BRAZIL - TARGET ECOMMERCE INDUSTRY BUSINESS ANALYSIS CASE STUDY

Sabarish S
selvamsabarish1998@gmail.com
9003589362

Abstract

- Section 1: Exploratory Analysis
- Section 2: In-depth Exploration
- Section 3: Evolution of E-commerce
- Section 4: Impact on Economy
- Section 5: Analysis based on sales, freight and delivery time.
- Section 6: Analysis based on the payments
- Section 7: Actionable Insights & Recommendations

Section 1: Exploratory Analysis

Questions

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.
    2. Get the time range between which the orders were placed.
    3. Count the Cities & States of customers who ordered during the given period.

Solution

One Additional table is created from the scraped data from internet (regarding the Brazil States) apart from the 8 Nos. CSV files that was provided with the Case Study. All these files are loaded into Big Query.

Abstract of Section 1

Target Dataset : Entity Relationship Diagram Visualised

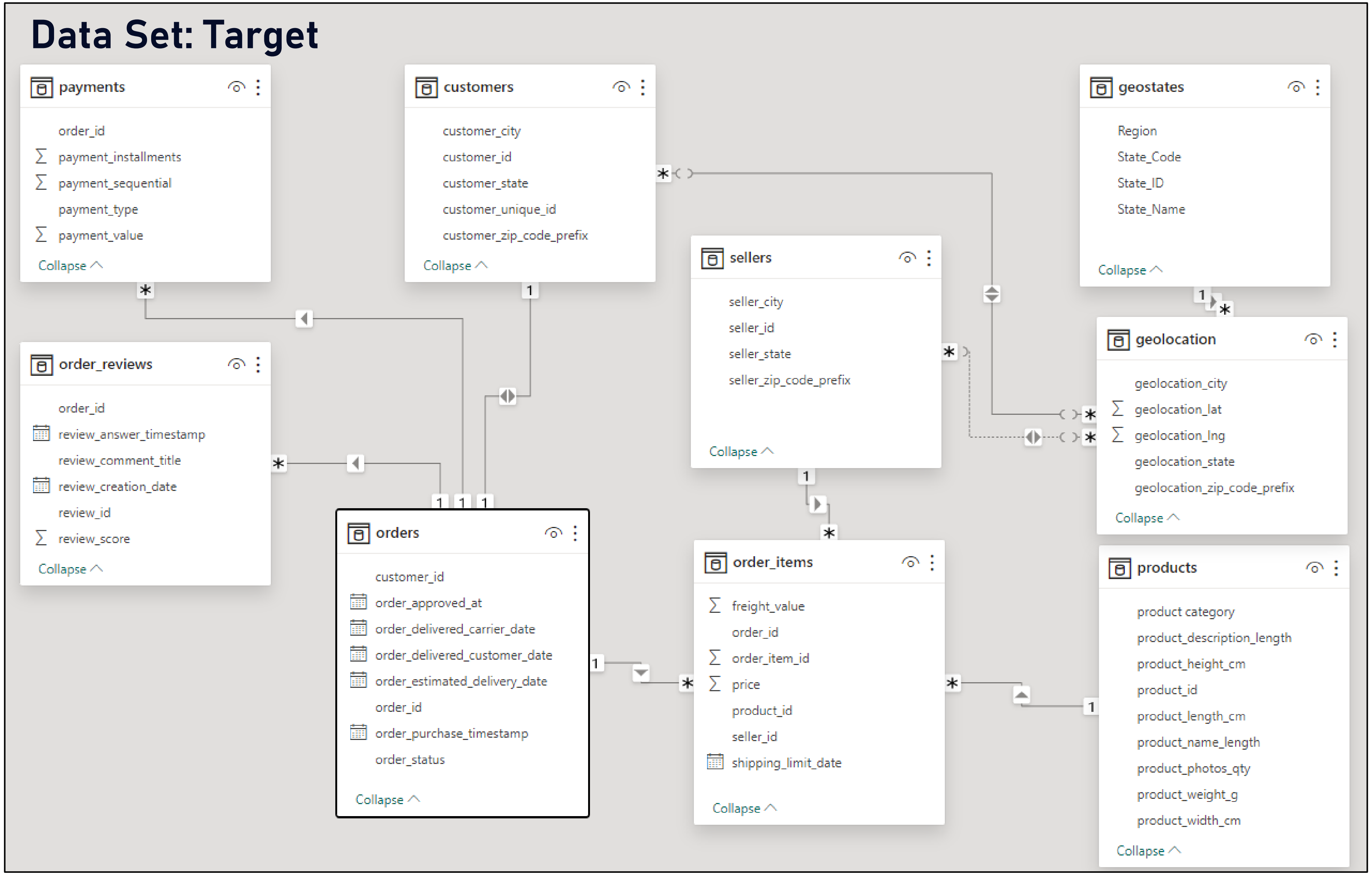
Keys and Cardinality : Tabulated

Query 1.1 : Target dataset’s : Information Schema

Query 1.2 : Order Time Range and its associated ad-hock analysis

Query 1.3 : Count of Geographical Location

Target Dataset – Entity Relationship Diagram



Data Set: Target

payments
- order_id
- payment_installments
- payment_sequential
- payment_type
- payment_value

customers
- customer_city
- customer_id
- customer_state
- customer_unique_id
- customer_zip_code_prefix

geostates
- Region
- State_Code
- State_ID
- State_Name

sellers
- seller_city
- seller_id
- seller_state
- seller_zip_code_prefix

geolocation
- geolocation_city
- geolocation_lat
- geolocation_lng
- geolocation_state
- geolocation_zip_code_prefix

order_reviews
- order_id
- review_answer_timestamp
- review_comment_title
- review_creation_date
- review_id
- review_score

orders
- customer_id
- order_approved_at
- order_delivered_carrier_date
- order_delivered_customer_date
- order_estimated_delivery_date
- order_id
- order_purchase_timestamp
- order_status

order_items
- freight_value
- order_id
- order_item_id
- price
- product_id
- seller_id
- shipping_limit_date

products
- product category
- product_description_length
- product_height_cm
- product_id
- product_length_cm
- product_name_length
- product_photos_qty
- product_weight_g
- product_width_cm

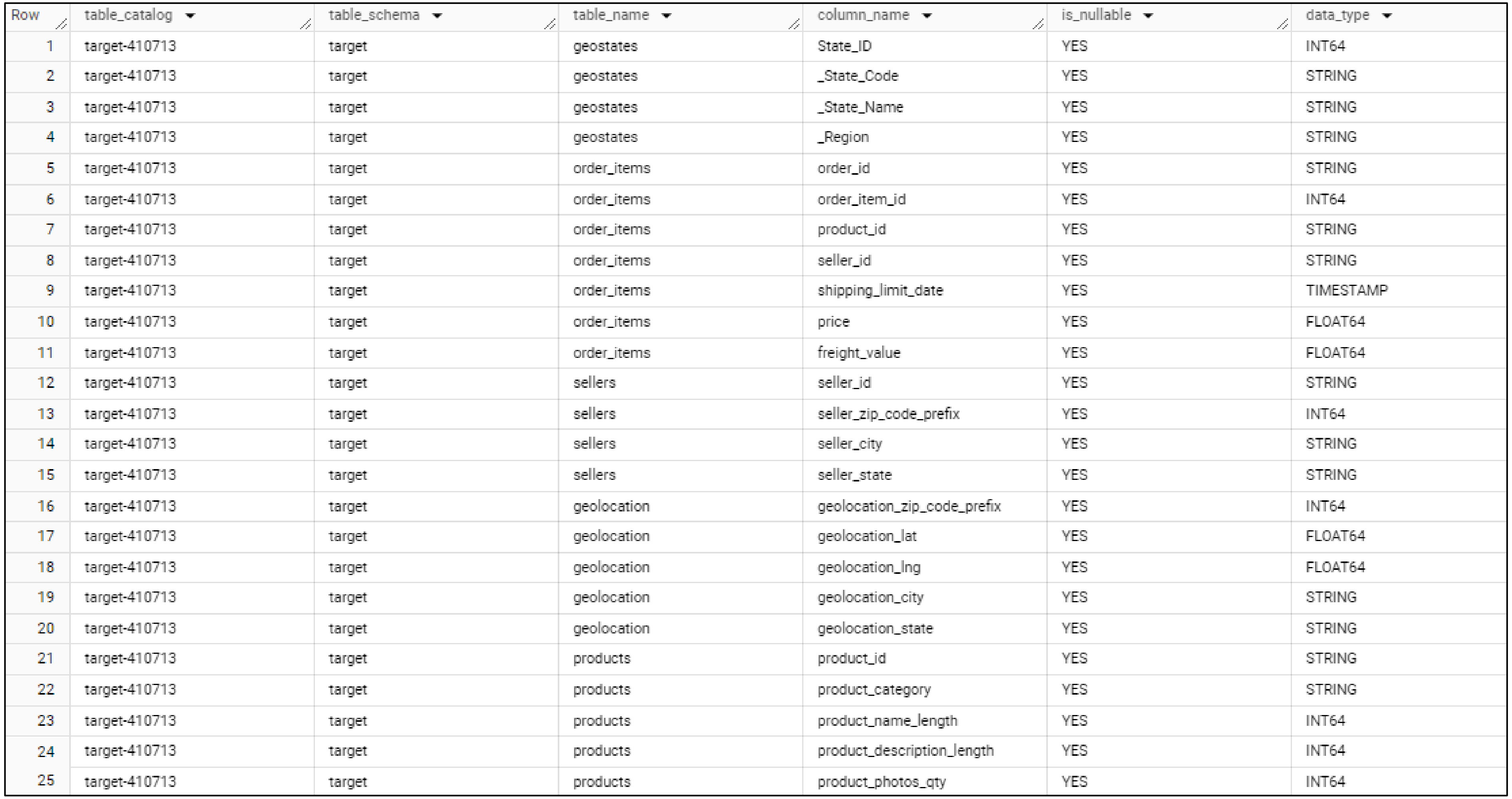
An extra “GeoStates" table is created within the dataset that contains “Regional“ and “State Names”, these attributes are connected to “Geolocation” table via “StateCode” attribute to the “Geolocation_state” attribute.

Keys and Cardinality

Table	Column	Primary Key	Foreign Key	Referenced Table.Column	Cardinality
geostates	State_ID	Yes	No		One-to-Many
order_items	order_item_id	Yes	No		One-to-Many
order_items	order_id	No	Yes	orders.order_id	Many-to-One
order_items	product_id	No	Yes	products.product_id	Many-to-One
order_items	seller_id	No	Yes	sellers.seller_id	Many-to-One
sellers	seller_id	Yes	No		One-to-Many
geolocation	geolocation_zip_code_prefix	Yes	No		One-to-Many
products	product_id	Yes	No		One-to-Many
orders	order_id	Yes	No		One-to-Many
orders	customer_id	No	Yes	customers.customer_id	Many-to-One
orders	order_status	No	No		One-to-Many
orders	order_purchase_timestamp	No	No		One-to-Many
orders	order_approved_at	No	No		One-to-Many
orders	order_delivered_carrier_date	No	No		One-to-Many
orders	order_delivered_customer_date	No	No		One-to-Many
orders	order_estimated_delivery_date	No	No		One-to-Many
payments	order_id	No	Yes	orders.order_id	Many-to-One
payments	payment_sequential	No	No		One-to-Many
payments	payment_type	No	No		One-to-Many
payments	payment_installments	No	No		One-to-Many
payments	payment_value	No	No		One-to-Many
customers	customer_id	Yes	No		One-to-Many
order_reviews	review_id	Yes	No		One-to-Many
order_reviews	order_id	No	Yes	orders.order_id	Many-to-One
order_reviews	review_score	No	No		One-to-Many
order_reviews	review_comment_title	No	No		One-to-Many
order_reviews	review_creation_date	No	No		One-to-Many
order_reviews	review_answer_timestamp	No	No		One-to-Many

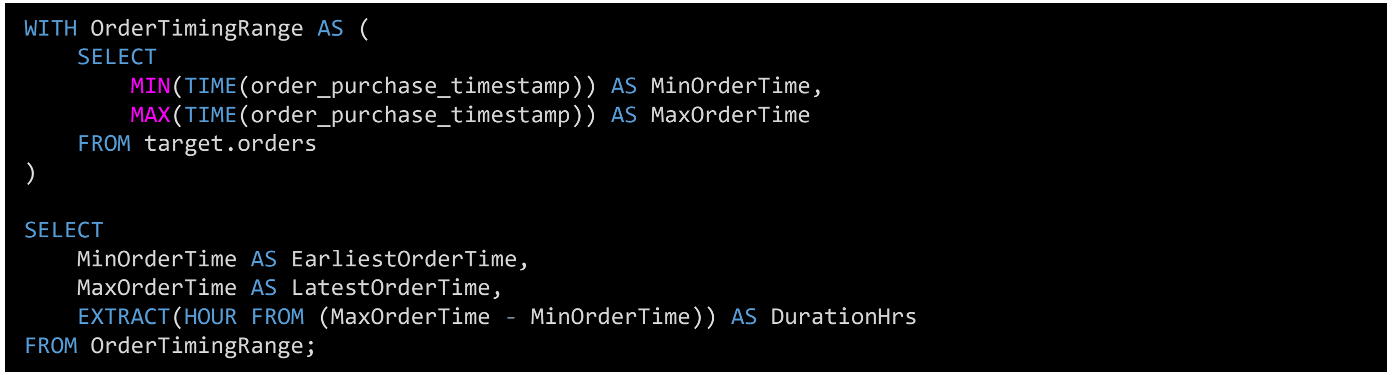
Exploratory Data Analysis; Query 1.1 Information Schema

```
SELECT
  table_catalog,
  table_schema,
  table_name,
  column_name,
  is_nullable,
  data_type
FROM
  target.INFORMATION_SCHEMA.COLUMNS;
```

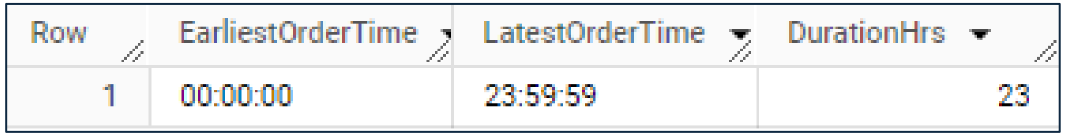


Row	table_catalog	table_schema	table_name	column_name	is_nullable	data_type
1	target-410713	target	geostates	State_ID	YES	INT64
2	target-410713	target	geostates	_State_Code	YES	STRING
3	target-410713	target	geostates	_State_Name	YES	STRING
4	target-410713	target	geostates	_Region	YES	STRING
5	target-410713	target	order_items	order_id	YES	STRING
6	target-410713	target	order_items	order_item_id	YES	INT64
7	target-410713	target	order_items	product_id	YES	STRING
8	target-410713	target	order_items	seller_id	YES	STRING
9	target-410713	target	order_items	shipping_limit_date	YES	TIMESTAMP
10	target-410713	target	order_items	price	YES	FLOAT64
11	target-410713	target	order_items	freight_value	YES	FLOAT64
12	target-410713	target	sellers	seller_id	YES	STRING
13	target-410713	target	sellers	seller_zip_code_prefix	YES	INT64
14	target-410713	target	sellers	seller_city	YES	STRING
15	target-410713	target	sellers	seller_state	YES	STRING
16	target-410713	target	geolocation	geolocation_zip_code_prefix	YES	INT64
17	target-410713	target	geolocation	geolocation_lat	YES	FLOAT64
18	target-410713	target	geolocation	geolocation_lng	YES	FLOAT64
19	target-410713	target	geolocation	geolocation_city	YES	STRING
20	target-410713	target	geolocation	geolocation_state	YES	STRING
21	target-410713	target	products	product_id	YES	STRING
22	target-410713	target	products	product_category	YES	STRING
23	target-410713	target	products	product_name_length	YES	INT64
24	target-410713	target	products	product_description_length	YES	INT64
25	target-410713	target	products	product_photos_qty	YES	INT64

Query 1.2



```sql
WITH OrderTimingRange AS (
    SELECT
        MIN(TIME(order_purchase_timestamp)) AS MinOrderTime,
        MAX(TIME(order_purchase_timestamp)) AS MaxOrderTime
    FROM target.orders
)

SELECT
    MinOrderTime AS EarliestOrderTime,
    MaxOrderTime AS LatestOrderTime,
    EXTRACT(HOUR FROM (MaxOrderTime - MinOrderTime)) AS DurationHrs
FROM OrderTimingRange;
```



Row	EarliestOrderTime	LatestOrderTime	DurationHrs
1	00:00:00	23:59:59	23

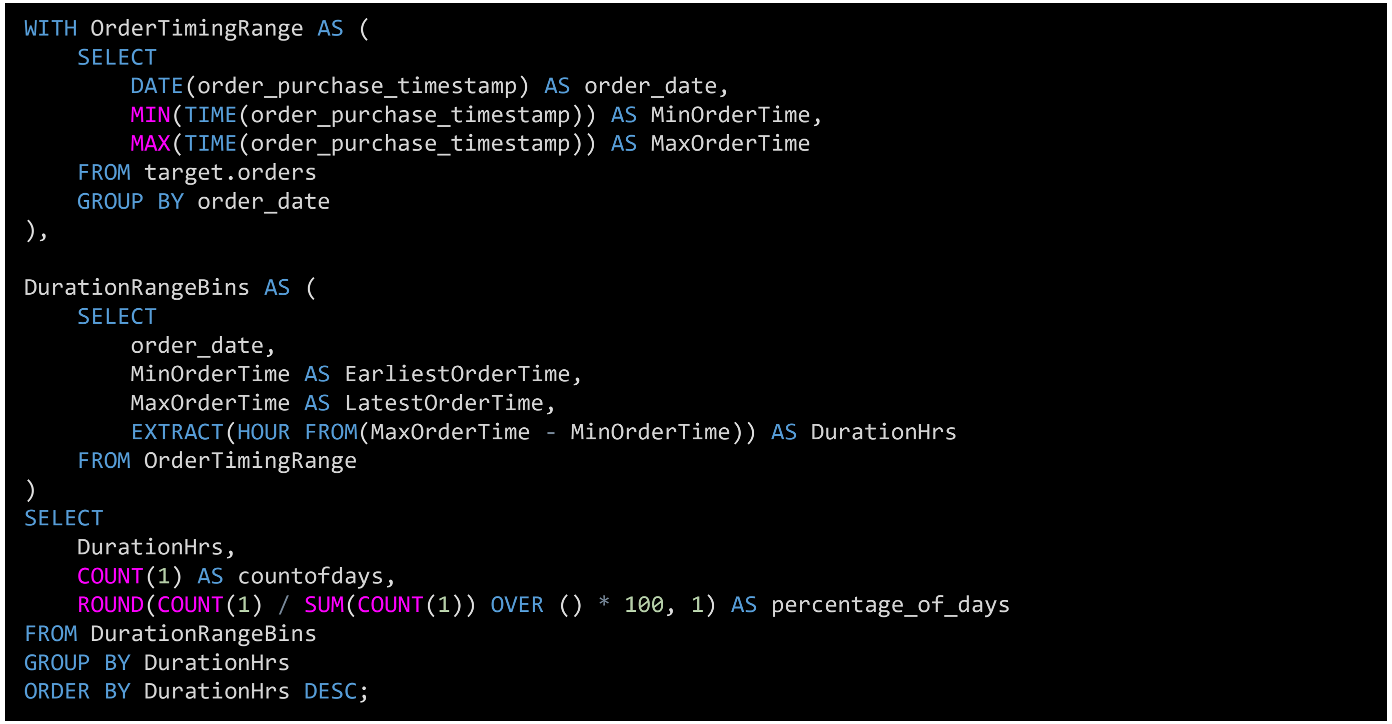
The hour part was observed as **'23 hours**' (within a range of 23 hours 59 minutes), prompting an investigation into whether this result was influenced by outliers.

The **percentage of days** within the dataset that fell into the 23-hour range was checked. 78% of the total number of days in the provided data were found to be within the **23-hour bucket**.

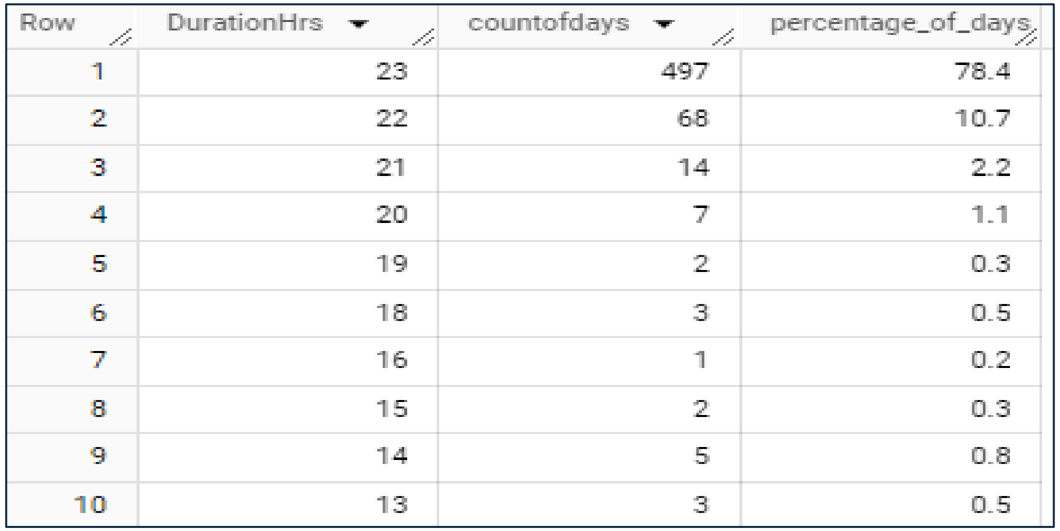
This indicated that it was not an outlier; instead, a majority of orders spanned for almost 23+ hours on **497 days**.

Thus it is conclusive to answer

The orders we placed between 12AM to 11:59 PM (23 Plus hours).

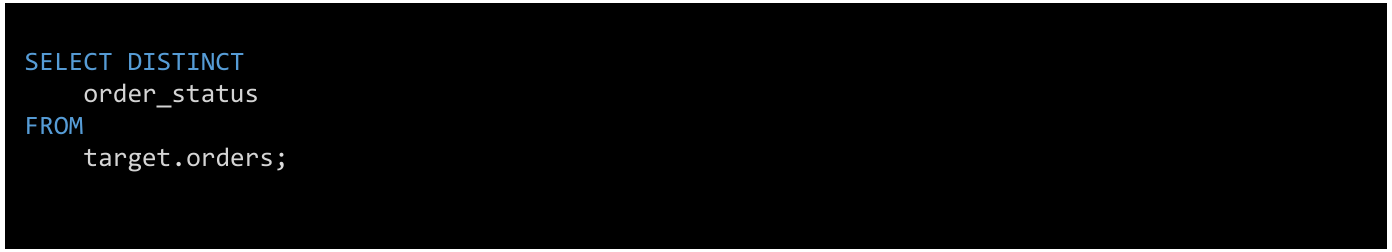


```sql
WITH OrderTimingRange AS (
    SELECT
        DATE(order_purchase_timestamp) AS order_date,
        MIN(TIME(order_purchase_timestamp)) AS MinOrderTime,
        MAX(TIME(order_purchase_timestamp)) AS MaxOrderTime
    FROM target.orders
    GROUP BY order_date
),

DurationRangeBins AS (
    SELECT
        order_date,
        MinOrderTime AS EarliestOrderTime,
        MaxOrderTime AS LatestOrderTime,
        EXTRACT(HOUR FROM(MaxOrderTime - MinOrderTime)) AS DurationHrs
    FROM OrderTimingRange
)
SELECT
    DurationHrs,
    COUNT(1) AS countofdays,
    ROUND(COUNT(1) / SUM(COUNT(1)) OVER () * 100, 1) AS percentage_of_days
FROM DurationRangeBins
GROUP BY DurationHrs
ORDER BY DurationHrs DESC;
```

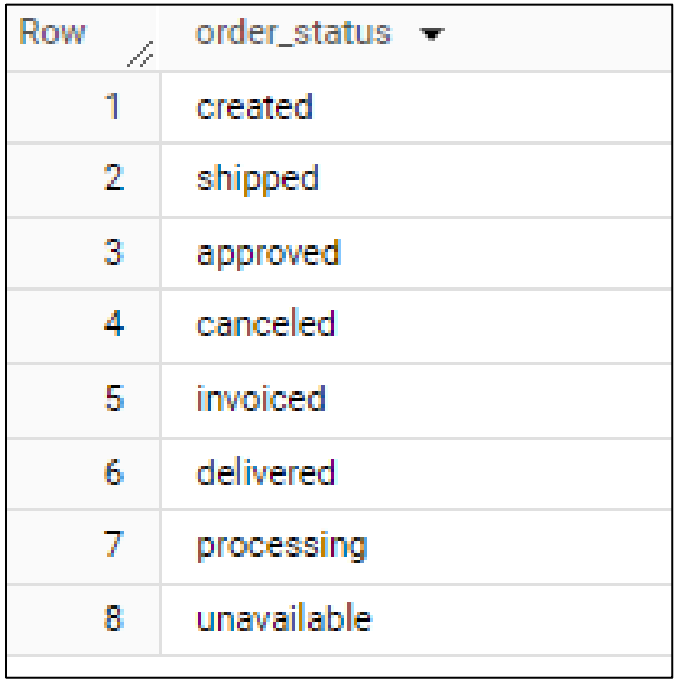


Row	DurationHrs	countofdays	percentage_of_days
1	23	497	78.4
2	22	68	10.7
3	21	14	2.2
4	20	7	1.1
5	19	2	0.3
6	18	3	0.5
7	16	1	0.2
8	15	2	0.3
9	14	5	0.8
10	13	3	0.5

Query 1.3



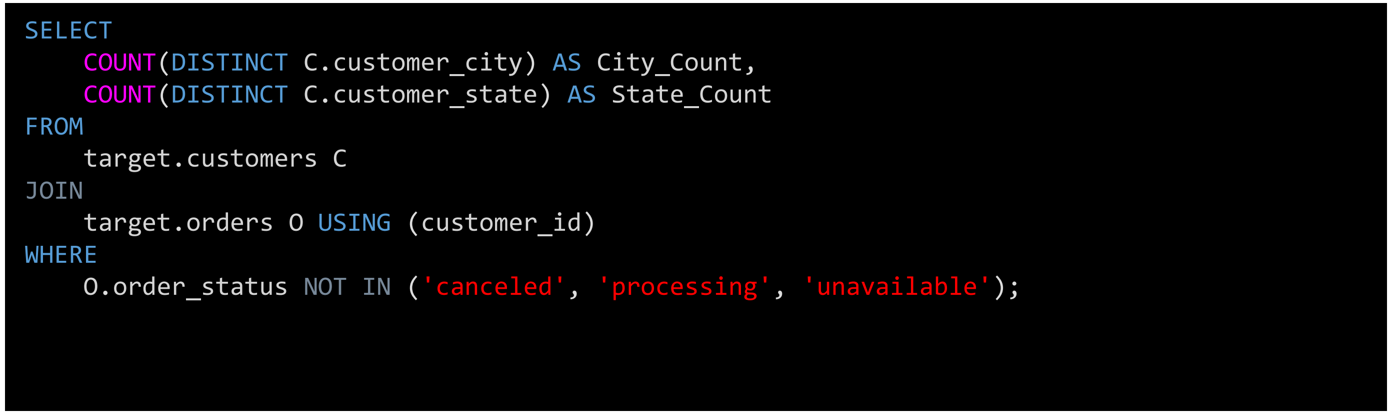
```sql
SELECT DISTINCT
    order_status
FROM
    target.orders;
```



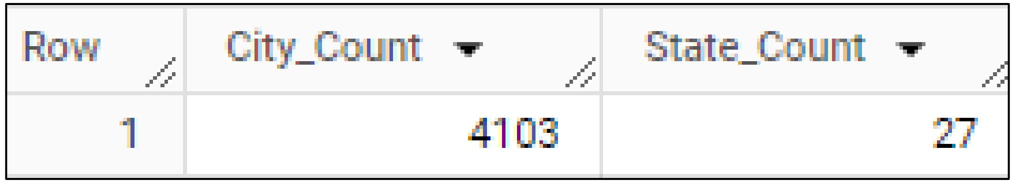
Row	order_status
1	created
2	shipped
3	approved
4	canceled
5	invoiced
6	delivered
7	processing
8	unavailable

This output provides insights into designing our filter condition. It is understood that we need to exclude records with **'canceled', 'processing', and 'unavailable'** as order statuses.

By applying this filter, we obtain 4103 unique cities across 27 states.



```sql
SELECT
    COUNT(DISTINCT C.customer_city) AS City_Count,
    COUNT(DISTINCT C.customer_state) AS State_Count
FROM
    target.customers C
JOIN
    target.orders O USING (customer_id)
WHERE
    O.order_status NOT IN ('canceled', 'processing', 'unavailable');
```



Row	City_Count	State_Count
1	4103	27

Section 2: In-depth Exploration

Questions

1. Is there a growing trend in the no. of orders placed over the past years?
2. Can we see some kind of monthly seasonality in terms of the no. of orders being placed?
3. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)
    1. 0-6 hrs : Dawn
    2. 7-12 hrs : Mornings
    3. 13-18 hrs : Afternoon
    4. 19-23 hrs : Night

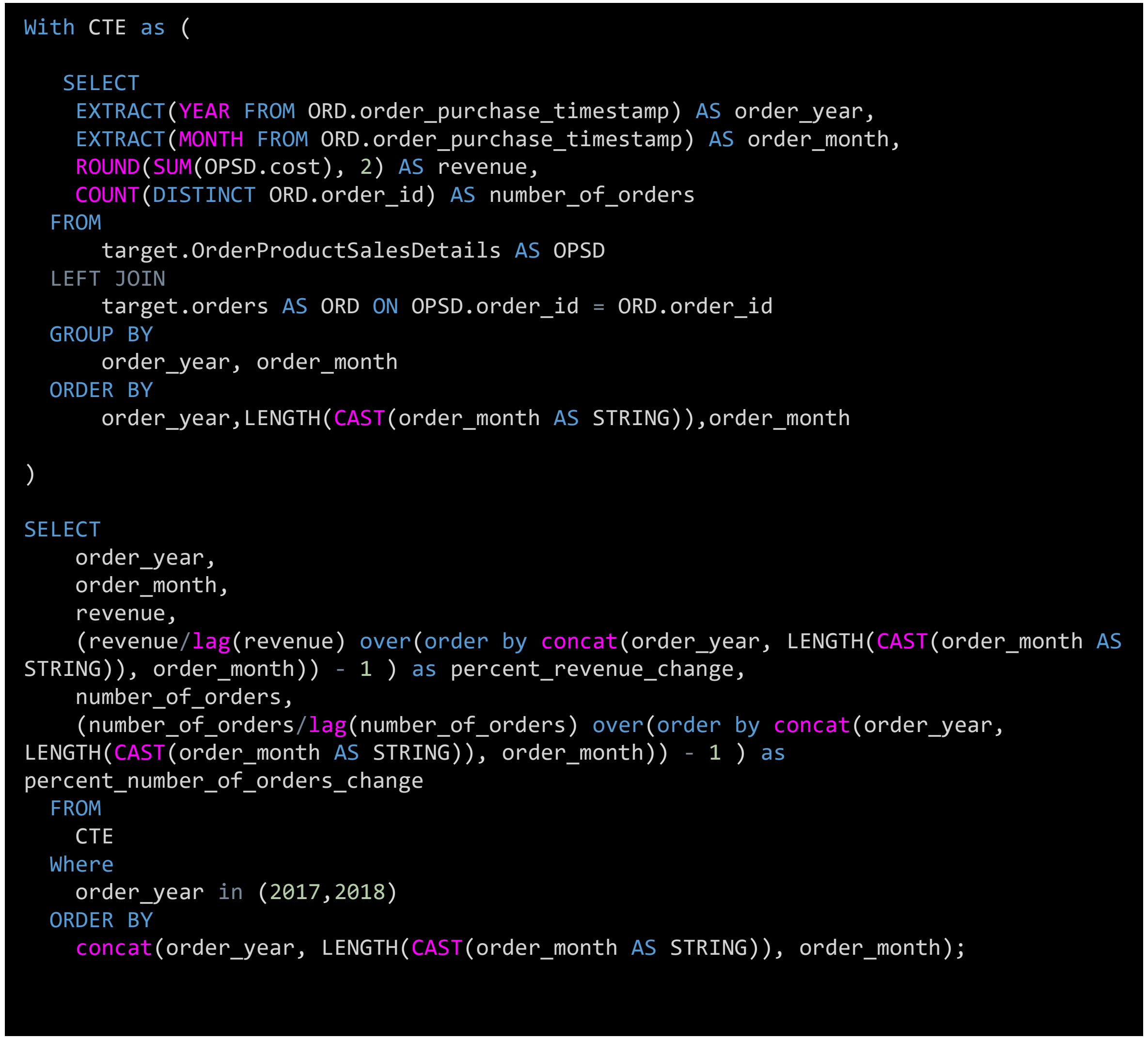
Solution

SQL Query 2.1 is designed to analyse the trend in the number of orders (and also revenue) over time. Specifically, it calculates the percent change compared to the previous month to understand the trend.

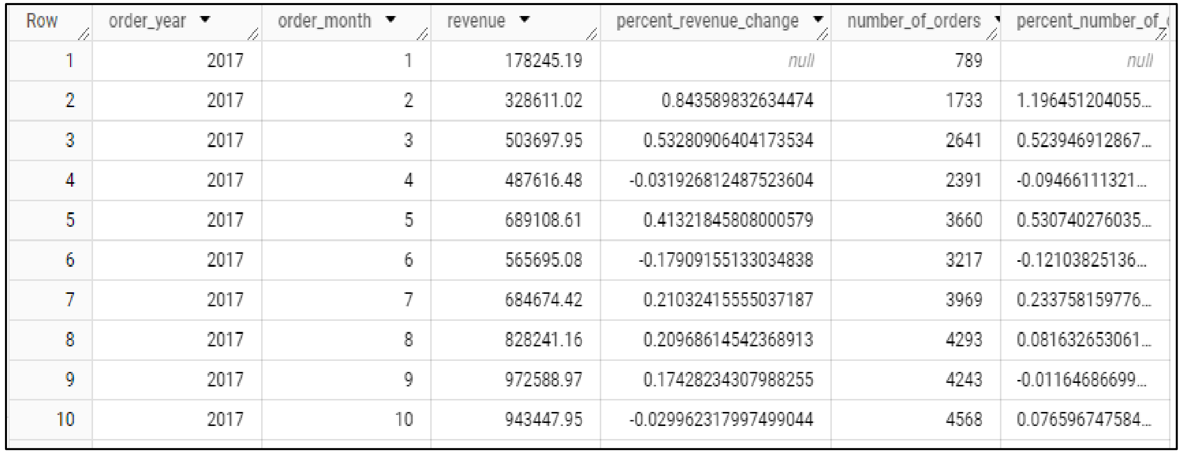
The retrieved data is then visualised in waterfall-chart to understand the growth trend and changes. This answers first 2 questions.

SQL Query 2.1 is structured to answer question 3, answering the number of orders placed in terms of numbers and percentage segmented by the ordering pattern of the customer.

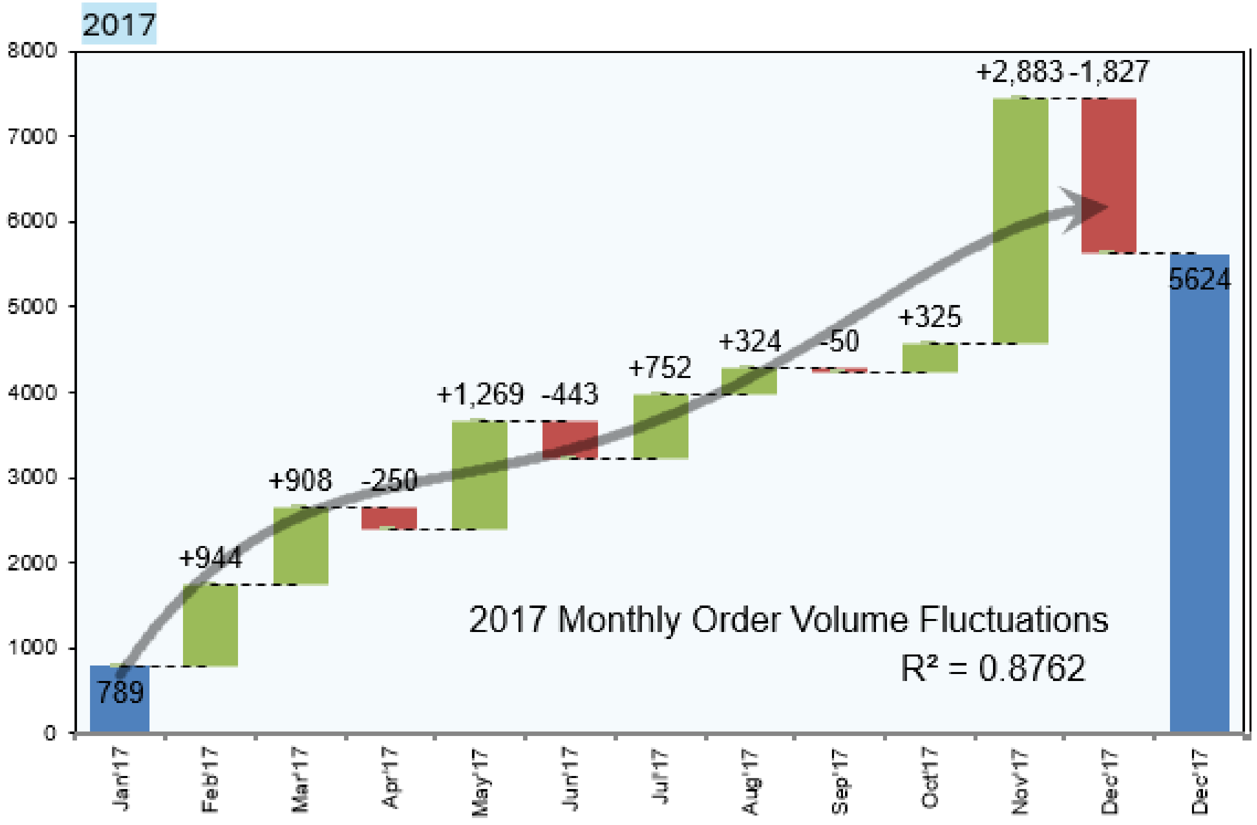
Query 2.1



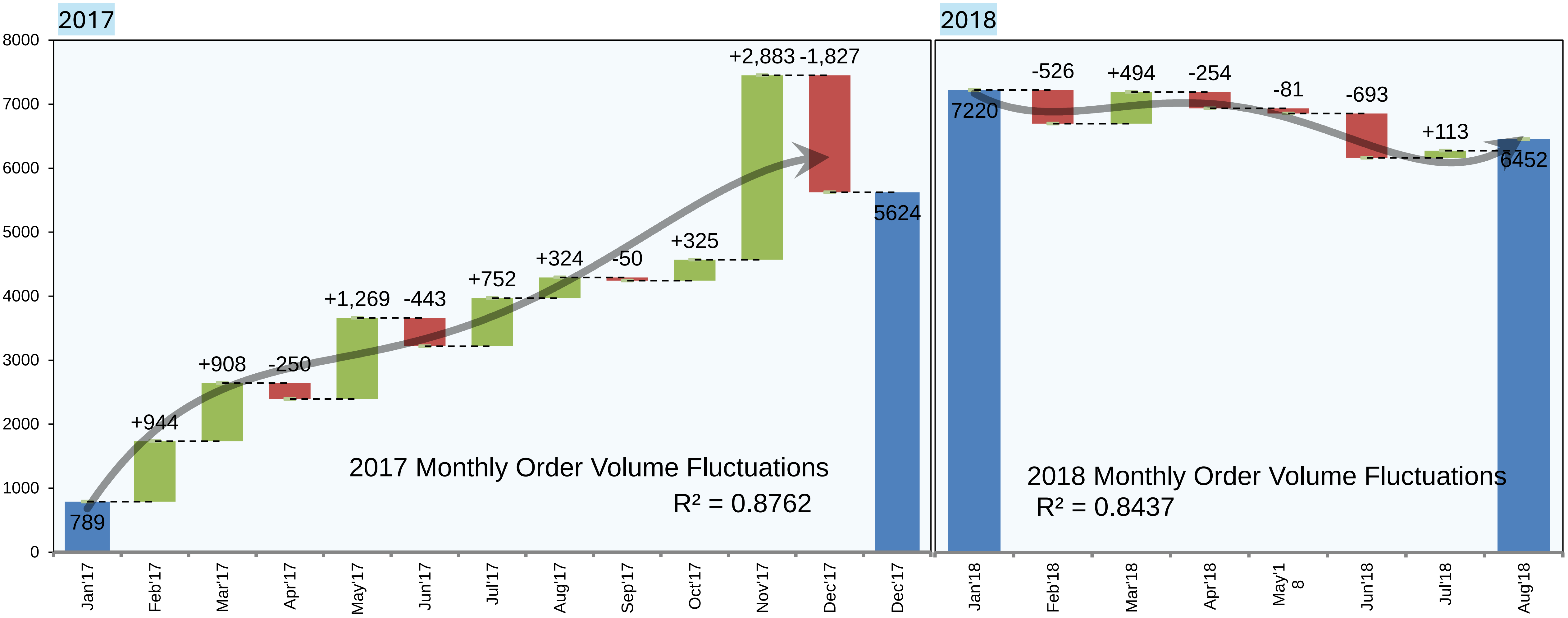
```
With CTE as (

   SELECT
    EXTRACT(YEAR FROM ORD.order_purchase_timestamp) AS order_year,
    EXTRACT(MONTH FROM ORD.order_purchase_timestamp) AS order_month,
    ROUND(SUM(OPSD.cost), 2) AS revenue,
    COUNT(DISTINCT ORD.order_id) AS number_of_orders
  FROM
      target.OrderProductSalesDetails AS OPSD
  LEFT JOIN
      target.orders AS ORD ON OPSD.order_id = ORD.order_id
  GROUP BY
      order_year, order_month
  ORDER BY
      order_year,LENGTH(CAST(order_month AS STRING)),order_month

)

SELECT
    order_year,
    order_month,
    revenue,
    (revenue/lag(revenue) over(order by concat(order_year, LENGTH(CAST(order_month AS
STRING)), order_month)) - 1 ) as percent_revenue_change,
    number_of_orders,
    (number_of_orders/lag(number_of_orders) over(order by concat(order_year,
LENGTH(CAST(order_month AS STRING)), order_month)) - 1 ) as
percent_number_of_orders_change
  FROM
    CTE
  Where
    order_year in (2017,2018)
  ORDER BY
    concat(order_year, LENGTH(CAST(order_month AS STRING)), order_month);
```



Row	order_year	order_month	revenue	percent_revenue_change	number_of_orders	percent_number_of_
1	2017	1	178245.19	null	789	null
2	2017	2	328611.02	0.843589832634474	1733	1.196451204055...
3	2017	3	503697.95	0.53280906404173534	2641	0.523946912867...
4	2017	4	487616.48	-0.031926812487523604	2391	-0.09466111321...
5	2017	5	689108.61	0.41321845808000579	3660	0.530740276035...
6	2017	6	565695.08	-0.17909155133034838	3217	-0.12103825136...
7	2017	7	684674.42	0.21032415555037187	3969	0.233758159776...
8	2017	8	828241.16	0.20968614542368913	4293	0.081632653061...
9	2017	9	972588.97	0.17428234307988255	4243	-0.01164686699...
10	2017	10	943447.95	-0.029962317997499044	4568	0.076596747584...



Order Volume Trend (in Nos.)



Quarter	2017			2018		
	% Change	Trend	Remarks	% Change	Trend	Remarks
Q1	N/A	Steady growth	Scope of market is open	6.50%	Recovery with moderate increase	Bouncing back from challenges faced in the previous quarter
Q2	130.90%	Continued growth	Strong growth, peak in May; potential increase due to seasonal trends	-14.40%	Decrease in orders	Market possibly saturated or influenced by fluctuations
Q3	52.80%	Consistent increase	Continued positive performance; customers consistently engaging	2.80%	Slight increase	Moderate positive performance; potential seasonal boost
Q4	-24.40%	Decline in December	Decline after peak in November; potential impact of holiday season.	N/A	-	No data available for Q4 2018; anticipating seasonal patterns

Inference from the Result

January 2017 to December 2017:

The number of orders increased from January to November 2017 by a factor of 8 times.

There is a significant increase in November 2017 (up by 63.11%).

However, there is a sharp decrease in December 2017 (down by 24.52%).

January 2018 to August 2018:

The trend in January 2018 shows an increase compared to December 2017.

February 2018 shows a slight decrease (down by 7.29%).

March 2018 sees an increase again (up by 7.38%).

From April to August 2018, there's a gradual decrease in the number of orders.

Observations:

The data suggests a cyclic pattern with peaks in November 2017 and January 2018.

The months of December 2017 and September 2018 show significant drops in the number of orders.

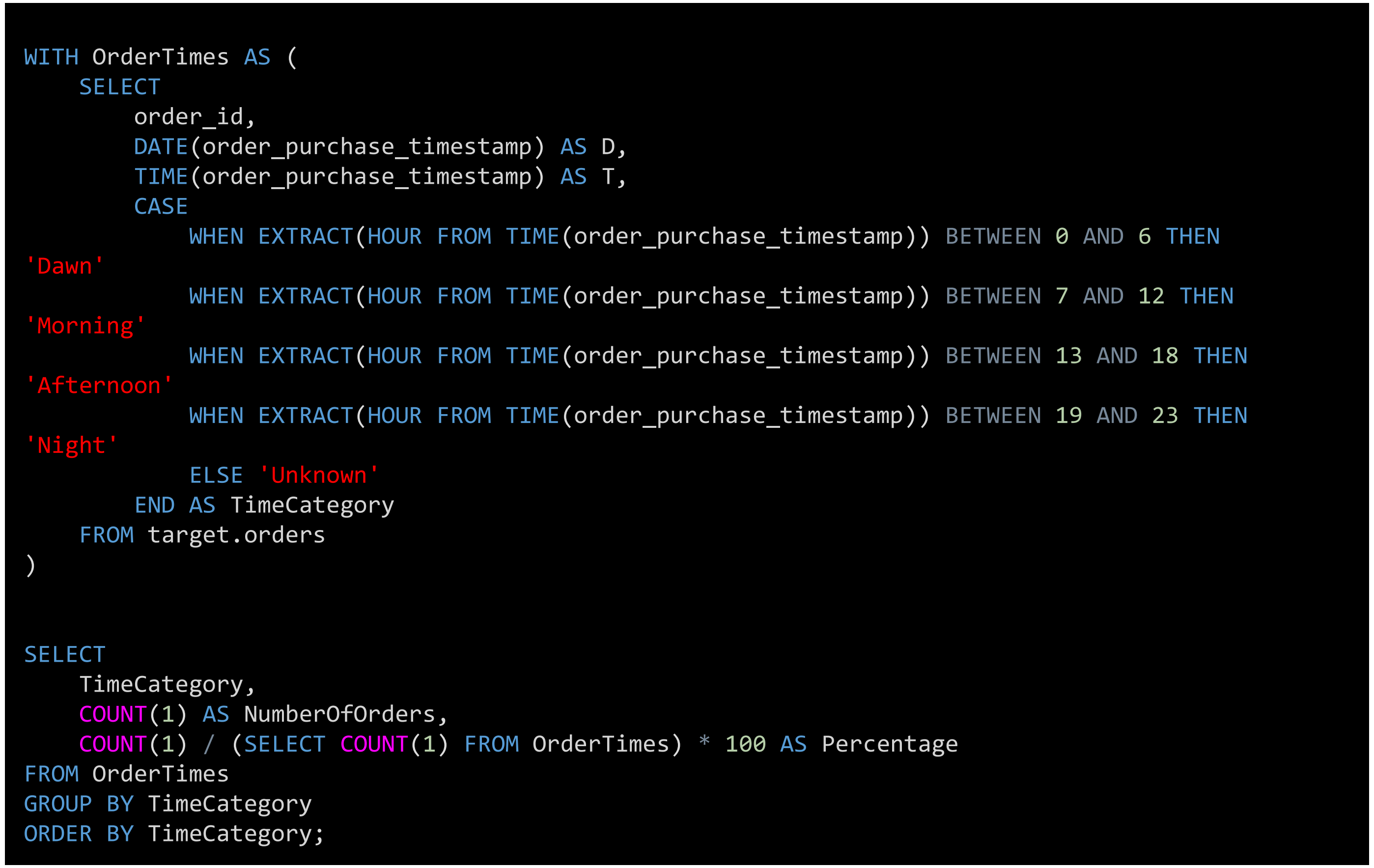
Conclusion:

While there was growth in the number of orders during certain months, there were also periods of decline.

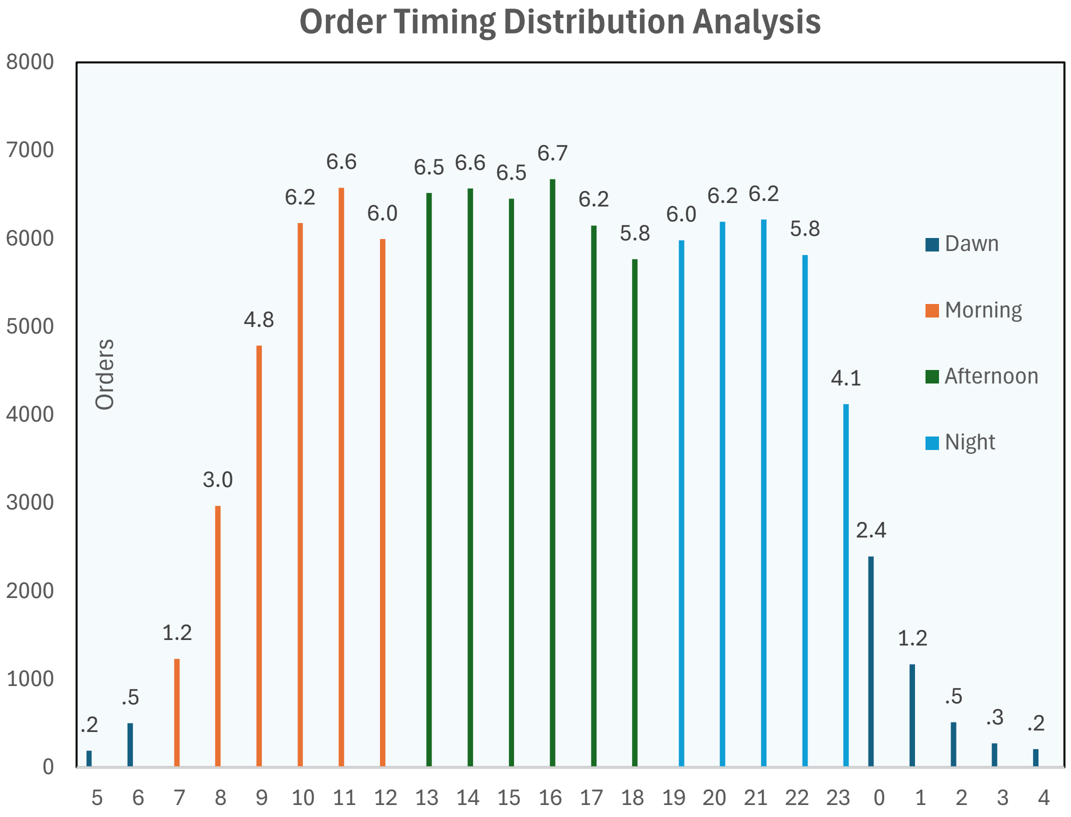
It's important to consider external factors such as holidays, promotions, or marketing campaigns that may have influenced these trends.

Query 2.2

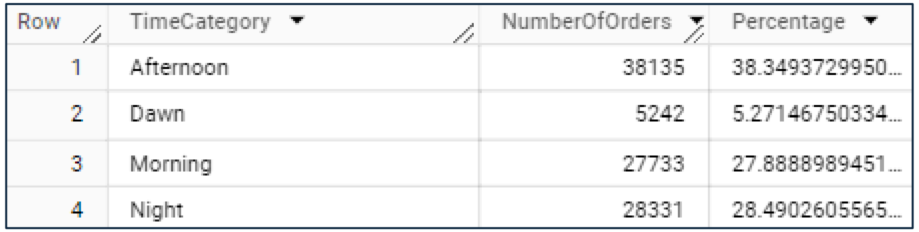
The analysis reveals that the peak order placement occurs consistently from 9 am to 10 pm, with order volumes ranging from 4.5K to 6.7K during this timeframe.



```sql
WITH OrderTimes AS (
    SELECT
        order_id,
        DATE(order_purchase_timestamp) AS D,
        TIME(order_purchase_timestamp) AS T,
        CASE
            WHEN EXTRACT(HOUR FROM TIME(order_purchase_timestamp)) BETWEEN 0 AND 6 THEN
'Dawn'
            WHEN EXTRACT(HOUR FROM TIME(order_purchase_timestamp)) BETWEEN 7 AND 12 THEN
'Morning'
            WHEN EXTRACT(HOUR FROM TIME(order_purchase_timestamp)) BETWEEN 13 AND 18 THEN
'Afternoon'
            WHEN EXTRACT(HOUR FROM TIME(order_purchase_timestamp)) BETWEEN 19 AND 23 THEN
'Night'
            ELSE 'Unknown'
        END AS TimeCategory
    FROM target.orders
)


SELECT
    TimeCategory,
    COUNT(1) AS NumberOfOrders,
    COUNT(1) / (SELECT COUNT(1) FROM OrderTimes) * 100 AS Percentage
FROM OrderTimes
GROUP BY TimeCategory
ORDER BY TimeCategory;
```



Order Timing Distribution Analysis



Row	TimeCategory	NumberOfOrders	Percentage
1	Afternoon	38135	38.3493729950…
2	Dawn	5242	5.27146750334…
3	Morning	27733	27.8888989451…
4	Night	28331	28.4902605565…

Inference: The distribution illustrates the variations in order placements throughout the day, providing insights into customer behaviour and preferences during different time categories. The Morning and Afternoon periods appear to be the busiest, while the Dawn and Night categories experience lower order activity.

Section 3: Evolution of E-commerce

Questions

Evolution of E-commerce orders in the Brazil region:

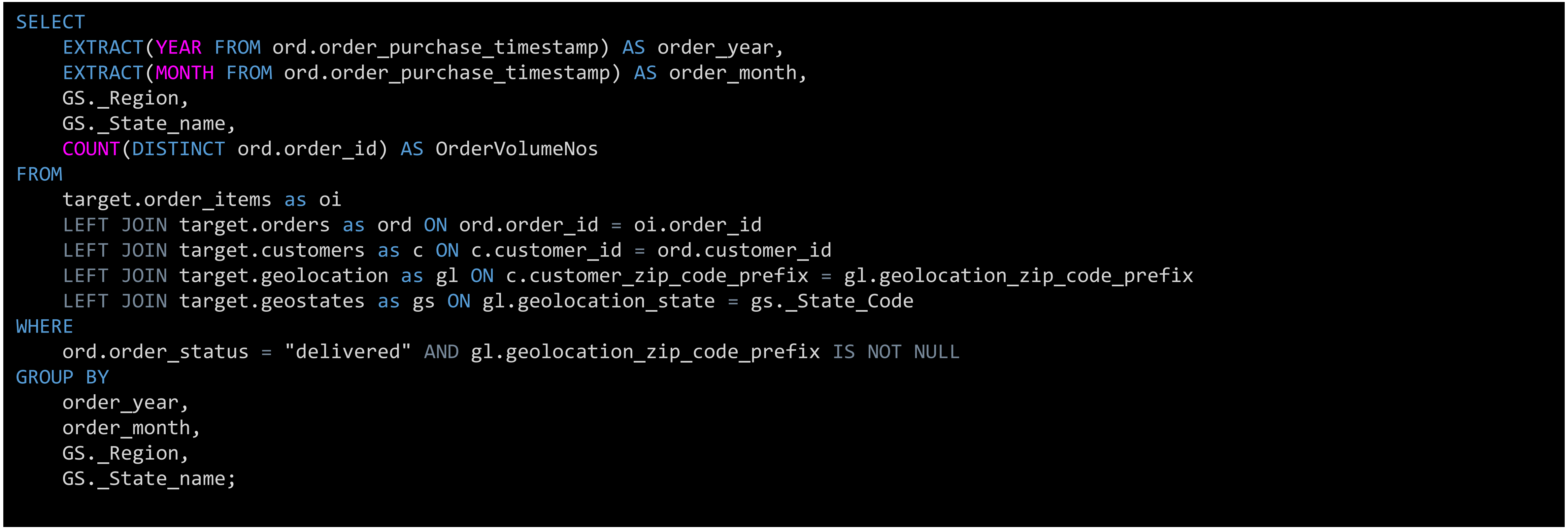
1. Get the month-on-month no. of orders placed in each state.
2. How are the customers distributed across all the states?

Solution

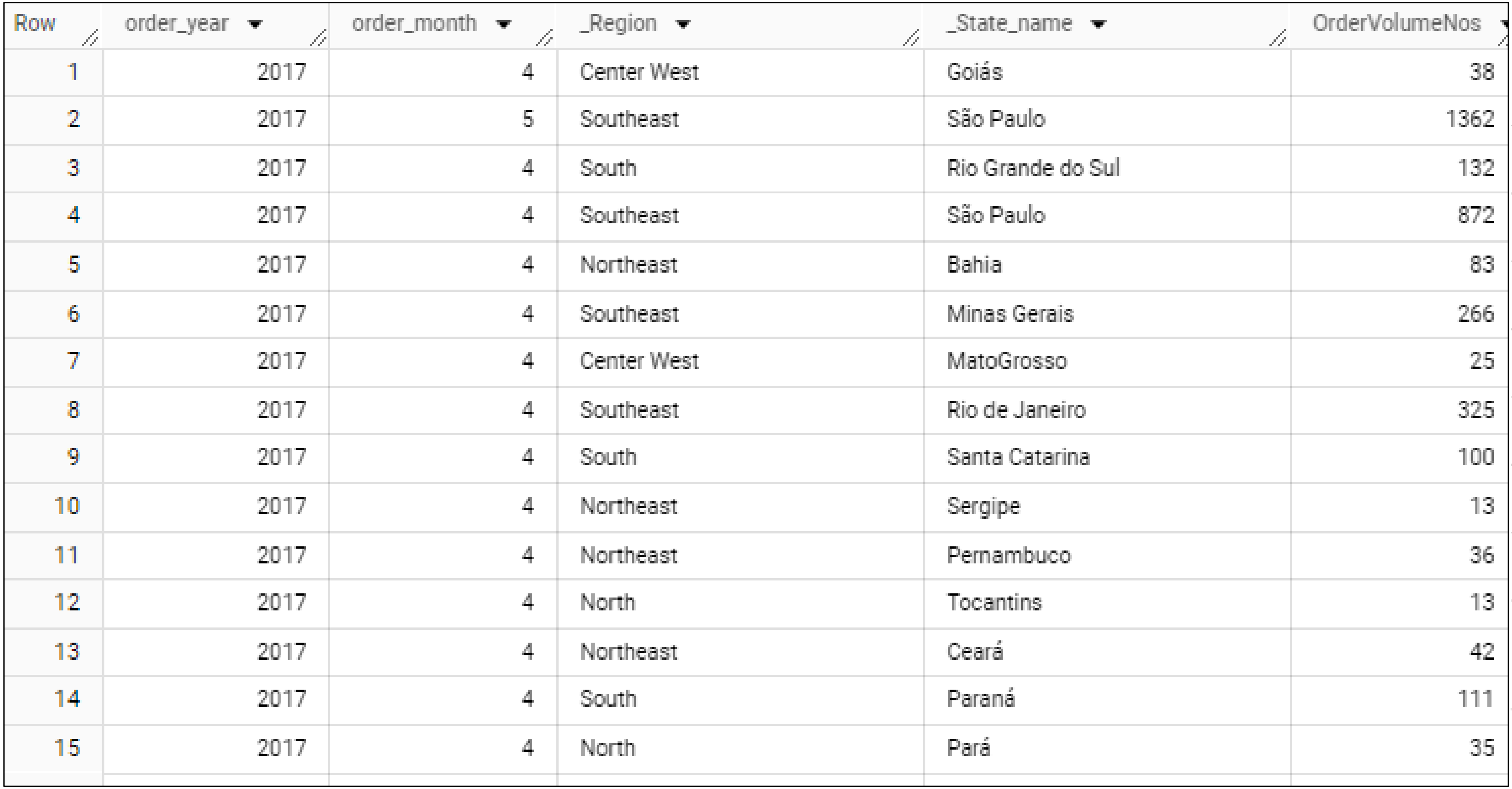
SQL Query 3.1 is structed to answer question 1 where Order_Purchase_TimeStamp is grouped to the granularity of Year, Months, Region and State to retrieve the count of orders that were placed and delivered (order_status=“delivered”).

SQL Query 3.2 is

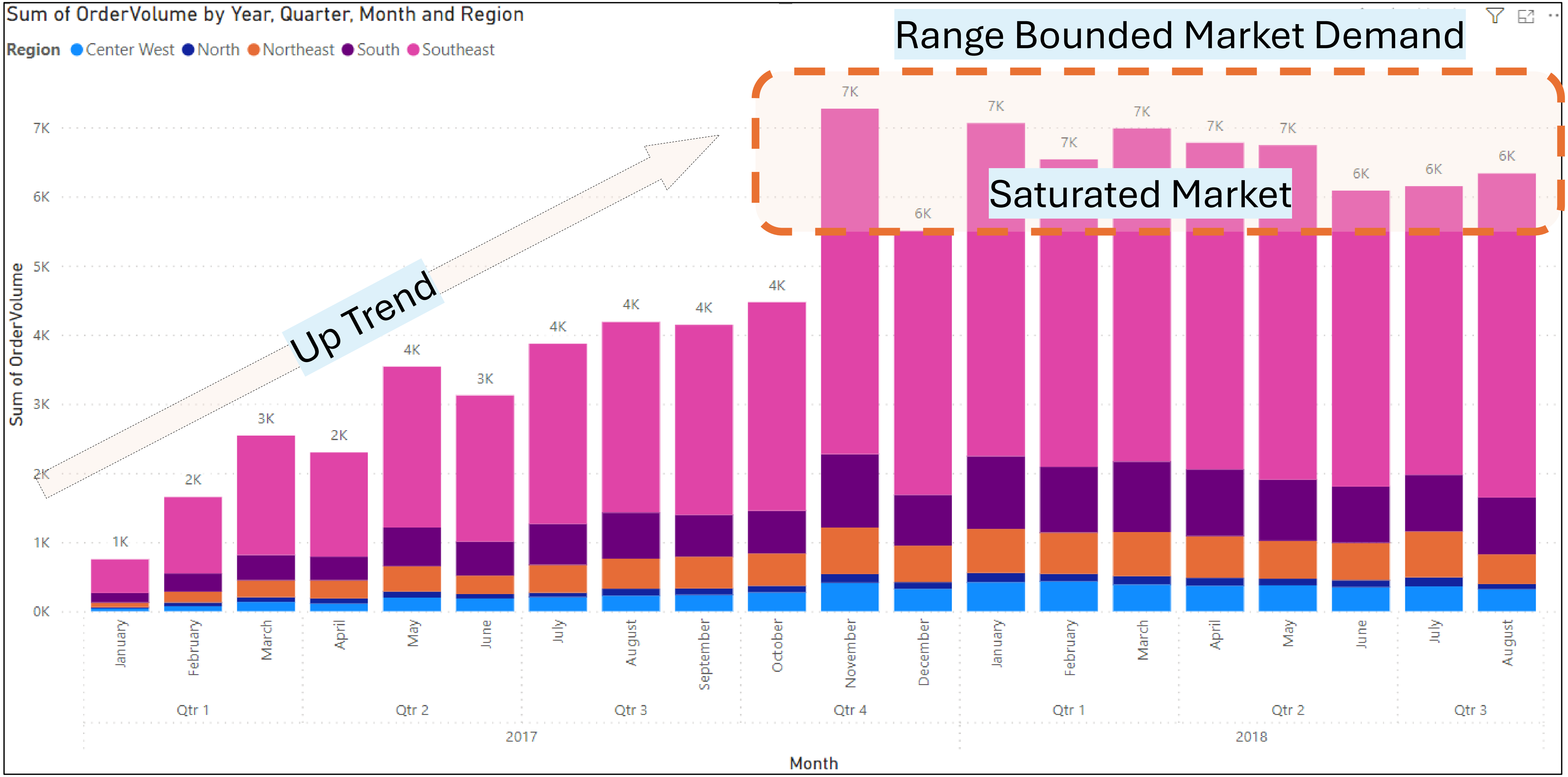
Query 3.1



```
SELECT
    EXTRACT(YEAR FROM ord.order_purchase_timestamp) AS order_year,
    EXTRACT(MONTH FROM ord.order_purchase_timestamp) AS order_month,
    GS._Region,
    GS._State_name,
    COUNT(DISTINCT ord.order_id) AS OrderVolumeNos
FROM
    target.order_items as oi
    LEFT JOIN target.orders as ord ON ord.order_id = oi.order_id
    LEFT JOIN target.customers as c ON c.customer_id = ord.customer_id
    LEFT JOIN target.geolocation as gl ON c.customer_zip_code_prefix = gl.geolocation_zip_code_prefix
    LEFT JOIN target.geostates as gs ON gl.geolocation_state = gs._State_Code
WHERE
    ord.order_status = "delivered" AND gl.geolocation_zip_code_prefix IS NOT NULL
GROUP BY
    order_year,
    order_month,
    GS._Region,
    GS._State_name;
```



Row	order_year	order_month	_Region	_State_name	OrderVolumeNos
1	2017	4	Center West	Goiás	38
2	2017	5	Southeast	São Paulo	1362
3	2017	4	South	Rio Grande do Sul	132
4	2017	4	Southeast	São Paulo	872
5	2017	4	Northeast	Bahia	83
6	2017	4	Southeast	Minas Gerais	266
7	2017	4	Center West	MatoGrosso	25
8	2017	4	Southeast	Rio de Janeiro	325
9	2017	4	South	Santa Catarina	100
10	2017	4	Northeast	Sergipe	13
11	2017	4	Northeast	Pernambuco	36
12	2017	4	North	Tocantins	13
13	2017	4	Northeast	Ceará	42
14	2017	4	South	Paraná	111
15	2017	4	North	Pará	35



Sum of OrderVolume by Year, Quarter, Month and Region

Range Bounded Market Demand

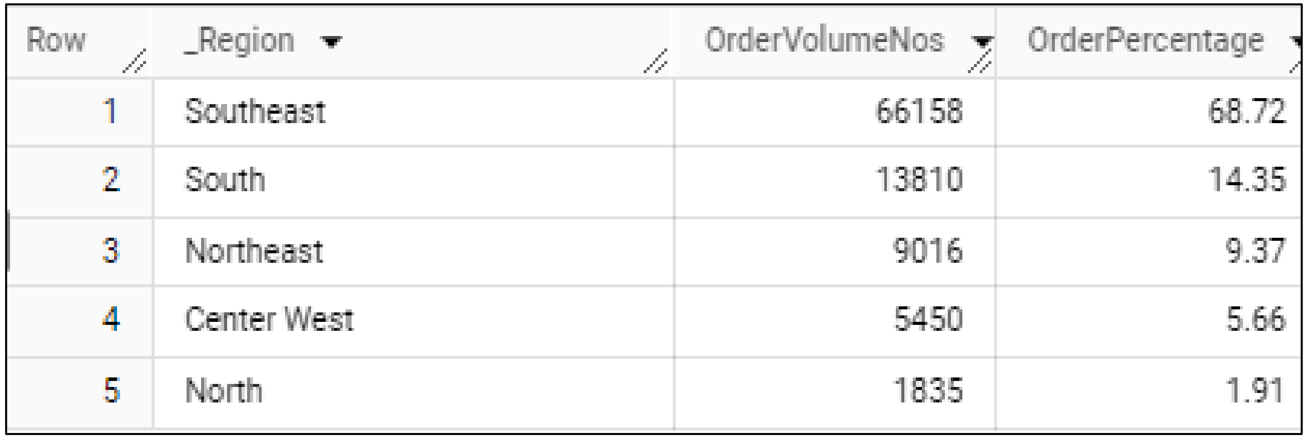
Up Trend

Saturated Market

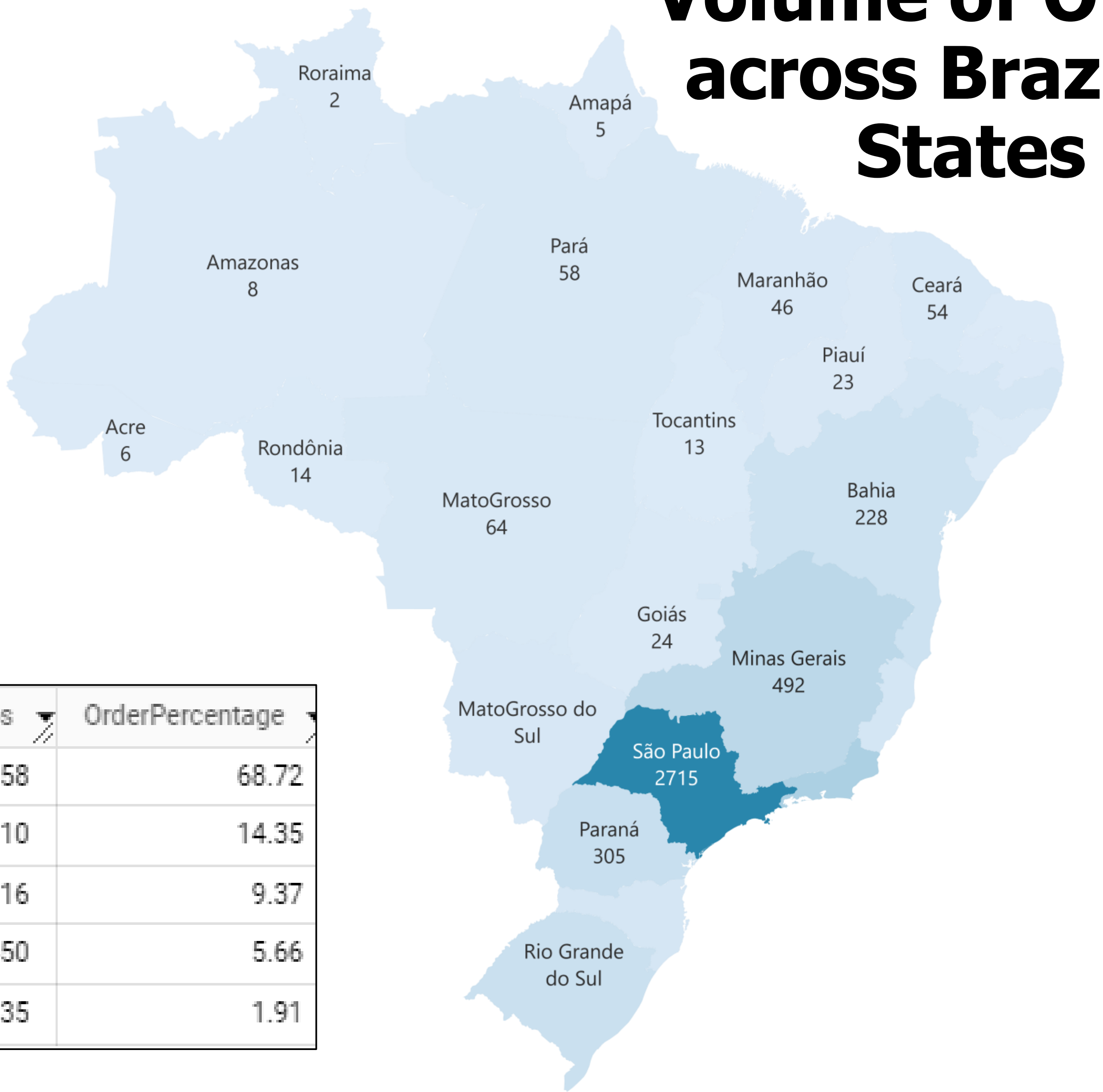
Volume of Orders across Brazilian States

Inference:

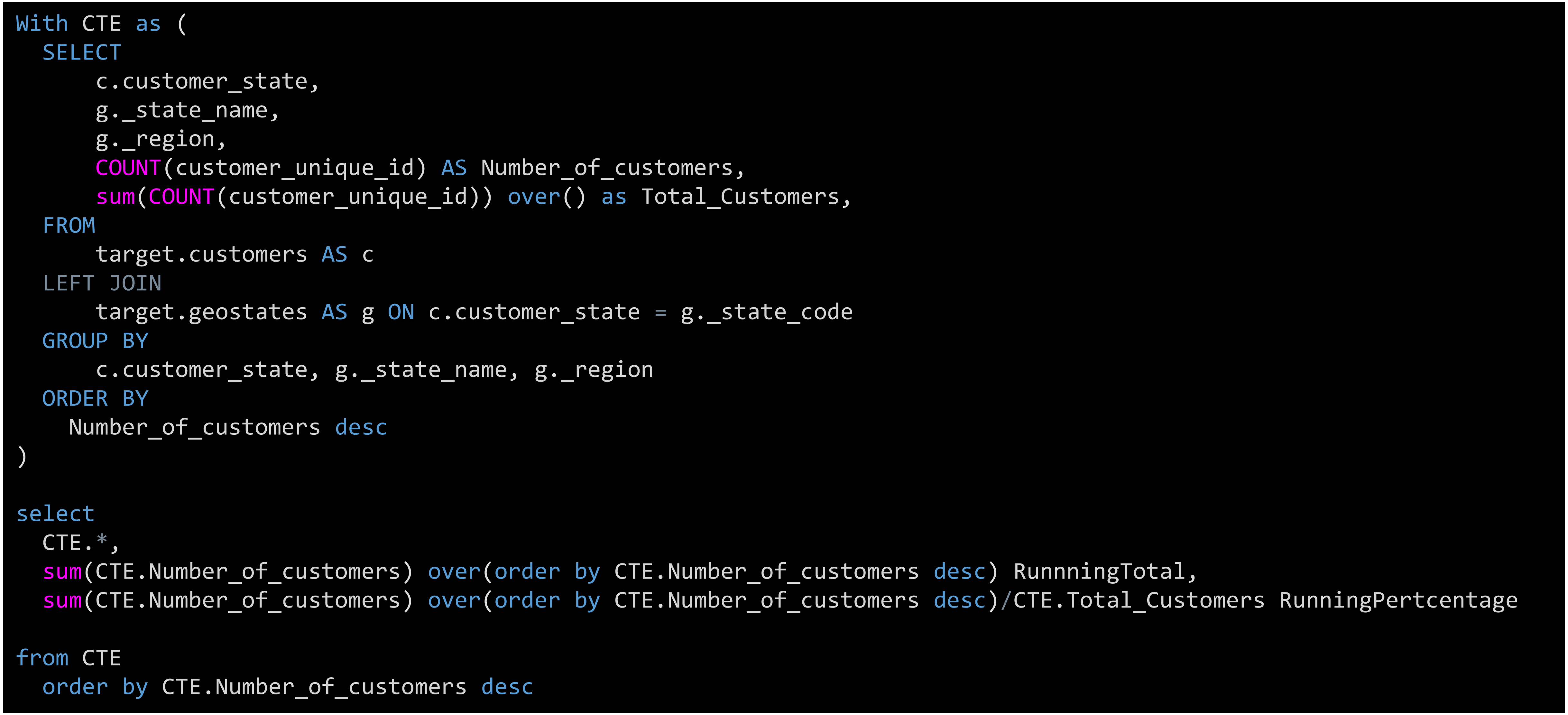
- The South-East region dominates with São Paulo, Rio de Janeiro, and Minas Gerais contributing to 68.7% of the total order volume.
- Followed by the South region, led by Rio Grande do Sul, Paraná, and Santa Catarina, accounts for 14.35%.
- Other regions, including North-East, Center West & North, contribute proportionally, while Northern states constitute the remaining small market.



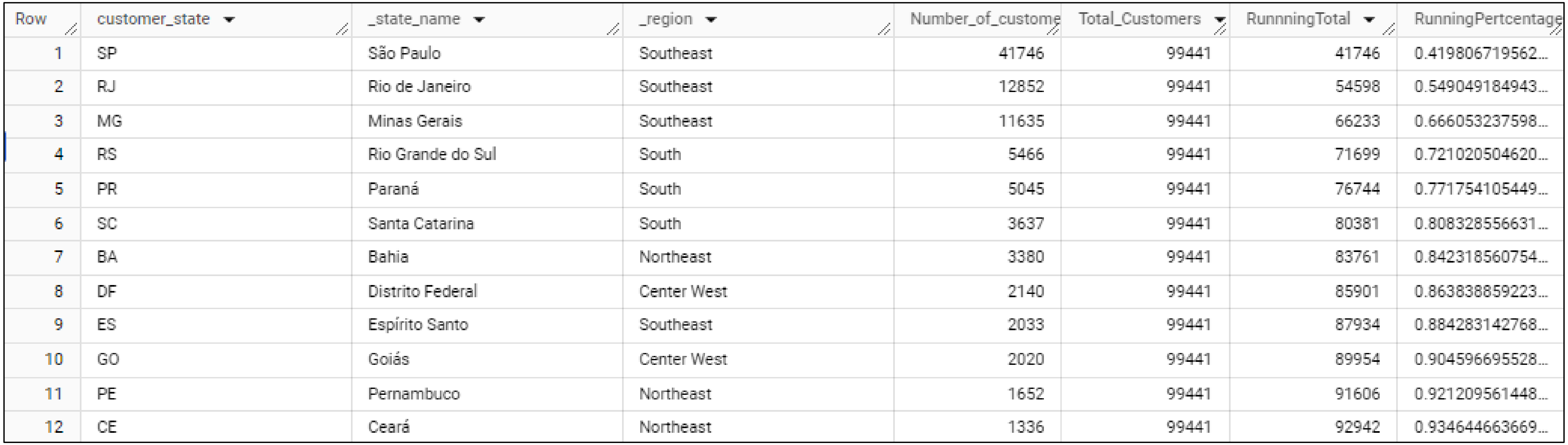
Row	_Region	OrderVolumeNos	OrderPercentage
1	Southeast	66158	68.72
2	South	13810	14.35
3	Northeast	9016	9.37
4	Center West	5450	5.66
5	North	1835	1.91



Query 3.2

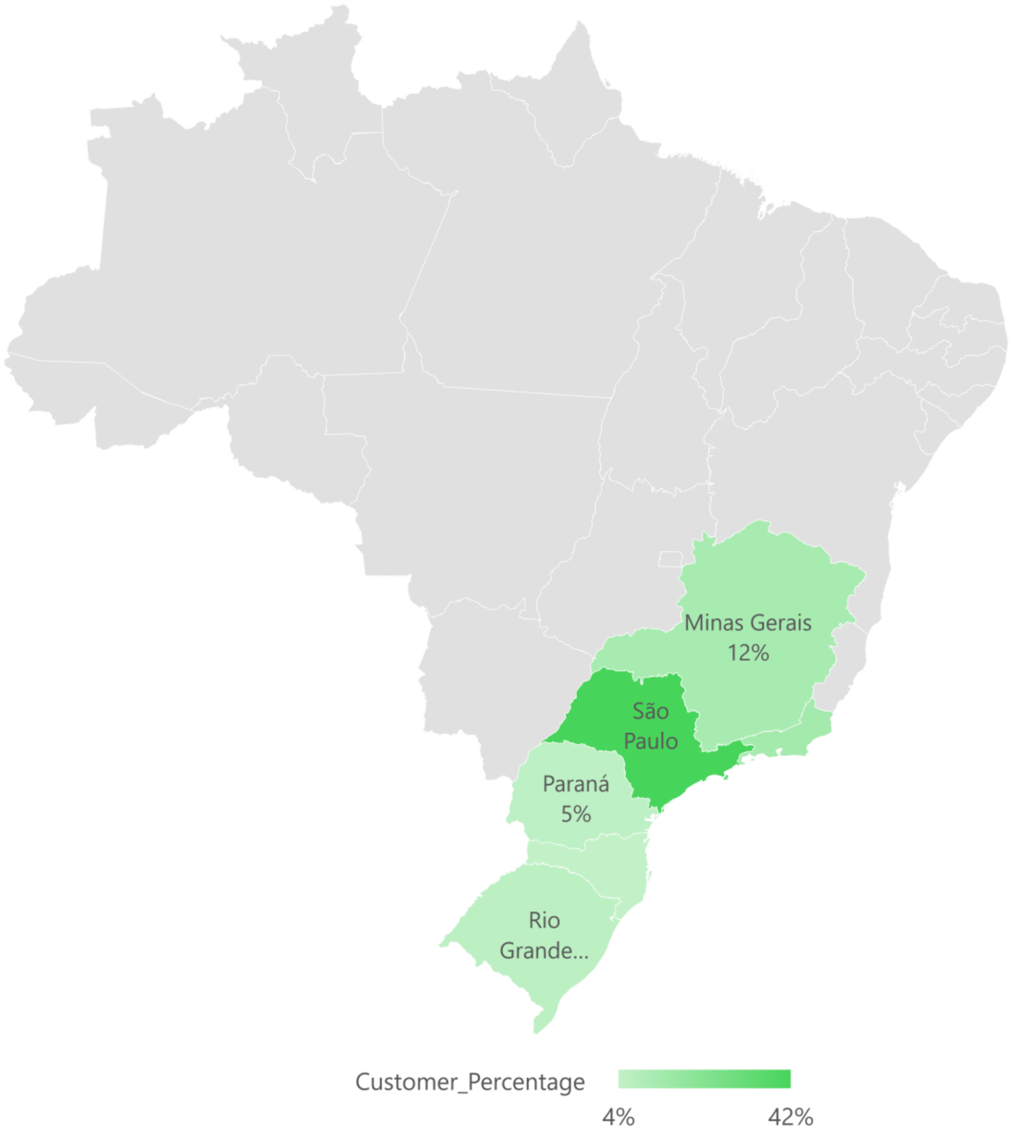


```sql
With CTE as (
  SELECT
      c.customer_state,
      g._state_name,
      g._region,
      COUNT(customer_unique_id) AS Number_of_customers,
      sum(COUNT(customer_unique_id)) over() as Total_Customers,
  FROM
      target.customers AS c
  LEFT JOIN
      target.geostates AS g ON c.customer_state = g._state_code
  GROUP BY
      c.customer_state, g._state_name, g._region
  ORDER BY
    Number_of_customers desc
)

select
  CTE.*,
  sum(CTE.Number_of_customers) over(order by CTE.Number_of_customers desc) RunnningTotal,
  sum(CTE.Number_of_customers) over(order by CTE.Number_of_customers desc)/CTE.Total_Customers RunningPertcentage

from CTE
  order by CTE.Number_of_customers desc
```

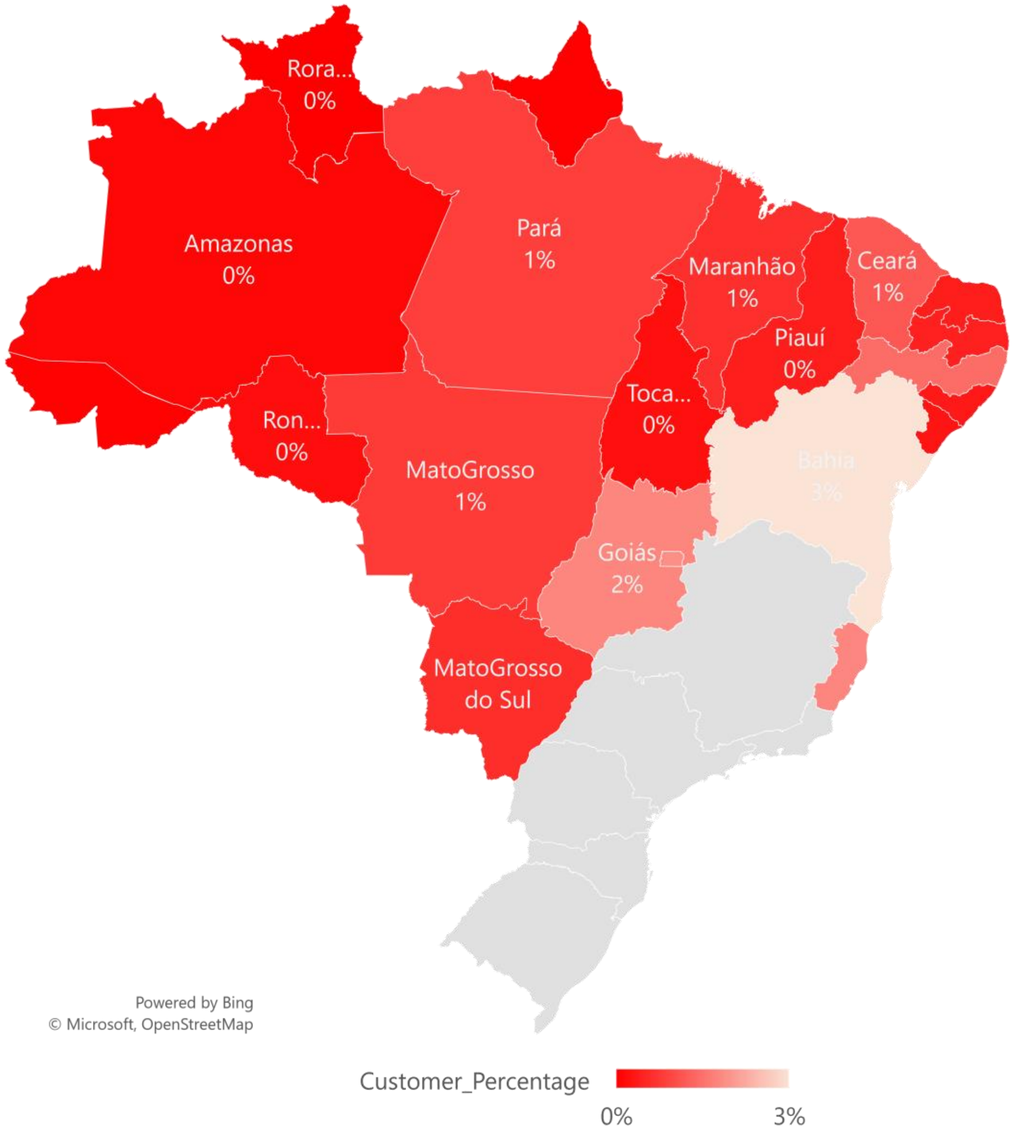


Row	customer_state	_state_name	_region	Number_of_custome	Total_Customers	RunnningTotal	RunningPertcentage
1	SP	São Paulo	Southeast	41746	99441	41746	0.419806719562...
2	RJ	Rio de Janeiro	Southeast	12852	99441	54598	0.549049184943...
3	MG	Minas Gerais	Southeast	11635	99441	66233	0.666053237598...
4	RS	Rio Grande do Sul	South	5466	99441	71699	0.721020504620...
5	PR	Paraná	South	5045	99441	76744	0.771754105449...
6	SC	Santa Catarina	South	3637	99441	80381	0.808328556631...
7	BA	Bahia	Northeast	3380	99441	83761	0.842318560754...
8	DF	Distrito Federal	Center West	2140	99441	85901	0.863838859223...
9	ES	Espírito Santo	Southeast	2033	99441	87934	0.884283142768...
10	GO	Goiás	Center West	2020	99441	89954	0.904596695528...
11	PE	Pernambuco	Northeast	1652	99441	91606	0.921209561448...
12	CE	Ceará	Northeast	1336	99441	92942	0.934644663669...

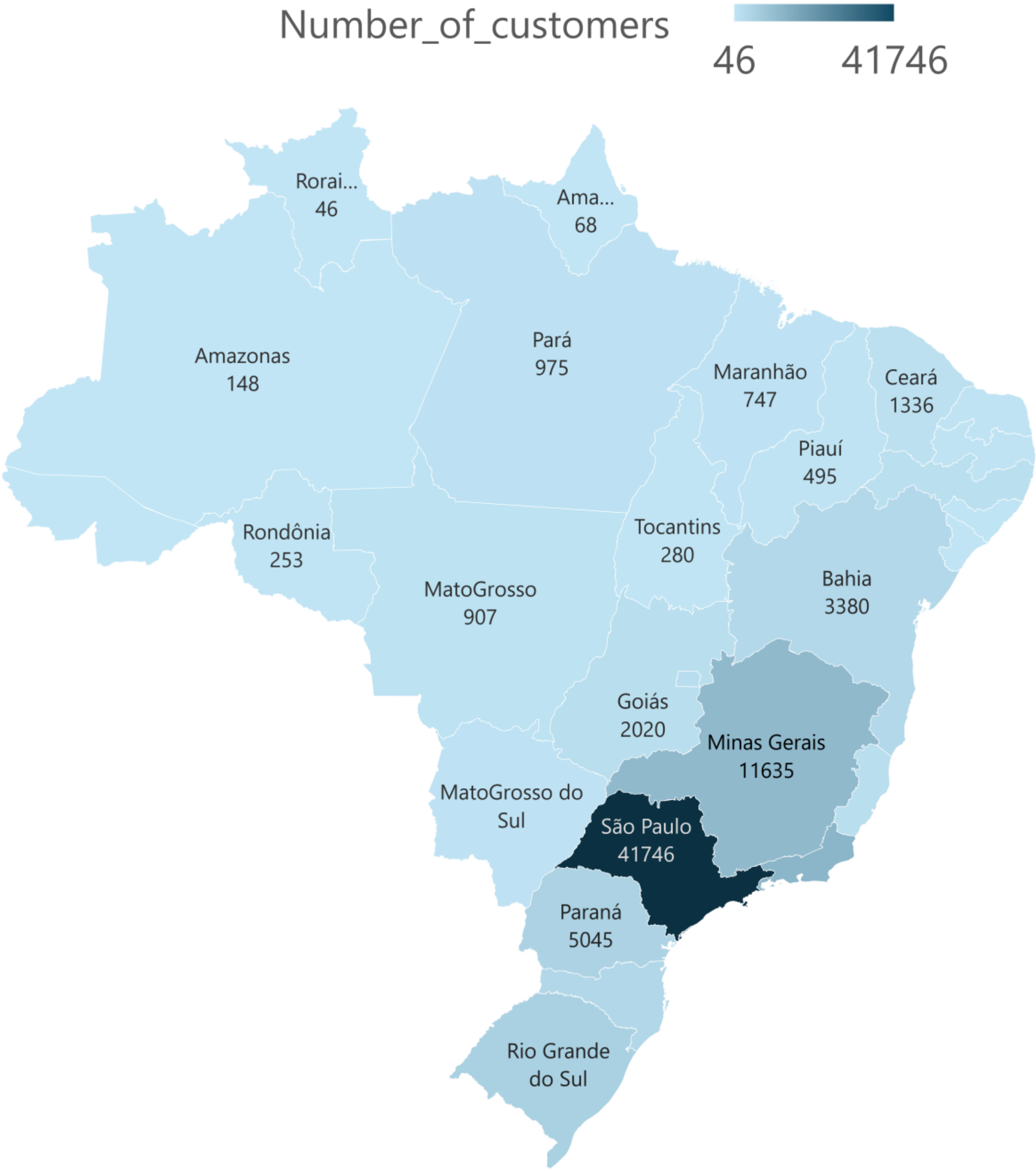
6 States (Southern and Southeastern) that contributes to 80% of the customer bases

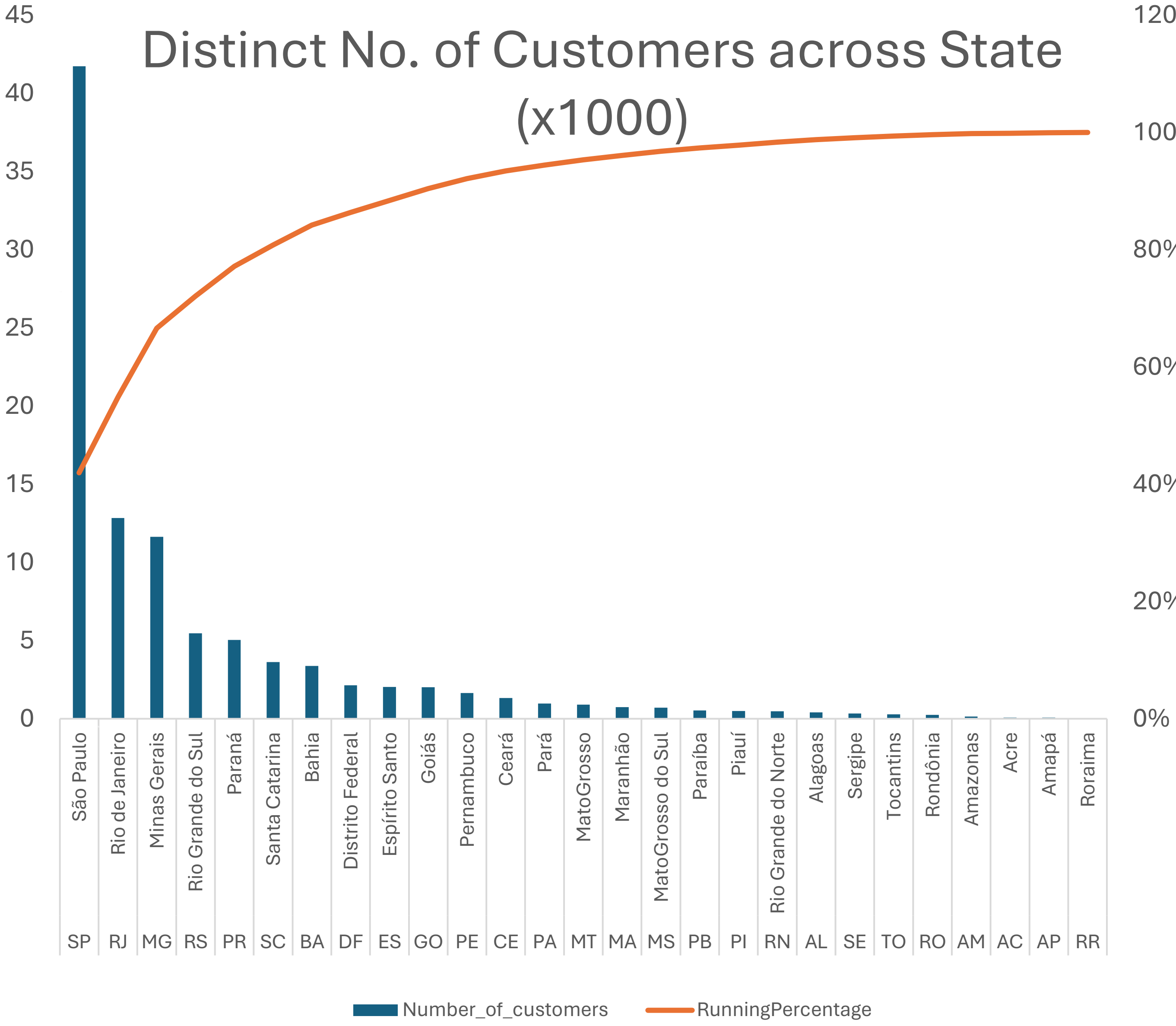


21 States (North, West) that contributes to the remaining 20% of the customer bases



Powered by Bing
© Microsoft, OpenStreetMap




Distinct No. of Customers across State (x1000)

Section 4: Impact on Economy

Question

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

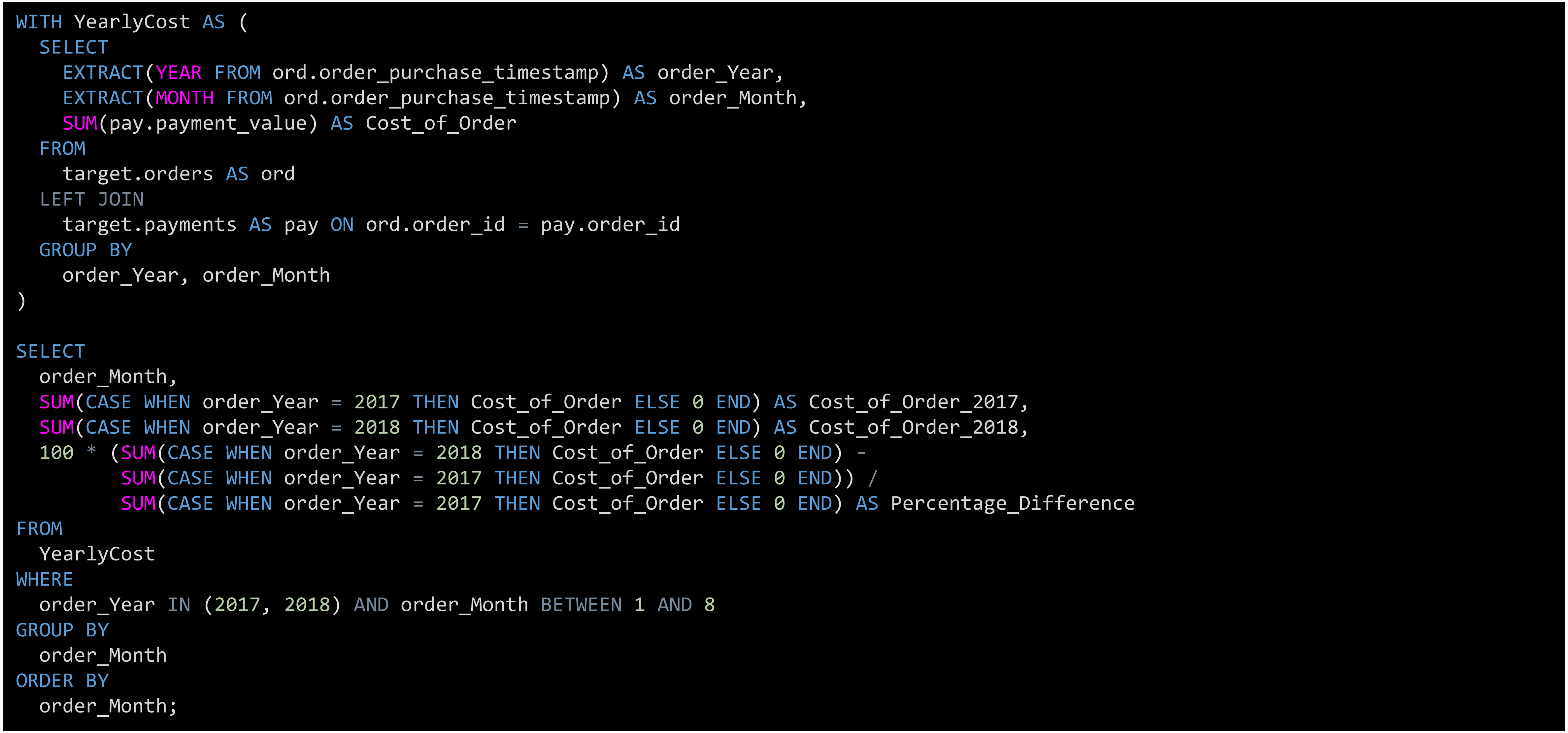
Solution

SQL Query 4.1: To answer the first question 2 Queries is structured “a” & “b”. Query A gives inference for how the parameter

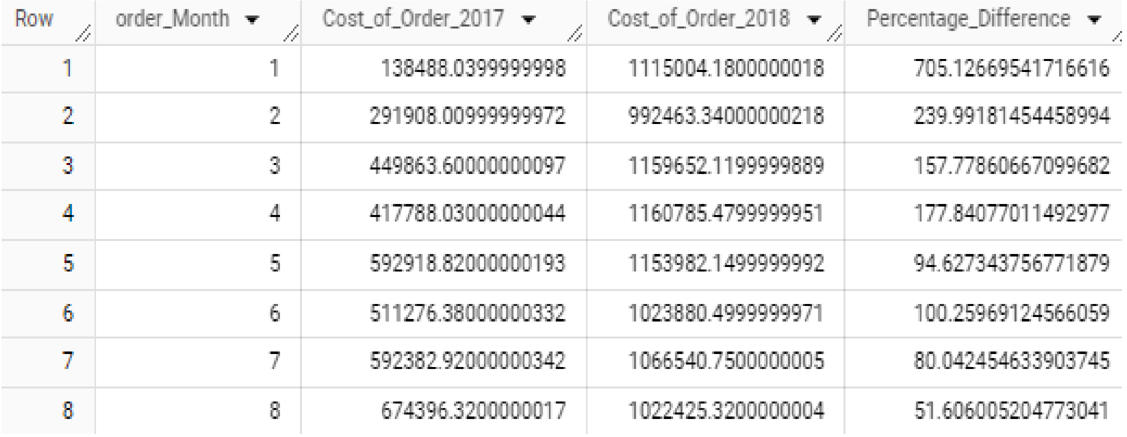
[% change] is visualised about how the economy of Brazil has performed over the year. Query B gives a boarder perspective of Query A

SQL Query 4.2 answers both question 2 and 3, aggregating the Total and Averages of Price and Freight Values State wise.

Query 4.1 a



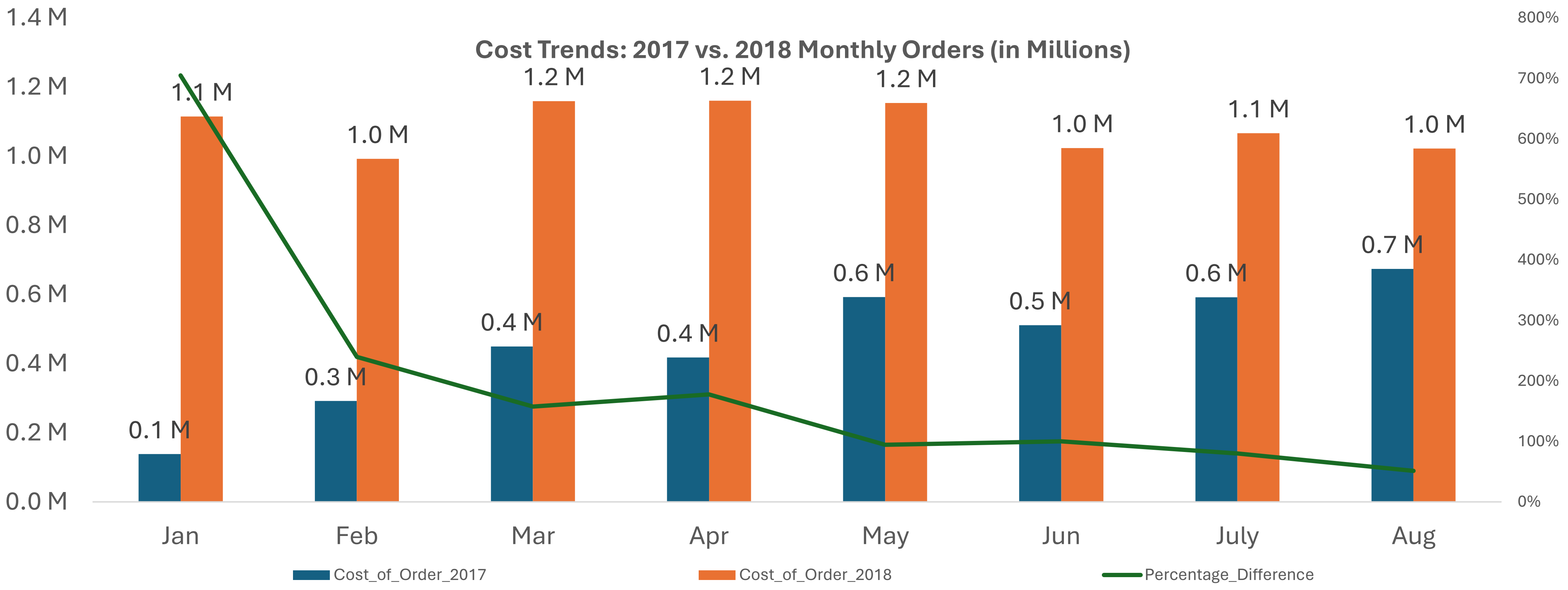
```sql
WITH YearlyCost AS (
  SELECT
    EXTRACT(YEAR FROM ord.order_purchase_timestamp) AS order_Year,
    EXTRACT(MONTH FROM ord.order_purchase_timestamp) AS order_Month,
    SUM(pay.payment_value) AS Cost_of_Order
  FROM
    target.orders AS ord
  LEFT JOIN
    target.payments AS pay ON ord.order_id = pay.order_id
  GROUP BY
    order_Year, order_Month
)

SELECT
  order_Month,
  SUM(CASE WHEN order_Year = 2017 THEN Cost_of_Order ELSE 0 END) AS Cost_of_Order_2017,
  SUM(CASE WHEN order_Year = 2018 THEN Cost_of_Order ELSE 0 END) AS Cost_of_Order_2018,
  100 * (SUM(CASE WHEN order_Year = 2018 THEN Cost_of_Order ELSE 0 END) -
         SUM(CASE WHEN order_Year = 2017 THEN Cost_of_Order ELSE 0 END)) /
         SUM(CASE WHEN order_Year = 2017 THEN Cost_of_Order ELSE 0 END) AS Percentage_Difference
FROM
  YearlyCost
WHERE
  order_Year IN (2017, 2018) AND order_Month BETWEEN 1 AND 8
GROUP BY
  order_Month
ORDER BY
  order_Month;
```



Row	order_Month	Cost_of_Order_2017	Cost_of_Order_2018	Percentage_Difference
1	1	138488.0399999998	1115004.1800000018	705.12669541716616
2	2	291908.00999999972	992463.34000000218	239.99181454458994
3	3	449863.60000000097	1159652.1199999889	157.77860667099682
4	4	417788.03000000044	1160785.4799999951	177.84077011492977
5	5	592918.82000000193	1153982.1499999992	94.627343756771879
6	6	511276.38000000332	1023880.4999999971	100.25969124566059
7	7	592382.92000000342	1066540.7500000005	80.042454633903745
8	8	674396.3200000017	1022425.3200000004	51.606005204773041

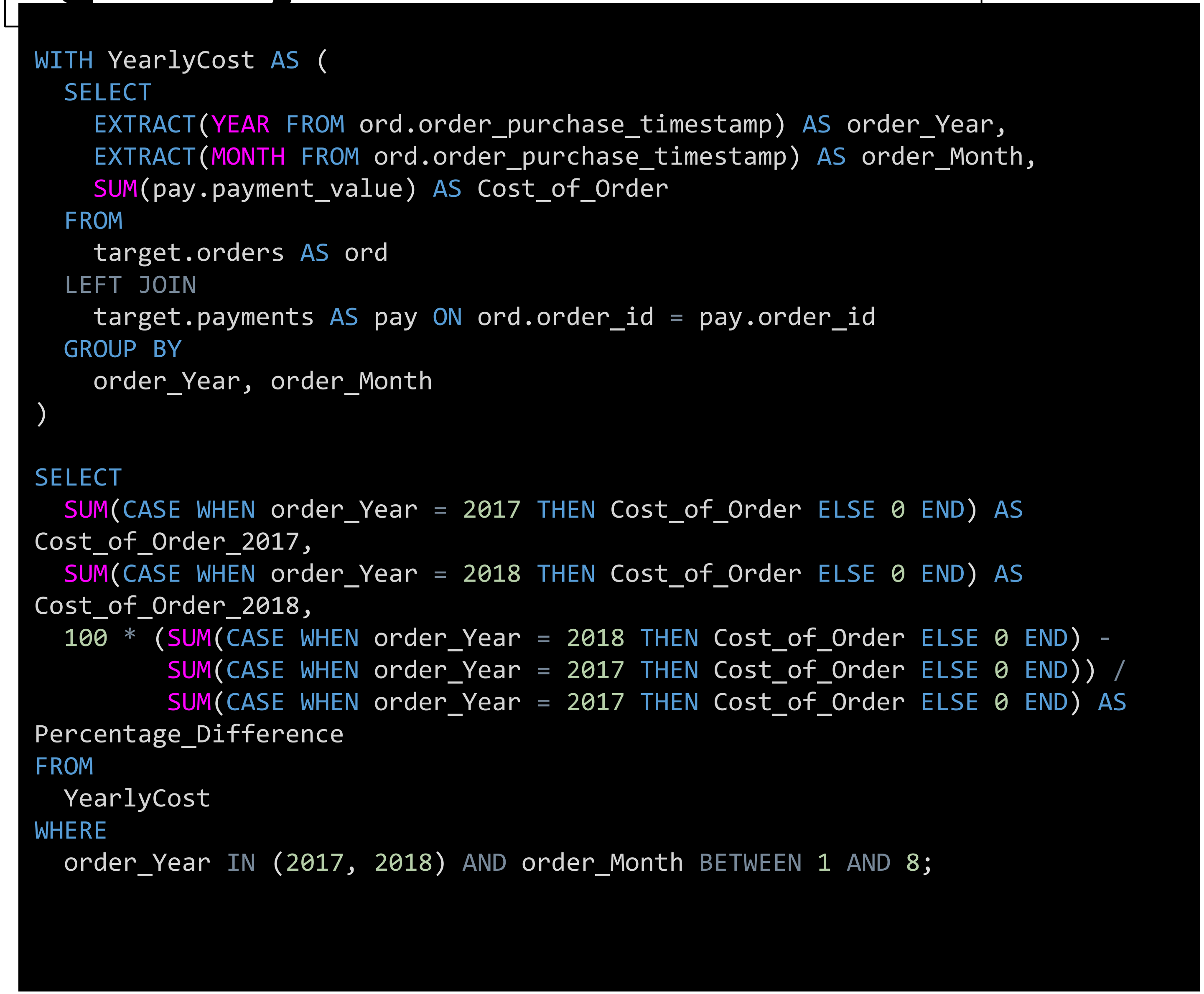
The cost of orders in January 2018 significantly increased by 705% compared to January 2017. This suggests a substantial spike in spending during this month.

The YOY percentage difference line has decreased from 700% to 50% in 8 Months Span rapidly, this suggest a significant reduction in the rate of growth. Market Saturation, or change in customer behaviour are some potential reasons for this decline.

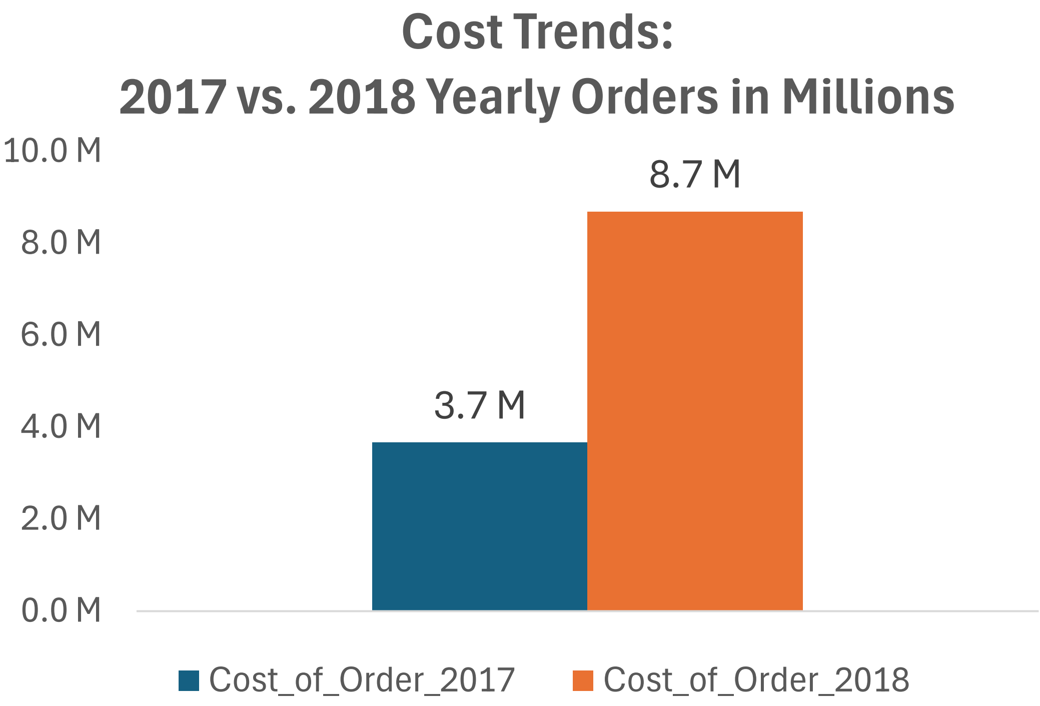


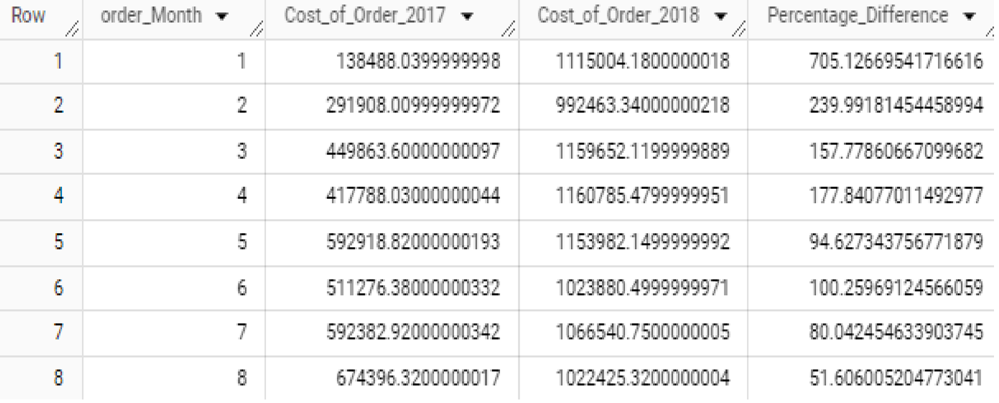
Cost Trends: 2017 vs. 2018 Monthly Orders (in Millions)

Query 4.1 b



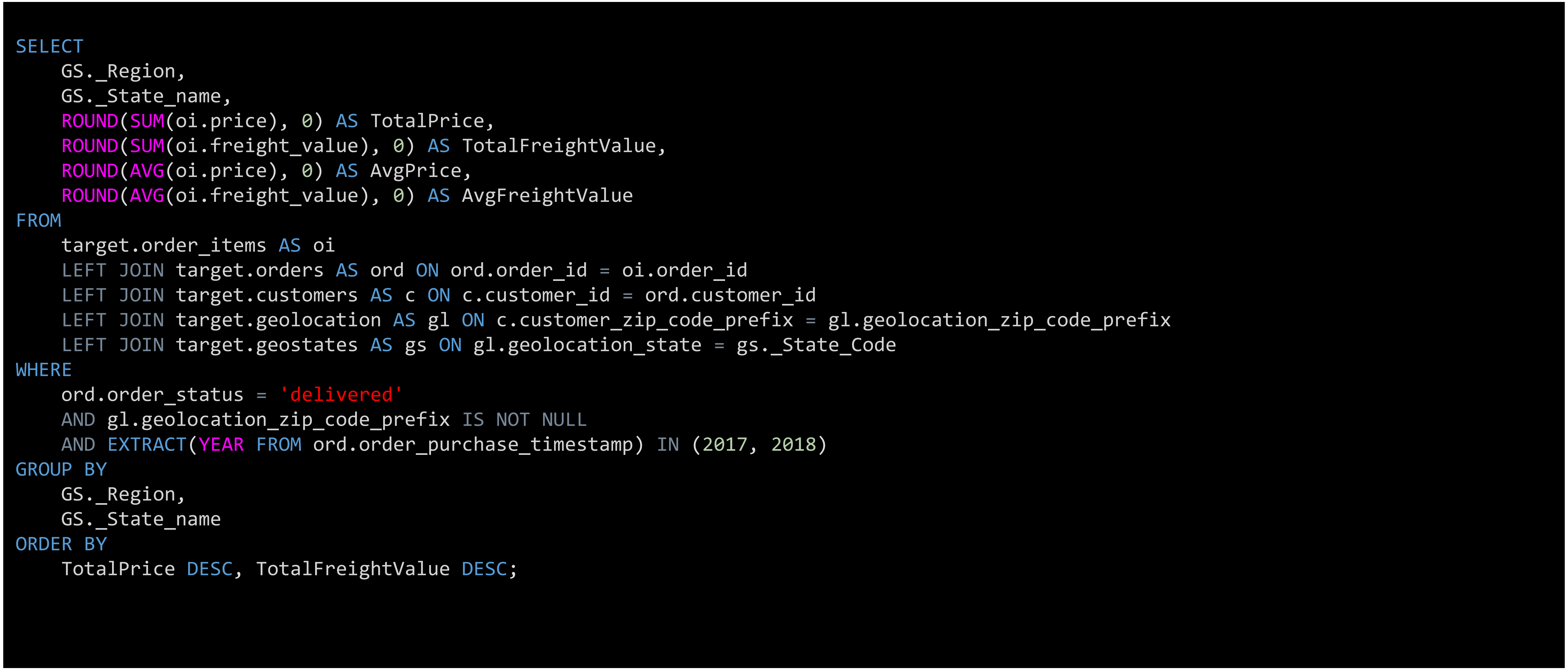
```sql
WITH YearlyCost AS (
  SELECT
    EXTRACT(YEAR FROM ord.order_purchase_timestamp) AS order_Year,
    EXTRACT(MONTH FROM ord.order_purchase_timestamp) AS order_Month,
    SUM(pay.payment_value) AS Cost_of_Order
  FROM
    target.orders AS ord
  LEFT JOIN
    target.payments AS pay ON ord.order_id = pay.order_id
  GROUP BY
    order_Year, order_Month
)

SELECT
  SUM(CASE WHEN order_Year = 2017 THEN Cost_of_Order ELSE 0 END) AS
Cost_of_Order_2017,
  SUM(CASE WHEN order_Year = 2018 THEN Cost_of_Order ELSE 0 END) AS
Cost_of_Order_2018,
  100 * (SUM(CASE WHEN order_Year = 2018 THEN Cost_of_Order ELSE 0 END) -
         SUM(CASE WHEN order_Year = 2017 THEN Cost_of_Order ELSE 0 END)) /
         SUM(CASE WHEN order_Year = 2017 THEN Cost_of_Order ELSE 0 END) AS
Percentage_Difference
FROM
  YearlyCost
WHERE
  order_Year IN (2017, 2018) AND order_Month BETWEEN 1 AND 8;
```





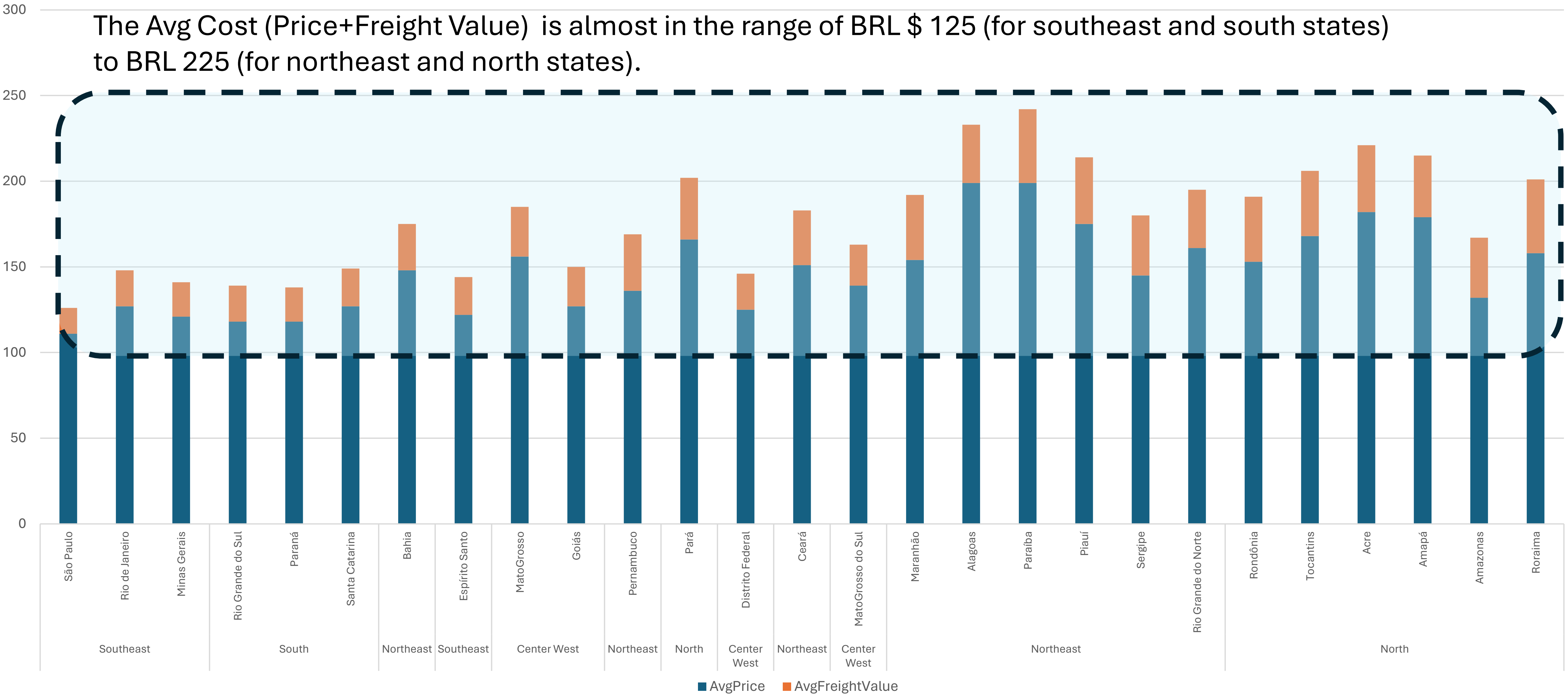
Row	order_Month	Cost_of_Order_2017	Cost_of_Order_2018	Percentage_Difference
1	1	138488.0399999998	1115004.1800000018	705.12669541716616
2	2	291908.00999999972	992463.34000000218	239.99181454458994
3	3	449863.60000000097	1159652.1199999889	157.77860667099682
4	4	417788.03000000044	1160785.4799999951	177.84077011492977
5	5	592918.82000000193	1153982.1499999992	94.627343756771879
6	6	511276.38000000332	1023880.4999999971	100.25969124566059
7	7	592382.92000000342	1066540.7500000005	80.042454633903745
8	8	674396.3200000017	1022425.3200000004	51.606005204773041

Query 4.2



```sql
SELECT
    GS._Region,
    GS._State_name,
    ROUND(SUM(oi.price), 0) AS TotalPrice,
    ROUND(SUM(oi.freight_value), 0) AS TotalFreightValue,
    ROUND(AVG(oi.price), 0) AS AvgPrice,
    ROUND(AVG(oi.freight_value), 0) AS AvgFreightValue
FROM
    target.order_items AS oi
    LEFT JOIN target.orders AS ord ON ord.order_id = oi.order_id
    LEFT JOIN target.customers AS c ON c.customer_id = ord.customer_id
    LEFT JOIN target.geolocation AS gl ON c.customer_zip_code_prefix = gl.geolocation_zip_code_prefix
    LEFT JOIN target.geostates AS gs ON gl.geolocation_state = gs._State_Code
WHERE
    ord.order_status = 'delivered'
    AND gl.geolocation_zip_code_prefix IS NOT NULL
    AND EXTRACT(YEAR FROM ord.order_purchase_timestamp) IN (2017, 2018)
GROUP BY
    GS._Region,
    GS._State_name
ORDER BY
    TotalPrice DESC, TotalFreightValue DESC;
```

Average Price of Order Price and Freight



The Avg Cost (Price+Freight Value) is almost in the range of BRL $ 125 (for southeast and south states) to BRL 225 (for northeast and north states).

Section 5: Analysis based on sales, freight and delivery time.

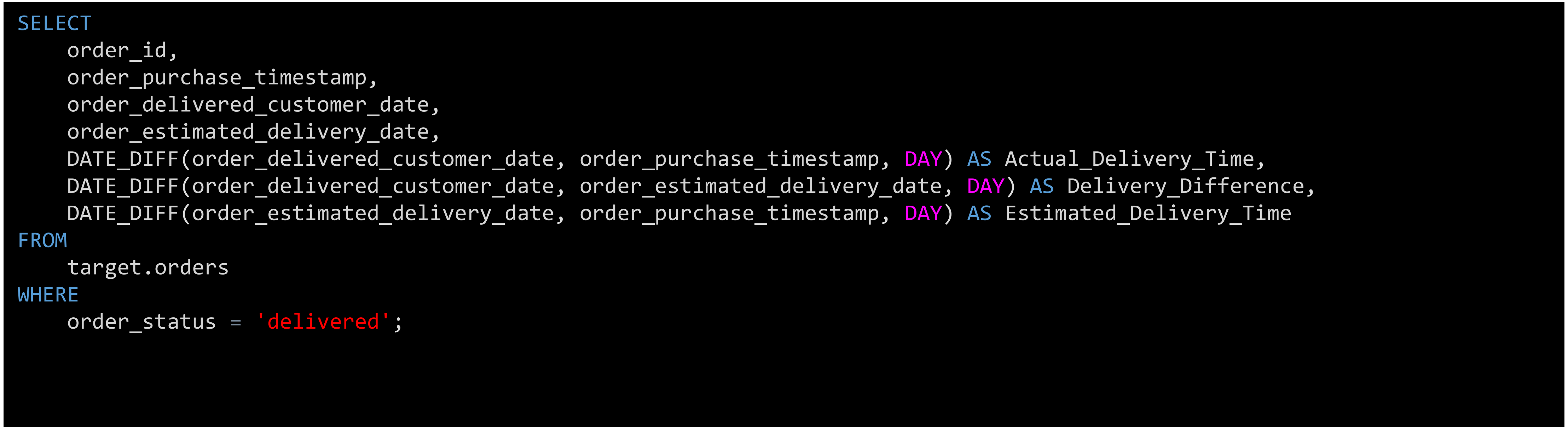
Question

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. [**Query 5.1**]

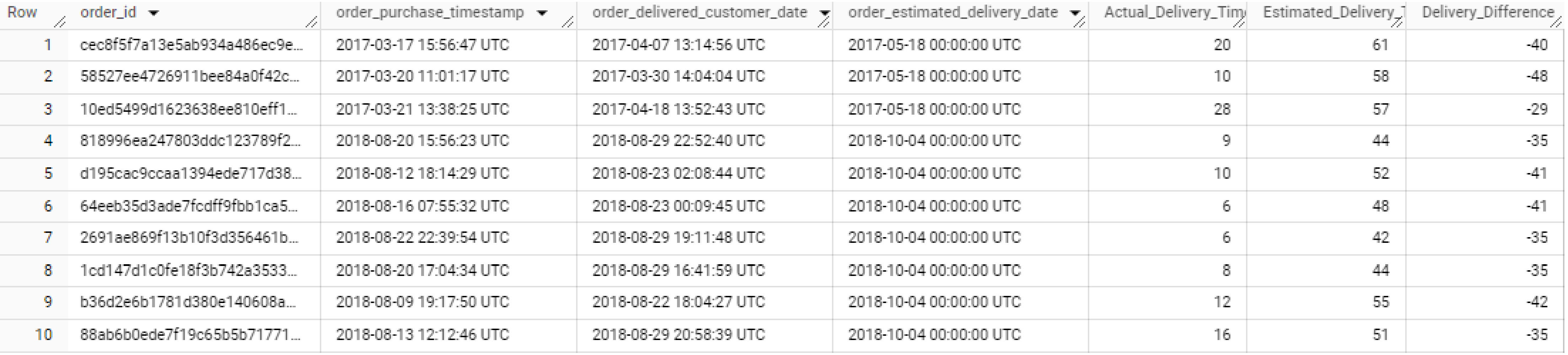
   You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

   1. **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
   2. **diff_estimated_delivery** = order_delivered_customer_date - order_estimated_delivery_date
2. Find out the top 5 states with the highest & lowest average freight value. [**Query 5.2 a,b**]
3. Find out the top 5 states with the highest & lowest average delivery time. [**Query 5.3 a,b**]
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.

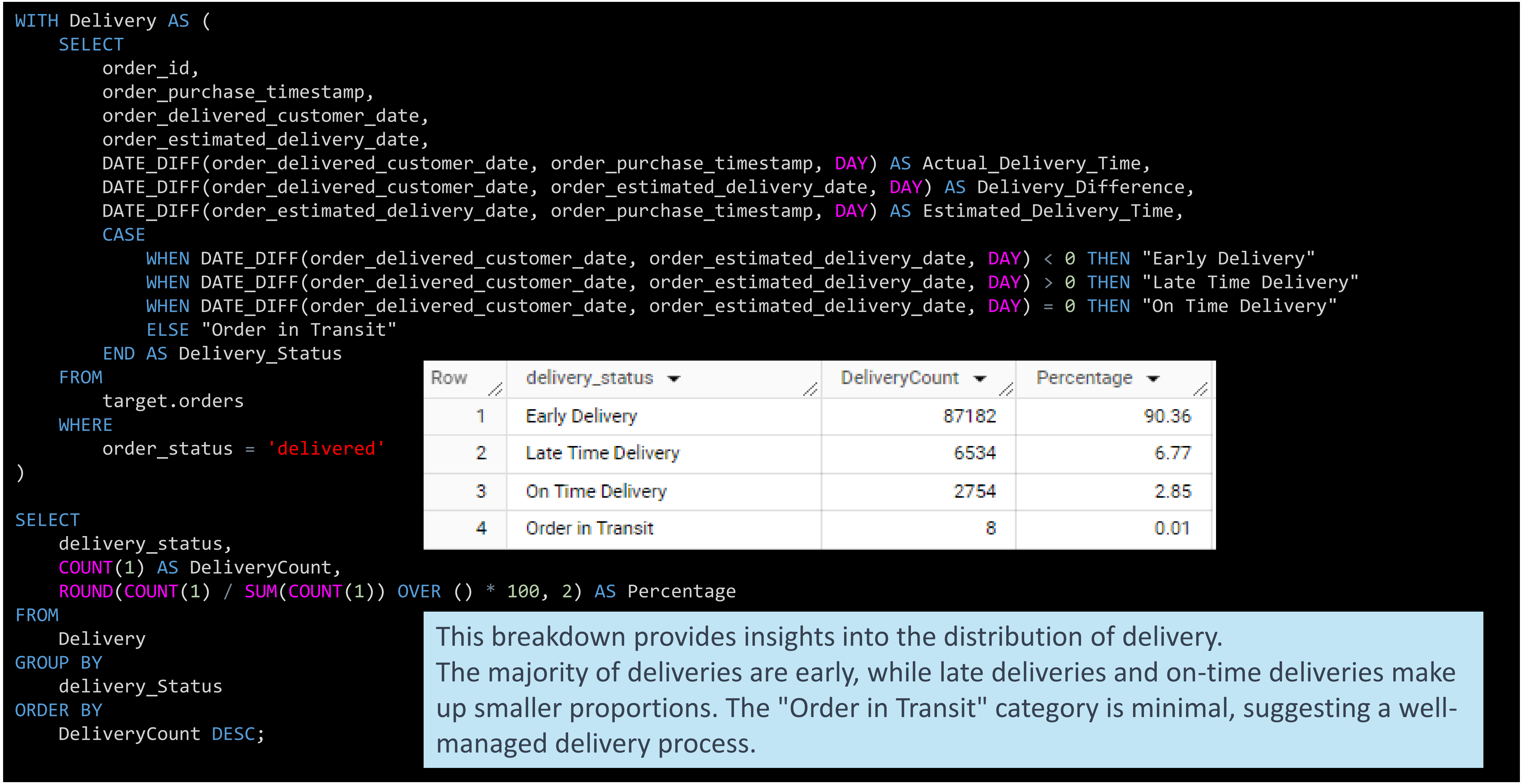
Query 5.1



```
SELECT
    order_id,
    order_purchase_timestamp,
    order_delivered_customer_date,
    order_estimated_delivery_date,
    DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) AS Actual_Delivery_Time,
    DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date, DAY) AS Delivery_Difference,
    DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, DAY) AS Estimated_Delivery_Time
FROM
    target.orders
WHERE
    order_status = 'delivered';
```



Row	order_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	Actual_Delivery_Tim	Estimated_Delivery_	Delivery_Difference
1	cec8f5f7a13e5ab934a486ec9e...	2017-03-17 15:56:47 UTC	2017-04-07 13:14:56 UTC	2017-05-18 00:00:00 UTC	20	61	-40
2	58527ee4726911bee84a0f42c...	2017-03-20 11:01:17 UTC	2017-03-30 14:04:04 UTC	2017-05-18 00:00:00 UTC	10	58	-48
3	10ed5499d1623638ee810eff1...	2017-03-21 13:38:25 UTC	2017-04-18 13:52:43 UTC	2017-05-18 00:00:00 UTC	28	57	-29
4	818996ea247803ddc123789f2...	2018-08-20 15:56:23 UTC	2018-08-29 22:52:40 UTC	2018-10-04 00:00:00 UTC	9	44	-35
5	d195cac9ccaa1394ede717d38...	2018-08-12 18:14:29 UTC	2018-08-23 02:08:44 UTC	2018-10-04 00:00:00 UTC	10	52	-41
6	64eeb35d3ade7fcdff9fbb1ca5...	2018-08-16 07:55:32 UTC	2018-08-23 00:09:45 UTC	2018-10-04 00:00:00 UTC	6	48	-41
7	2691ae869f13b10f3d356461b...	2018-08-22 22:39:54 UTC	2018-08-29 19:11:48 UTC	2018-10-04 00:00:00 UTC	6	42	-35
8	1cd147d1c0fe18f3b742a3533...	2018-08-20 17:04:34 UTC	2018-08-29 16:41:59 UTC	2018-10-04 00:00:00 UTC	8	44	-35
9	b36d2e6b1781d380e140608a...	2018-08-09 19:17:50 UTC	2018-08-22 18:04:27 UTC	2018-10-04 00:00:00 UTC	12	55	-42
10	88ab6b0ede7f19c65b5b71771...	2018-08-13 12:12:46 UTC	2018-08-29 20:58:39 UTC	2018-10-04 00:00:00 UTC	16	51	-35



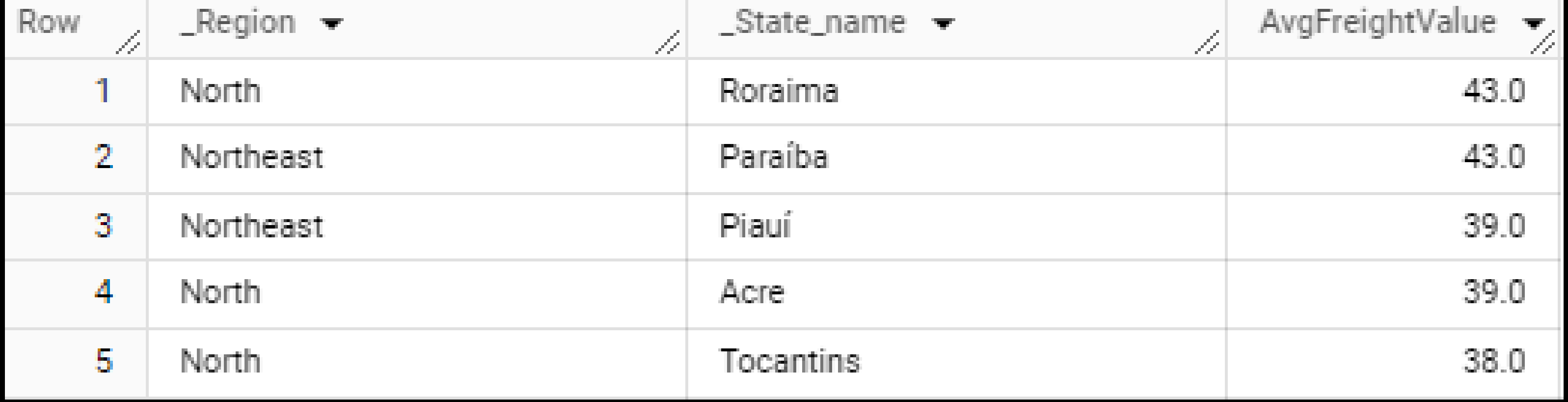
```sql
WITH Delivery AS (
    SELECT
        order_id,
        order_purchase_timestamp,
        order_delivered_customer_date,
        order_estimated_delivery_date,
        DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) AS Actual_Delivery_Time,
        DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date, DAY) AS Delivery_Difference,
        DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, DAY) AS Estimated_Delivery_Time,
        CASE
            WHEN DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date, DAY) < 0 THEN "Early Delivery"
            WHEN DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date, DAY) > 0 THEN "Late Time Delivery"
            WHEN DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date, DAY) = 0 THEN "On Time Delivery"
            ELSE "Order in Transit"
        END AS Delivery_Status
    FROM
        target.orders
    WHERE
        order_status = 'delivered'
)

SELECT
    delivery_status,
    COUNT(1) AS DeliveryCount,
    ROUND(COUNT(1) / SUM(COUNT(1)) OVER () * 100, 2) AS Percentage
FROM
    Delivery
GROUP BY
    delivery_Status
ORDER BY
    DeliveryCount DESC;
```

Row	delivery_status	DeliveryCount	Percentage
1	Early Delivery	87182	90.36
2	Late Time Delivery	6534	6.77
3	On Time Delivery	2754	2.85
4	Order in Transit	8	0.01

This breakdown provides insights into the distribution of delivery.
The majority of deliveries are early, while late deliveries and on-time deliveries make up smaller proportions. The "Order in Transit" category is minimal, suggesting a well-managed delivery process.

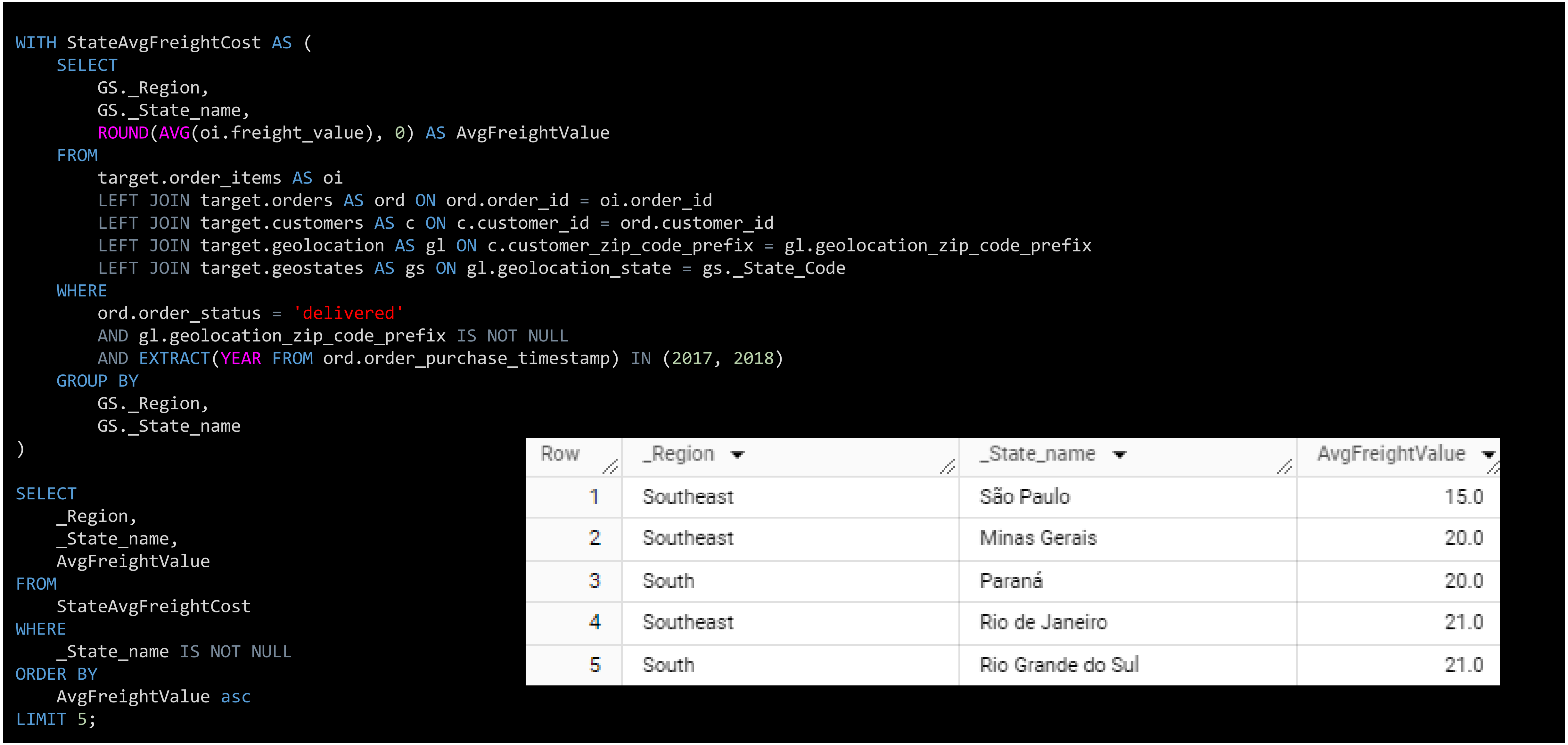
Query 5.2 a - Highest Avg. Freight Value

```
WITH StateAvgFreightCost AS (
    SELECT
        GS._Region,
        GS._State_name,
        ROUND(AVG(oi.freight_value), 0) AS AvgFreightValue
    FROM
        target.order_items AS oi
        LEFT JOIN target.orders AS ord ON ord.order_id = oi.order_id
        LEFT JOIN target.customers AS c ON c.customer_id = ord.customer_id
        LEFT JOIN target.geolocation AS gl ON c.customer_zip_code_prefix = gl.geolocation_zip_code_prefix
        LEFT JOIN target.geostates AS gs ON gl.geolocation_state = gs._State_Code
    WHERE
        ord.order_status = 'delivered'
        AND gl.geolocation_zip_code_prefix IS NOT NULL
        AND EXTRACT(YEAR FROM ord.order_purchase_timestamp) IN (2017, 2018)
    GROUP BY
        GS._Region,
        GS._State_name
)

SELECT
    _Region,
    _State_name,
    AvgFreightValue
FROM
    StateAvgFreightCost
WHERE
    _State_name IS NOT NULL
ORDER BY
    AvgFreightValue DESC
LIMIT 5;
```



Row	_Region	_State_name	AvgFreightValue
1	North	Roraima	43.0
2	Northeast	Paraíba	43.0
3	Northeast	Piauí	39.0
4	North	Acre	39.0
5	North	Tocantins	38.0

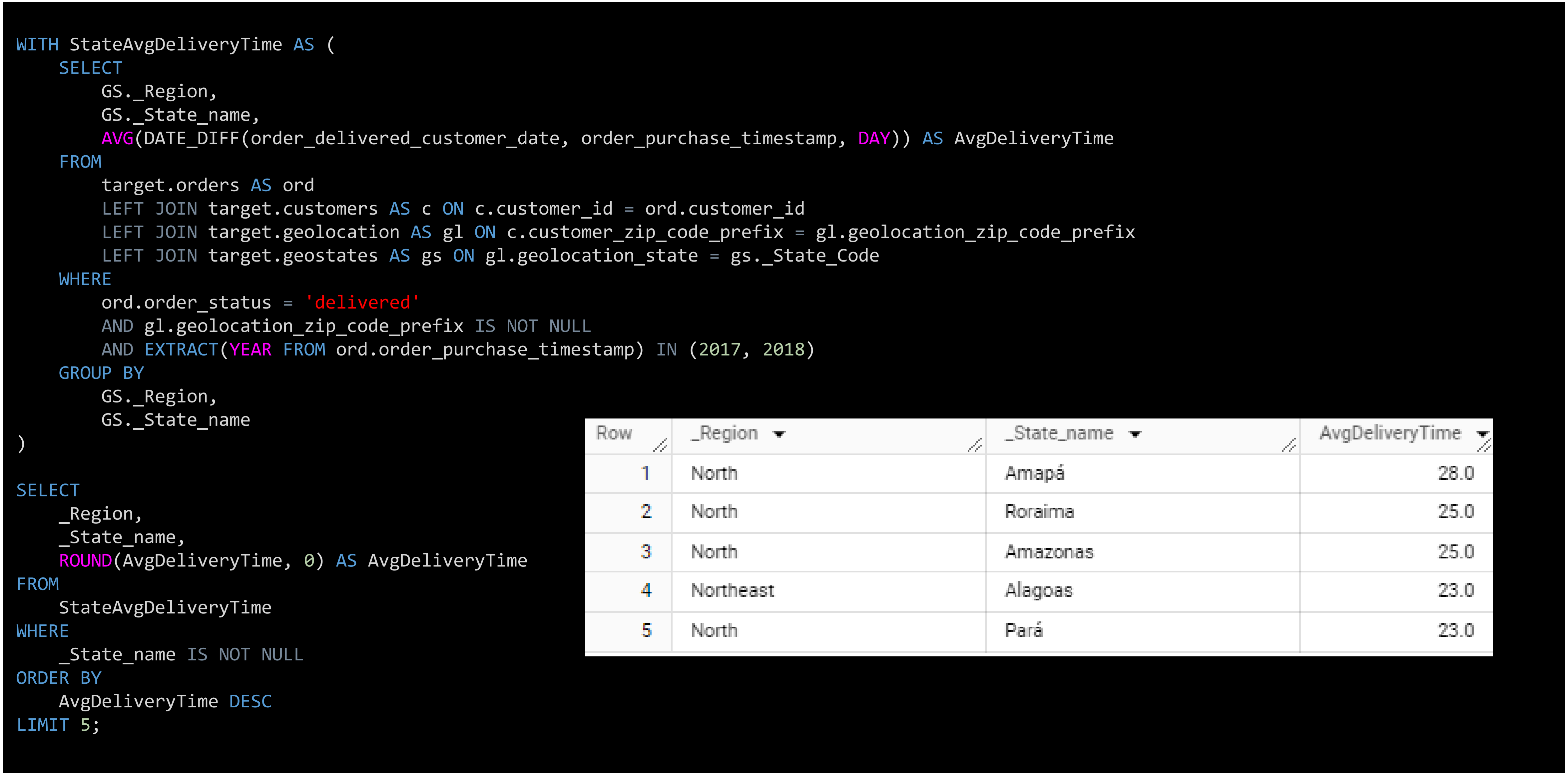
Query 5.2 b - Lowest Avg. Freight Value



```
WITH StateAvgFreightCost AS (
    SELECT
        GS._Region,
        GS._State_name,
        ROUND(AVG(oi.freight_value), 0) AS AvgFreightValue
    FROM
        target.order_items AS oi
        LEFT JOIN target.orders AS ord ON ord.order_id = oi.order_id
        LEFT JOIN target.customers AS c ON c.customer_id = ord.customer_id
        LEFT JOIN target.geolocation AS gl ON c.customer_zip_code_prefix = gl.geolocation_zip_code_prefix
        LEFT JOIN target.geostates AS gs ON gl.geolocation_state = gs._State_Code
    WHERE
        ord.order_status = 'delivered'
        AND gl.geolocation_zip_code_prefix IS NOT NULL
        AND EXTRACT(YEAR FROM ord.order_purchase_timestamp) IN (2017, 2018)
    GROUP BY
        GS._Region,
        GS._State_name
)

SELECT
    _Region,
    _State_name,
    AvgFreightValue
FROM
    StateAvgFreightCost
WHERE
    _State_name IS NOT NULL
ORDER BY
    AvgFreightValue asc
LIMIT 5;
```

Row	_Region	_State_name	AvgFreightValue
1	Southeast	São Paulo	15.0
2	Southeast	Minas Gerais	20.0
3	South	Paraná	20.0
4	Southeast	Rio de Janeiro	21.0
5	South	Rio Grande do Sul	21.0

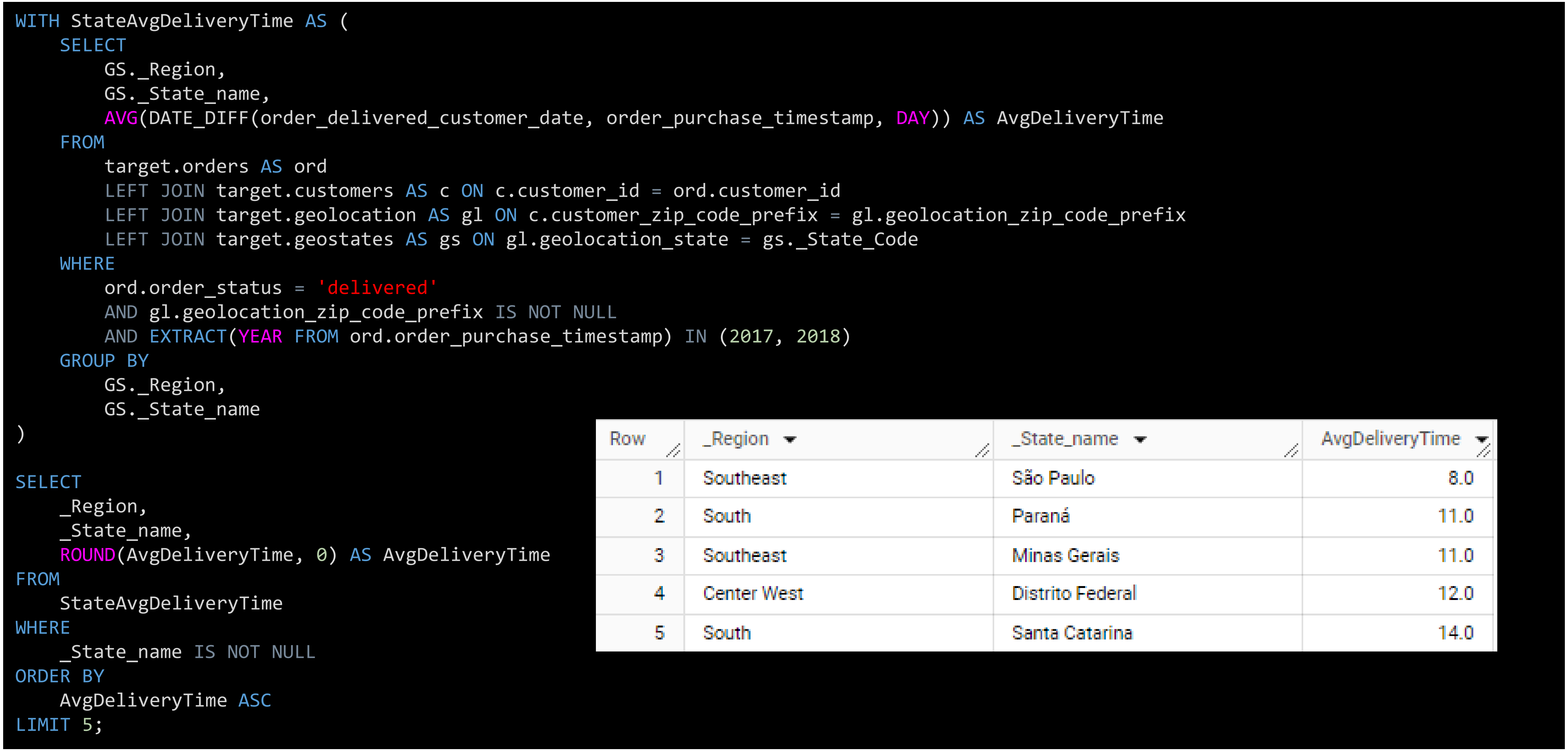
Query 5.3 a - Highest Avg. Delivery Time



```
WITH StateAvgDeliveryTime AS (
    SELECT
        GS._Region,
        GS._State_name,
        AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)) AS AvgDeliveryTime
    FROM
        target.orders AS ord
        LEFT JOIN target.customers AS c ON c.customer_id = ord.customer_id
        LEFT JOIN target.geolocation AS gl ON c.customer_zip_code_prefix = gl.geolocation_zip_code_prefix
        LEFT JOIN target.geostates AS gs ON gl.geolocation_state = gs._State_Code
    WHERE
        ord.order_status = 'delivered'
        AND gl.geolocation_zip_code_prefix IS NOT NULL
        AND EXTRACT(YEAR FROM ord.order_purchase_timestamp) IN (2017, 2018)
    GROUP BY
        GS._Region,
        GS._State_name
)

SELECT
    _Region,
    _State_name,
    ROUND(AvgDeliveryTime, 0) AS AvgDeliveryTime
FROM
    StateAvgDeliveryTime
WHERE
    _State_name IS NOT NULL
ORDER BY
    AvgDeliveryTime DESC
LIMIT 5;
```

Row	_Region	_State_name	AvgDeliveryTime
1	North	Amapá	28.0
2	North	Roraima	25.0
3	North	Amazonas	25.0
4	Northeast	Alagoas	23.0
5	North	Pará	23.0

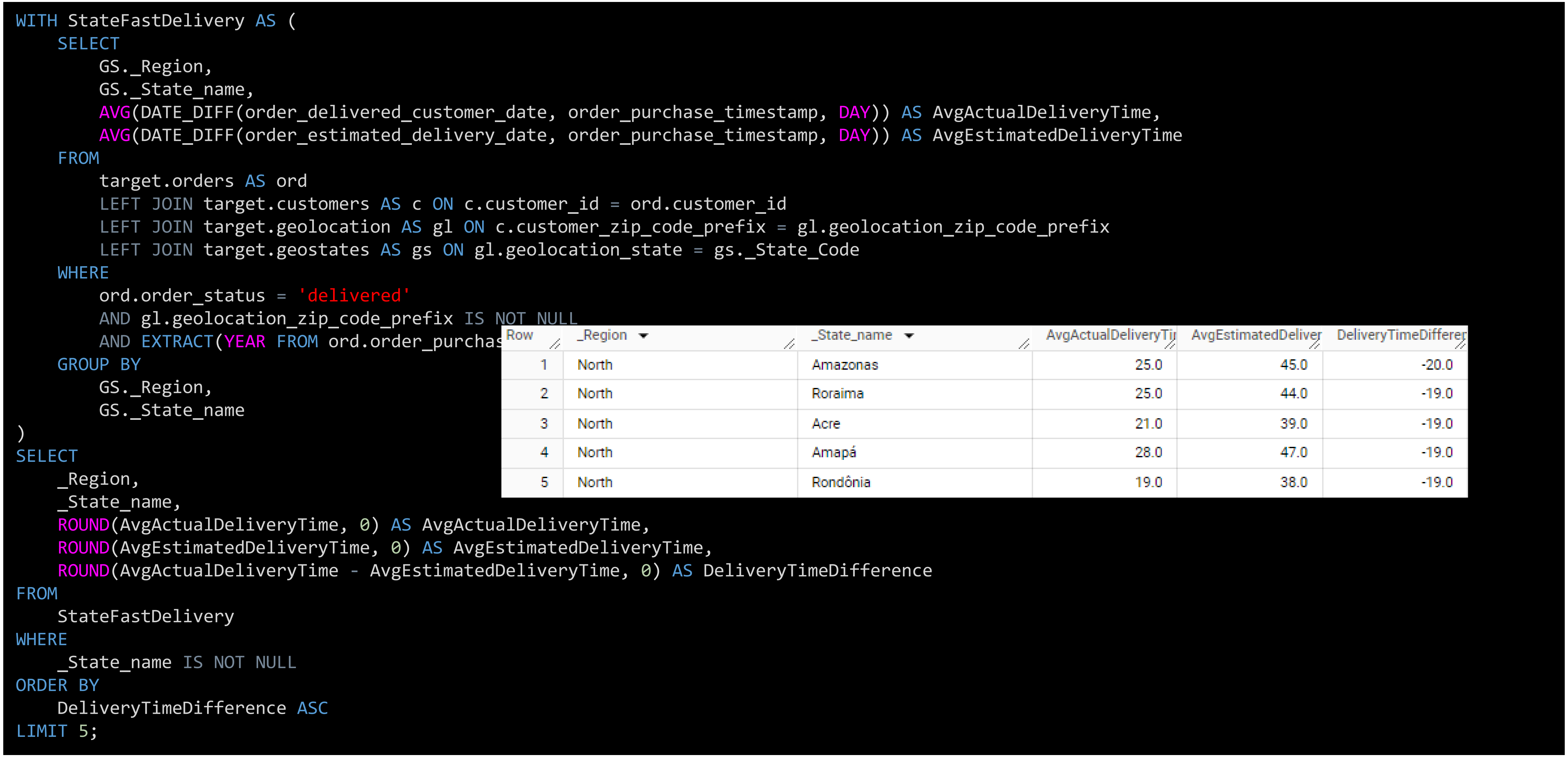
Query 5.3 b – Lowest Avg. Delivery Time



```
WITH StateAvgDeliveryTime AS (
    SELECT
        GS._Region,
        GS._State_name,
        AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)) AS AvgDeliveryTime
    FROM
        target.orders AS ord
        LEFT JOIN target.customers AS c ON c.customer_id = ord.customer_id
        LEFT JOIN target.geolocation AS gl ON c.customer_zip_code_prefix = gl.geolocation_zip_code_prefix
        LEFT JOIN target.geostates AS gs ON gl.geolocation_state = gs._State_Code
    WHERE
        ord.order_status = 'delivered'
        AND gl.geolocation_zip_code_prefix IS NOT NULL
        AND EXTRACT(YEAR FROM ord.order_purchase_timestamp) IN (2017, 2018)
    GROUP BY
        GS._Region,
        GS._State_name
)

SELECT
    _Region,
    _State_name,
    ROUND(AvgDeliveryTime, 0) AS AvgDeliveryTime
FROM
    StateAvgDeliveryTime
WHERE
    _State_name IS NOT NULL
ORDER BY
    AvgDeliveryTime ASC
LIMIT 5;
```

Row	_Region	_State_name	AvgDeliveryTime
1	Southeast	São Paulo	8.0
2	South	Paraná	11.0
3	Southeast	Minas Gerais	11.0
4	Center West	Distrito Federal	12.0
5	South	Santa Catarina	14.0

Query 5.4 – Actual Time < Estimated Time



```
WITH StateFastDelivery AS (
    SELECT
        GS._Region,
        GS._State_name,
        AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)) AS AvgActualDeliveryTime,
        AVG(DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, DAY)) AS AvgEstimatedDeliveryTime
    FROM
        target.orders AS ord
        LEFT JOIN target.customers AS c ON c.customer_id = ord.customer_id
        LEFT JOIN target.geolocation AS gl ON c.customer_zip_code_prefix = gl.geolocation_zip_code_prefix
        LEFT JOIN target.geostates AS gs ON gl.geolocation_state = gs._State_Code
    WHERE
        ord.order_status = 'delivered'
        AND gl.geolocation_zip_code_prefix IS NOT NULL
        AND EXTRACT(YEAR FROM ord.order_purchas
    GROUP BY
        GS._Region,
        GS._State_name
)
SELECT
    _Region,
    _State_name,
    ROUND(AvgActualDeliveryTime, 0) AS AvgActualDeliveryTime,
    ROUND(AvgEstimatedDeliveryTime, 0) AS AvgEstimatedDeliveryTime,
    ROUND(AvgActualDeliveryTime - AvgEstimatedDeliveryTime, 0) AS DeliveryTimeDifference
FROM
    StateFastDelivery
WHERE
    _State_name IS NOT NULL
ORDER BY
    DeliveryTimeDifference ASC
LIMIT 5;
```

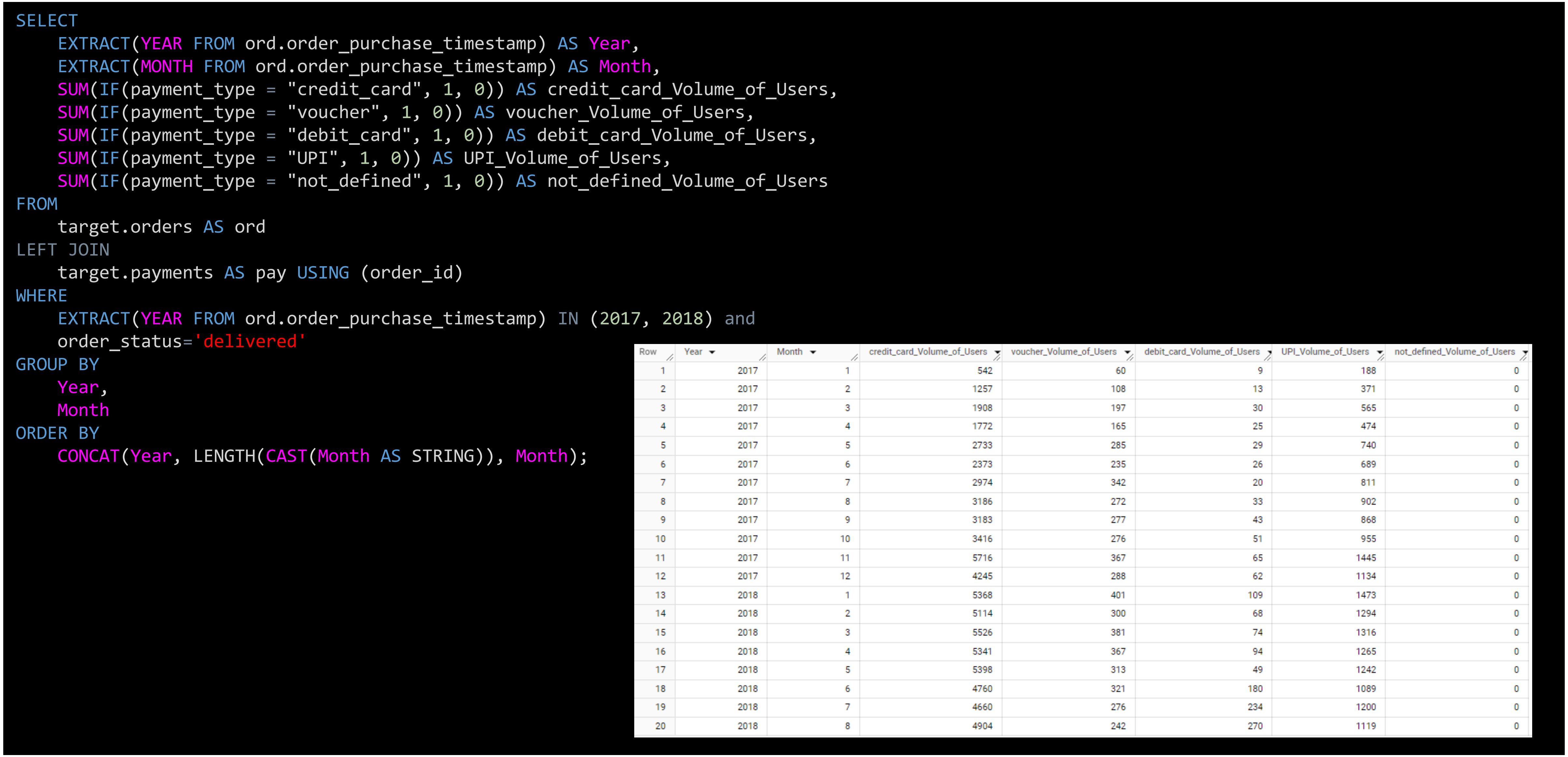
Row	_Region	_State_name	AvgActualDeliveryTi	AvgEstimatedDeliver	DeliveryTimeDifferer
1	North	Amazonas	25.0	45.0	-20.0
2	North	Roraima	25.0	44.0	-19.0
3	North	Acre	21.0	39.0	-19.0
4	North	Amapá	28.0	47.0	-19.0
5	North	Rondônia	19.0	38.0	-19.0

Section 6: Analysis based on the payments

Question

1. Find the month on month no. of orders placed using different payment types. [**Query 6.1**]
2. Find the no. of orders placed on the basis of the payment installments that have been paid. [**Query 6.2**]

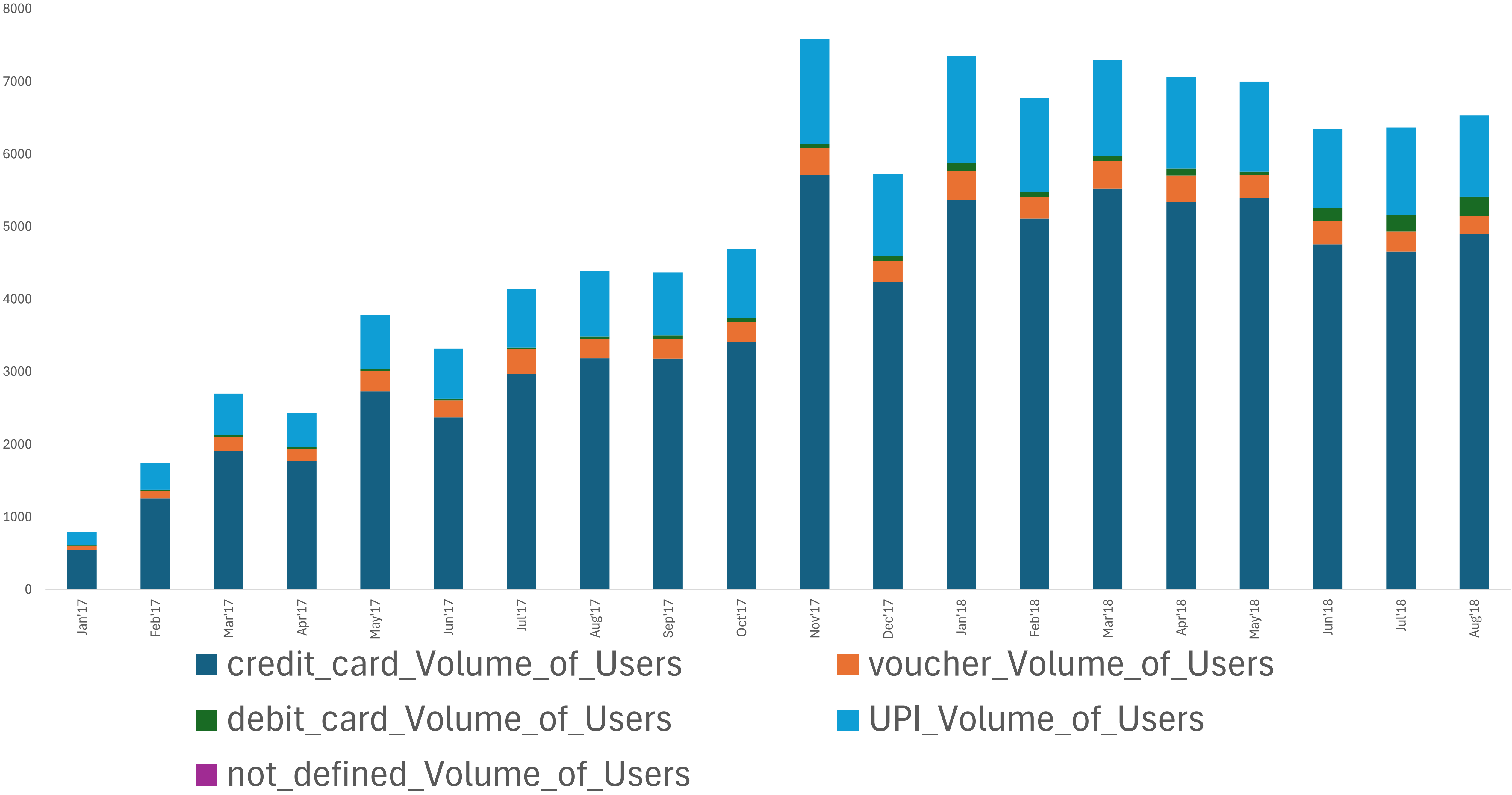
Query 6.1 – Payment Types



```sql
SELECT
    EXTRACT(YEAR FROM ord.order_purchase_timestamp) AS Year,
    EXTRACT(MONTH FROM ord.order_purchase_timestamp) AS Month,
    SUM(IF(payment_type = "credit_card", 1, 0)) AS credit_card_Volume_of_Users,
    SUM(IF(payment_type = "voucher", 1, 0)) AS voucher_Volume_of_Users,
    SUM(IF(payment_type = "debit_card", 1, 0)) AS debit_card_Volume_of_Users,
    SUM(IF(payment_type = "UPI", 1, 0)) AS UPI_Volume_of_Users,
    SUM(IF(payment_type = "not_defined", 1, 0)) AS not_defined_Volume_of_Users
FROM
    target.orders AS ord
LEFT JOIN
    target.payments AS pay USING (order_id)
WHERE
    EXTRACT(YEAR FROM ord.order_purchase_timestamp) IN (2017, 2018) and
    order_status='delivered'
GROUP BY
    Year,
    Month
ORDER BY
    CONCAT(Year, LENGTH(CAST(Month AS STRING)), Month);
```

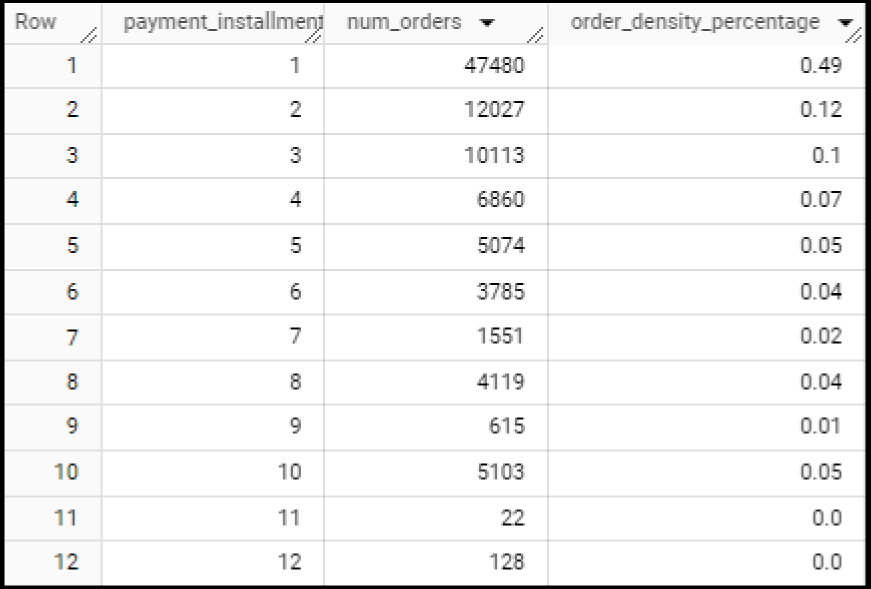
Row	Year	Month	credit_card_Volume_of_Users	voucher_Volume_of_Users	debit_card_Volume_of_Users	UPI_Volume_of_Users	not_defined_Volume_of_Users
1	2017	1	542	60	9	188	0
2	2017	2	1257	108	13	371	0
3	2017	3	1908	197	30	565	0
4	2017	4	1772	165	25	474	0
5	2017	5	2733	285	29	740	0
6	2017	6	2373	235	26	689	0
7	2017	7	2974	342	20	811	0
8	2017	8	3186	272	33	902	0
9	2017	9	3183	277	43	868	0
10	2017	10	3416	276	51	955	0
11	2017	11	5716	367	65	1445	0
12	2017	12	4245	288	62	1134	0
13	2018	1	5368	401	109	1473	0
14	2018	2	5114	300	68	1294	0
15	2018	3	5526	381	74	1316	0
16	2018	4	5341	367	94	1265	0
17	2018	5	5398	313	49	1242	0
18	2018	6	4760	321	180	1089	0
19	2018	7	4660	276	234	1200	0
20	2018	8	4904	242	270	1119	0

Volume of orders placed in different payment methods



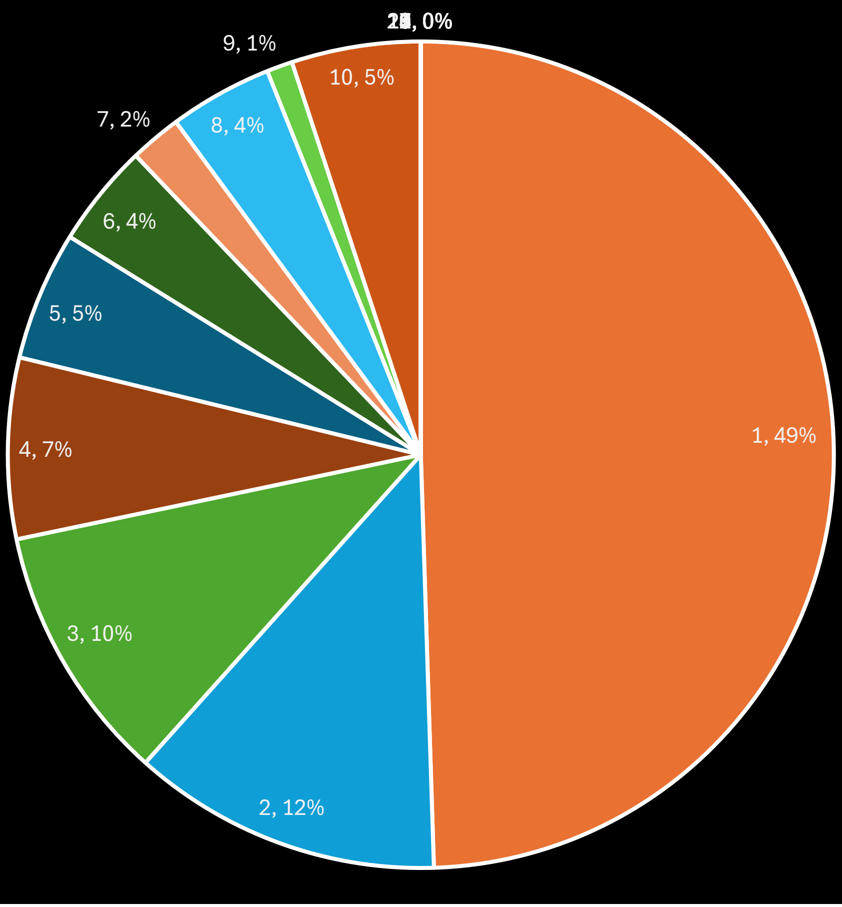
Query 6.2 – Payment Instalments

```
SELECT
    payment_installments,
    COUNT(DISTINCT ord.order_id) AS num_orders,
    ROUND((COUNT(DISTINCT ord.order_id) / SUM(COUNT(DISTINCT ord.order_id)) OVER ()), 2) AS order_density_percentage
FROM
    target.orders AS ord
LEFT JOIN
    target.payments AS pay ON ord.order_id = pay.order_id
WHERE
    EXTRACT(YEAR FROM ord.order_purchase_timestamp) IN (2017, 2018) and
    order_status='delivered' and
    payment_installments>0

GROUP BY
    payment_installments
ORDER BY
    payment_installments;
```



Row	payment_installment	num_orders ▾	order_density_percentage ▾
1	1	47480	0.49
2	2	12027	0.12
3	3	10113	0.1
4	4	6860	0.07
5	5	5074	0.05
6	6	3785	0.04
7	7	1551	0.02
8	8	4119	0.04
9	9	615	0.01
10	10	5103	0.05
11	11	22	0.0
12	12	128	0.0

Payment Installment Types Distibution

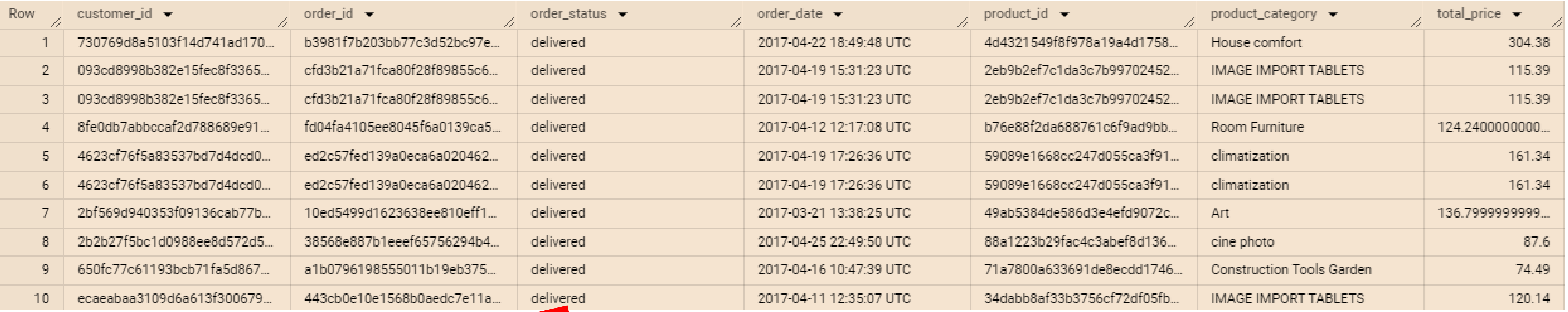


Section 7: Actionable Insights & Recommendations

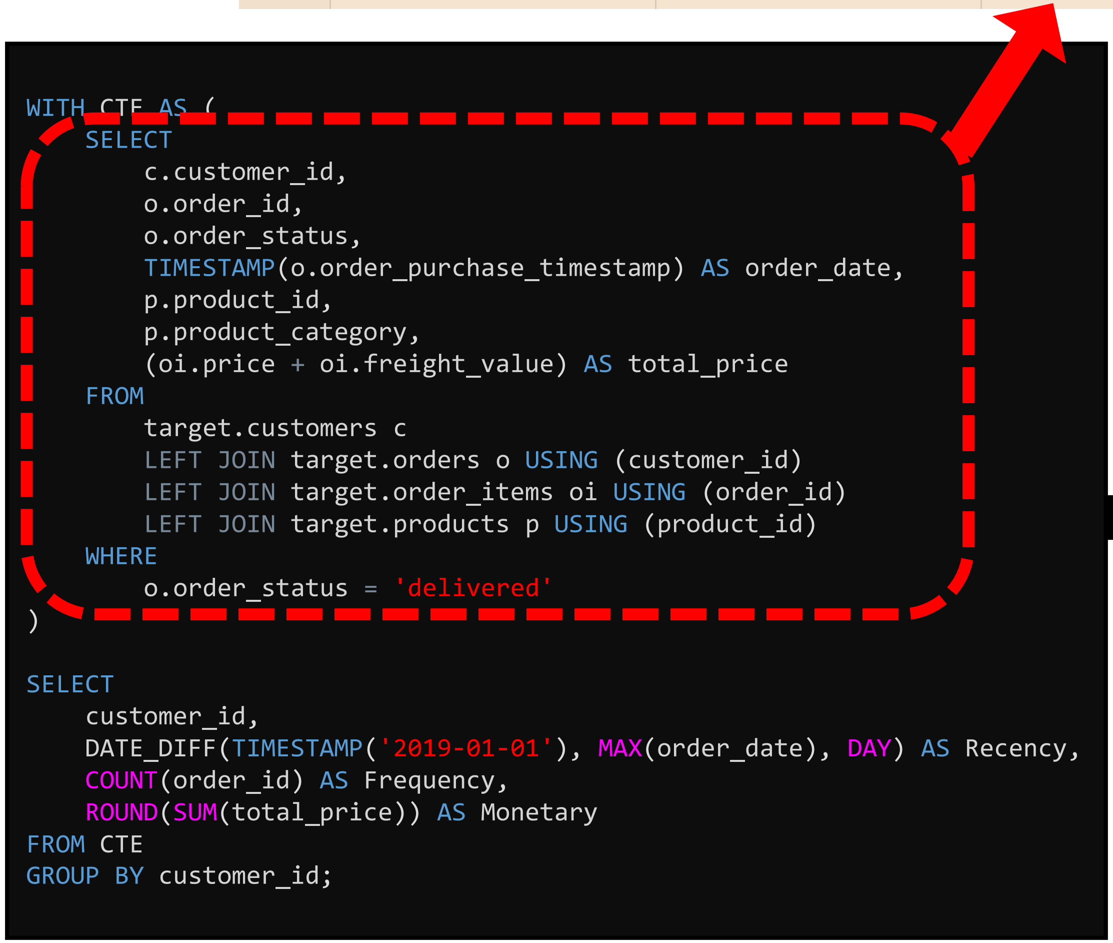
RFM Analysis

- RFM analysis is a customer segmentation technique used in marketing to categorize customers based on their purchasing behavior.
- RFM stands for Recency, Frequency, and Monetary Value. These three factors help businesses understand and identify their most valuable customers.

1. **Recency (R):**
   This measures how recently a customer has made a purchase. It's based on the principle that customers who have made a purchase more recently are likely to be more engaged and responsive.
2. **Frequency (F):**
   This measures how often a customer makes a purchase. Customers who make frequent purchases are often more loyal and contribute more to the business's revenue.
3. **Monetary Value (M):**
   This measures the total monetary value of a customer's purchases. It represents the overall value a customer brings to the business.

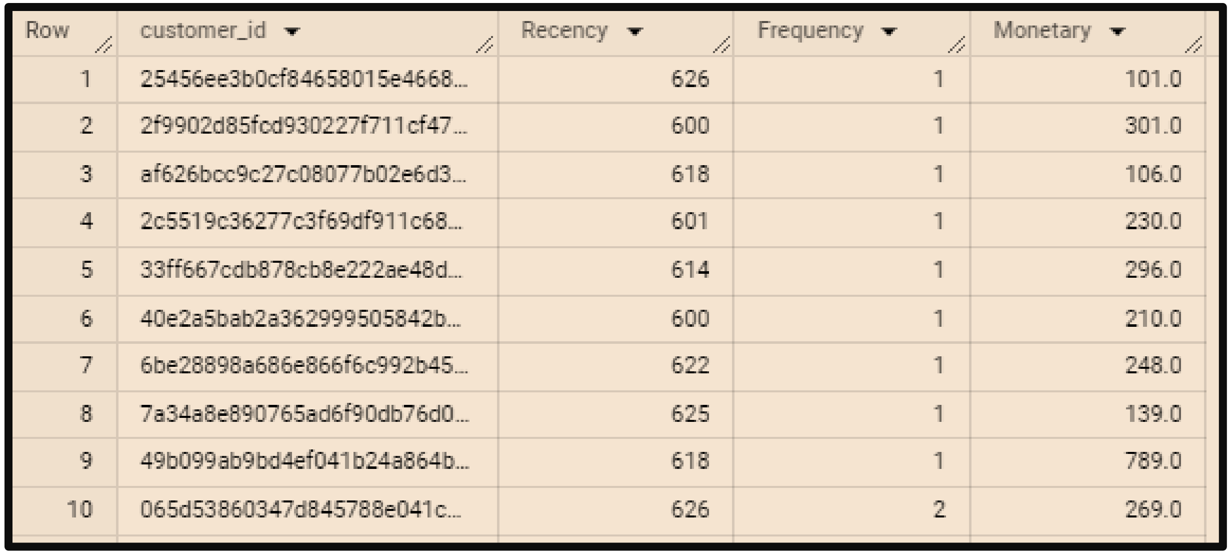


Row	customer_id	order_id	order_status	order_date	product_id	product_category	total_price
1	730769d8a5103f14d741ad170...	b3981f7b203bb77c3d52bc97e...	delivered	2017-04-22 18:49:48 UTC	4d4321549f8f978a19a4d1758...	House comfort	304.38
2	093cd8998b382e15fec8f3365...	cfd3b21a71fca80f28f89855c6...	delivered	2017-04-19 15:31:23 UTC	2eb9b2ef7c1da3c7b99702452...	IMAGE IMPORT TABLETS	115.39
3	093cd8998b382e15fec8f3365...	cfd3b21a71fca80f28f89855c6...	delivered	2017-04-19 15:31:23 UTC	2eb9b2ef7c1da3c7b99702452...	IMAGE IMPORT TABLETS	115.39
4	8fe0db7abbccaf2d788689e91...	fd04fa4105ee8045f6a0139ca5...	delivered	2017-04-12 12:17:08 UTC	b76e88f2da688761c6f9ad9bb...	Room Furniture	124.2400000000...
5	4623cf76f5a83537bd7d4dcd0...	ed2c57fed139a0eca6a020462...	delivered	2017-04-19 17:26:36 UTC	59089e1668cc247d055ca3f91...	climatization	161.34
6	4623cf76f5a83537bd7d4dcd0...	ed2c57fed139a0eca6a020462...	delivered	2017-04-19 17:26:36 UTC	59089e1668cc247d055ca3f91...	climatization	161.34
7	2bf569d940353f09136cab77b...	10ed5499d1623638ee810eff1...	delivered	2017-03-21 13:38:25 UTC	49ab5384de586d3e4efd9072c...	Art	136.7999999999...
8	2b2b27f5bc1d0988ee8d572d5...	38568e887b1eeef65756294b4...	delivered	2017-04-25 22:49:50 UTC	88a1223b29fac4c3abef8d136...	cine photo	87.6
9	650fc77c61193bcb71fa5d867...	a1b0796198555011b19eb375...	delivered	2017-04-16 10:47:39 UTC	71a7800a633691de8ecdd1746...	Construction Tools Garden	74.49
10	ecaeabaa3109d6a613f300679...	443cb0e10e1568b0aedc7e11a...	delivered	2017-04-11 12:35:07 UTC	34dabb8af33b3756cf72df05fb...	IMAGE IMPORT TABLETS	120.14

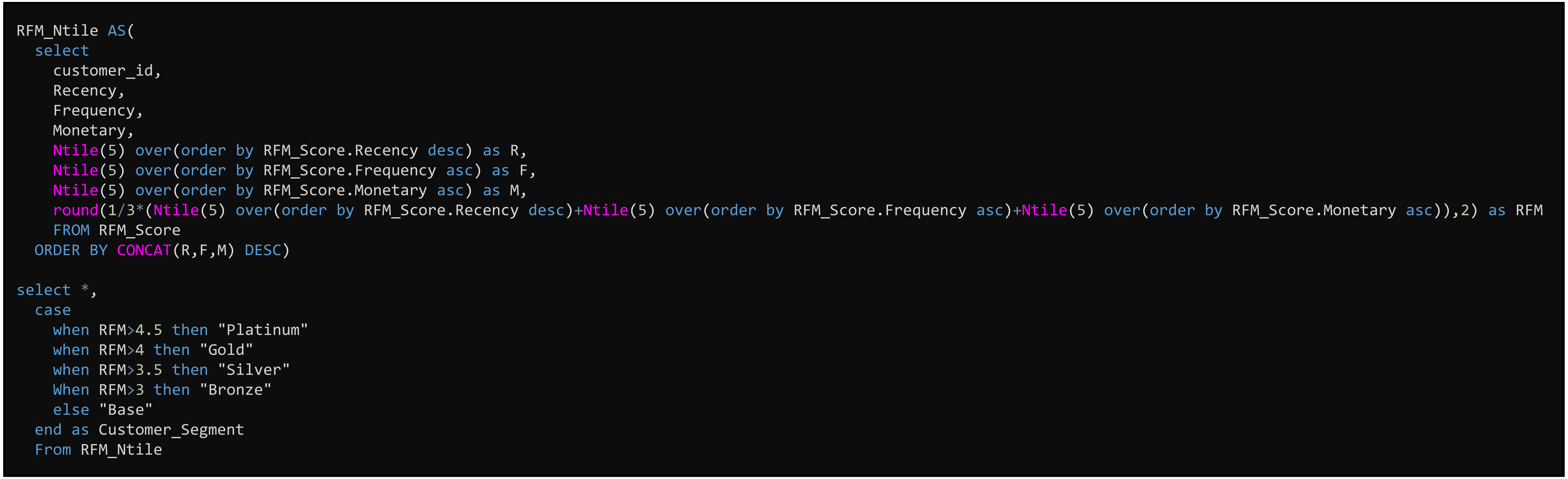


```
WITH CTE AS (
    SELECT
        c.customer_id,
        o.order_id,
        o.order_status,
        TIMESTAMP(o.order_purchase_timestamp) AS order_date,
        p.product_id,
        p.product_category,
        (oi.price + oi.freight_value) AS total_price
    FROM
        target.customers c
        LEFT JOIN target.orders o USING (customer_id)
        LEFT JOIN target.order_items oi USING (order_id)
        LEFT JOIN target.products p USING (product_id)
    WHERE
        o.order_status = 'delivered'
)

SELECT
    customer_id,
    DATE_DIFF(TIMESTAMP('2019-01-01'), MAX(order_date), DAY) AS Recency,
    COUNT(order_id) AS Frequency,
    ROUND(SUM(total_price)) AS Monetary
FROM CTE
GROUP BY customer_id;
```

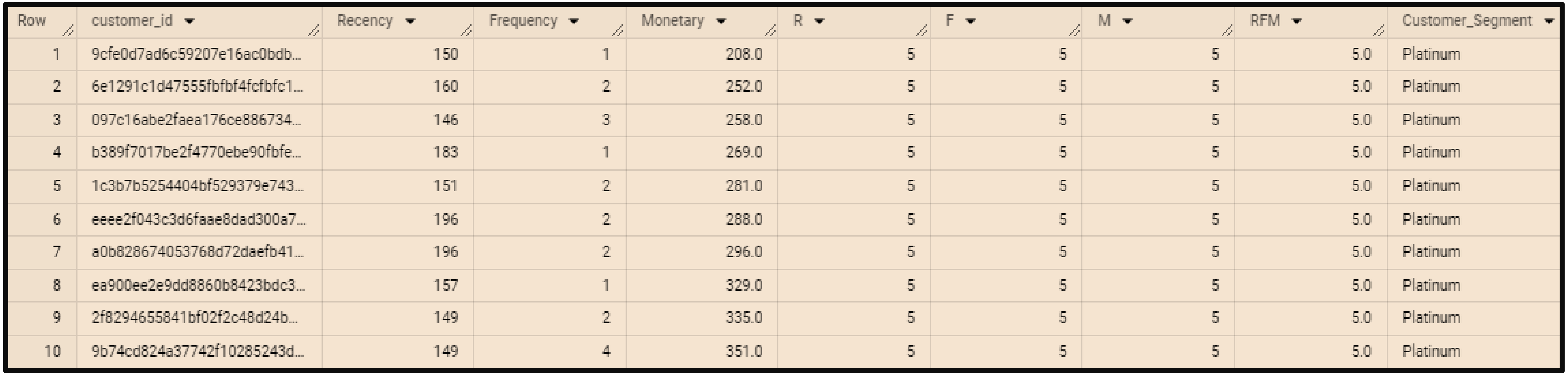
Storing the result as “RFM_Score”



Row	customer_id	Recency	Frequency	Monetary
1	25456ee3b0cf84658015e4668...	626	1	101.0
2	2f9902d85fcd930227f711cf47...	600	1	301.0
3	af626bcc9c27c08077b02e6d3...	618	1	106.0
4	2c5519c36277c3f69df911c68...	601	1	230.0
5	33ff667cdb878cb8e222ae48d...	614	1	296.0
6	40e2a5bab2a362999505842b...	600	1	210.0
7	6be28898a686e866f6c992b45...	622	1	248.0
8	7a34a8e890765ad6f90db76d0...	625	1	139.0
9	49b099ab9bd4ef041b24a864b...	618	1	789.0
10	065d53860347d845788e041c...	626	2	269.0

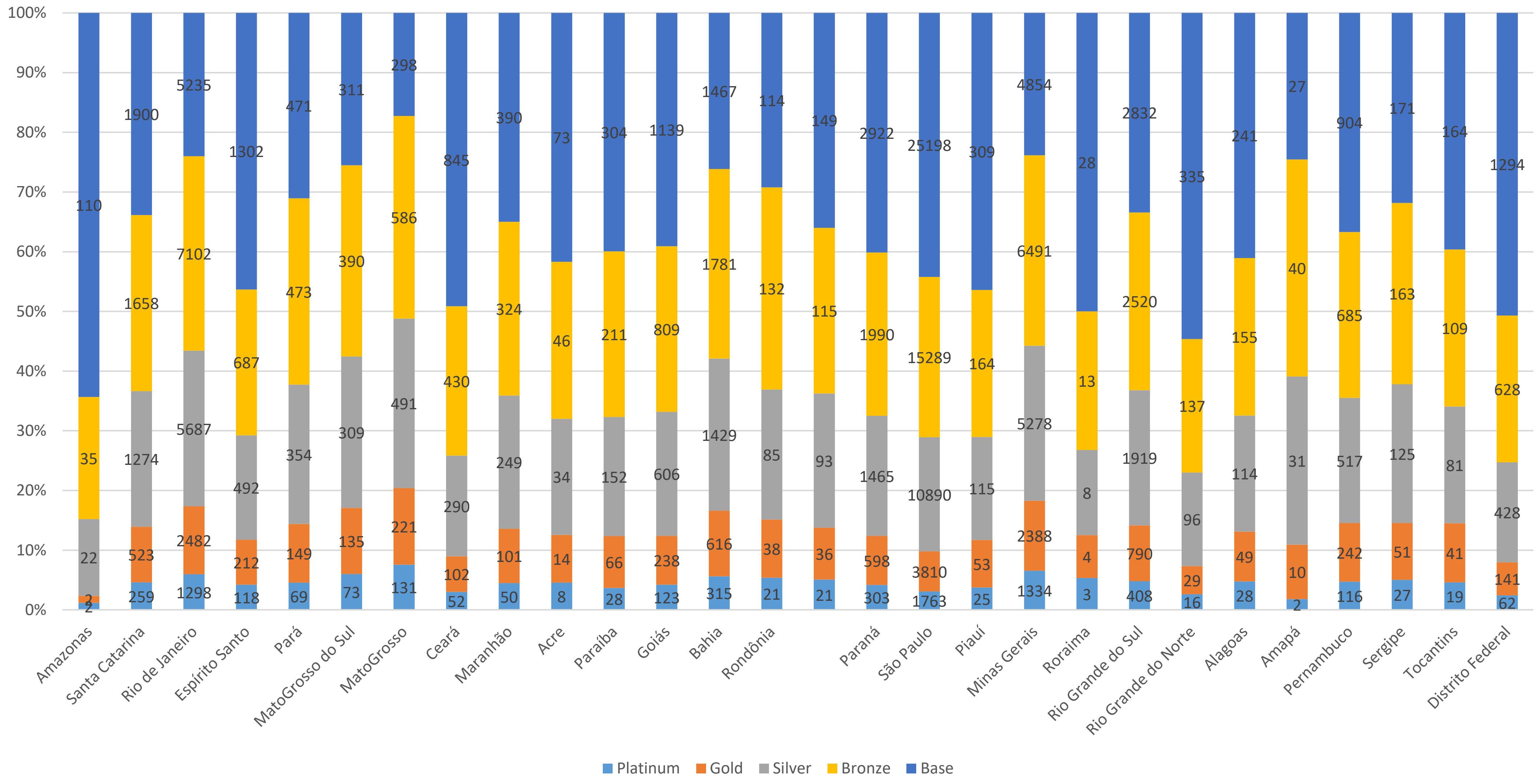


```
RFM_Ntile AS(
  select
    customer_id,
    Recency,
    Frequency,
    Monetary,
    Ntile(5) over(order by RFM_Score.Recency desc) as R,
    Ntile(5) over(order by RFM_Score.Frequency asc) as F,
    Ntile(5) over(order by RFM_Score.Monetary asc) as M,
    round(1/3*(Ntile(5) over(order by RFM_Score.Recency desc)+Ntile(5) over(order by RFM_Score.Frequency asc)+Ntile(5) over(order by RFM_Score.Monetary asc)),2) as RFM
    FROM RFM_Score
  ORDER BY CONCAT(R,F,M) DESC)

select *,
  case
    when RFM>4.5 then "Platinum"
    when RFM>4 then "Gold"
    when RFM>3.5 then "Silver"
    When RFM>3 then "Bronze"
    else "Base"
  end as Customer_Segment
  From RFM_Ntile
```



Row	customer_id	Recency	Frequency	Monetary	R	F	M	RFM	Customer_Segment
1	9cfe0d7ad6c59207e16ac0bdb...	150	1	208.0	5	5	5	5.0	Platinum
2	6e1291c1d47555fbfbf4fcfbfc1...	160	2	252.0	5	5	5	5.0	Platinum
3	097c16abe2faea176ce886734...	146	3	258.0	5	5	5	5.0	Platinum
4	b389f7017be2f4770ebe90fbfe...	183	1	269.0	5	5	5	5.0	Platinum
5	1c3b7b5254404bf529379e743...	151	2	281.0	5	5	5	5.0	Platinum
6	eeee2f043c3d6faae8dad300a7...	196	2	288.0	5	5	5	5.0	Platinum
7	a0b828674053768d72daefb41...	196	2	296.0	5	5	5	5.0	Platinum
8	ea900ee2e9dd8860b8423bdc3...	157	1	329.0	5	5	5	5.0	Platinum
9	2f8294655841bf02f2c48d24b...	149	2	335.0	5	5	5	5.0	Platinum
10	9b74cd824a37742f10285243d...	149	4	351.0	5	5	5	5.0	Platinum

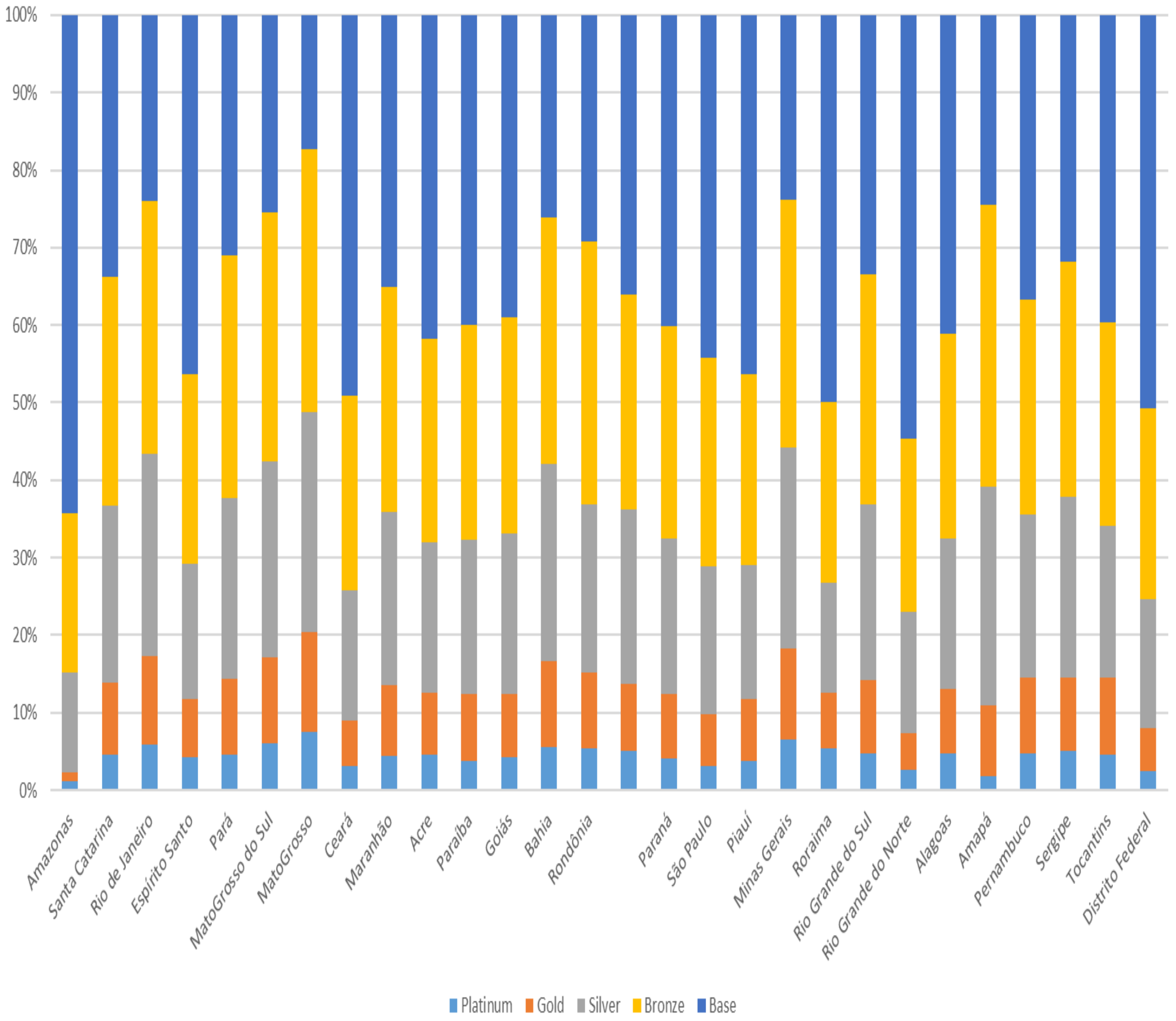
RFM Analysis And Customer Segmentation



Strategizing the Opportunity For Market Caputure

1. **Retain Platinum and Gold Customers:**
    1. Implement loyalty programs with exclusive benefits for Platinum and Gold customers (As they Being the minoring across all states).
    2. Offer personalized discounts, early access to new products, or free shipping to incentivize repeat purchases.

2. **Convert Silver and Bronze Customers:**
    1. Identify Silver and Bronze customers who are close to moving up in their RFM segments.
    2. Implement targeted marketing campaigns to encourage more frequent purchases, higher order values, or engagement with your brand.



Cont..
Strategizing the Opportunity For Market Capture

3. Re-engage Base Customers:

1. Identify Base customers who haven't made a purchase recently.
2. Launch reactivation campaigns with special offers, personalized recommendations, or reminders about the value your products/services bring.

4. Personalized Marketing:

1. Leverage the insights from RFM analysis to create personalized marketing messages.
2. Send targeted emails or promotions based on each customer's specific RFM segment.

5. Customer Segmentation:

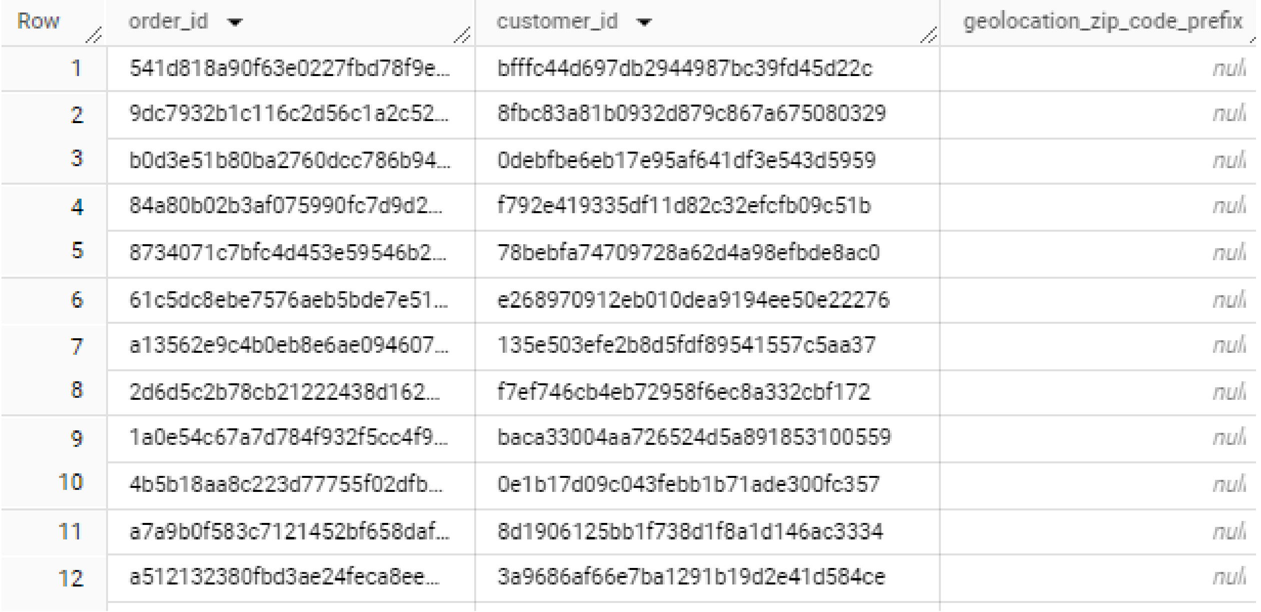
1. Further analyze customer behavior to identify patterns within each RFM segment.
2. Create sub-segments based on additional factors such as product preferences, channel preferences, or interaction history.

*****

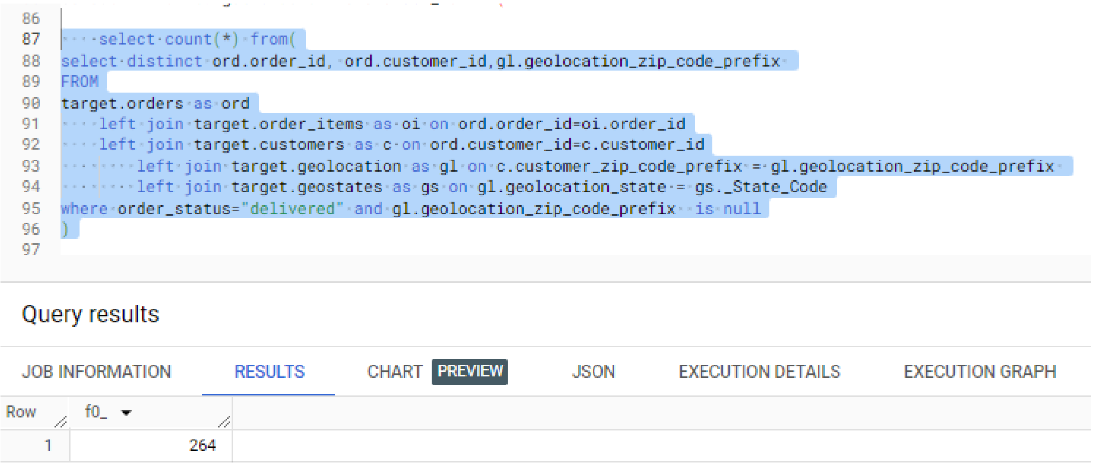
Data Discrepancy – No ZIP CODE for 264 Records

```
select distinct ord.order_id, ord.customer_id,gl.geolocation_zip_code_prefix
FROM
target.orders as ord
    left join target.order_items as oi on ord.order_id=oi.order_id
    left join target.customers as c on ord.customer_id=c.customer_id
        left join target.geolocation as gl on c.customer_zip_code_prefix = gl.geolocation_zip_code_prefix
        left join target.geostates as gs on gl.geolocation_state = gs._State_Code
where order_status="delivered" and gl.geolocation_zip_code_prefix  is null
```

In many of my queries I have included a filter condition to remove the record where zip codes is null. The reason behind this was, During the EDA it was found that the orders having ‘ delivered’ as status should ideally have address, but 264 records had null.



Row	order_id	customer_id	geolocation_zip_code_prefix
1	541d818a90f63e0227fbd78f9e...	bfffc44d697db2944987bc39fd45d22c	nul
2	9dc7932b1c116c2d56c1a2c52...	8fbc83a81b0932d879c867a675080329	nul
3	b0d3e51b80ba2760dcc786b94...	0debfbe6eb17e95af641df3e543d5959	nul
4	84a80b02b3af075990fc7d9d2...	f792e419335df11d82c32efcfb09c51b	nul
5	8734071c7bfc4d453e59546b2...	78bebfa74709728a62d4a98efbde8ac0	nul
6	61c5dc8ebe7576aeb5bde7e51...	e268970912eb010dea9194ee50e22276	nul
7	a13562e9c4b0eb8e6ae094607...	135e503efe2b8d5fdf89541557c5aa37	nul
8	2d6d5c2b78cb21222438d162...	f7ef746cb4eb72958f6ec8a332cbf172	nul
9	1a0e54c67a7d784f932f5cc4f9...	baca33004aa726524d5a891853100559	nul
10	4b5b18aa8c223d77755f02dfb...	0e1b17d09c043febb1b71ade300fc357	nul
11	a7a9b0f583c7121452bf658daf...	8d1906125bb1f738d1f8a1d146ac3334	nul
12	a512132380fbd3ae24feca8ee...	3a9686af66e7ba1291b19d2e41d584ce	nul

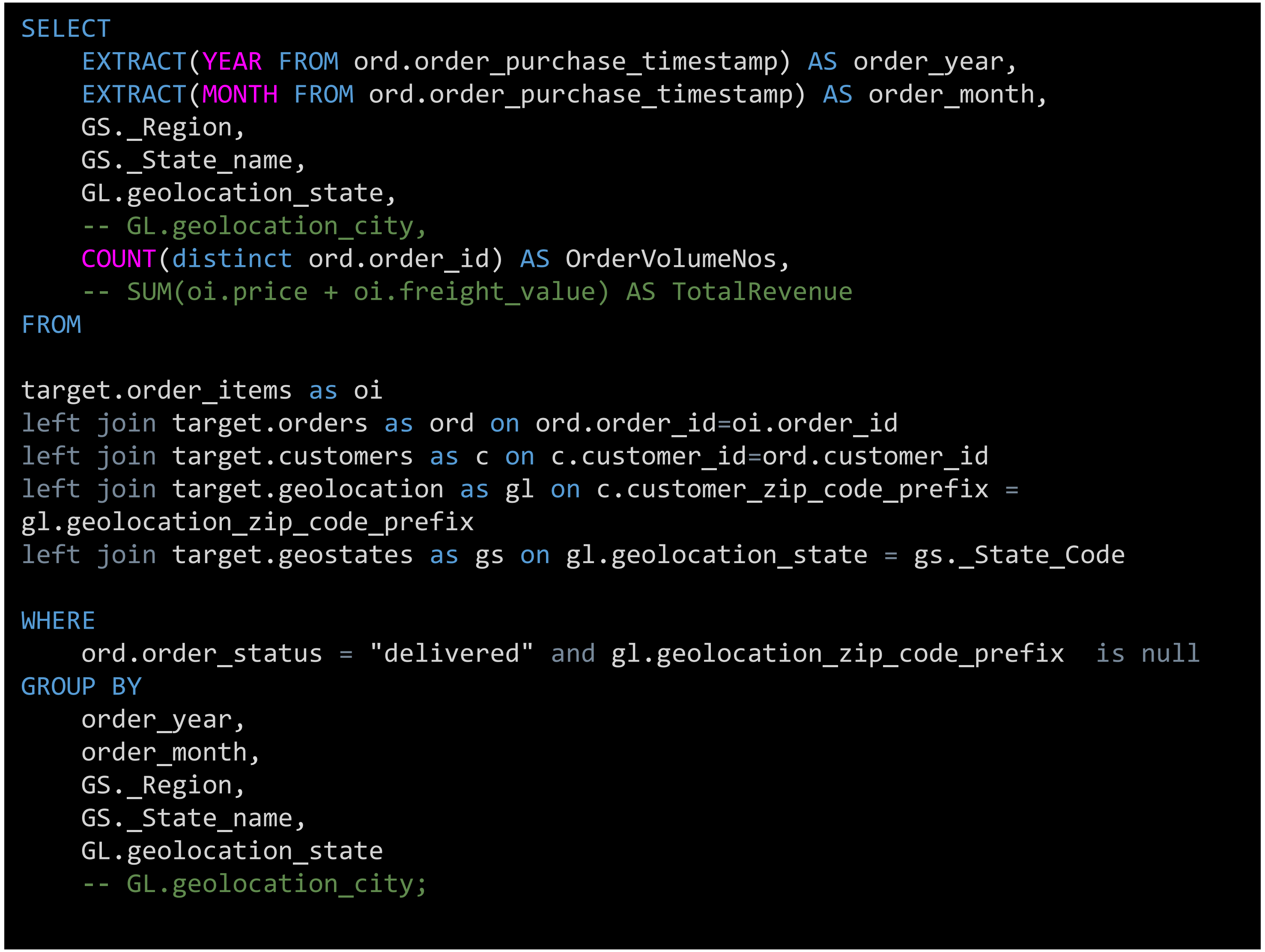


```
    select count(*) from(
select distinct ord.order_id, ord.customer_id,gl.geolocation_zip_code_prefix
FROM
target.orders as ord
    left join target.order_items as oi on ord.order_id=oi.order_id
    left join target.customers as c on ord.customer_id=c.customer_id
        left join target.geolocation as gl on c.customer_zip_code_prefix = gl.geolocation_zip_code_prefix
        left join target.geostates as gs on gl.geolocation_state = gs._State_Code
where order_status="delivered" and gl.geolocation_zip_code_prefix  is null
)
```

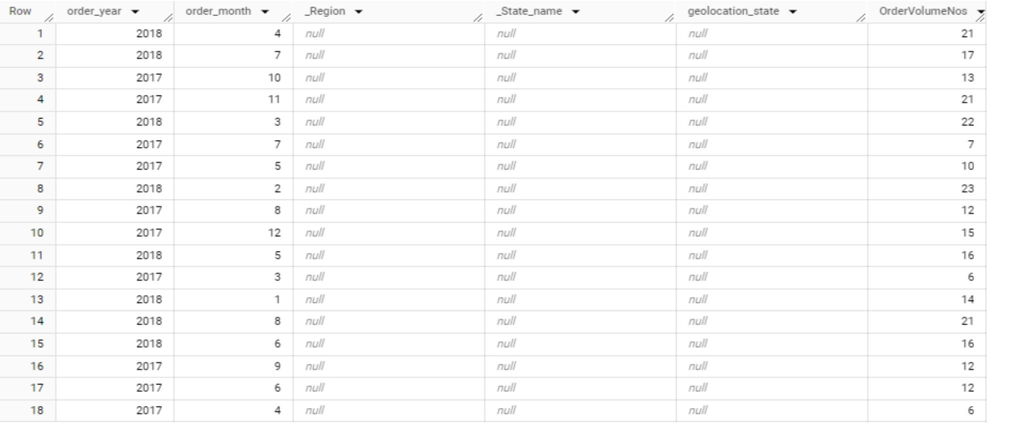
Query results

Row	f0_
1	264

Cont... Data Discrepancy – No ZIP CODE for 264 Records



```
SELECT
    EXTRACT(YEAR FROM ord.order_purchase_timestamp) AS order_year,
    EXTRACT(MONTH FROM ord.order_purchase_timestamp) AS order_month,
    GS._Region,
    GS._State_name,
    GL.geolocation_state,
    -- GL.geolocation_city,
    COUNT(distinct ord.order_id) AS OrderVolumeNos,
    -- SUM(oi.price + oi.freight_value) AS TotalRevenue
FROM

target.order_items as oi
left join target.orders as ord on ord.order_id=oi.order_id
left join target.customers as c on c.customer_id=ord.customer_id
left join target.geolocation as gl on c.customer_zip_code_prefix =
gl.geolocation_zip_code_prefix
left join target.geostates as gs on gl.geolocation_state = gs._State_Code

WHERE
    ord.order_status = "delivered" and gl.geolocation_zip_code_prefix  is null
GROUP BY
    order_year,
    order_month,
    GS._Region,
    GS._State_name,
    GL.geolocation_state
    -- GL.geolocation_city;
```



Row	order_year	order_month	_Region	_State_name	geolocation_state	OrderVolumeNos
1	2018	4	null	null	null	21
2	2018	7	null	null	null	17
3	2017	10	null	null	null	13
4	2017	11	null	null	null	21
5	2018	3	null	null	null	22
6	2017	7	null	null	null	7
7	2017	5	null	null	null	10
8	2018	2	null	null	null	23
9	2017	8	null	null	null	12
10	2017	12	null	null	null	15
11	2018	5	null	null	null	16
12	2017	3	null	null	null	6
13	2018	1	null	null	null	14
14	2018	8	null	null	null	21
15	2018	6	null	null	null	16
16	2017	9	null	null	null	12
17	2017	6	null	null	null	12
18	2017	4	null	null	null	6

Thank You