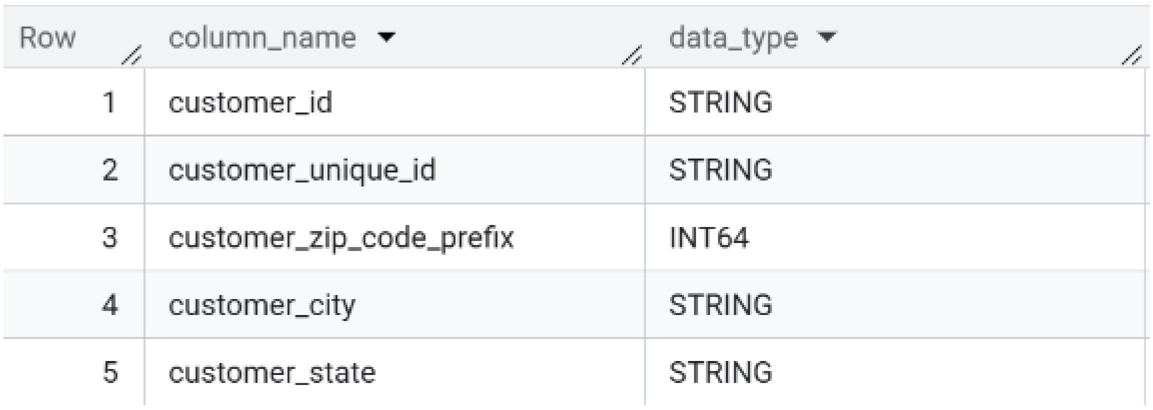
TARGET SQL PROJECT

# Data type of all columns in the "customers" table.

Code syntax:

```
SELECT
  column_name,
  data_type
FROM
  `scaler-dsml-sql-467408.Target.INFORMATION_SCHEMA.COLUMNS`
WHERE
  table_name = 'customers'
```

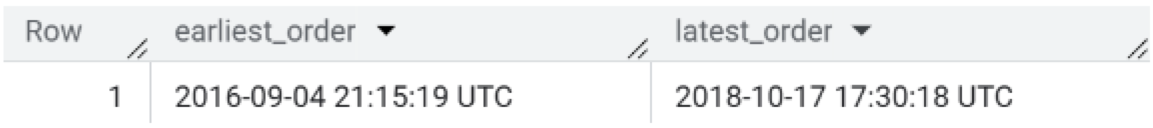


Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

# The time range between which the orders were placed.

Syntax:

```
SELECT
  MIN(order_purchase_timestamp) AS earliest_order,
  MAX(order_purchase_timestamp) AS latest_order
FROM
 `Target.orders
```

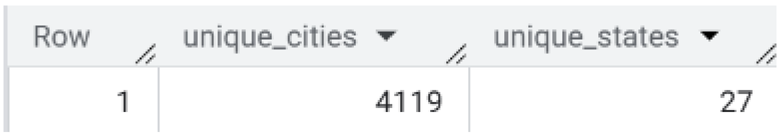


Row	earliest_order	latest_order
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

#Number of the Cities & States of customers who ordered during the given period

Syntax:

```
SELECT
  COUNT(DISTINCT c.customer_city) AS unique_cities,
  COUNT(DISTINCT c.customer_state) AS unique_states
FROM
  `scaler-dsml-sql-467408.Target.orders` o
JOIN
  `scaler-dsml-sql-467408.Target.customers` c
ON
  o.customer_id = c.customer_id
WHERE
  o.order_purchase_timestamp BETWEEN TIMESTAMP('2016-09-04') AND
TIMESTAMP('2018-10-17')
```



Row	unique_cities	unique_states
1	4119	27

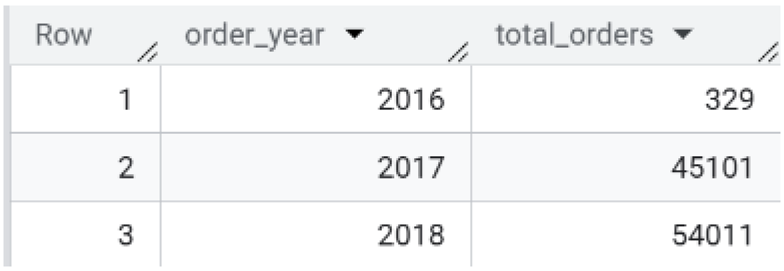
In-depth Exploration

# growing trend in the no. of orders placed over the past years

Syntax:

```
SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
  COUNT(*) AS total_orders
```

```
FROM
  `Target.orders`
GROUP BY
  order_year
ORDER BY
  order_year;
```

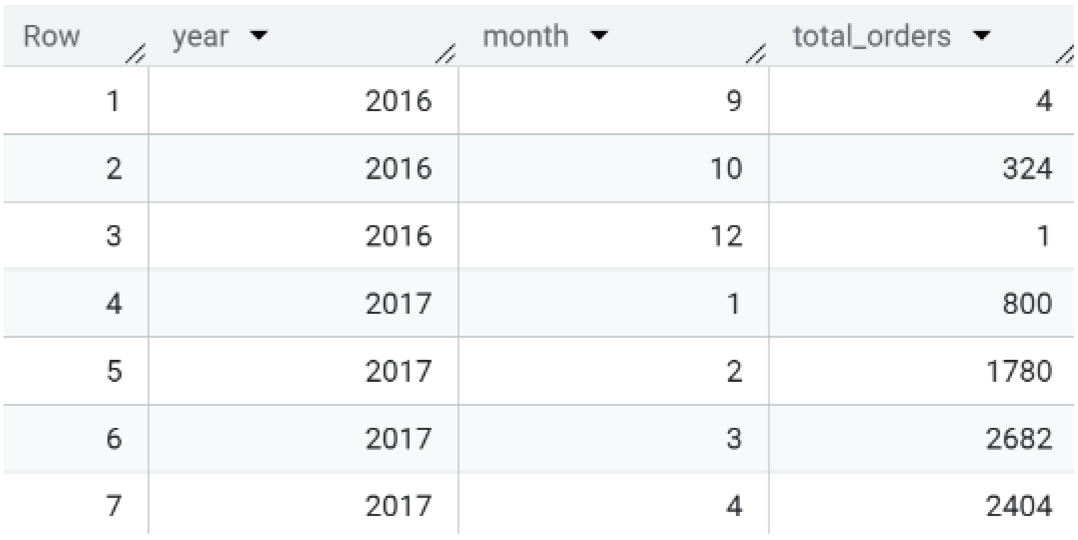


Row	order_year	total_orders
1	2016	329
2	2017	45101
3	2018	54011

# Monthly seasonality in terms of the no. of orders being placed

Syntax:

```
SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
  COUNT(*) AS total_orders
FROM
  `Target.orders`
GROUP BY
  year, month
ORDER BY
  year, month
```



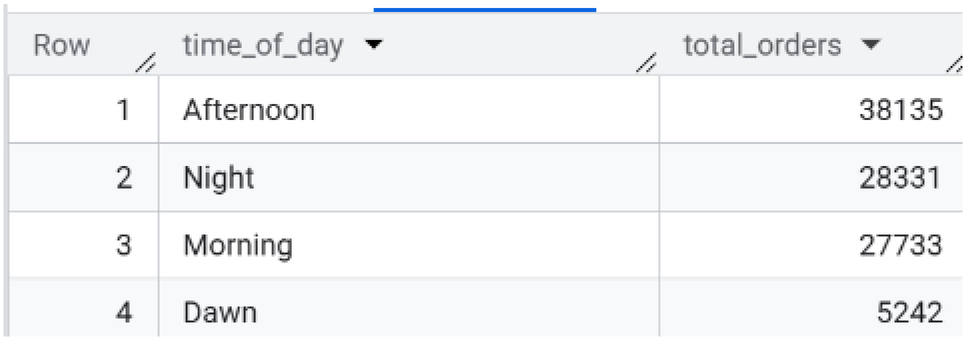
Row	year	month	total_orders
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404

# Time of the day, the Brazilian customers mostly place their orders(Dawn, Morning, Afternoon or Night)

Syntax:

```
SELECT
  CASE
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 6 THEN 'Dawn'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN 'Morning'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN 'Afternoon'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 19 AND 23 THEN 'Night'
    ELSE 'Unknown'
  END AS time_of_day,
  COUNT(*) AS total_orders
FROM
  `Target.orders`
GROUP BY
  time_of_day
ORDER BY
```

```
  total_orders DESC
```



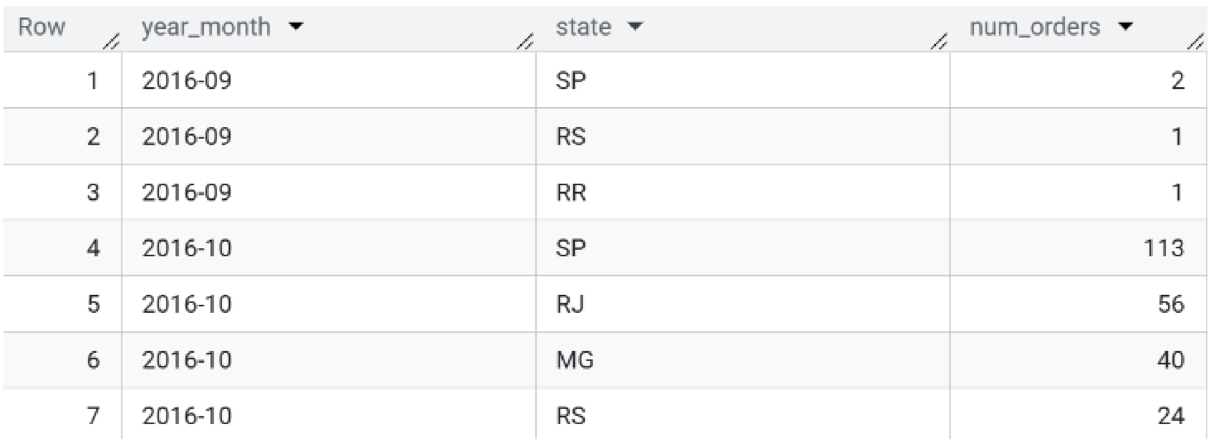
Row	time_of_day	total_orders
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

Evolution of E-commerce orders in the Brazil region

# The month on month no. of orders placed in each state.

Syntax:

```
WITH customer_orders AS (
  SELECT
    FORMAT_TIMESTAMP('%Y-%m', o.order_purchase_timestamp) AS year_month,
    c.customer_state AS state,
    o.order_id
  FROM `scaler-dsml-sql-467408.Target.orders` AS o
  JOIN `scaler-dsml-sql-467408.Target.customers` AS c
    ON o.customer_id = c.customer_id
)
SELECT
  year_month,
  state,
  COUNT(DISTINCT order_id) AS num_orders
FROM customer_orders
GROUP BY year_month, state
ORDER BY year_month, num_orders DESC;
```

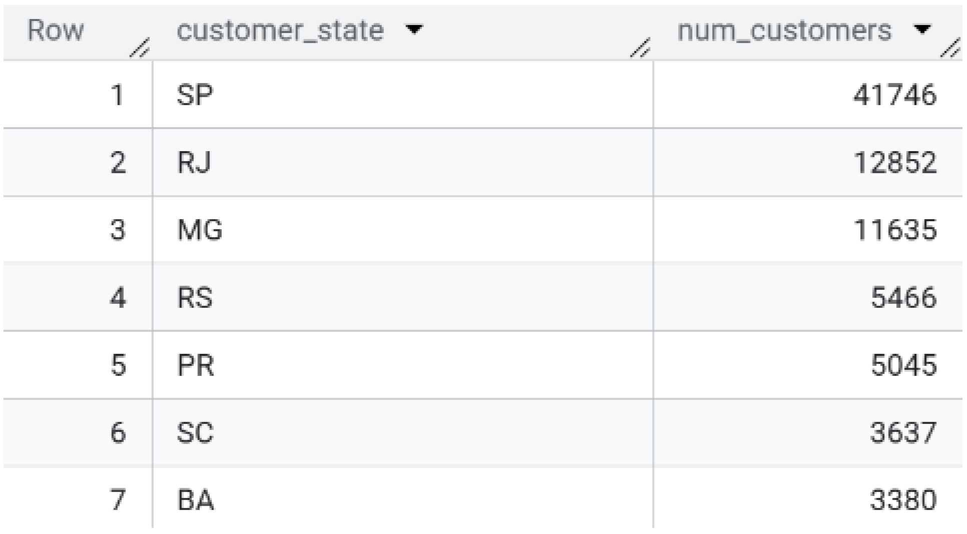


Row	year_month	state	num_orders
1	2016-09	SP	2
2	2016-09	RS	1
3	2016-09	RR	1
4	2016-10	SP	113
5	2016-10	RJ	56
6	2016-10	MG	40
7	2016-10	RS	24

# The customers distributed across all the states.

Syntax:

```
SELECT
  customer_state,
  COUNT(DISTINCT customer_id) AS num_customers
FROM
  `scaler-dsml-sql-467408.Target.customers`
GROUP BY
  customer_state
ORDER BY
  num_customers DESC;
```



Row	customer_state	num_customers
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380

Impact on Economy

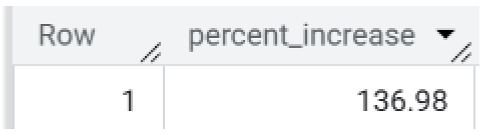
# The % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).
You can use the "payment_value" column in the payments table to get the cost of orders

Syntax:

```
WITH filtered_orders AS (
 SELECT
  p.payment_value,
  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month
 FROM
  `Target.orders` o
 JOIN
  `Target.payments` p
 ON
  o.order_id = p.order_id
  WHERE
    EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
    AND EXTRACT(YEAR FROM o.order_purchase_timestamp) IN (2017, 2018)
),

yearly_totals AS (
  SELECT
    order_year,
    SUM(payment_value) AS total_payment
  FROM
    filtered_orders
  GROUP BY
    order_year
)

SELECT
  ROUND(
    SAFE_DIVIDE(
      (MAX(CASE WHEN order_year = 2018 THEN total_payment END) -
       MAX(CASE WHEN order_year = 2017 THEN total_payment END)),
      MAX(CASE WHEN order_year = 2017 THEN total_payment END)
    ) * 100,
    2
  ) AS percent_increase
FROM
  yearly_totals;
```

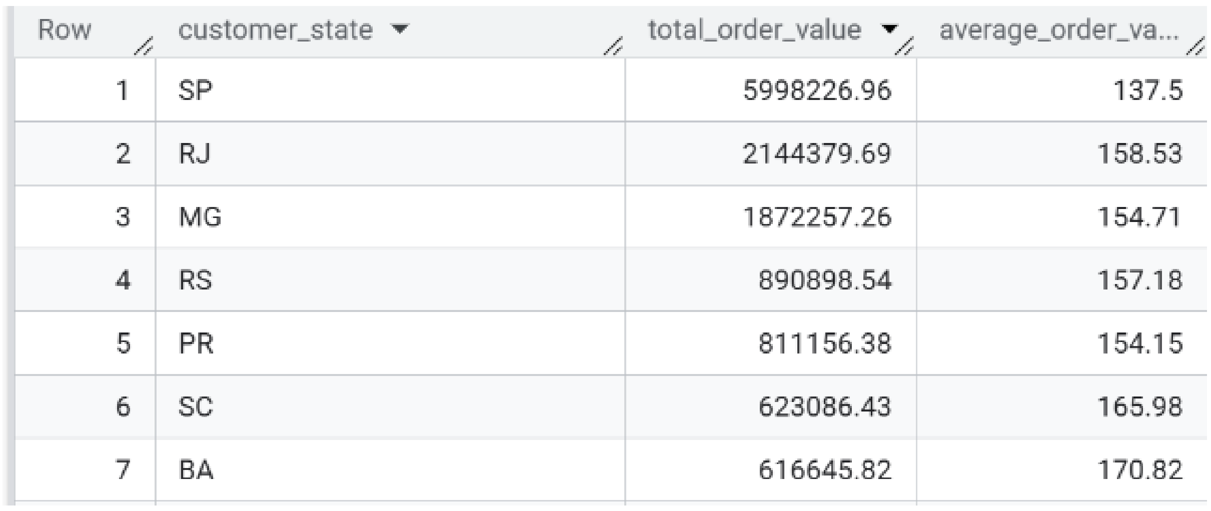


Row	percent_increase
1	136.98

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

Syntax:

```
SELECT
  c.customer_state,
  ROUND(SUM(p.payment_value), 2) AS total_order_value,
  ROUND(AVG(p.payment_value), 2) AS average_order_value
FROM
  `Target.orders` o
JOIN
  `Target.payments` p
ON
  o.order_id = p.order_id
JOIN
  `Target.customers` c
ON
  o.customer_id = c.customer_id
GROUP BY
  c.customer_state
ORDER BY
  total_order_value DESC;
```



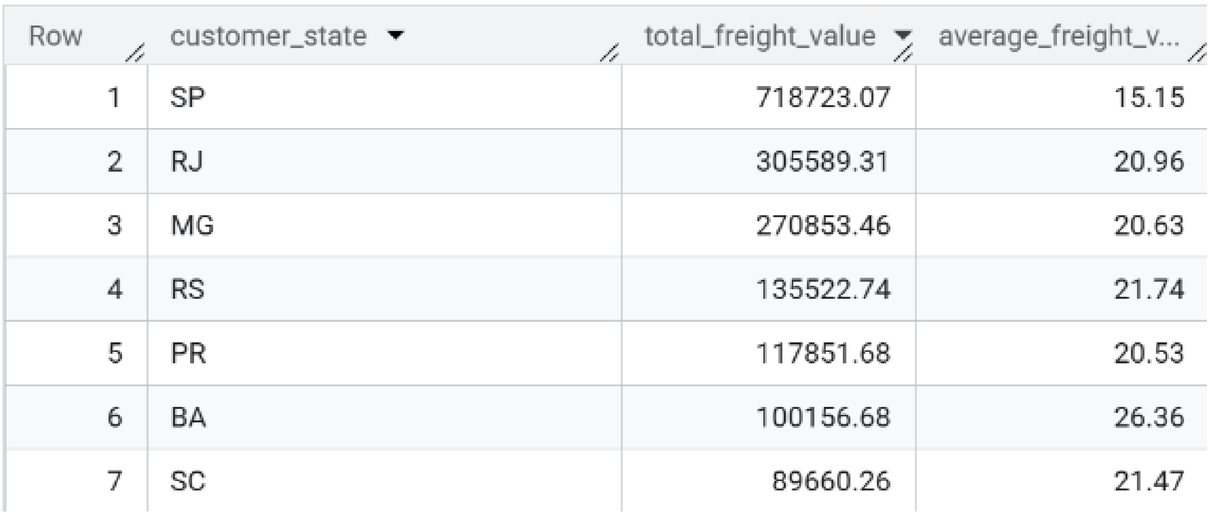
Row	customer_state	total_order_value	average_order_va...
1	SP	5998226.96	137.5
2	RJ	2144379.69	158.53
3	MG	1872257.26	154.71
4	RS	890898.54	157.18
5	PR	811156.38	154.15
6	SC	623086.43	165.98
7	BA	616645.82	170.82

# The Total & Average value of order freight for each state

Syntax:

```
SELECT
  c.customer_state,
  ROUND(SUM(oi.freight_value), 2) AS total_freight_value,
  ROUND(AVG(oi.freight_value), 2) AS average_freight_value
FROM
  `Target.orders` o
JOIN
  `Target.order_items` oi
ON
  o.order_id = oi.order_id
JOIN
  `Target.customers` c
ON
  o.customer_id = c.customer_id
GROUP BY
  c.customer_state
ORDER BY
```

  total_freight_value DESC;



Row	customer_state	total_freight_value	average_freight_v...
1	SP	718723.07	15.15
2	RJ	305589.31	20.96
3	MG	270853.46	20.63
4	RS	135522.74	21.74
5	PR	117851.68	20.53
6	BA	100156.68	26.36
7	SC	89660.26	21.47

# Analysis based on sales, freight and delivery time

# 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.

Syntax:

SELECT

  order_id,

  DATE(order_purchase_timestamp) AS purchase_date,

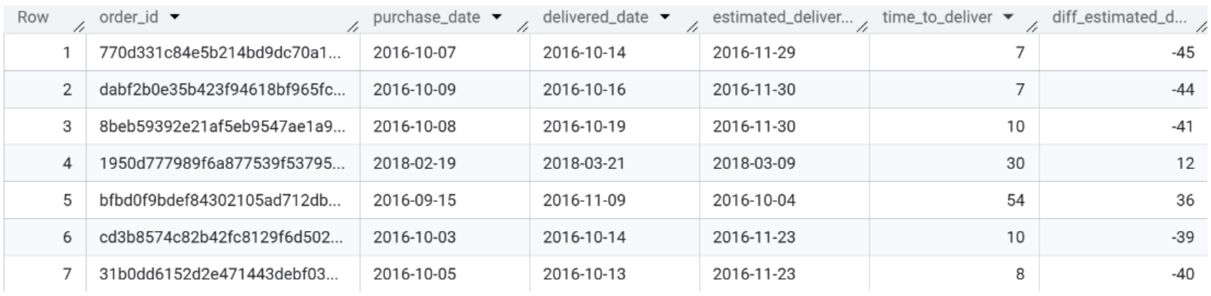
  DATE(order_delivered_customer_date) AS delivered_date,

  DATE(order_estimated_delivery_date) AS estimated_delivery_date,

  DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) AS time_to_deliver,

  DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date, DAY) AS diff_estimated_delivery

```
    FROM
     `Target.orders`
    WHERE
     order_delivered_customer_date IS NOT NULL
     AND order_purchase_timestamp IS NOT NULL
     AND order_estimated_delivery_date IS NOT NULL;
```



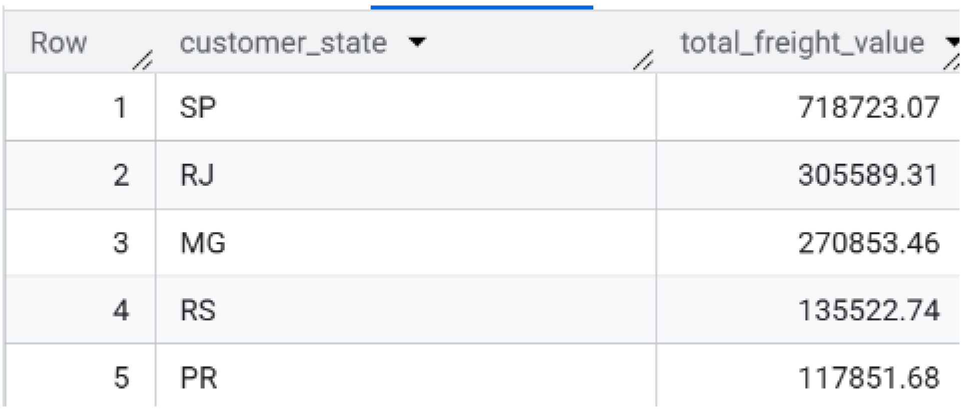
Row	order_id	purchase_date	delivered_date	estimated_deliver...	time_to_deliver	diff_estimated_d...
1	770d331c84e5b214bd9dc70a1...	2016-10-07	2016-10-14	2016-11-29	7	-45
2	dabf2b0e35b423f94618bf965fc...	2016-10-09	2016-10-16	2016-11-30	7	-44
3	8beb59392e21af5eb9547ae1a9...	2016-10-08	2016-10-19	2016-11-30	10	-41
4	1950d777989f6a877539f53795...	2018-02-19	2018-03-21	2018-03-09	30	12
5	bfbd0f9bdef84302105ad712db...	2016-09-15	2016-11-09	2016-10-04	54	36
6	cd3b8574c82b42fc8129f6d502...	2016-10-03	2016-10-14	2016-11-23	10	-39
7	31b0dd6152d2e471443debf03...	2016-10-05	2016-10-13	2016-11-23	8	-40

# Top 5 States with Highest Total Freight Value

Syntax:

```
SELECT
 c.customer_state,
 ROUND(SUM(oi.freight_value), 2) AS total_freight_value
FROM
 `Target.orders` o
JOIN
 `Target.order_items` oi ON o.order_id = oi.order_id
JOIN
 `Target.customers` c ON o.customer_id = c.customer_id
GROUP BY
 c.customer_state
ORDER BY
 total_freight_value DESC
```

LIMIT 5;

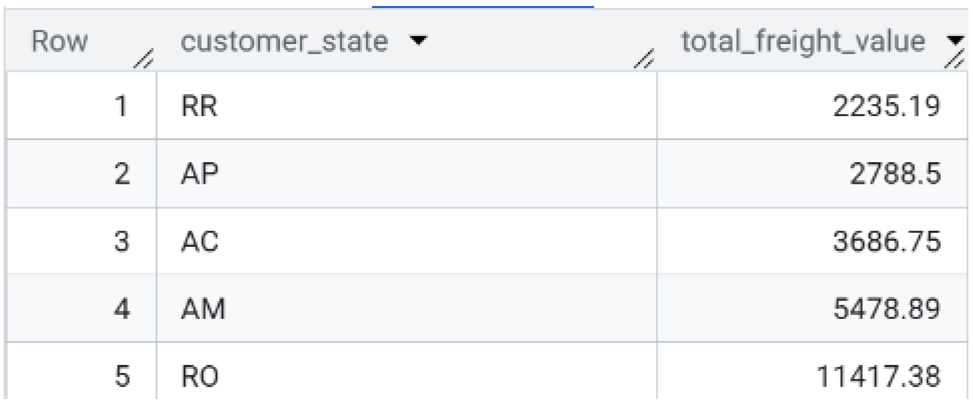


Row	customer_state	total_freight_value
1	SP	718723.07
2	RJ	305589.31
3	MG	270853.46
4	RS	135522.74
5	PR	117851.68

# Top 5 States with Lowest Total Freight Value

Syntax:

```
SELECT
  c.customer_state,
  ROUND(SUM(oi.freight_value), 2) AS total_freight_value
FROM
  `Target.orders` o
JOIN
  `Target.order_items` oi ON o.order_id = oi.order_id
JOIN
  `Target.customers` c ON o.customer_id = c.customer_id
GROUP BY
  c.customer_state
ORDER BY
  total_freight_value ASC
limit 5;
```

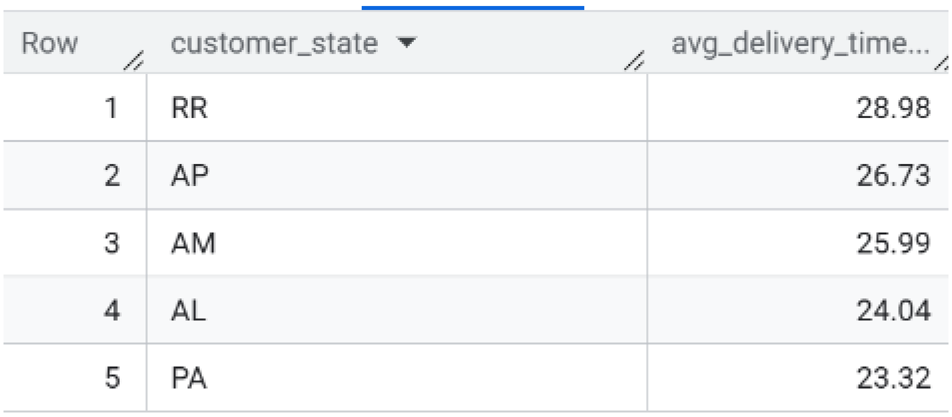


Row	customer_state	total_freight_value
1	RR	2235.19
2	AP	2788.5
3	AC	3686.75
4	AM	5478.89
5	RO	11417.38

# Top 5 States with Highest Average Delivery Time

Syntax:

```
SELECT
  c.customer_state,
  ROUND(AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, 
DAY)), 2) AS avg_delivery_time_days
FROM
  `Target.orders` o
JOIN
  `Target.customers` c ON o.customer_id = c.customer_id
WHERE
  order_delivered_customer_date IS NOT NULL
  AND order_purchase_timestamp IS NOT NULL
GROUP BY
  c.customer_state
ORDER BY
  avg_delivery_time_days DESC
LIMIT 5;
```

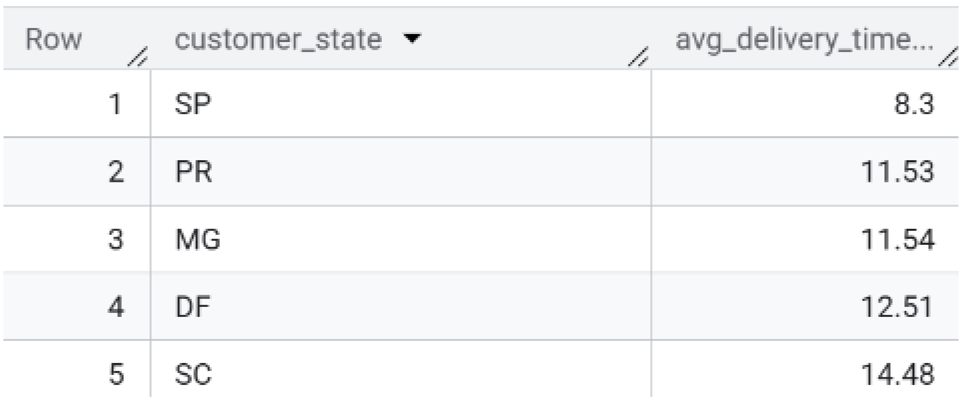


Row	customer_state	avg_delivery_time...
1	RR	28.98
2	AP	26.73
3	AM	25.99
4	AL	24.04
5	PA	23.32

#Top 5 States with Lowest Average Delivery Time

Syntax:

```
SELECT
  c.customer_state,
  ROUND(AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,
DAY)), 2) AS avg_delivery_time_days
FROM
  `Target.orders` o
JOIN
  `Target.customers` c ON o.customer_id = c.customer_id
WHERE
  order_delivered_customer_date IS NOT NULL
  AND order_purchase_timestamp IS NOT NULL
GROUP BY
  c.customer_state
ORDER BY
  avg_delivery_time_days ASC
LIMIT 5;
```

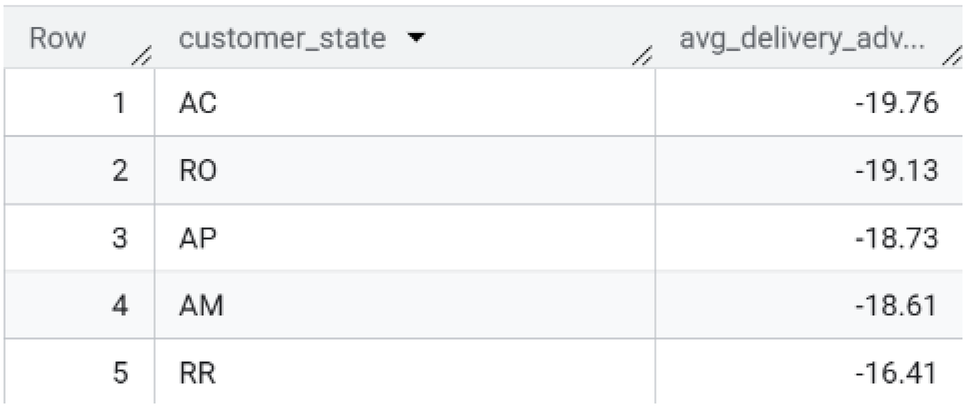


Row	customer_state	avg_delivery_time...
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48

# The top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

Syntax:

```
SELECT
  c.customer_state,
  ROUND(AVG(DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date, DAY)), 2) AS avg_delivery_advance_days
FROM
  `Target.orders` o
JOIN
  `Target.customers` c ON o.customer_id = c.customer_id
WHERE
  order_delivered_customer_date IS NOT NULL
  AND order_estimated_delivery_date IS NOT NULL
GROUP BY
  c.customer_state
ORDER BY
  avg_delivery_advance_days ASC
LIMIT 5;
```



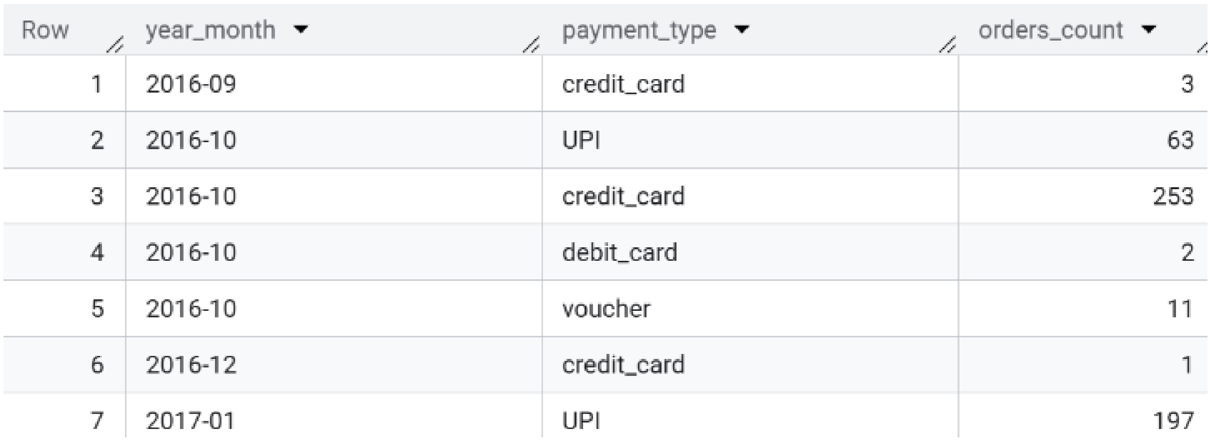
Row	customer_state	avg_delivery_adv...
1	AC	-19.76
2	RO	-19.13
3	AP	-18.73
4	AM	-18.61
5	RR	-16.41

# Analysis based on the payments

# The month on month no. of orders placed using different payment types

Syntax:

```
SELECT
  FORMAT_TIMESTAMP('%Y-%m', o.order_purchase_timestamp) AS year_month,
  p.payment_type,
  COUNT(DISTINCT o.order_id) AS orders_count
FROM
  `Target.orders` o
JOIN
  `Target.payments` p
ON
  o.order_id = p.order_id
GROUP BY
  year_month,
  p.payment_type
ORDER BY
  year_month,
  p.payment_type;
```

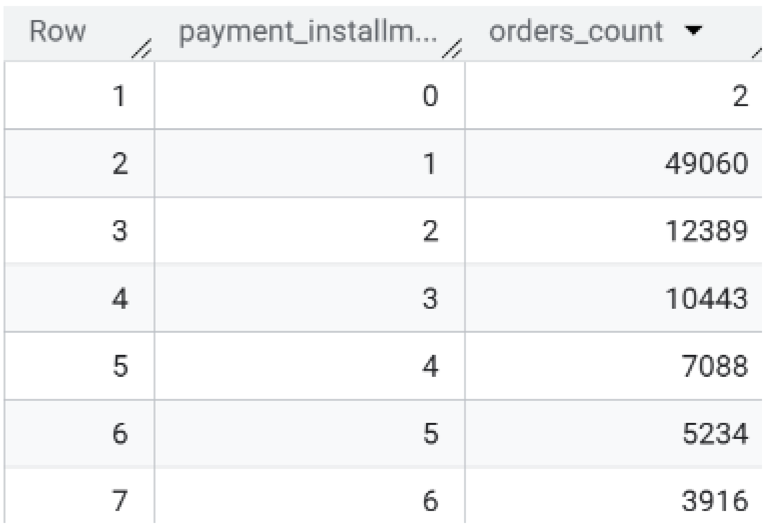


Row	year_month	payment_type	orders_count
1	2016-09	credit_card	3
2	2016-10	UPI	63
3	2016-10	credit_card	253
4	2016-10	debit_card	2
5	2016-10	voucher	11
6	2016-12	credit_card	1
7	2017-01	UPI	197

# The no. of orders placed on the basis of the payment installments that have been paid.

Syntax:

```
SELECT
  payment_installments,
  COUNT(DISTINCT order_id) AS orders_count
FROM
  `Target.payments`
GROUP BY
  payment_installments
ORDER BY
  payment_installments;
```



Row	payment_installm...	orders_count
1	0	2
2	1	49060
3	2	12389
4	3	10443
5	4	7088
6	5	5234
7	6	3916

Actionable insights

1. **Order Growth:** There is a measurable increase in order cost from 2017 to 2018, indicating growing demand.
2. **Time of Day:** Orders peak during specific time windows (e.g., morning or afternoon), showing customer buying behavior patterns.
3. **Freight Variability:** Freight costs vary significantly by state, suggesting logistics cost differences due to geography or infrastructure.
4. **Delivery Times:** Some states experience faster deliveries, while others show delays relative to estimated delivery dates.
5. **Payment Types:** Payment preferences vary month-to-month, indicating seasonal or promotional impacts on payment behavior.
6. **Installments Usage:** A sizable portion of orders is paid in installments, highlighting the importance of EMI options.
7. **Delivery Accuracy:** Some states consistently deliver faster than estimated, which could enhance customer satisfaction.

8. **Customer Distribution:** Most orders come from a handful of states, which dominate sales and logistics efforts.
9. **Freight vs. Order Value:** States with higher average freight costs might see lower average order values, potentially affecting profitability.
10. **Review Scores & Delivery:** Faster delivery times often correlate with higher review scores, emphasizing timely delivery's impact on customer satisfaction.

Recommendations

1. **Optimize Logistics in High-Freight States:** Investigate ways to reduce freight costs in states with high shipping charges (e.g., by partnering with local carriers or using regional warehouses).
2. **Target Peak Purchase Times:** Schedule promotions or inventory replenishment aligned with peak order times (e.g., mornings or afternoons).
3. **Promote Faster Delivery Regions:** Use faster delivery states as benchmarks to improve slower regions through process improvements or better carrier selection.
4. **Leverage Payment Preferences:** Offer tailored payment options during months when certain payment types peak to boost conversion rates.
5. **Enhance EMI Offers:** Since installments are popular, promote EMI plans with attractive terms to increase average order size.
6. **Improve Estimated Delivery Accuracy:** Analyze data from states with large discrepancies between estimated and actual delivery to refine estimated delivery algorithms.

7. **Focus on High-Volume States:** Prioritize marketing and logistics investments in states contributing the majority of sales.
8. **Monitor Freight Impact on Sales:** Track if high freight costs correlate with cart abandonment or lower repeat purchases and adjust pricing or shipping policies accordingly.
9. **Use Delivery Performance in Customer Feedback:** Highlight on-time and faster deliveries in marketing communications to build trust and encourage positive reviews.
10. **Personalize Customer Experience by Region:** Use geographic insights to customize promotions, shipping offers, and product assortments for different states.