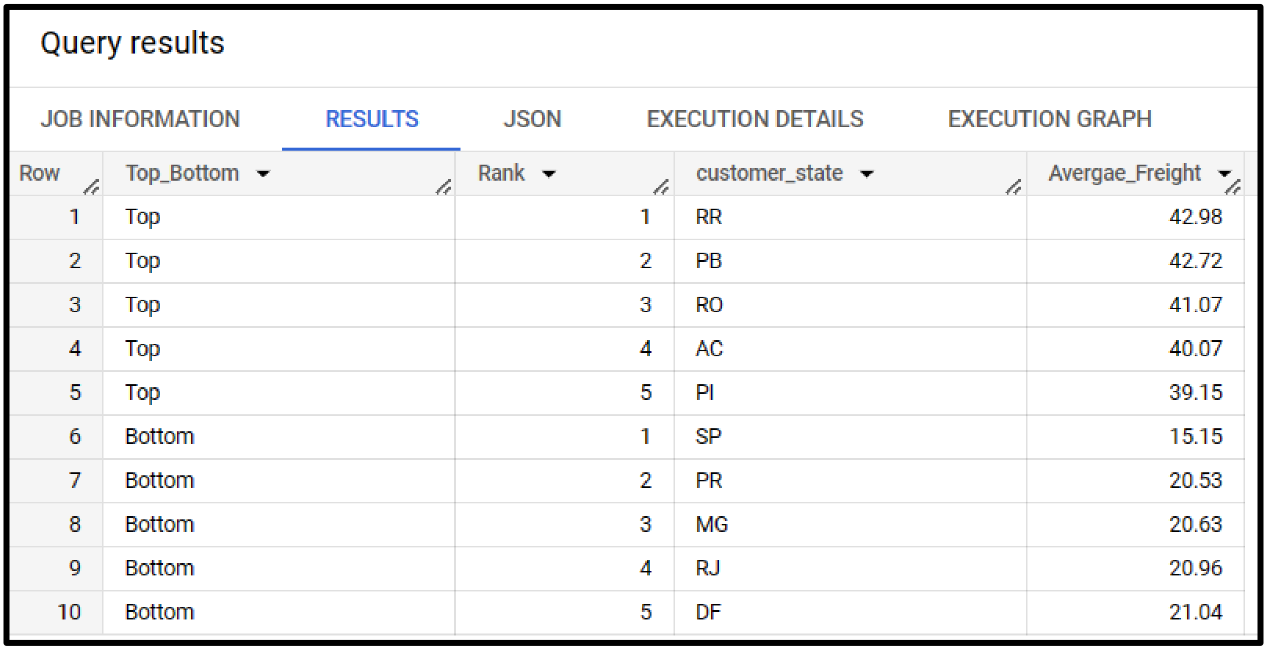
```
SELECT
  Table1.Actual_Delivery_Time,
  Table1.Promisied_Delivery_Time,
  Table1.Estimated_Delivery_Time ,
  CASE
  WHEN Table1.Actual_Delivery_Time = 0 THEN 'On-Time'
  WHEN Table1.Actual_Delivery_Time > 0  THEN concat( Table1.Actual_Delivery_Time,' Days
Fast')
  ELSE  concat(abs(Table1.Actual_Delivery_Time),' Days Late')
END as Delivery_Status
FROM
(
  SELECT
    o.order_id,
    timestamp_diff(o.order_estimated_delivery_date,o.order_purchase_timestamp,day) as
Estimated_Delivery_Time,
    timestamp_diff(o.order_delivered_customer_date,o.order_purchase_timestamp, day) as
Promisied_Delivery_Time,
    timestamp_diff(o.order_estimated_delivery_date,order_delivered_customer_date,day) as
Actual_Delivery_Time
  FROM `SQL_Casestudy.orders`  o
  WHERE o.order_delivered_customer_date is not null AND o.order_purchase_timestamp is not
null AND o.order_estimated_delivery_date is not null
) as Table1;
```



Row	Actual_Delivery_Tim	Promisied_Delivery_	Estimated_Delivery_	Delivery_Status
1	45	7	52	45 Days Fast
2	-12	30	17	12 Days Late
3	44	7	51	44 Days Fast
4	41	10	52	41 Days Fast
5	-5	12	7	5 Days Late
6	-4	43	39	4 Days Late
7	29	6	36	29 Days Fast
8	40	20	61	40 Days Fast
9	-4	40	36	4 Days Late
10	48	10	58	48 Days Fast

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

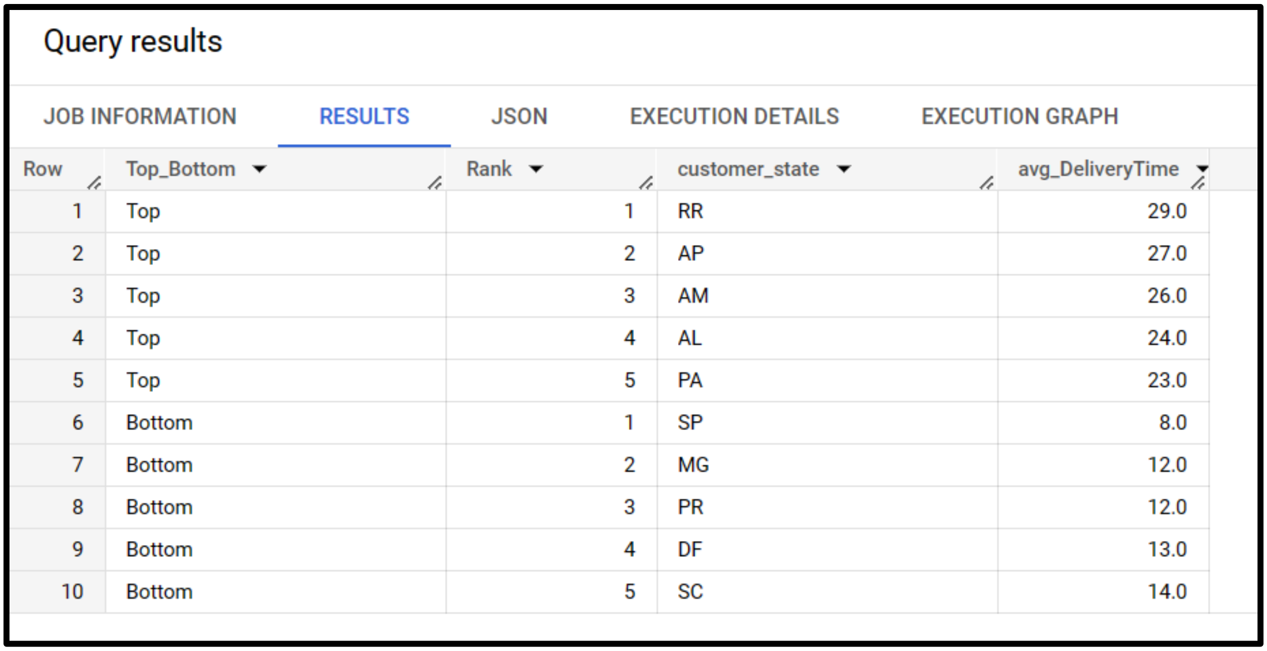
```
SELECT

   T1.Top_Bottom,T1.rn as Rank,T1.customer_state,T1.Avergae_Freight
FROM
  (SELECT
    c.customer_state,
    'Top' as Top_Bottom,
    round(AVG(oi.freight_value),2) as Avergae_Freight,
    row_number() over(order by AVG(oi.freight_value) desc) as rn
  FROM `SQL_Casestudy.customers` c
    INNER JOIN `SQL_Casestudy.orders` o ON c.customer_id = o.customer_id
    INNER JOIN `SQL_Casestudy.order_items` oi ON o.order_id = oi.order_id
  GROUP BY c.customer_state
  ) as T1
WHERE rn < 6
UNION ALL
SELECT
   T1.Top_Bottom,T1.rn as Rank,T1.customer_state,T1.Avergae_Freight
FROM
  (SELECT
    c.customer_state,
    'Bottom' as Top_Bottom,
    round(AVG(oi.freight_value),2) as Avergae_Freight,
    row_number() over(order by AVG(oi.freight_value) ) as rn
  FROM `SQL_Casestudy.customers` c
    INNER JOIN `SQL_Casestudy.orders` o ON c.customer_id = o.customer_id
    INNER JOIN `SQL_Casestudy.order_items` oi ON o.order_id = oi.order_id
  GROUP BY c.customer_state
  ) as T1
WHERE rn < 6;
```



Row	Top_Bottom	Rank	customer_state	Avergae_Freight
1	Top	1	RR	42.98
2	Top	2	PB	42.72
3	Top	3	RO	41.07
4	Top	4	AC	40.07
5	Top	5	PI	39.15
6	Bottom	1	SP	15.15
7	Bottom	2	PR	20.53
8	Bottom	3	MG	20.63
9	Bottom	4	RJ	20.96
10	Bottom	5	DF	21.04

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

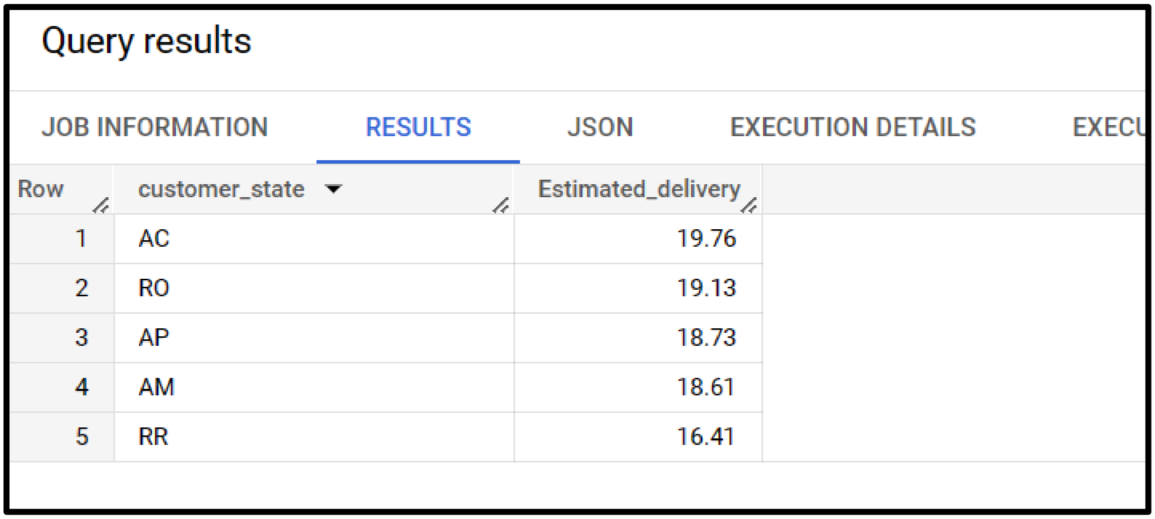
```
SELECT
 T1.Top_Bottom,T1.rn as Rank,T1.customer_state,T1.avg_DeliveryTime
FROM
 (
  SELECT
   distinct c.customer_state,
    'Top' as Top_Bottom,
    round(avg(timestamp_diff(o.order_delivered_customer_date,o.orde
r_purchase_timestamp, day))) as avg_DeliveryTime,
    row_number() over(order by
round(avg(timestamp_diff(o.order_delivered_customer_date,o.order_purcha
se_timestamp, day)))desc ) as rn
  FROM `SQL_Casestudy.customers` c INNER JOIN `SQL_Casestudy.orders`
o ON c.customer_id = o.customer_id
  GROUP BY c.customer_state
  ORDER BY rn
 ) as T1
WHERE T1.rn < 6
UNION ALL
SELECT
 T1.Top_Bottom,T1.rn as Rank,T1.customer_state,T1.avg_DeliveryTime
FROM
 (
  SELECT
   distinct c.customer_state,
    'Bottom' as Top_Bottom,
    round(avg(timestamp_diff(o.order_delivered_customer_date,o.orde
r_purchase_timestamp, day))) as avg_DeliveryTime,
    row_number() over(order by
round(avg(timestamp_diff(o.order_delivered_customer_date,o.order_purcha
se_timestamp, day))) ) as rn
  FROM `SQL_Casestudy.customers` c INNER JOIN `SQL_Casestudy.orders`
o ON c.customer_id = o.customer_id
  GROUP BY c.customer_state
  ORDER BY rn
 ) as T1
WHERE T1.rn < 6;
```



Row	Top_Bottom	Rank	customer_state	avg_DeliveryTime
1	Top	1	RR	29.0
2	Top	2	AP	27.0
3	Top	3	AM	26.0
4	Top	4	AL	24.0
5	Top	5	PA	23.0
6	Bottom	1	SP	8.0
7	Bottom	2	MG	12.0
8	Bottom	3	PR	12.0
9	Bottom	4	DF	13.0
10	Bottom	5	SC	14.0

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

```
SELECT
  customer_state,
  round(Table1.EstimatedDelivery,2) as Estimated_delivery
FROM
(
  SELECT
    c.customer_state,
    AVG(timestamp_diff(o.order_estimated_delivery_date,order_deli
vered_customer_date,day)) as EstimatedDelivery,
  FROM `SQL_Casestudy.orders`  o INNER JOIN
`SQL_Casestudy.customers` c ON o.customer_id = c.customer_id
  WHERE o.order_delivered_customer_date is not null AND
o.order_purchase_timestamp is not null AND
o.     order_estimated_delivery_date is not null AND
    o.order_status= 'delivered'
  GROUP BY customer_state

) as Table1
GROUP BY customer_state,EstimatedDelivery
ORDER BY EstimatedDelivery desc
LIMIT 5
```

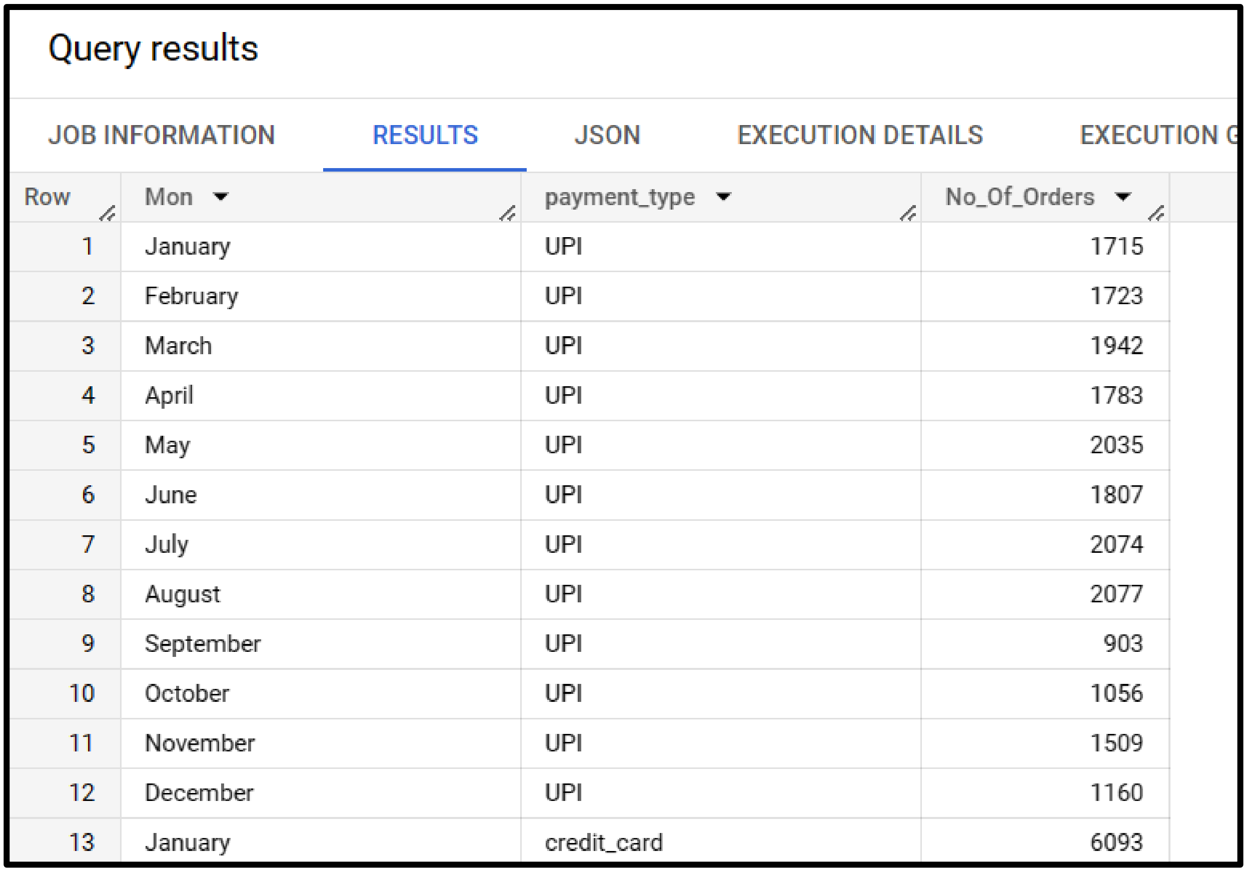


Row	customer_state	Estimated_delivery
1	AC	19.76
2	RO	19.13
3	AP	18.73
4	AM	18.61
5	RR	16.41

VI. Analysis based on the payments:

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

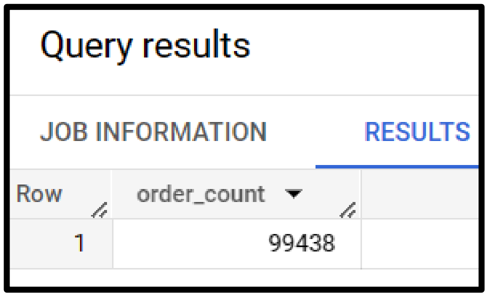
```
WITH CTE AS
(SELECT
  distinct p.payment_type as payment_type,
  extract(month from o.order_purchase_timestamp) as M_N,
  format_timestamp('%B', o.order_purchase_timestamp) as Mon,
  count(distinct o.order_id) as No_Of_Orders
FROM `SQL_Casestudy.payments` p INNER JOIN `SQL_Casestudy.orders`
o ON p.order_id = o.order_id
  GROUP BY Mon,p.payment_type,M_N
   ORDER BY p.payment_type,M_N,No_Of_Orders desc,Mon
)
SELECT Mon,payment_type,No_Of_Orders
FROM CTE
```



Row	Mon	payment_type	No_Of_Orders
1	January	UPI	1715
2	February	UPI	1723
3	March	UPI	1942
4	April	UPI	1783
5	May	UPI	2035
6	June	UPI	1807
7	July	UPI	2074
8	August	UPI	2077
9	September	UPI	903
10	October	UPI	1056
11	November	UPI	1509
12	December	UPI	1160
13	January	credit_card	6093

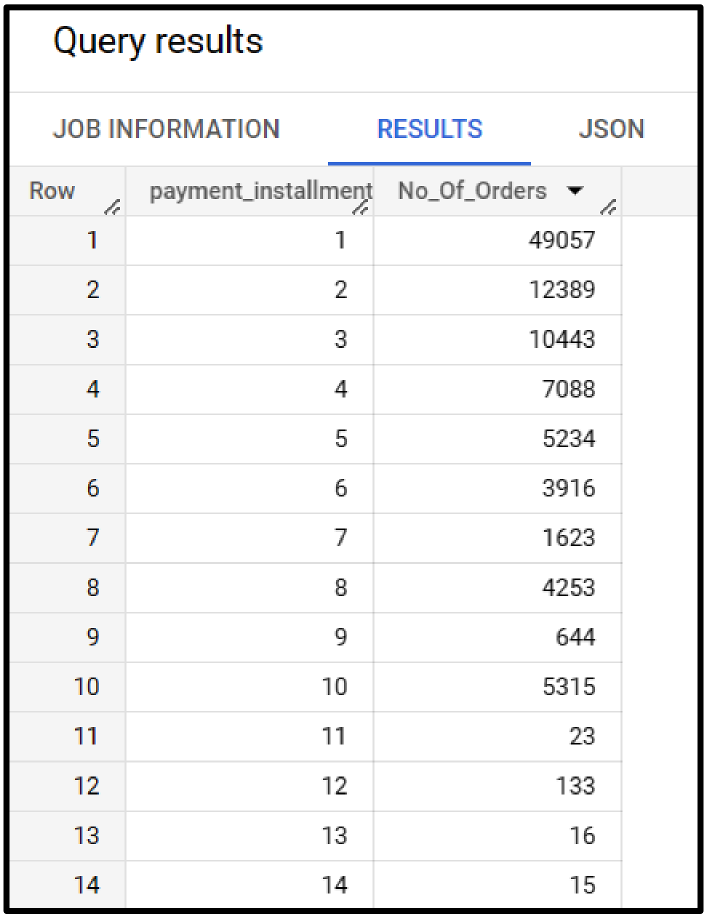
B. Find the no. of orders placed on the basis of the payment installments that have been paid.

```
SELECT
  COUNT(DISTINCT order_id) AS order_count
FROM
  `SQL_Casestudy.payments`
WHERE
  payment_installments >= 1
```



Row	order_count
1	99438

```
-- Installments wise
SELECT
    p.payment_installments,
    count(distinct o.order_id) as No_Of_Orders
  FROM `SQL_Casestudy.payments` p INNER JOIN
`SQL_Casestudy.orders` o ON p.order_id = o.order_id
  WHERE p.payment_installments > 0 AND p.payment_value > 0
  GROUP BY p.payment_installments
  ORDER BY payment_installments
```

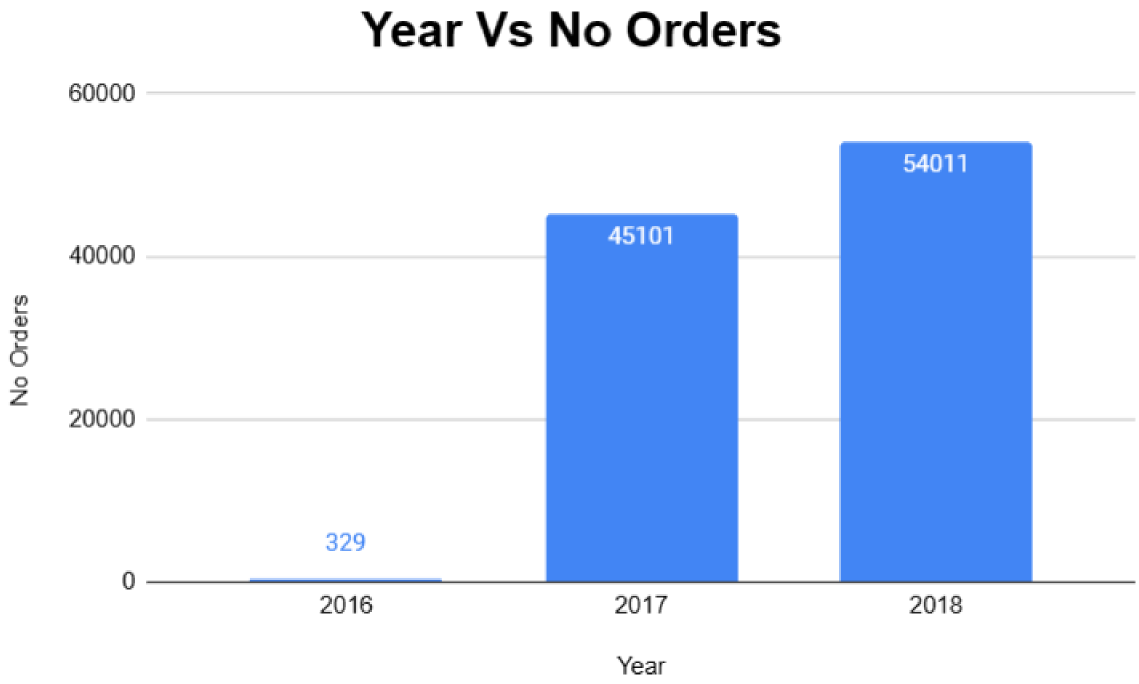


Row	payment_installment	No_Of_Orders
1	1	49057
2	2	12389
3	3	10443
4	4	7088
5	5	5234
6	6	3916
7	7	1623
8	8	4253
9	9	644
10	10	5315
11	11	23
12	12	133
13	13	16
14	14	15

Insights and Recommendations:

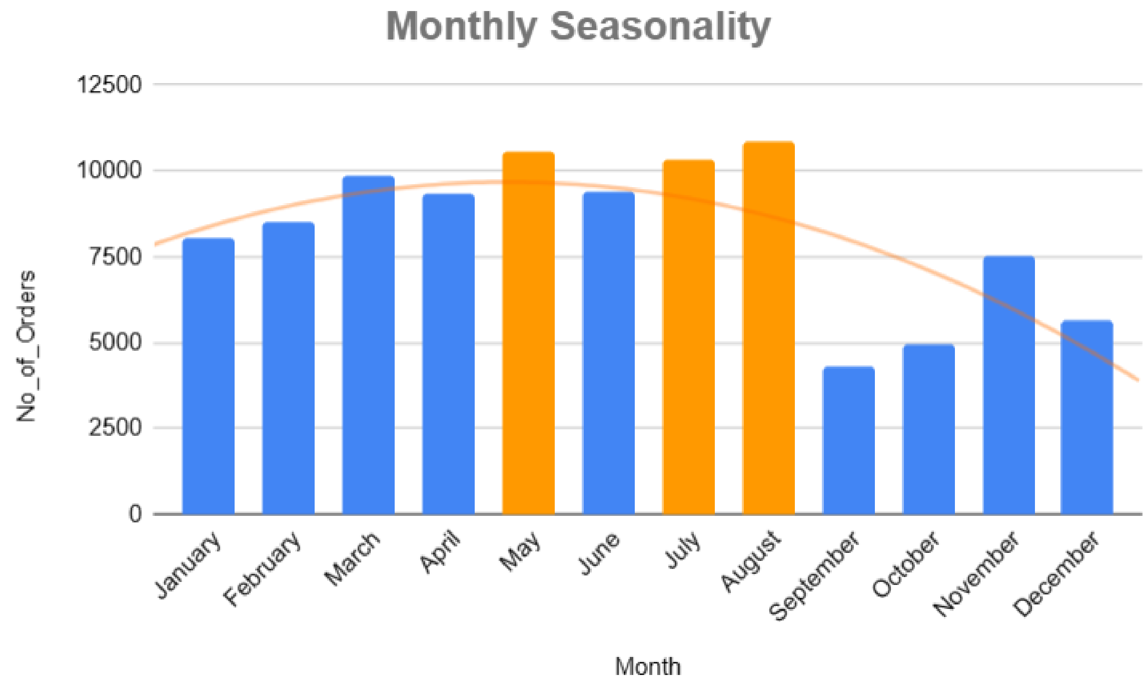
1. Year Wise Orders:

We can see clearly Increasing trend in all 3 years
With massive growth rate of **13608% in FY 2016-17** and well maintained **19% in FY 2017-18**

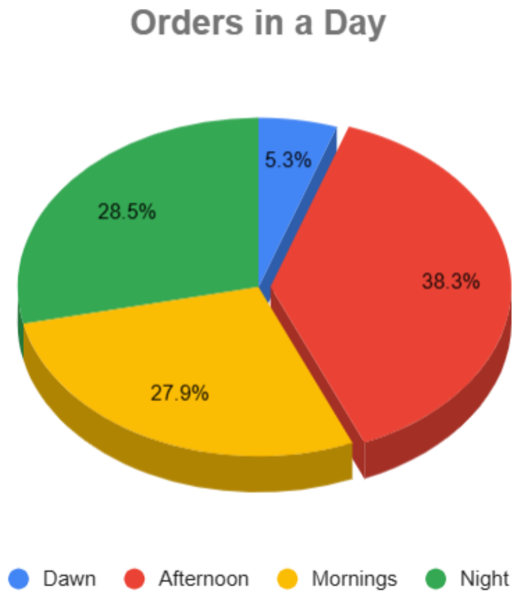


2. Monthly Seasonality

- We can clearly **Autumn & Mid-Winter** are most profitable seasons with **Median of 100K orders**
- **Spring is the Least profitable** season Avg of 500K Orders

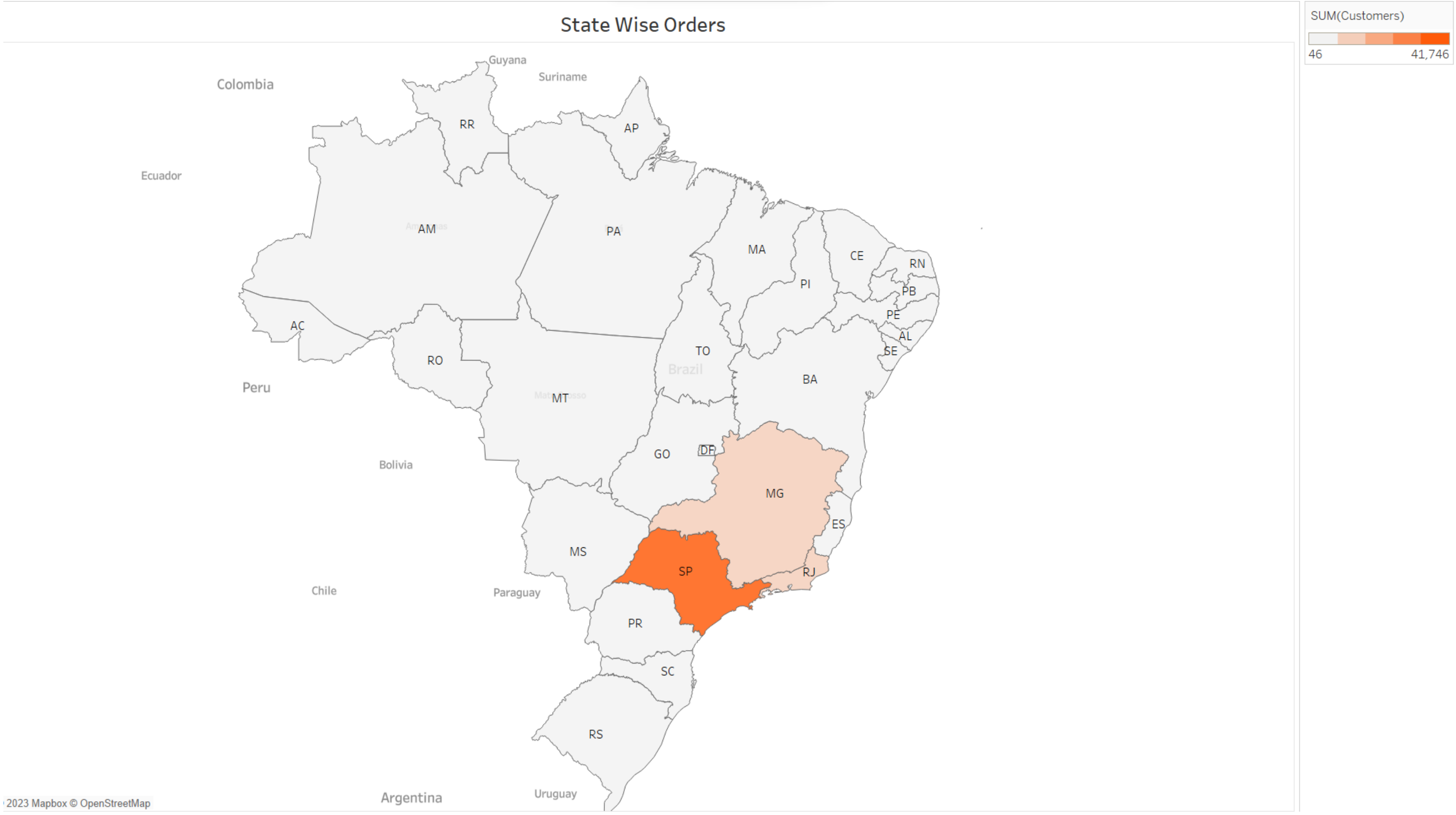


3. Orders in Day



4. State-wise customers distribution

São Paulo State has Highest Customers Followed by Minas Gerais since these 2 are the Highest Population States in Brazil

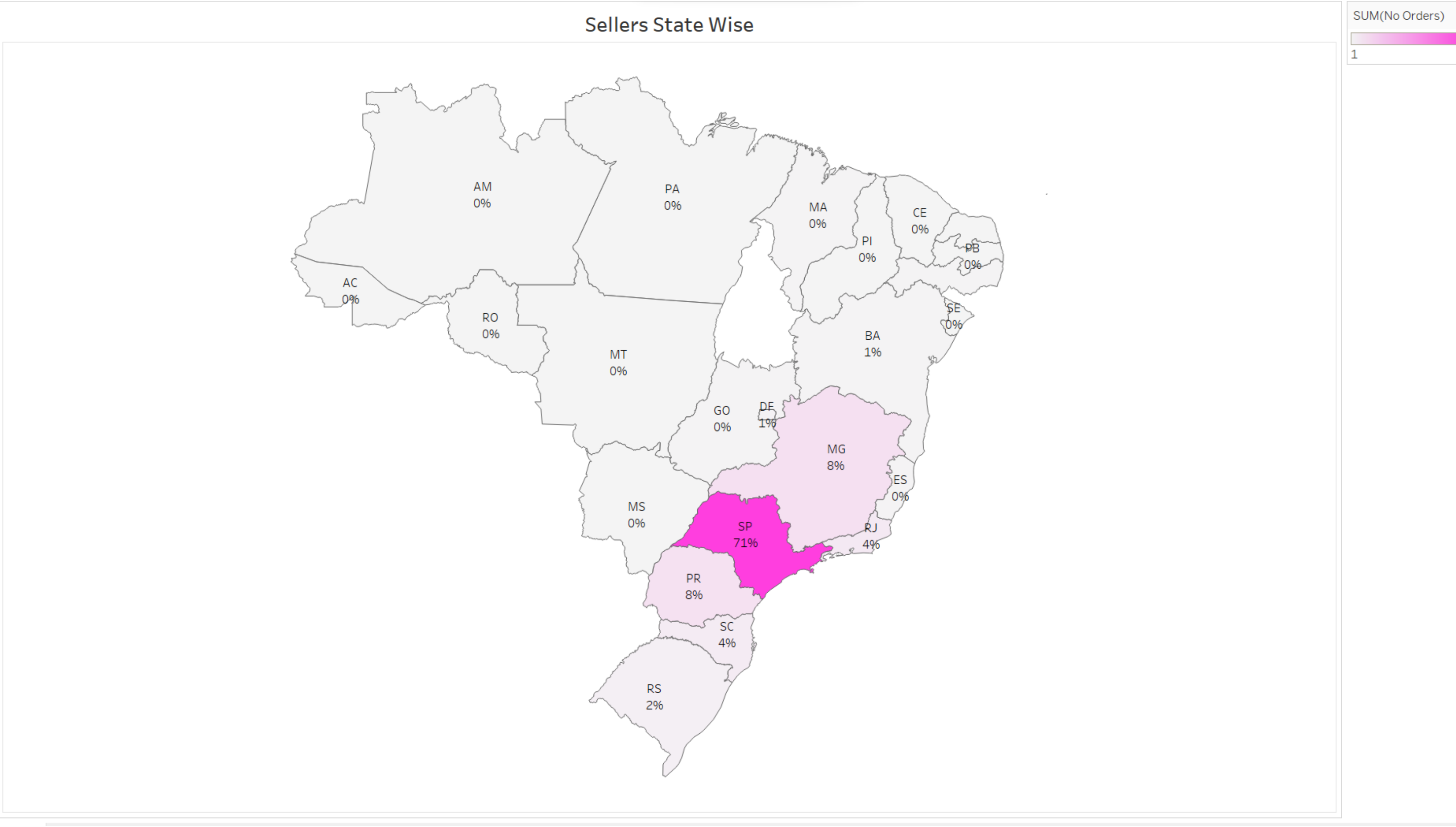


5. Sellers State wise

Among all states in Brazil, we can clearly see Sellers and Customers are from São Paulo State & Minas Gerais since these 2 are have largest customers for our business with almost **79% share**

Recommendation:

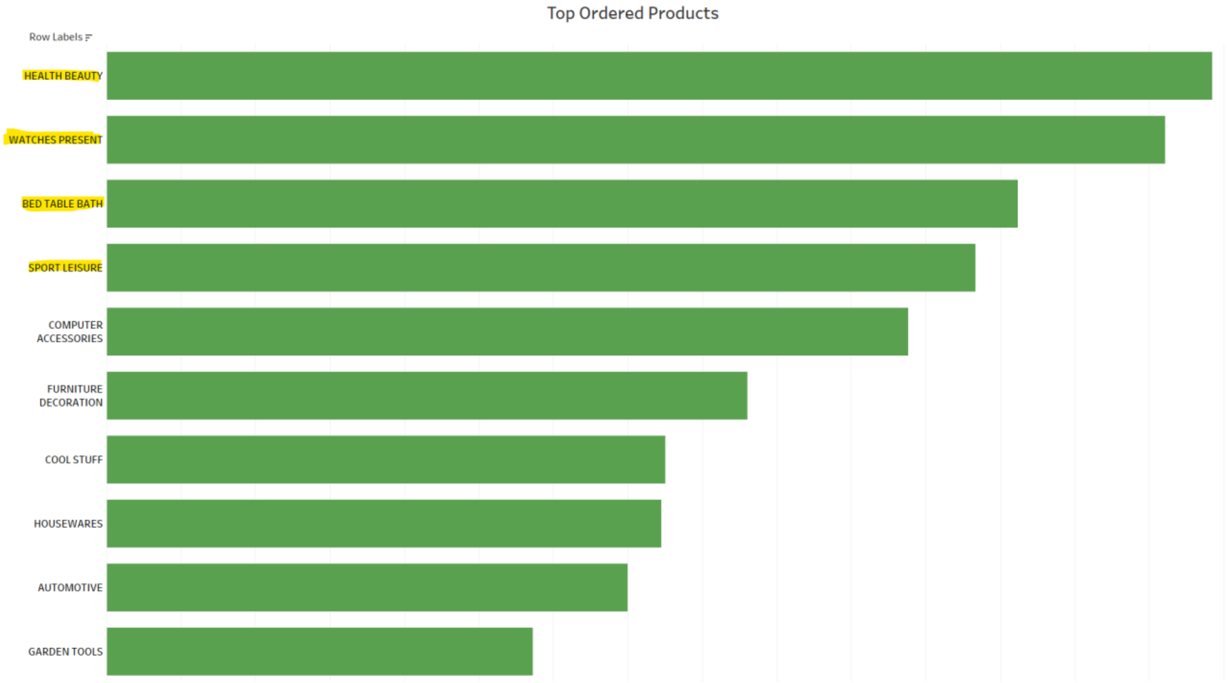
we have both sellers and customers are residing in SP& MG states
We have good scope to **increase our business to** nearby states in circular manner mostly **North Region** they don't have any contribution to our business in terms of customer and Revenue



6.Top Ordered Products

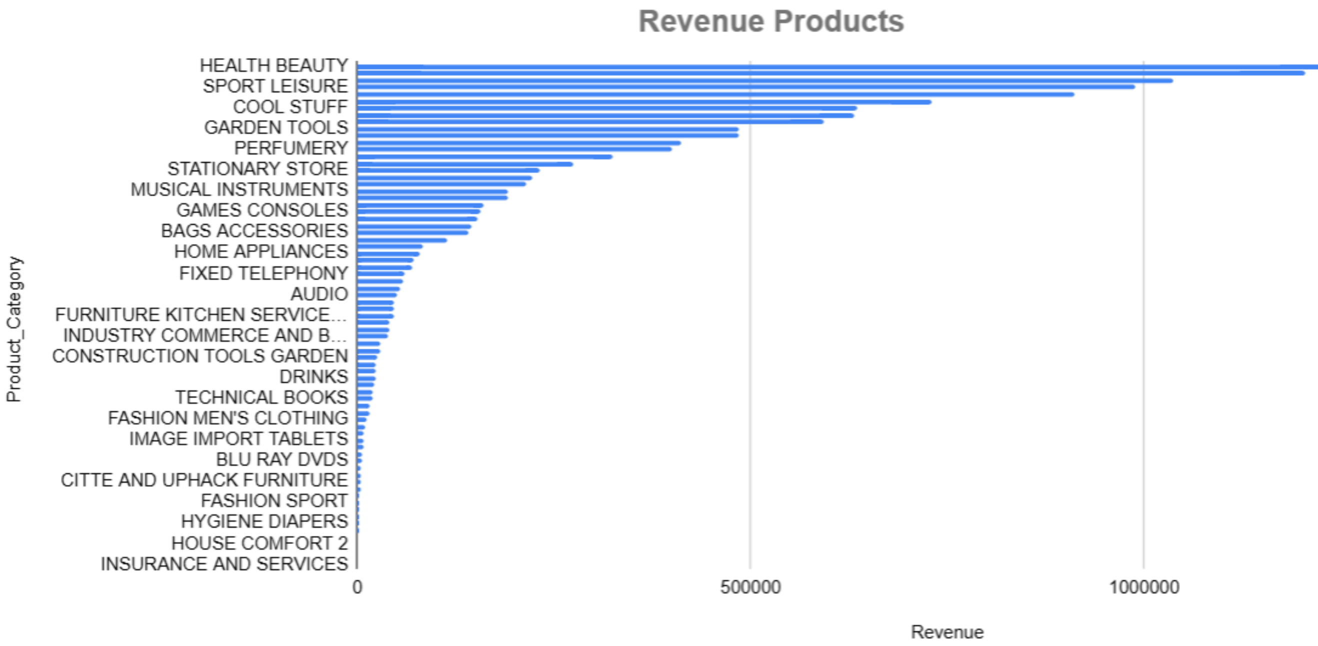
Among 73 Unique products

Personal Care products ordered most such as Healthcare, Beds etc.



7.Top Revenue Generated Products

In terms of Revenue most products are from Personal Care followed by Electronic Gadgets



Recommendations:

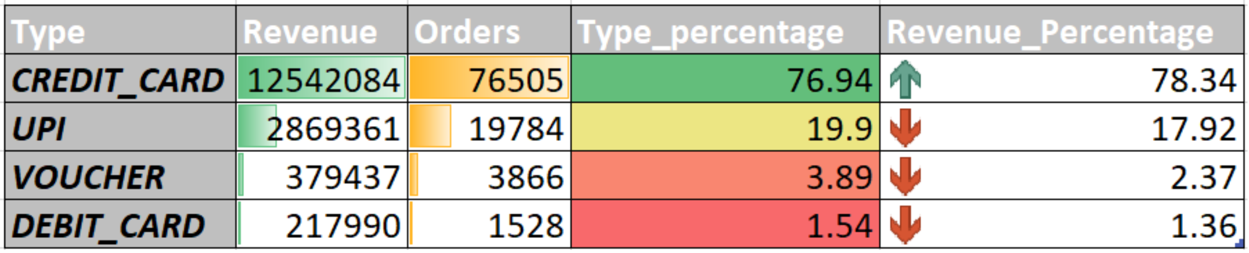
- There is evidence Customers are Buying Healthcare, Electronics and Personal care products more than **25%**
- In Terms of Revenue also **healthcare** products are Major contributors
- But if we observe products which are not contributing to revenue and less sold are occasional products

  We can do 3 things here

  a. We can give offers & promocodes on medium revenue generated products
  b. We can remove from cart which are least contributors and in special occasions we can promote that product to customers
  c. Most Important is we need to **deliver** the Top-Rated products **on time** so that our revenue growth consistent

8.Payments:

```
SELECT
  distinct UPPER(payment_type) as Type,
  round(sum(payment_value) over(partition by payment_type)) as Amount,
  count(distinct order_id) over(partition by payment_type) as
No_of_Orders,
  concat(round((count(distinct order_id) over(partition by
payment_type)/count(distinct order_id) over())*100,2),'%') as
Payment_Type_percentage,
  concat(round((sum(payment_value) over(partition by
payment_type)/sum(payment_value) over()*100),2),'%') as
Amount_Percentage
FROM `scaler-sql-sessions.SQL_Casestudy.payments`
WHERE payment_type <> 'not_defined'
order by Amount desc
```



Type	Revenue	Orders	Type_percentage	Revenue_Percentage
CREDIT_CARD	12542084	76505	76.94	↑ 78.34
UPI	2869361	19784	19.9	↓ 17.92
VOUCHER	379437	3866	3.89	↓ 2.37
DEBIT_CARD	217990	1528	1.54	↓ 1.36

Recommendations:

- **In** Payments Credit Card is leading payment gateway with >75% orders & Revenue Percentage that’s good sign
- UPI is Second place with approx. 20% share there is **huge advantage** we need to promote UPI on our platform because of its features like
  a. Globally Adaptable
  b. Instant payment gateway
  c. No extra charges
  d. We can target customers who are not using Credit and Debit cards
- We can add EMI and Buy now Pay later option to **attract Young People**
- If we follow above points, we can easily increase **voucher** payment **revenue** automatically