Business Case Study

Company Name **:** TARGET

Data collected between 2016 & 2018

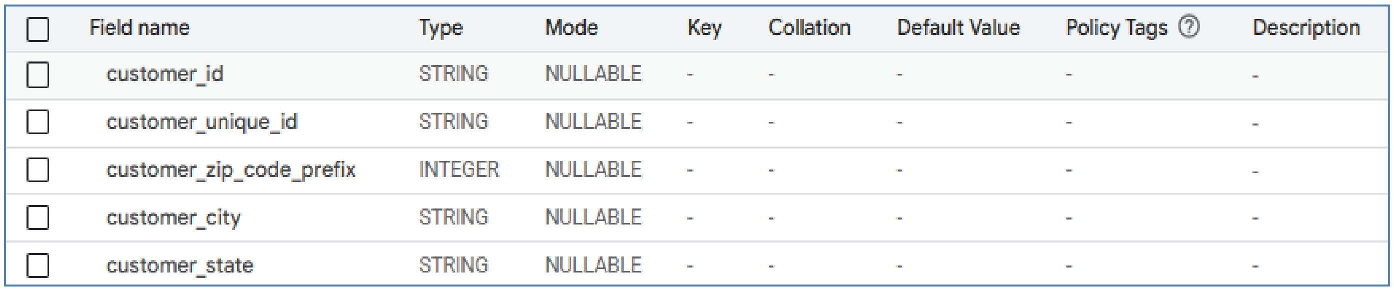
Problem Statement:

Assuming you are a data analyst/ scientist at Target, you have been assigned the task of analyzing the given dataset to extract valuable insights and provide actionable recommendations.

What does 'good' look like?

1. **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.**

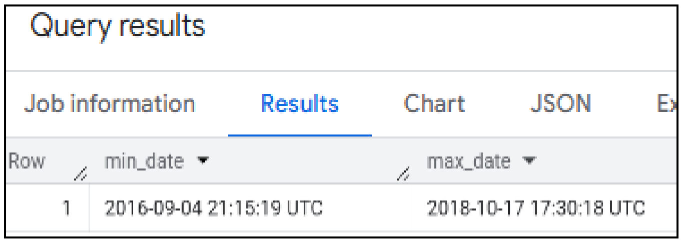


☐	Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Description
☐	customer_id	STRING	NULLABLE	-	-	-	-	-
☐	customer_unique_id	STRING	NULLABLE	-	-	-	-	-
☐	customer_zip_code_prefix	INTEGER	NULLABLE	-	-	-	-	-
☐	customer_city	STRING	NULLABLE	-	-	-	-	-
☐	customer_state	STRING	NULLABLE	-	-	-	-	-

- Four columns of string data type (**customer_id, customer_unique_id, customer_city, customer_state**)
- One column of integer data type (**customer_zip_code_prefix**)

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

```
SELECT
    MIN(order_purchase_timestamp) AS min_date,
    MAX(order_purchase_timestamp) AS max_date
FROM
  `target.orders`
```

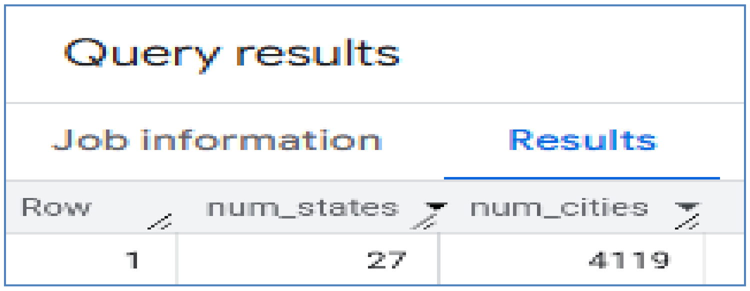


```
Row  min_date                  max_date
1    2016-09-04 21:15:19 UTC   2018-10-17 17:30:18 UTC
```

First order was placed on 04 Sep 2016 and last order on 17 Oct 2018 between year 2016 and 2018.

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

```
SELECT
    COUNT (DISTINCT customer_state) AS num_states,
    COUNT (DISTINCT customer_city) AS num_cities
FROM
    `target.customers` c JOIN
`target.orders` o ON c.customer_id = o.customer_id
```


Query results

Job information | Results

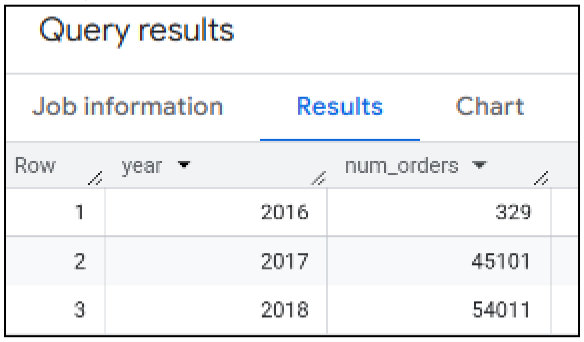
Row	num_states	num_cities
1	27	4119

Customers from all states across 4119 cities in Brazil ordered during the given period.

2. In-depth-Exploration:

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

```
SELECT
      EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
      COUNT(*) as num_orders
FROM
  `target.orders`
GROUP BY 1
ORDER BY 1;
```


Query results

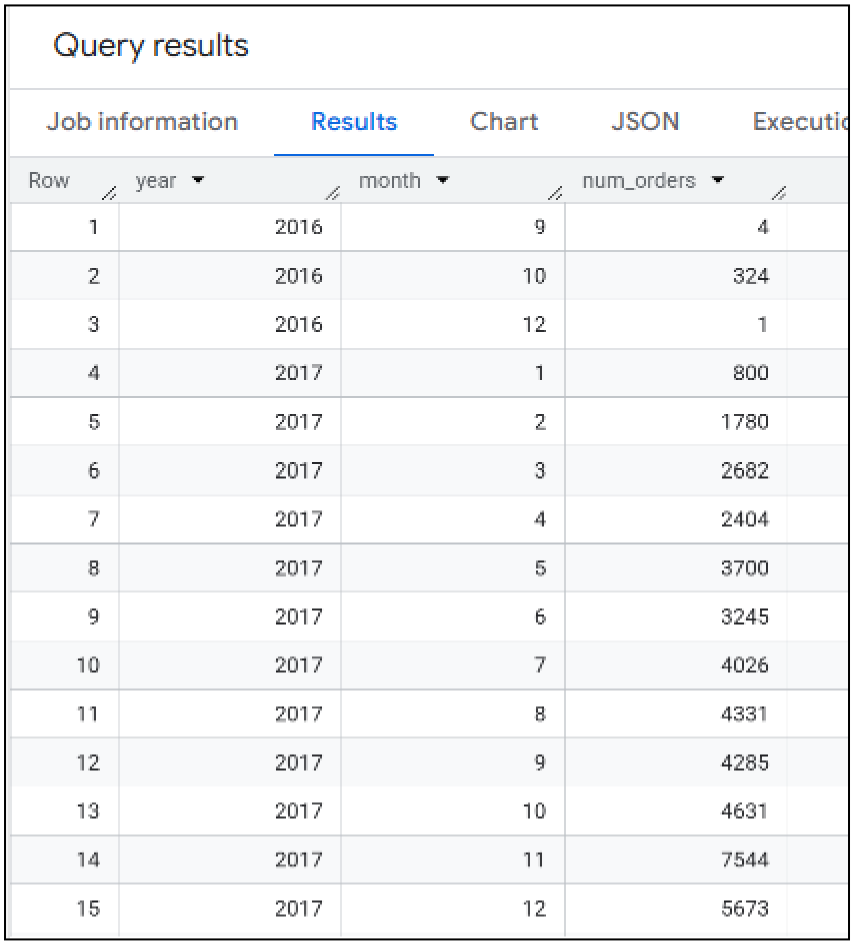
Job information | Results | Chart

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

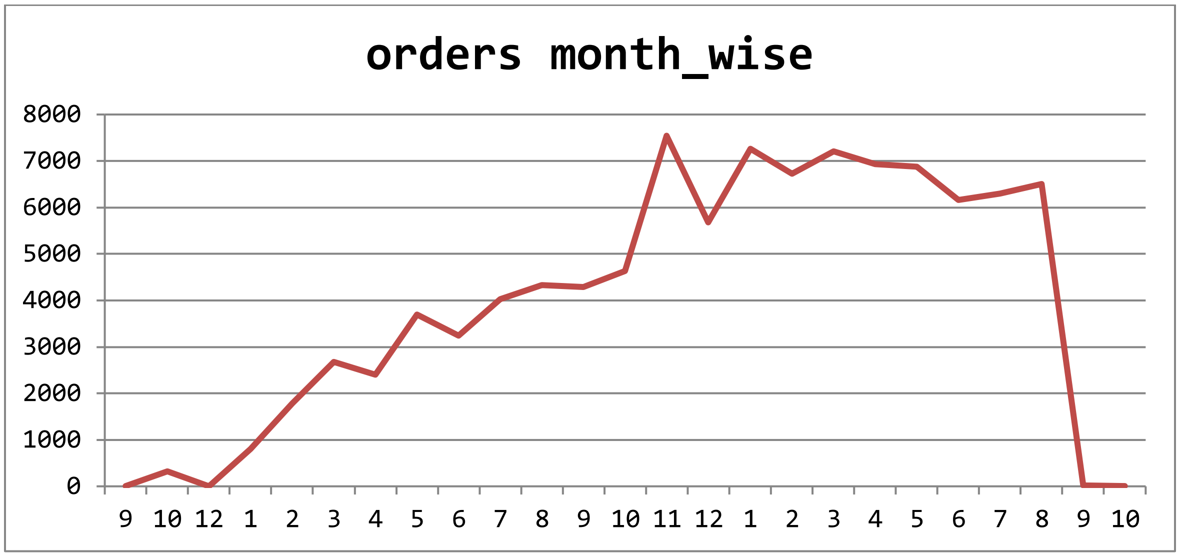
Yes, there is a growing trend in the number of orders placed over the past years. Year 2017 has seen a **12463%** rise in the orders placed compared to year 2016. Year 2018 has seen a **19.75 %** rise in the orders placed compared to year 2017.(**Note** :2016 has only 3 months available for calculation)

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

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



Row	year	month	num_orders
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
14	2017	11	7544
15	2017	12	5673

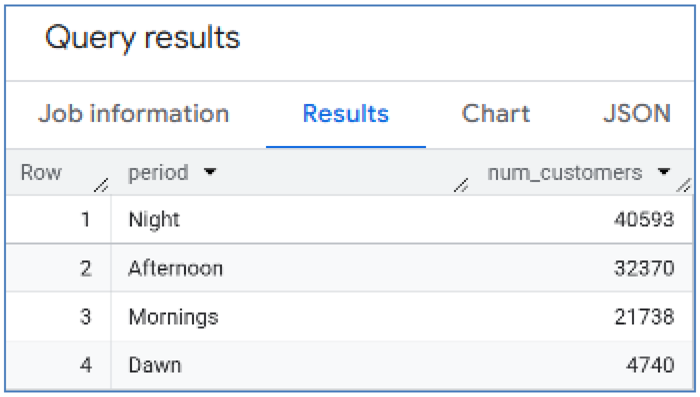


There is no visible monthly trend in the orders placed. However months of July, August have better sales in the past 2 years.

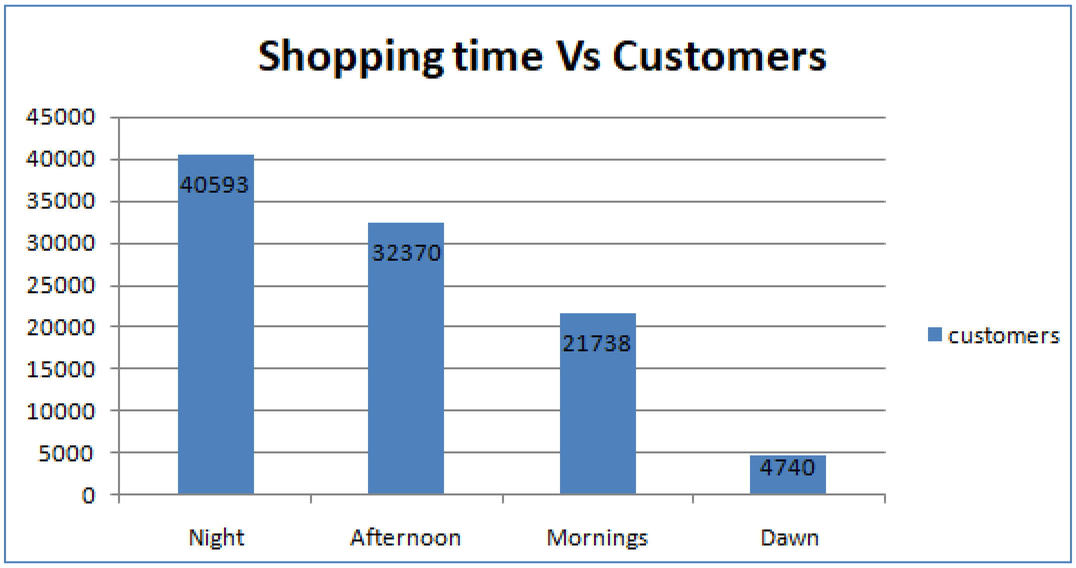
3. **During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)**

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

```
SELECT
      period,
      COUNT(*)AS num_customers
FROM
      (SELECT
        customer_id,

        CASE WHEN(EXTRACT(time FROM order_purchase_timestamp ))
      BETWEEN '00:00:00' AND '06:00:00' THEN 'Dawn'
             WHEN(EXTRACT(time FROM order_purchase_timestamp ))
      BETWEEN '07:00:00' AND '12:00:00' THEN 'Mornings'
             WHEN(EXTRACT(time FROM order_purchase_timestamp ))
      BETWEEN '13:00:00' AND '18:00:00' THEN 'Afternoon'
             ELSE 'Night'   END AS period
      FROM
        `target.orders`)
GROUP BY 1
ORDER BY num_customers DESC;
```


Query results

Row	period	num_customers
1	Night	40593
2	Afternoon	32370
3	Mornings	21738
4	Dawn	4740



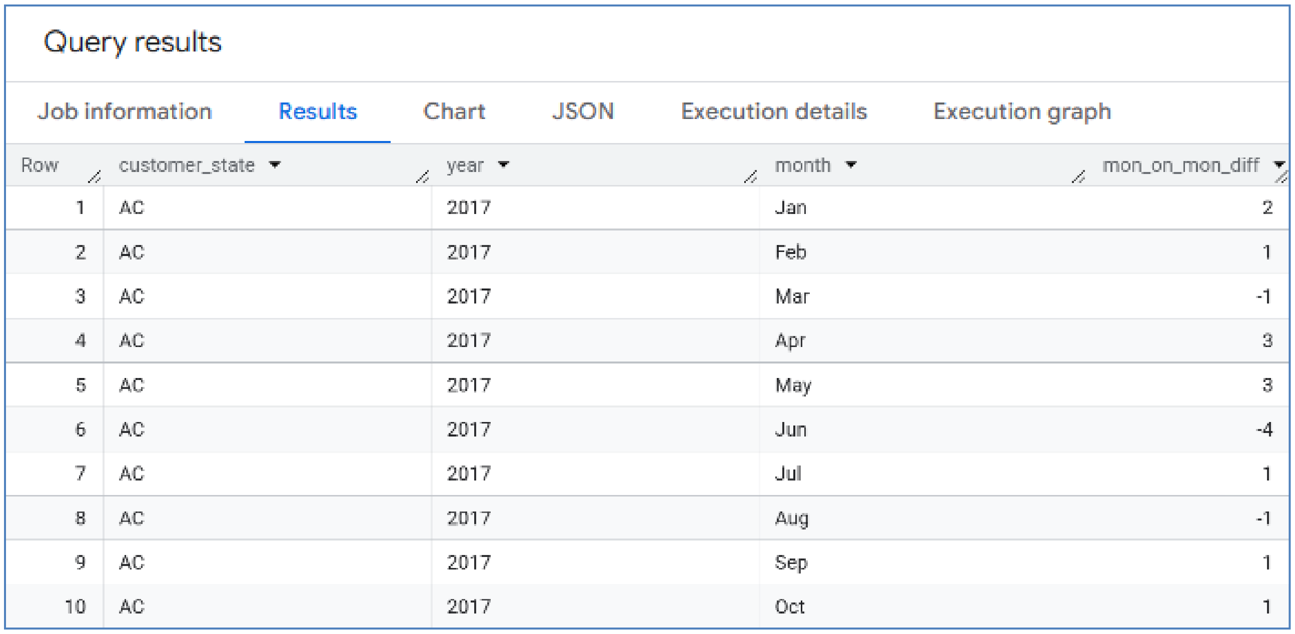
Most of the customers prefer to shop in the night preferably after office timings and also in afternoon period. Please ensure that staff availability and stock is sufficient in these periods to cater for the rush of customers.

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

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

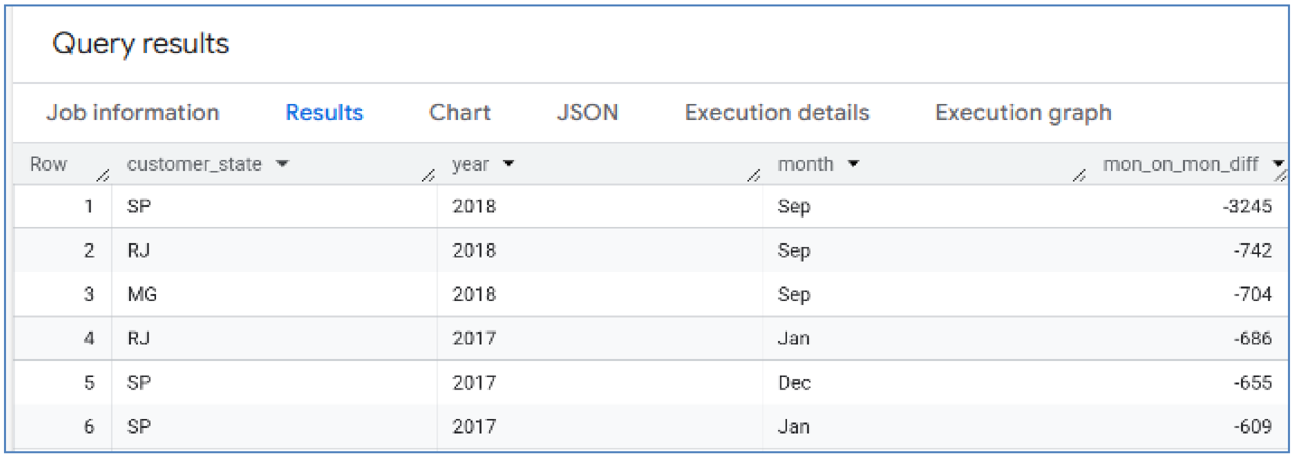
```
WITH base AS
          ( SELECT
              COUNT(DISTINCT o.order_id) AS order_count,
              FORMAT_DATETIME("%b", order_purchase_timestamp) AS month,
              FORMAT_DATETIME("%Y", order_purchase_timestamp) AS year,
              customer_state
           FROM
                `target.orders`o
              LEFT JOIN `target.order_items` oi ON
                     o.order_id = oi.order_id
              LEFT JOIN `target.customers` c ON
```

```
                          o.customer_id = c.customer_id
                GROUP BY 2,3,4
                ORDER BY 4,3,2
                ),

filter1 AS
        (SELECT
               *,

         FROM base
         ORDER BY month,year),

filter2 AS
        (SELECT
               *,
         FROM filter1
         ORDER BY customer_state,year),

filter3 AS
        (SELECT

              customer_state,

              year,

              month,

              order_count AS month1,

             LAG(order_count) over (PARTITION BY customer_state ORDER BY

             year) AS month2,

         FROM filter2

         ORDER BY customer_state),


final AS
        (SELECT
              *,
              IFNULL(month2,0)AS mod_month2
         FROM filter3)

SELECT
     customer_state,
     year,
     month,
     month1 -mod_month2 AS mon_on_mon_diff
FROM final
```


Query results

Job information | Results | Chart | JSON | Execution details | Execution graph

Row	customer_state	year	month	mon_on_mon_diff
1	AC	2017	Jan	2
2	AC	2017	Feb	1
3	AC	2017	Mar	-1
4	AC	2017	Apr	3
5	AC	2017	May	3
6	AC	2017	Jun	-4
7	AC	2017	Jul	1
8	AC	2017	Aug	-1
9	AC	2017	Sep	1
10	AC	2017	Oct	1


Query results

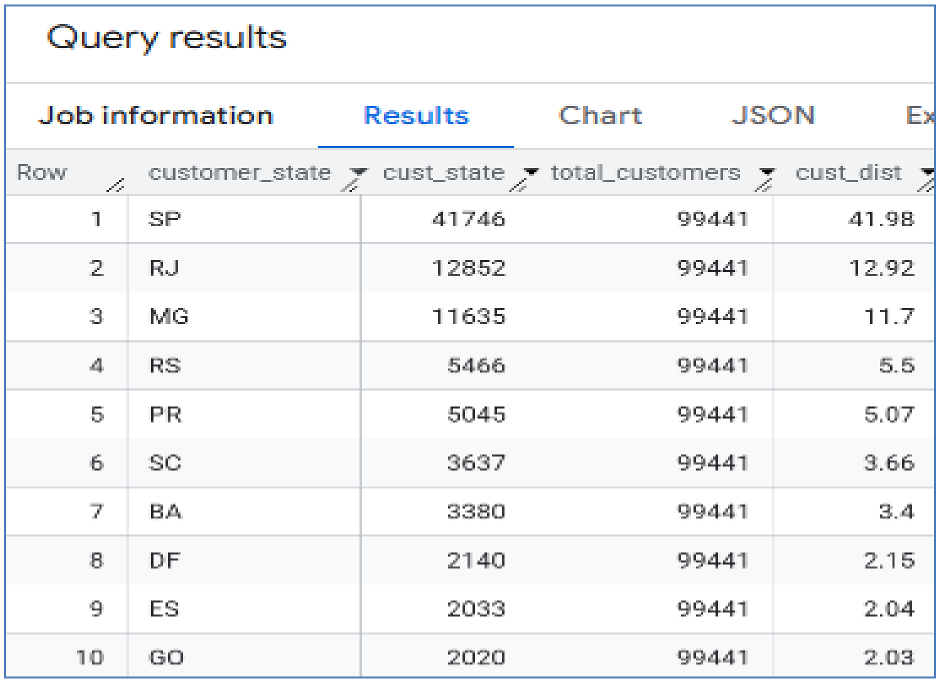
Job information | Results | Chart | JSON | Execution details | Execution graph

Row	customer_state	year	month	mon_on_mon_diff
1	SP	2018	Sep	-3245
2	RJ	2018	Sep	-742
3	MG	2018	Sep	-704
4	RJ	2017	Jan	-686
5	SP	2017	Dec	-655
6	SP	2017	Jan	-609

(SP)Sao Paulo has the worst month-on-month order difference (-3245) followed by (RJ)Rio de Janeiro (-742) and (MG)Minas Garais(-704)

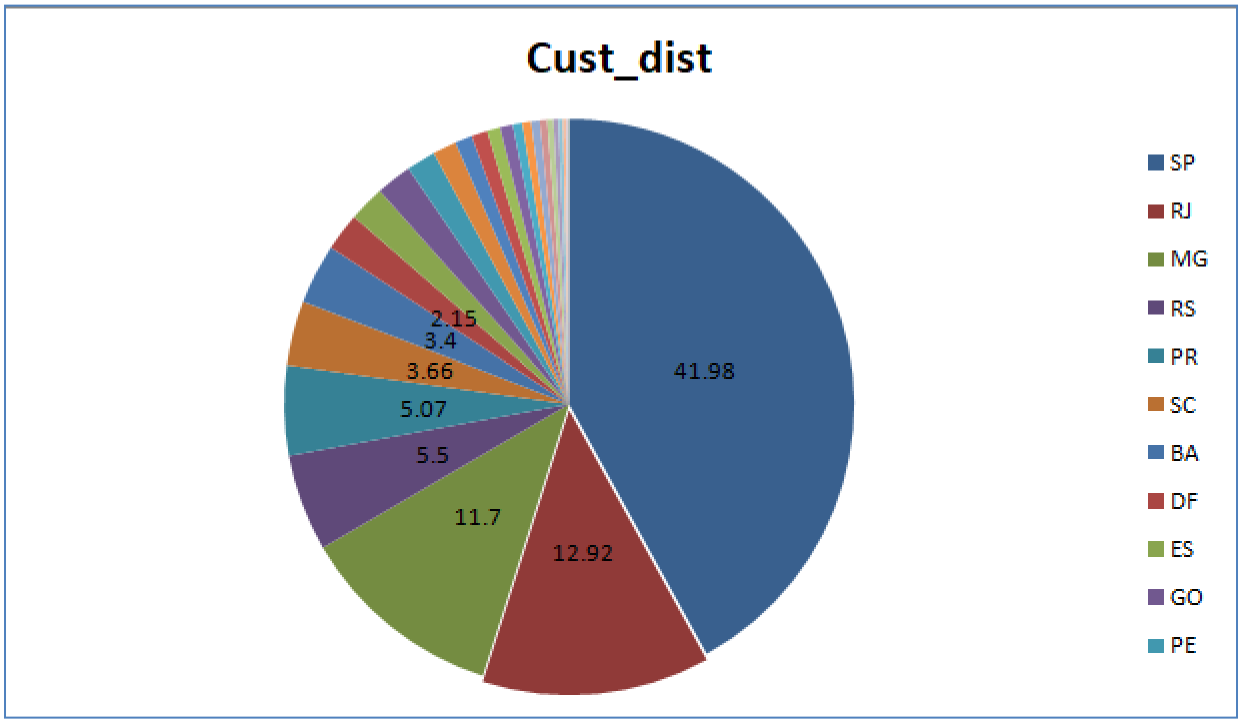
2. How are the customers distributed across all the states?

```
SELECT
   *,
   ROUND((cust_state/total_customers)*100,2) AS cust_dist
FROM
   ( SELECT
      DISTINCT customer_state,
      COUNT(customer_id) OVER (PARTITION BY customer_state)AS cust_state,

      COUNT(customer_id) OVER () AS total_customers
     FROM
        `target.customers`)
ORDER BY cust_dist DESC;
```


Query results

Job information | Results | Chart | JSON | Ex

Row	customer_state	cust_state	total_customers	cust_dist
1	SP	41746	99441	41.98
2	RJ	12852	99441	12.92
3	MG	11635	99441	11.7
4	RS	5466	99441	5.5
5	PR	5045	99441	5.07
6	SC	3637	99441	3.66
7	BA	3380	99441	3.4
8	DF	2140	99441	2.15
9	ES	2033	99441	2.04
10	GO	2020	99441	2.03

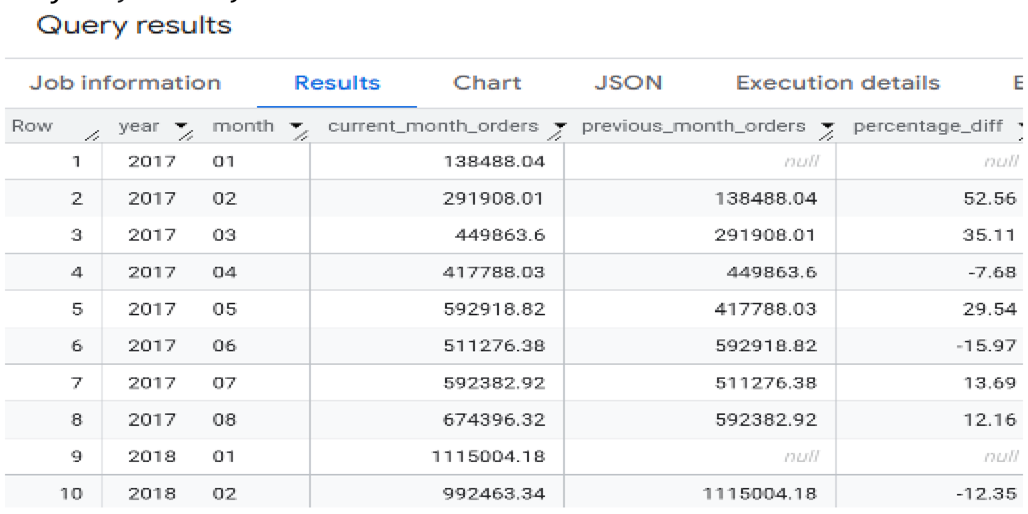


(SP)Sao Paulo has the highest customer-base followed by (RJ)Rio de Janeiro (MG)Minas Garais. But ironically these three states have the lowest month-on-month orders. Customers in these states are not happy with the service which should be looked into.

4. **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). You can use the "payment_value" column in the payments table to get the cost of orders.

```
SELECT
 year,
 month,
 current_month_orders,
 previous_month_orders,
    ROUND(((current_month_orders - previous_month_orders)/current_month_orders) *
                                                  100,2)             AS percentage_diff
FROM
   (SELECT
      *,
      LAG(current_month_orders,1) OVER (PARTITION BY year ORDER BY month) AS
      previous_month_orders
   FROM
      (SELECT * FROM
            (SELECT
            EXTRACT(year from order_purchase_timestamp) AS year,
            FORMAT_DATETIME("%m",order_purchase_timestamp ) AS month,
            ROUND(SUM(payment_value),2) AS current_month_orders
            FROM
            `target.orders` o
            LEFT JOIN `target.payments`p ON
            o.order_id = p.order_id
            GROUP BY 1,2
            )
      WHERE year IN (2017,2018) AND month IN (
                                                    SELECT
                                                    month
                                                    FROM target.orders
                                                    WHERE month BETWEEN "01" AND "08"
                                                    )))
      ORDER BY year,month ;
```


Query results

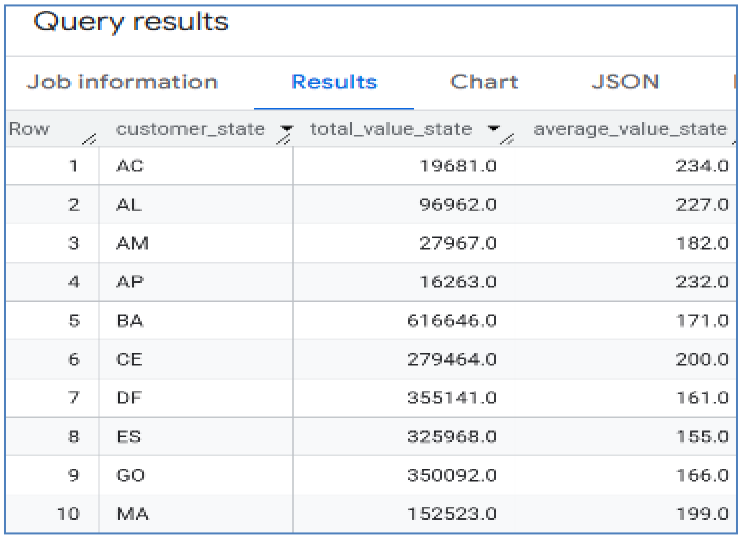
Job information | Results | Chart | JSON | Execution details | E

Row	year	month	current_month_orders	previous_month_orders	percentage_diff
1	2017	01	138488.04	null	null
2	2017	02	291908.01	138488.04	52.56
3	2017	03	449863.6	291908.01	35.11
4	2017	04	417788.03	449863.6	-7.68
5	2017	05	592918.82	417788.03	29.54
6	2017	06	511276.38	592918.82	-15.97
7	2017	07	592382.92	511276.38	13.69
8	2017	08	674396.32	592382.92	12.16
9	2018	01	1115004.18	null	null
10	2018	02	992463.34	1115004.18	-12.35

In year 2017, month of February saw highest percentage increase in orders and in year 2018, only March saw a positive increase in orders, whereas rest of the months orders decreased compared to previous month.

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

```
SELECT
      customer_state,
      ROUND(SUM(payment_value))AS total_value_state,
      ROUND(AVG(payment_value))AS average_value_state
FROM `target.customers`c
JOIN `target.orders`o ON c.customer_id = o.customer_id
JOIN `target.payments`p ON o.order_id = p.order_id
GROUP BY 1
ORDER BY 1;
```

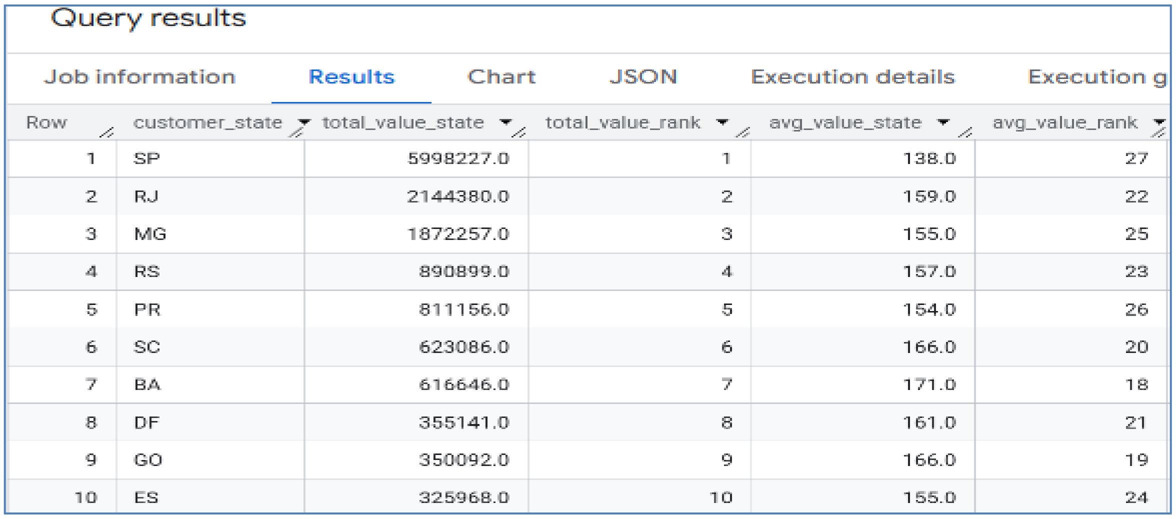

Query results

Job information | Results | Chart | JSON

Row	customer_state	total_value_state	average_value_state
1	AC	19681.0	234.0
2	AL	96962.0	227.0
3	AM	27967.0	182.0
4	AP	16263.0	232.0
5	BA	616646.0	171.0
6	CE	279464.0	200.0
7	DF	355141.0	161.0
8	ES	325968.0	155.0
9	GO	350092.0	166.0
10	MA	152523.0	199.0

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

```
WITH base AS
     (SELECT
           customer_state,
           payment_value AS order_price
     FROM `target.customers`c
     JOIN `target.orders`o ON c.customer_id = o.customer_id
     JOIN `target.payments`p ON o.order_id = p.order_id
     ),

total_value AS
     (SELECT
           customer_state,
            ROUND(SUM(order_price))AS total_value_state
      FROM base
      GROUP BY 1
      ),

rank1 AS
      (SELECT
            *,
            ROW_NUMBER() OVER (ORDER BY total_value_state DESC) AS
total_value_rank
            FROM total_value
      ),

avg_value AS
      (SELECT
            customer_state,
            ROUND(AVG(order_price))AS avg_value_state
      FROM base
      GROUP BY 1
      ORDER BY 2 DESC),

rank2 AS
      (SELECT
            *,
            ROW_NUMBER() OVER (ORDER BY avg_value_state DESC) AS
avg_value_rank
      FROM avg_value
      )

SELECT
    r1.customer_state,
    total_value_state,
    total_value_rank,
    avg_value_state,
    avg_value_rank
FROM rank1 r1
FULL OUTER JOIN rank2 r2 ON
r1.customer_state = r2.customer_state
ORDER BY 3;
```



Query results

Row	customer_state	total_value_state	total_value_rank	avg_value_state	avg_value_rank
1	SP	5998227.0	1	138.0	27
2	RJ	2144380.0	2	159.0	22
3	MG	1872257.0	3	155.0	25
4	RS	890899.0	4	157.0	23
5	PR	811156.0	5	154.0	26
6	SC	623086.0	6	166.0	20
7	BA	616646.0	7	171.0	18
8	DF	355141.0	8	161.0	21
9	GO	350092.0	9	166.0	19
10	ES	325968.0	10	155.0	24

Top 3 Countries having maximum orders are not having good record in average orders as these countries have higher customer base compare to other countries.

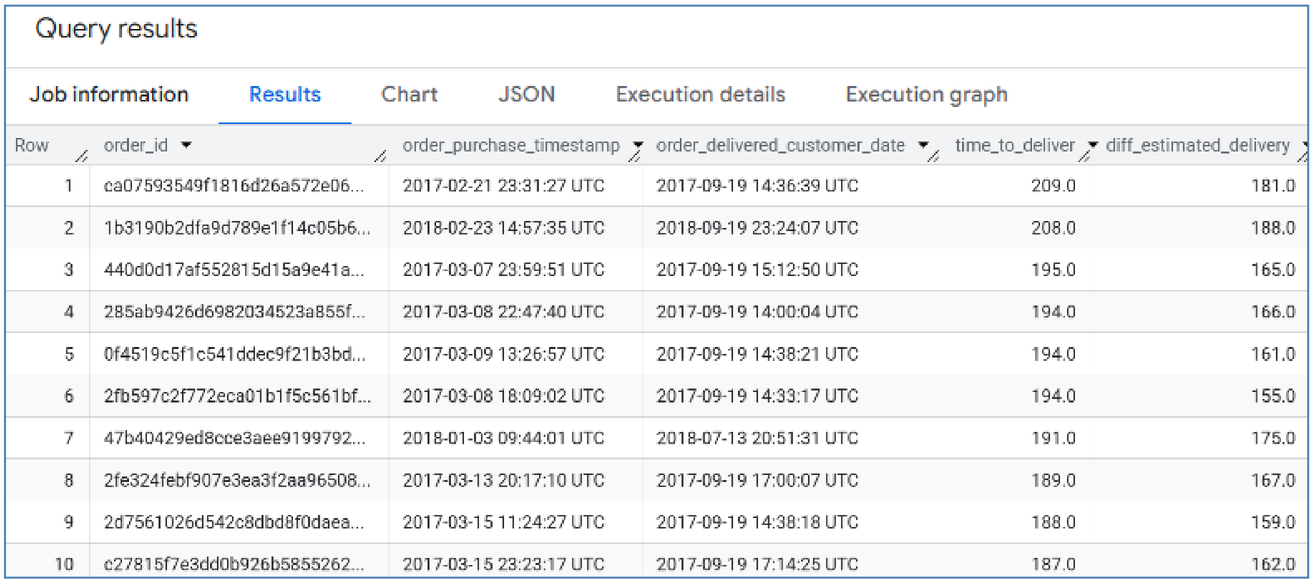
5. **Analysis based on sales, freight and delivery time.**

   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. Do this in a single query**.

      You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

      - **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
      - **diff_estimated_delivery** = order_delivered_customer_date - order_estimated_delivery_date

```
SELECT
    order_id,
    order_purchase_timestamp,
    ROUND(TIMESTAMP_DIFF(order_delivered_customer_date,order_pu
rchase_timestamp,DAY)) AS time_to_deliver,
    ROUND(TIMESTAMP_DIFF(order_delivered_customer_date,order_es
timated_delivery_date,DAY)) AS diff_estimated_delivery
FROM `target.orders`
WHERE
   order_status = 'delivered'
ORDER BY 4 DESC;
```

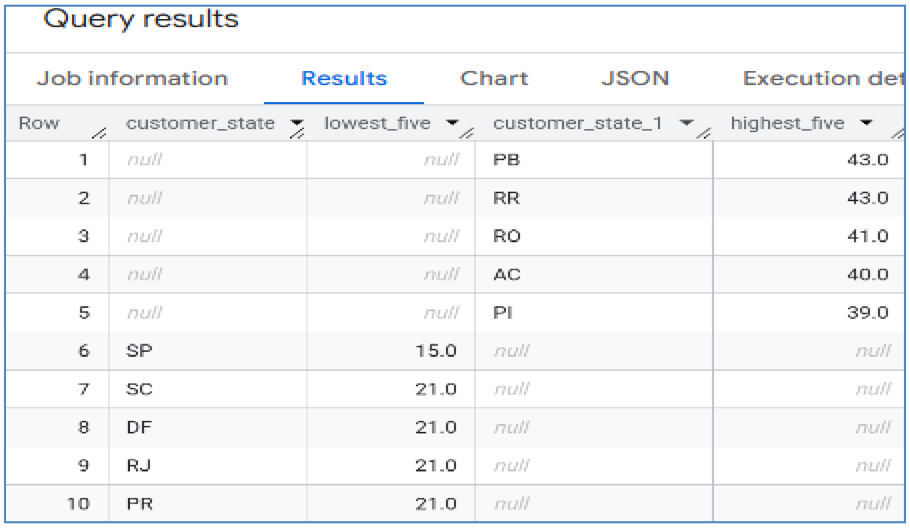


Row	order_id	order_purchase_timestamp	order_delivered_customer_date	time_to_deliver	diff_estimated_delivery
1	ca07593549f1816d26a572e06...	2017-02-21 23:31:27 UTC	2017-09-19 14:36:39 UTC	209.0	181.0
2	1b3190b2dfa9d789e1f14c05b6...	2018-02-23 14:57:35 UTC	2018-09-19 23:24:07 UTC	208.0	188.0
3	440d0d17af552815d15a9e41a...	2017-03-07 23:59:51 UTC	2017-09-19 15:12:50 UTC	195.0	165.0
4	285ab9426d6982034523a855f...	2017-03-08 22:47:40 UTC	2017-09-19 14:00:04 UTC	194.0	166.0
5	0f4519c5f1c541ddec9f21b3bd...	2017-03-09 13:26:57 UTC	2017-09-19 14:38:21 UTC	194.0	161.0
6	2fb597c2f772eca01b1f5c561bf...	2017-03-08 18:09:02 UTC	2017-09-19 14:33:17 UTC	194.0	155.0
7	47b40429ed8cce3aee9199792...	2018-01-03 09:44:01 UTC	2018-07-13 20:51:31 UTC	191.0	175.0
8	2fe324febf907e3ea3f2aa96508...	2017-03-13 20:17:10 UTC	2017-09-19 17:00:07 UTC	189.0	167.0
9	2d7561026d542c8dbd8f0daea...	2017-03-15 11:24:27 UTC	2017-09-19 14:38:18 UTC	188.0	159.0
10	c27815f7e3dd0b926b5855262...	2017-03-15 23:23:17 UTC	2017-09-19 17:14:25 UTC	187.0	162.0

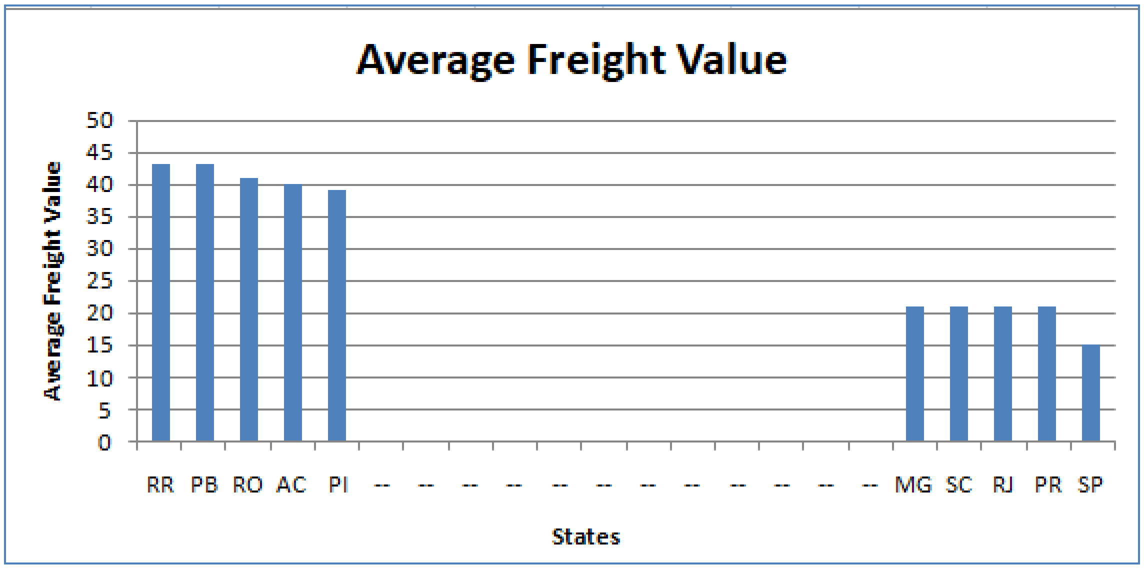
Some orders have a huge difference between estimated delivery date and actual delivery date which is a big concern and all efforts should be made to reduce **diff_estimated_delivery.**

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

```
WITH min_avg AS
        (
          SELECT
            c.customer_state,
            ROUND(AVG(freight_value)) AS lowest_five
          FROM `target.customers`c JOIN `target.orders` o ON
            c.customer_id = o.customer_id
          JOIN `target.order_items`oi ON
            o.order_id = oi.order_id
          GROUP BY 1
          ORDER BY 2
          LIMIT 5
        ),
max_avg AS
        (
          SELECT
            c.customer_state,
            ROUND(AVG(freight_value)) AS highest_five
          FROM `target.customers`c JOIN `target.orders` o ON
            c.customer_id = o.customer_id
          JOIN `target.order_items`oi ON
            o.order_id = oi.order_id
          GROUP BY 1
          ORDER BY 2 DESC
          LIMIT 5
        )
   SELECT avg1.*, avg2.*
FROM min_avg avg1 LEFT JOIN max_avg avg2 ON
                   avg1.customer_state=avg2.customer_state
UNION DISTINCT
   SELECT avg1.*, avg2.*
FROM min_avg avg1 RIGHT JOIN max_avg avg2 ON
                   avg1.customer_state=avg2.customer_state;
```

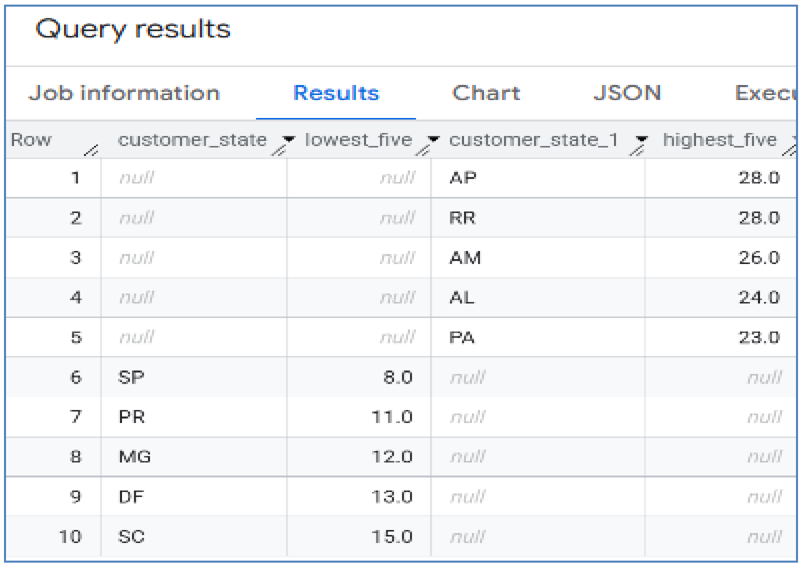


Row	customer_state	lowest_five	customer_state_1	highest_five
1	null	null	PB	43.0
2	null	null	RR	43.0
3	null	null	RO	41.0
4	null	null	AC	40.0
5	null	null	PI	39.0
6	SP	15.0	null	null
7	SC	21.0	null	null
8	DF	21.0	null	null
9	RJ	21.0	null	null
10	PR	21.0	null	null

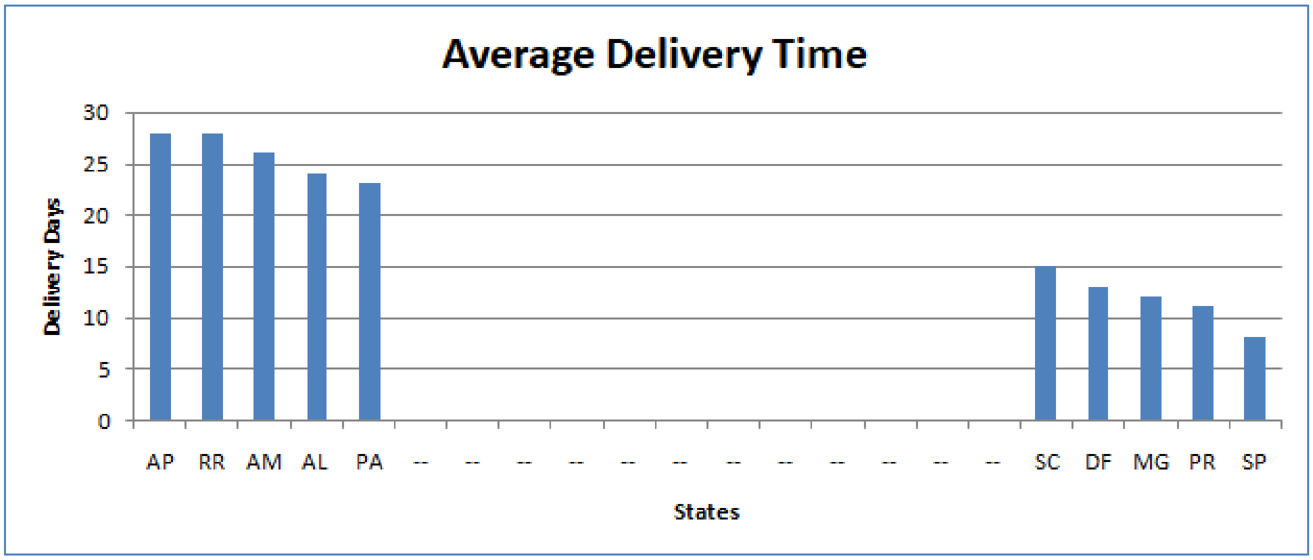


3. **Find out the top 5 states with the highest & lowest average delivery time.**

```
WITH min_avg AS
        (
          SELECT
            c.customer_state,
            ROUND(AVG(TIMESTAMP_DIFF(order_delivered_customer_date,order_purchase_ti
mestamp,DAY))) AS lowest_five
          FROM `target.customers`c JOIN `target.orders` o ON
            c.customer_id = o.customer_id
          JOIN `target.order_items`oi ON
            o.order_id = oi.order_id
          GROUP BY 1
          ORDER BY 2
          LIMIT 5
        ),
max_avg AS
        (
          SELECT
            c.customer_state,
            ROUND(AVG(TIMESTAMP_DIFF(order_delivered_customer_date,order_purchase_ti
mestamp,DAY))) AS highest_five
          FROM `target.customers`c JOIN `target.orders` o ON
            c.customer_id = o.customer_id
          JOIN `target.order_items`oi ON
            o.order_id = oi.order_id
          GROUP BY 1
          ORDER BY 2 DESC
          LIMIT 5
        )
SELECT avg1.*, avg2.*
FROM min_avg avg1 LEFT JOIN max_avg avg2 ON
  avg1.customer_state=avg2.customer_state
UNION DISTINCT
SELECT avg1.*, avg2.*
FROM min_avg avg1 RIGHT JOIN max_avg avg2 ON
  avg1.customer_state=avg2.customer_state ;
```



Row	customer_state	lowest_five	customer_state_1	highest_five
1	null	null	AP	28.0
2	null	null	RR	28.0
3	null	null	AM	26.0
4	null	null	AL	24.0
5	null	null	PA	23.0
6	SP	8.0	null	null
7	PR	11.0	null	null
8	MG	12.0	null	null
9	DF	13.0	null	null
10	SC	15.0	null	null

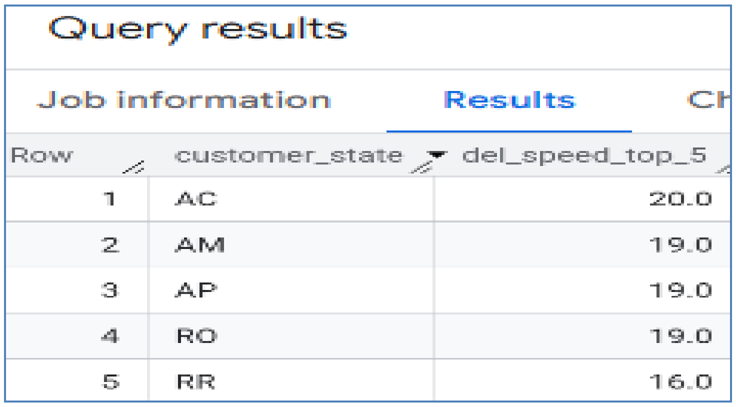


Sao Paulo has the lowest average delivery time and Amapa has the highest average delivery time.

4. **Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery. You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.**

```
SELECT
      c.customer_state,
      ROUND(AVG(TIMESTAMP_DIFF(order_estimated_delivery_date,order_d
      elivered_customer_date,DAY))) AS del_speed_top_5
FROM `target.customers`c JOIN `target.orders` o ON
```

```
            c.customer_id = o.customer_id
GROUP BY 1
ORDER BY 2 DESC
LIMIT 5 ;
```



Query results

Job information   Results   Ch

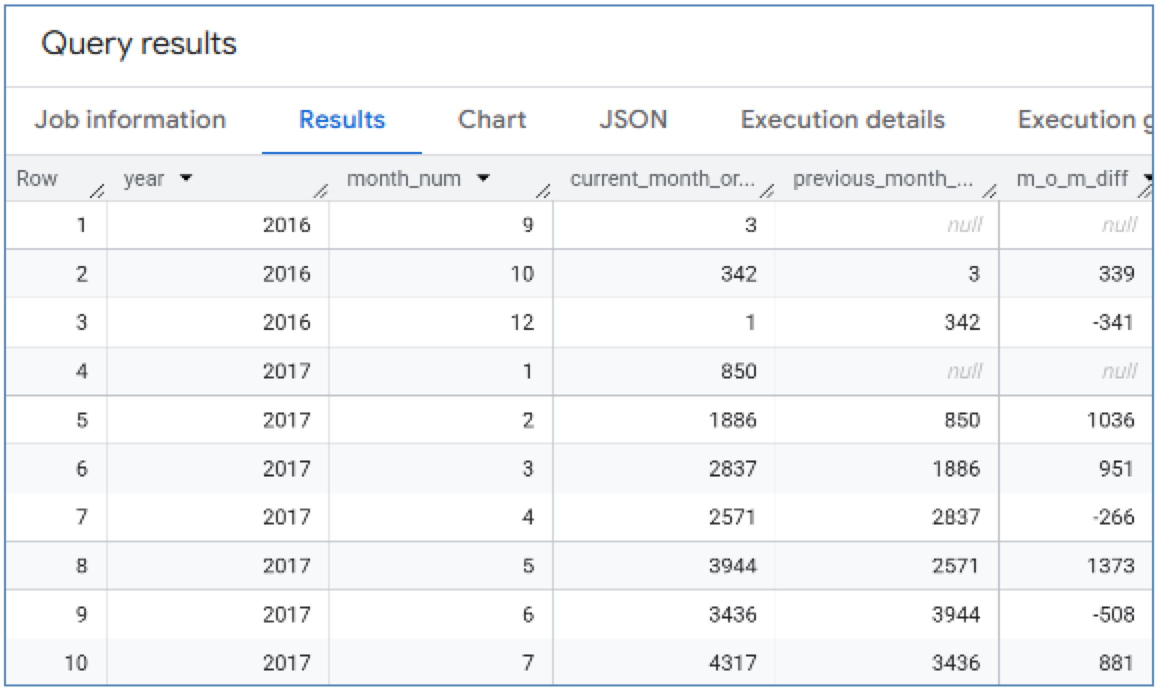
Row	customer_state	del_speed_top_5
1	AC	20.0
2	AM	19.0
3	AP	19.0
4	RO	19.0
5	RR	16.0

States with good track record in fast delivery are **Acre, Amazonas, Amapa, Rondonia and Roraima** with Acre has managed to deliver 20 days earlier than estimated delivery date in average.

6. **Analysis based on the payments:**

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

```
WITH base AS
      (SELECT
            o.order_id,
            EXTRACT(YEAR FROM order_purchase_timestamp) AS
year,
            EXTRACT(MONTH FROM order_purchase_timestamp) AS
month_num,
            payment_type
      FROM
            `target.orders`o
      LEFT JOIN `target.payments`p ON
            o.order_id = p.order_id
      WHERE payment_type IS NOT NULL AND payment_type !='not
defined'
      ORDER BY 2,3),

filter1 AS
      (
      SELECT
            year,
                month_num,
                COUNT(payment_type) AS current_month_orders
            FROM base
            GROUP BY 1,2
            ORDER BY 1,2
            ),
    filter2 AS
          (
            SELECT
                year,
                month_num,
                current_month_orders,
                LAG(current_month_orders) OVER (PARTITION BY year
    ORDER BY month_num) AS previous_month_orders
            FROM filter1
            GROUP BY 1,2,3
            ORDER BY 1,2,3
            )
    SELECT

            *,

           current_month_orders - previous_month_orders  AS

           m_o_m_diff

    FROM filter2;
```



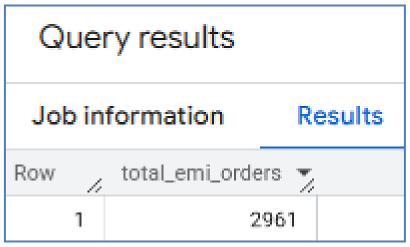
Query results

Row	year	month_num	current_month_or...	previous_month_...	m_o_m_diff
1	2016	9	3	null	null
2	2016	10	342	3	339
3	2016	12	1	342	-341
4	2017	1	850	null	null
5	2017	2	1886	850	1036
6	2017	3	2837	1886	951
7	2017	4	2571	2837	-266
8	2017	5	3944	2571	1373
9	2017	6	3436	3944	-508
10	2017	7	4317	3436	881

* September (2018) seen a sharpest decline in the number od orders placed compared to previous month of same year. Moreover in 2018 orders declined throughout most of the months.

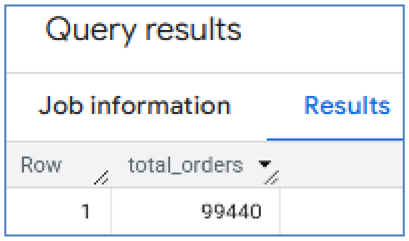
2. **Find the no. of orders placed on the basis of the payment installments that have been paid.**

```
SELECT COUNT(num_emi_per_order) AS total_emi_orders FROM
     (SELECT
            DISTINCT order_id,
            COUNT(payment_installments) AS num_emi_per_order
     FROM
            `target.payments`
     GROUP BY 1 )
WHERE num_emi_per_order > 1
```


Query results

Job information Results

Row	total_emi_orders
1	2961


Query results

Job information Results

Row	total_orders
1	99440

Orders based on EMI constitute only 2.9 % of total order.

****