Target SQL Business Case

Prepared by: Laxmi Sharma

Tool Used: Google BigQuery

Duration: 1 Week

Submission Date: 05 August 2025

1. Introduction

This case study explores the e-commerce operations of Target in Brazil using real-world datasets. The aim is to perform a series of SQL-based data explorations and analyses to uncover insights related to customer behavior, order trends, payment patterns, and delivery performance. Google BigQuery was used to run SQL queries over 8 CSV tables including customers, orders, payments, products, and more.

The project covers exploratory data analysis, order trends, customer distribution, payment behavior, delivery performance, and economic indicators

2. Dataset Overview

The analysis is based on the following tables:
- customers.csv
- geolocation.csv
- order_items.csv
- payments.csv
- reviews.csv
- orders.csv
- products.csv
- sellers.csv

These tables provide information about:

- Customer location and zip codes
- Order and delivery timestamps
- Product categories, weights, and sizes
- Seller regions
- Payment amounts and types
- Review scores and comment

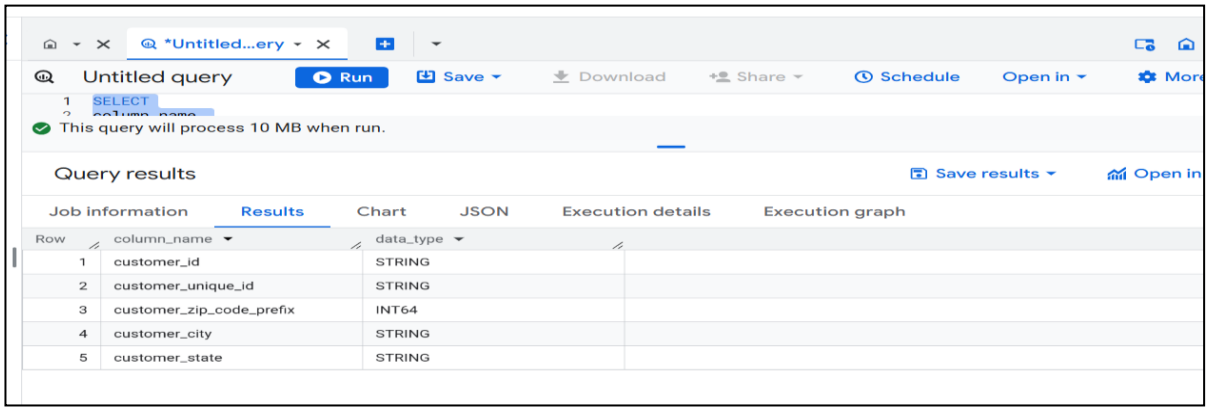
3. Analysis & Insights

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

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

```
SELECT column_name,
data_type
FROM scaler-dsml-sql-463209.target.INFORMATION_SCHEMA.COLUMNS
WHERE table_name = "customers"
```

Output :-

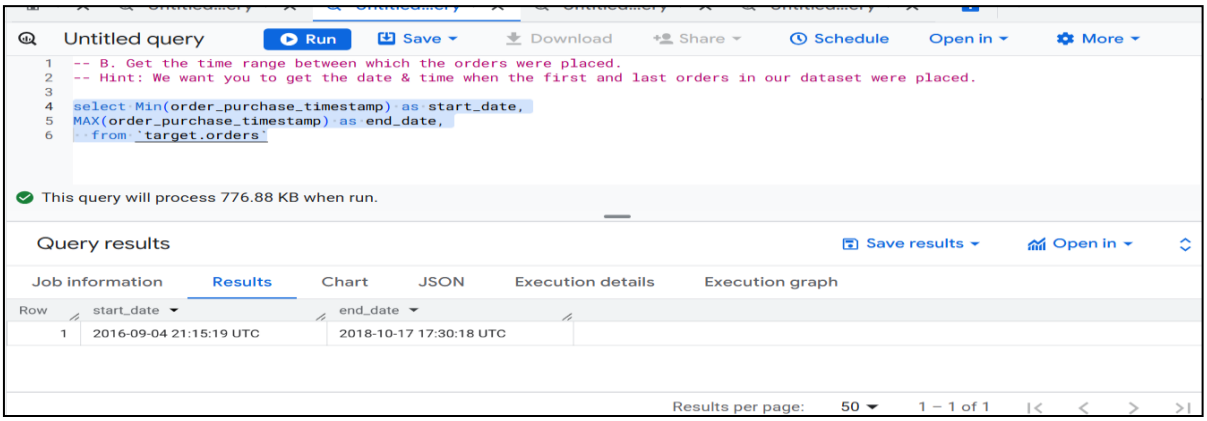


B. Get the time range between which the orders were placed.

Query :-

```
select
Min(order_purchase_timestamp) as start_date,
MAX(order_purchase_timestamp) as end_date,
from `target.orders`
```

Output :-

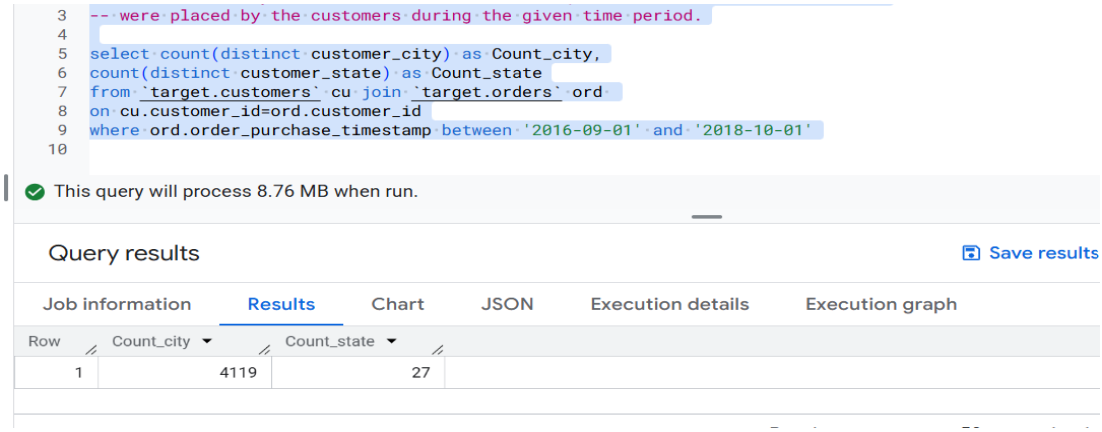


C. Count the Cities & States of customers who ordered during the given period.

Query :-

```
select count(distinct customer_city) as Count_city,
count(distinct customer_state) as Count_state
from `target.customers` cu join `target.orders` ord
on cu.customer_id=ord.customer_id
where ord.order_purchase_timestamp between ‘2016-09-01’ and ‘2018-10-01’
```

Output:-

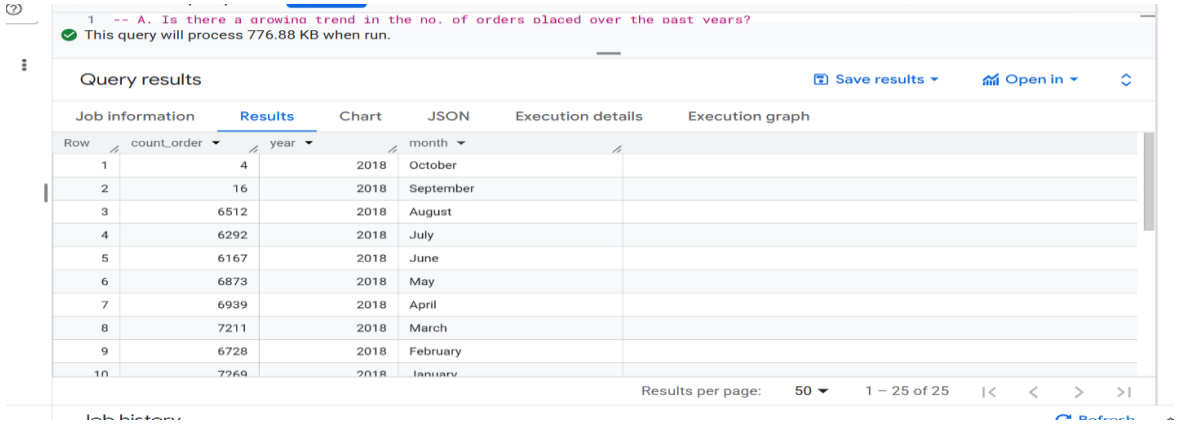


II. In-depth Exploration:

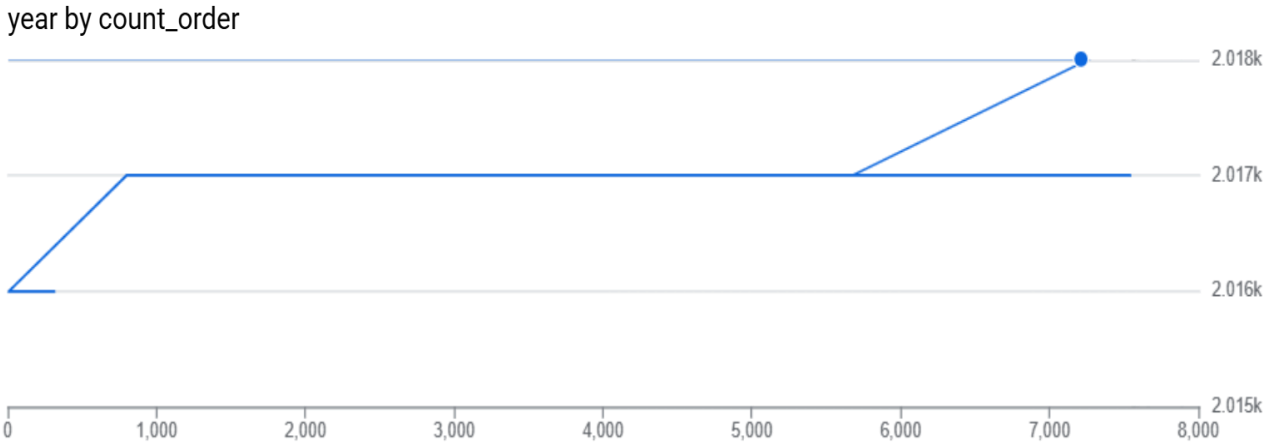
A. Is there a growing trend in the no. of orders placed over the past years?
Query :-

```
select count(*) count_order,
extract(YEAR from  order_purchase_timestamp) year,
format_date ('%B', order_purchase_timestamp) month,
from target.orders
group by year, month
order by year desc , parse_date ('%B', month) desc
```

Output :-



Line Chart :-

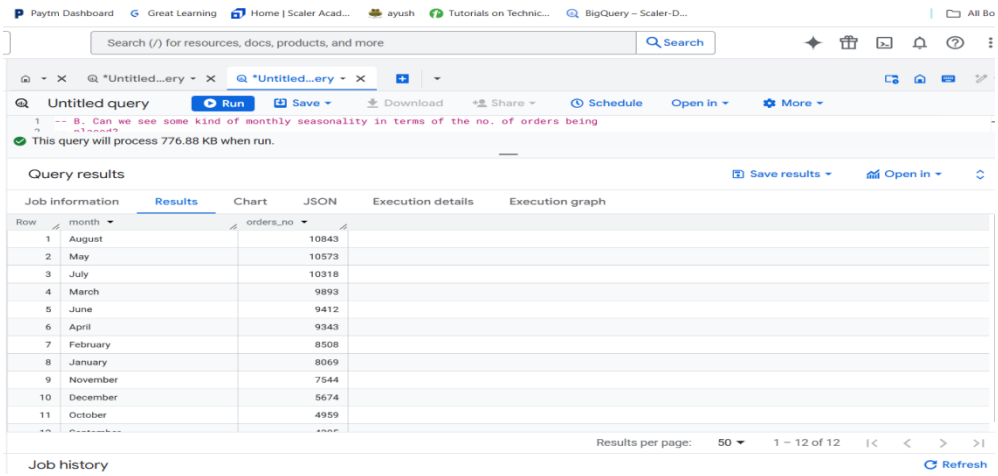


B. Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

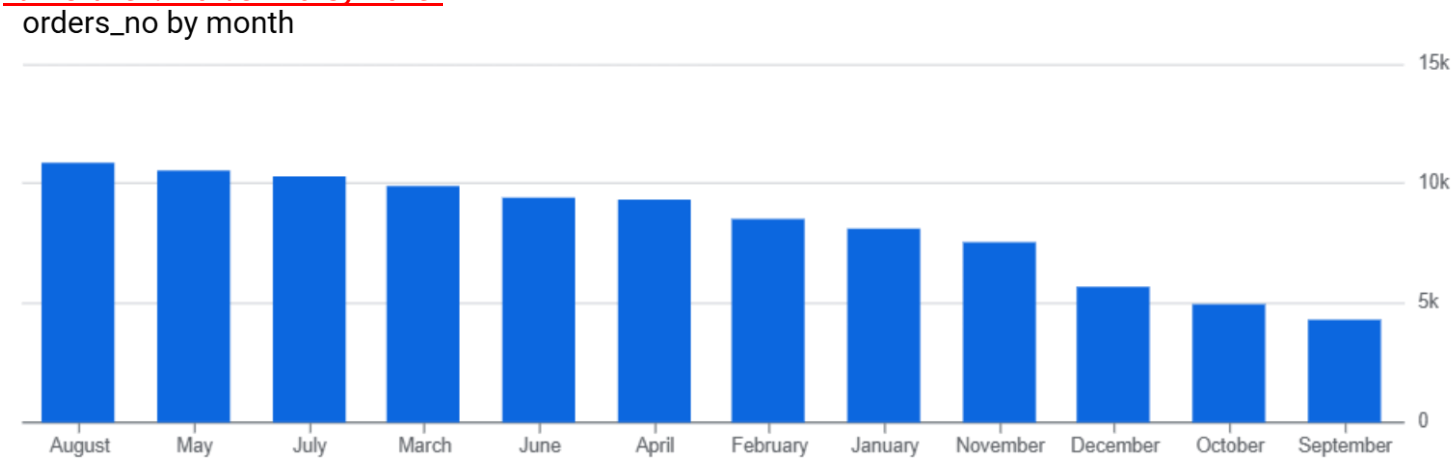
Query :-

```
select
 format_date( '%B',order_purchase_timestamp) as month,
 count (*) orders_no
 from `target.orders`
 group by month
 order by orders_no desc
```

Output :-



Bar Chart :- Order No by Month



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

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

Query :-

```
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 'Mornings'
when extract (HOUR from order_purchase_timestamp ) between 13 and 18 then 'Afternoon'
else 'Night'
end) as intervals_time,
count (*) order_placed
from `target.orders`
group by intervals_time
order by order_placed desc
```

Output:-

Untitled query

Run | Save | Download | Share | Schedule | Open in | More

```
-- C. During what time of the day, do the Brazilian customers mostly place their   -- orders? (Dawn, Morning, Afternoon or Night)
-- • 0-6 hrs : Dawn -- • 7-12 hrs : Mornings   -- • 13-18 hrs : Afternoon     -- • 19-23 hrs : Night
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 'Mornings'
when extract (HOUR from order_purchase_timestamp ) between 13 and 18 then 'Afternoon'
else 'Night'
end) as intervals_time,
count (*) order_placed
from `target.orders`
group by intervals_time
```

This query will process 776.88 KB when run.

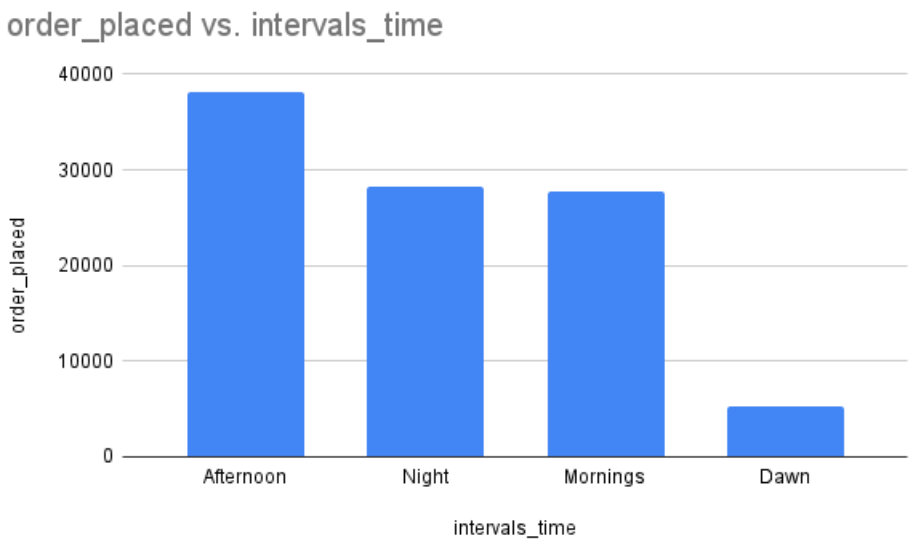
Query results

Save results | Open in

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

Row	intervals_time	order_placed
1	Afternoon	38135
2	Night	28331
3	Mornings	27733
4	Dawn	5242

Bar Chart- Order Placed by Interval time :-



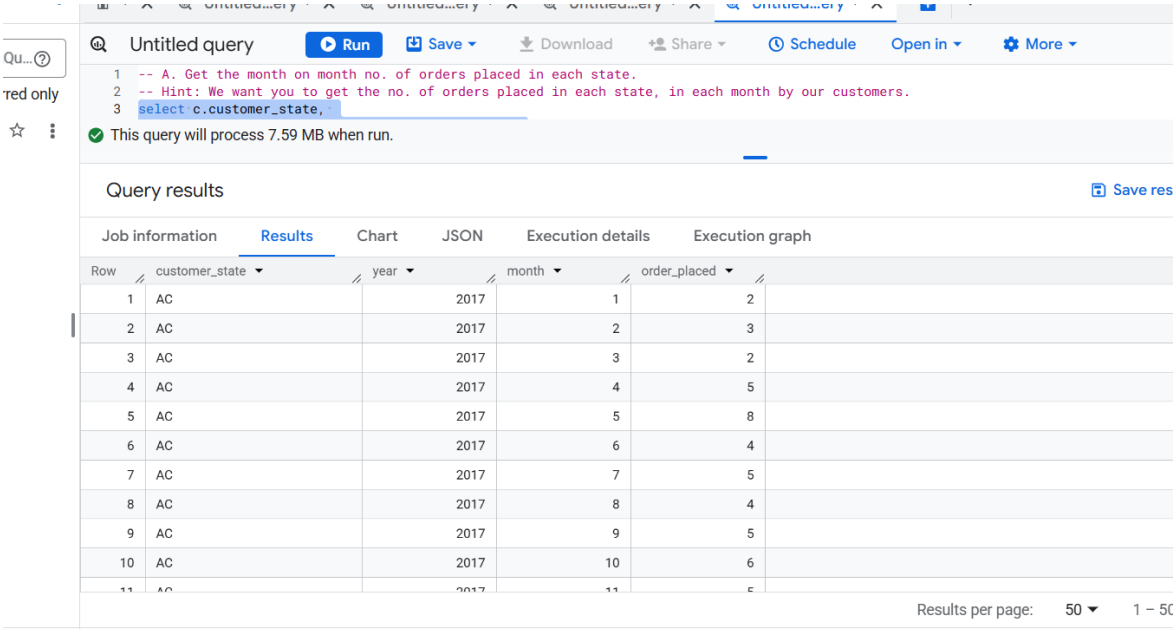
III. Evolution of E-commerce orders in the Brazil region:

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

Query :-

```
select c.customer_state,
extract(year from order_purchase_timestamp) year,
extract(month from order_purchase_timestamp) month,
count (*) order_placed
from `target.customers` c join `target.orders` o
on  c.customer_id = o.customer_id
group by c.customer_state, year,month
order by c.customer_state, year,month
```

Output :-

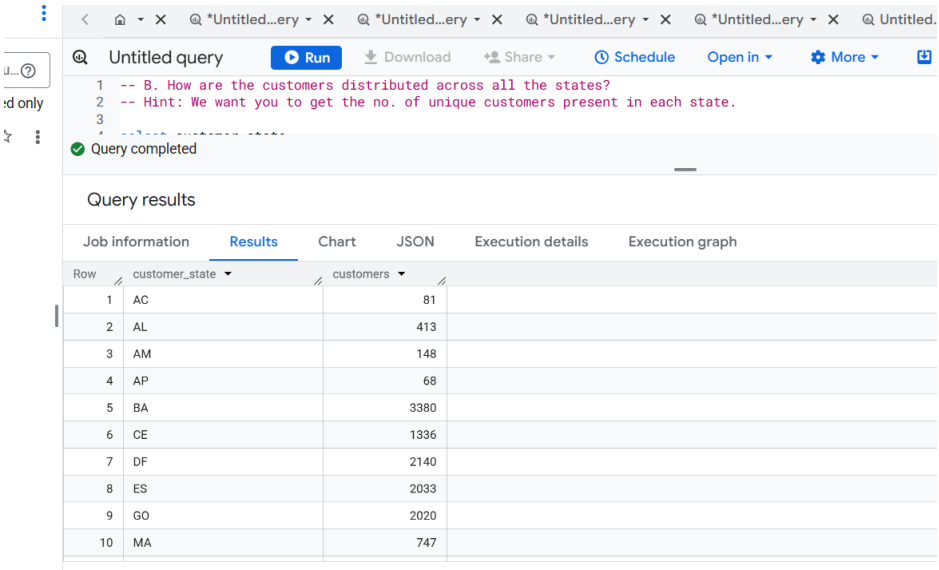


B. How are the customers distributed across all the states?

Query :-

```
select customer_state,
count (distinct customer_id) customers,
from `target.customers`
group by customer_state
order by customer_state
```

Output :-



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

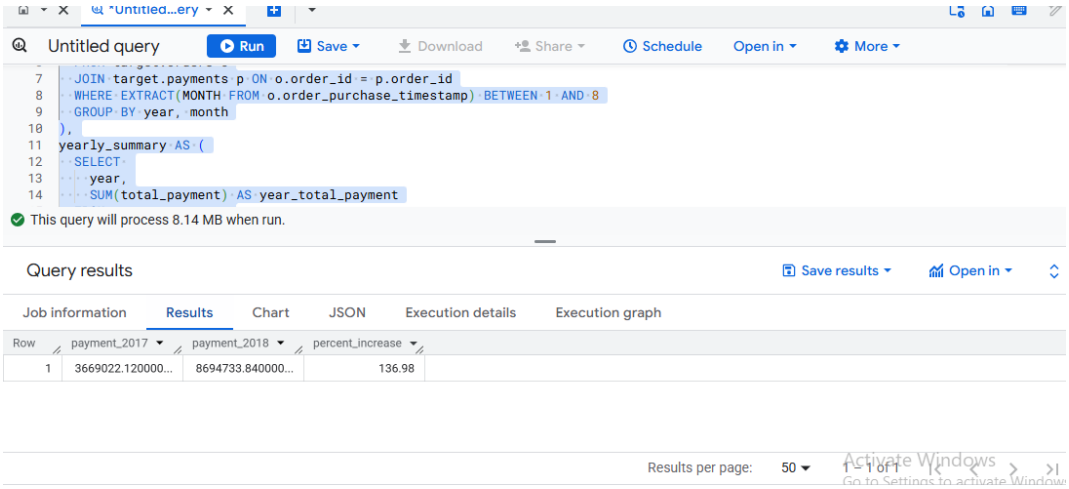
A. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

Query :-

```
WITH payment_summary AS(
  SELECT
    EXTRACT(YEAR FROM o.order_purchase_timestamp)AS year,
    EXTRACT(MONTH FROM o.order_purchase_timestamp)AS month,
    SUM(p.payment_value)AS total_payment
  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
  GROUP BY year, month
),
yearly_summary AS(
  SELECT
    year,
    SUM(total_payment) AS year_total_payment
  FROM payment_summary
  GROUP BY year
)
SELECT
  y2017.year_total_payment AS payment_2017,
  y2018.year_total_payment AS payment_2018,
  ROUND(((y2018.year_total_payment-y2017.year_total_payment)/y2017.year_total_payment)*100,
2)ASpercent_increase
FROM
  (SELECT year_total_payment FROM yearly_summary WHERE year = 2017) y2017,
  (SELECT year_total_payment FROM yearly_summary WHERE year = 2018) y2018
```

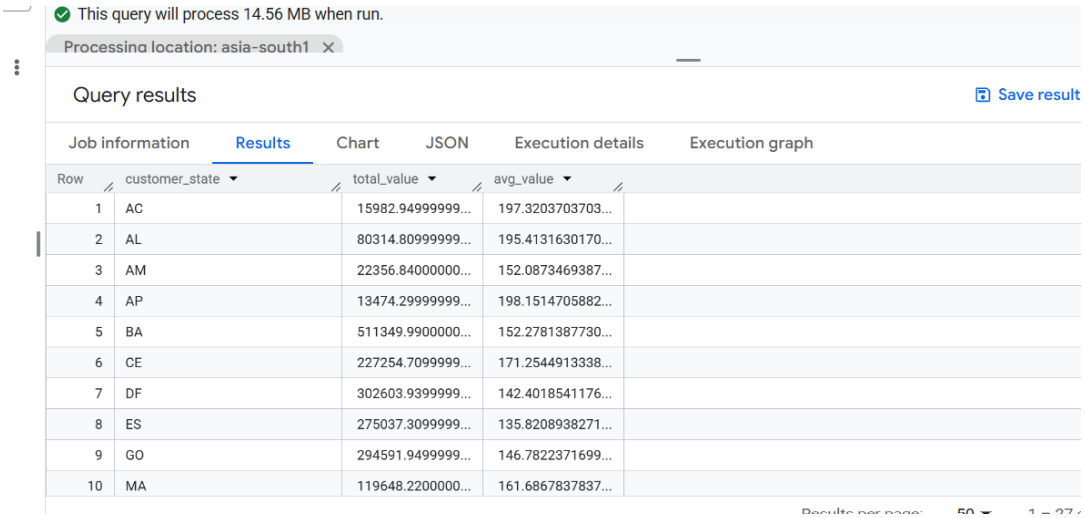
Output :-



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

```
select c.customer_state, sum (oi.price) total_value , sum (oi.price)/ count (distinct
oi.order_id)  avg_value
 from `target.order_items` oi join `target.orders` o
 on oi.order_id = o.order_id
 join `target.customers` c
 on  o.customer_id = c.customer_id
 group by c.customer_state
 order by c.customer_state
```

Output :-



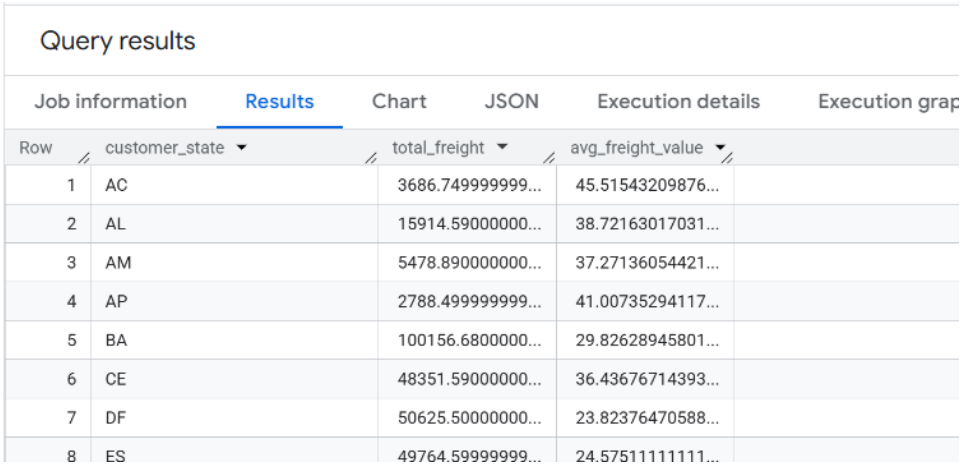
Row	customer_state	total_value	avg_value
1	AC	15982.94999999...	197.3203703703...
2	AL	80314.80999999...	195.4131630170...
3	AM	22356.84000000...	152.0873469387...
4	AP	13474.29999999...	198.1514705882...
5	BA	511349.9900000...	152.2781387730...
6	CE	227254.7099999...	171.2544913338...
7	DF	302603.9399999...	142.4018541176...
8	ES	275037.3099999...	135.8208938271...
9	GO	294591.9499999...	146.7822371699...
10	MA	119648.2200000...	161.6867837837...

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

Query :-

```
select c.customer_state, sum (oi.freight_value) total_freight ,
sum (oi.freight_value)/ count (distinct oi.order_id)  avg_freight_value
from `target.order_items` oi join `target.orders` o
on oi.order_id = o.order_id
join `target.customers` c
on  o.customer_id = c.customer_id
group by c.customer_state
order by c.customer_state
```

Output :-



Row	customer_state	total_freight	avg_freight_value
1	AC	3686.749999999...	45.51543209876...
2	AL	15914.59000000...	38.72163017031...
3	AM	5478.890000000...	37.27136054421...
4	AP	2788.499999999...	41.00735294117...
5	BA	100156.6800000...	29.82628945801...
6	CE	48351.59000000...	36.43676714393...
7	DF	50625.50000000...	23.82376470588...
8	ES	49764.59999999...	24.57511111111...

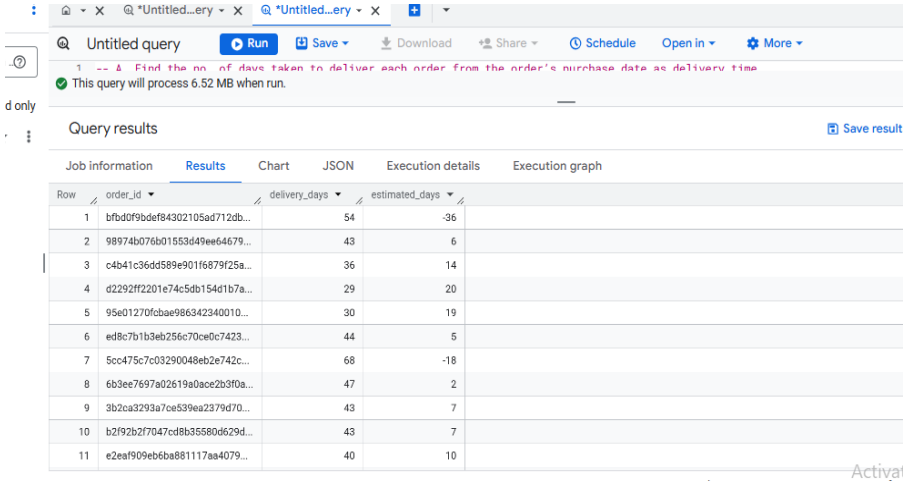
V. Analysis based on sales, freight and delivery time.

A. Find the no. of days taken to deliver each order from the order's purchase date as delivery time. Also, calculate the difference (in days) between the estimated & actual delivery date of an order. Do this in a single query.

Query :-

```
Select order_id,
date_diff(order_delivered_customer_date , order_purchase_timestamp, day)delivery_days,
date_diff(order_estimated_delivery_date ,order_delivered_customer_date , day)estimated_days
from`target.orders`
where order_status = 'delivered'
```

Output :-



Row	order_id	delivery_days	estimated_days
1	bfbd0f9bdef84302105ad712db...	54	-36
2	98974b076b01553d49ee64679...	43	6
3	c4b41c36dd589e901f6879f25a...	36	14
4	d2292ff2201e74c5db154d1b7a...	29	20
5	95e01270fcbae986342340010...	30	19
6	ed8c7b1b3eb256c70ce0c7423...	44	5
7	5cc475c7c03290048eb2e742c...	68	-18
8	6b3ee7697a02619a0ace2b3f0a...	47	2
9	3b2ca3293a7ce539ea2379d70...	43	7
10	b2f92b2f7047cd8b35580d629d...	43	7
11	e2eaf909eb6ba881117aa4079...	40	10

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

Query:-

```
with highest as(select'highest freight'as category ,
c.customer_state ,
avg(freight_value)avg_freight_value
from `target.order_items`oi join `target.orders`o
on oi.order_id = o.order_id
join  `target.customers`c on o.customer_id = c.customer_id
group by customer_state
order by avg_freight_value desc
limit 5),
lowest as(select'lowest freight'as category ,
c.customer_state ,
avg(freight_value)avg_freight_value
from`target.order_items`oi join `target.orders`o
on oi.order_id = o.order_id
join  `target.customers`c on o.customer_id = c.customer_id
group by customer_state
order by avg_freight_value asc
limit 5)

select * from lowest
union all
select * from highest
order by avg_freight_value asc
```

Output :-

This query will process 14.56 MB when run.

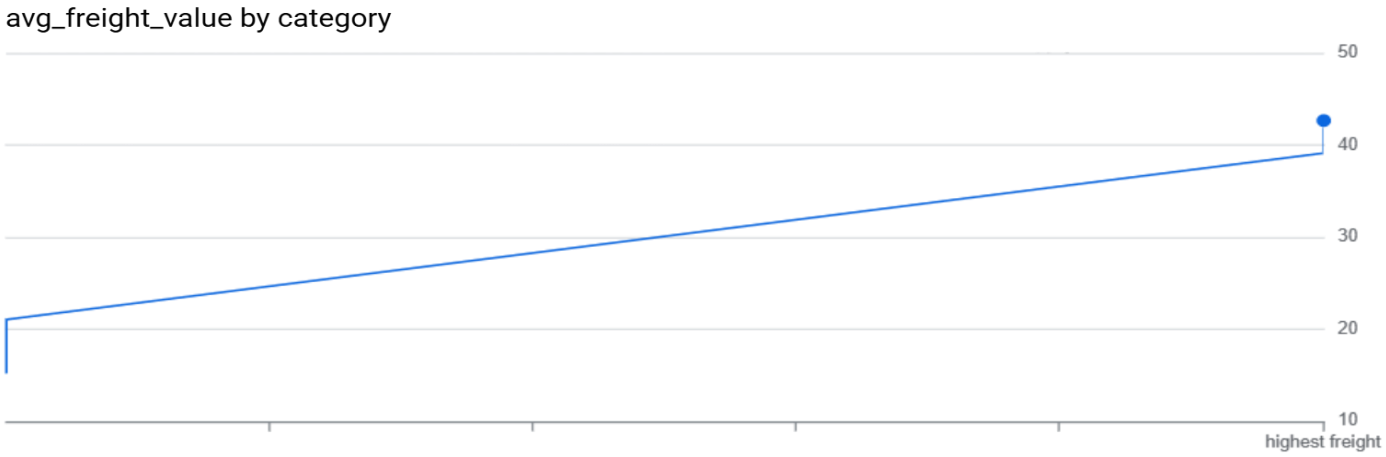
Query results

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

Row	category	customer_state	avg_freight_value
2	lowest freight	PR	20.53165156794...
3	lowest freight	MG	20.63016680630...
4	lowest freight	RJ	20.96092393168...
5	lowest freight	DF	21.04135494596...
6	highest freight	PI	39.14797047970...
7	highest freight	AC	40.07336956521...
8	highest freight	RO	41.06971223021...
9	highest freight	PB	42.72380398671...
10	highest freight	RR	42.98442307692...

Results per page: 50

Line Graph :-



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

```
with lowest_delivery as (select 'lowest_delivery_time'as category, customer_state,
Avg(date_diff(order_delivered_customer_date, order_purchase_timestamp, day)) as
avg_delivery_time
 from `target.customers` c join `target.orders` o
on  c.customer_id = o.customer_id
group by customer_state
order by avg_delivery_time asc
limit 5 ),
highest_delivery as (select 'Highest_delivery_time'as category, customer_state,
Avg(date_diff(order_delivered_customer_date, order_purchase_timestamp, day)) as
avg_delivery_time
 from `target.customers` c join `target.orders` o
on  c.customer_id = o.customer_id
group by customer_state
order by avg_delivery_time desc
limit 5 )
select * from lowest_delivery
union all
select * from  highest_delivery
order by avg_delivery_time asc
```

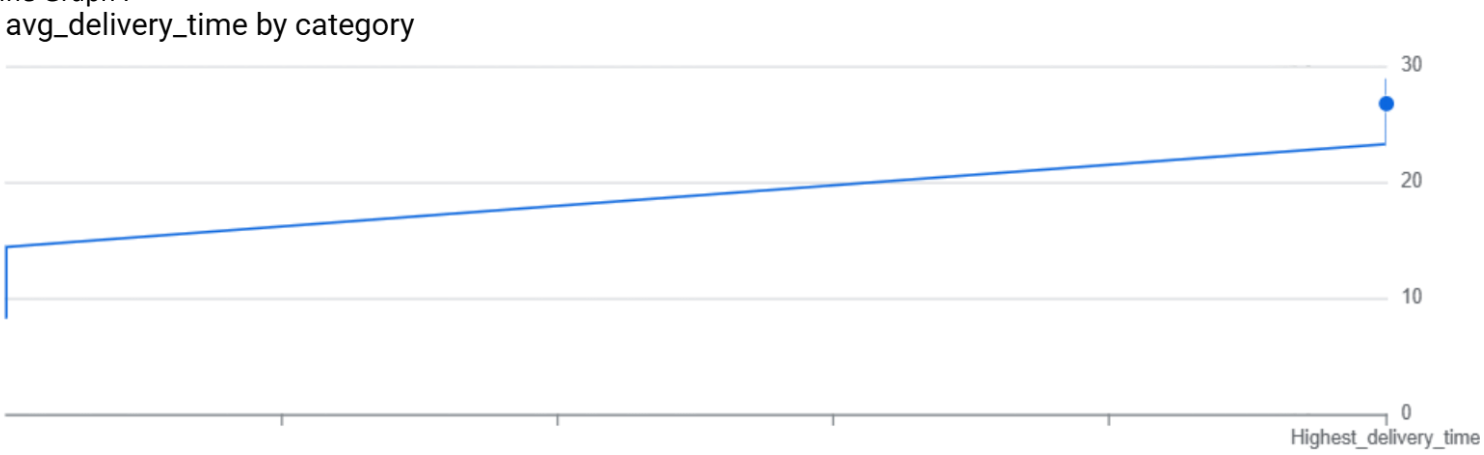
OutPut :-

Query results

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

Row	category	customer_state	avg_delivery_time
1	lowest_delivery_time	SP	8.298061489072...
2	lowest_delivery_time	PR	11.52671135486...
3	lowest_delivery_time	MG	11.54381329810...
4	lowest_delivery_time	DF	12.50913461538...
5	lowest_delivery_time	SC	14.47956019171...
6	Highest_delivery_time	PA	23.31606765327...
7	Highest_delivery_time	AL	24.04030226700...
8	Highest_delivery_time	AM	25.98620689655...
9	Highest_delivery_time	AP	26.73134328358...
10	Highest_delivery_time	RR	28.97560975609...

Line Graph :-

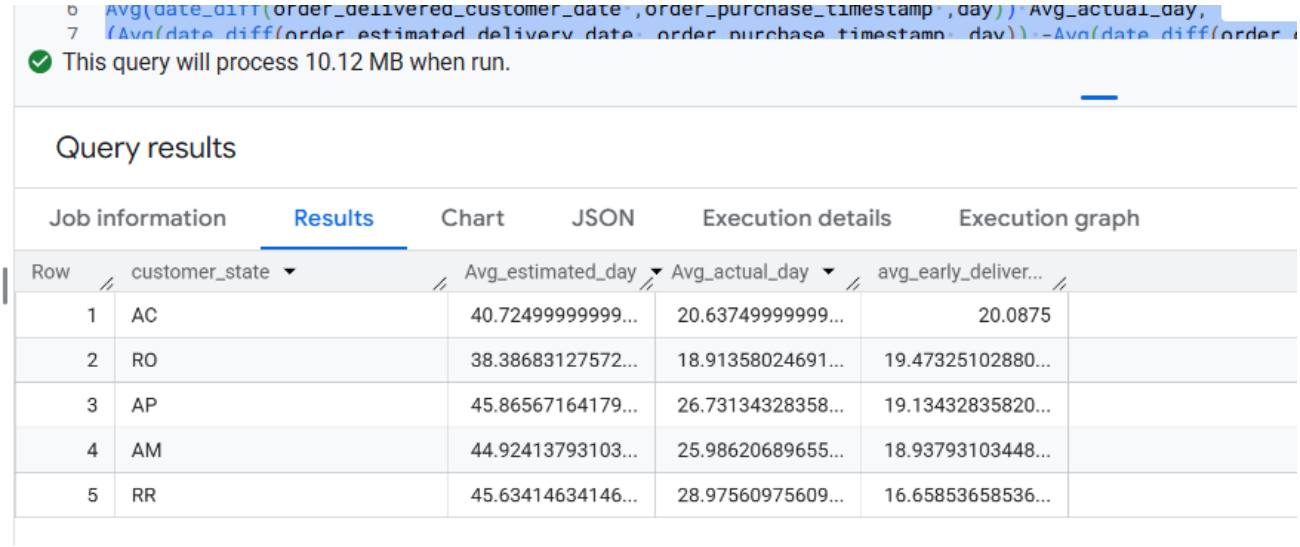


D. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery. You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

Query :-

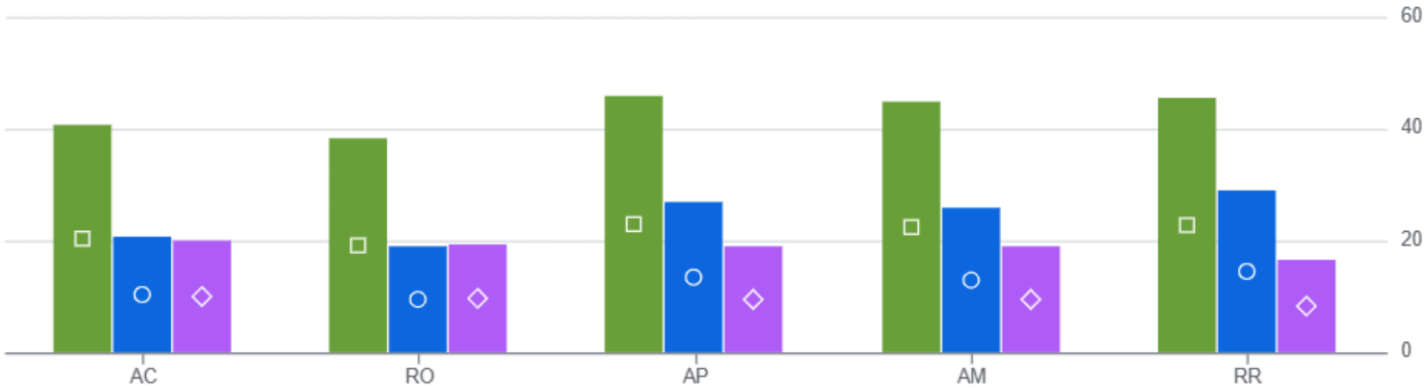
```
select c.customer_state,
Avg(date_diff(order_estimated_delivery_date ,order_purchase_timestamp ,day)) Avg_estimated_day,
Avg(date_diff(order_delivered_customer_date ,order_purchase_timestamp ,day)) Avg_actual_day,
(Avg(date_diff(order_estimated_delivery_date ,order_purchase_timestamp ,day)) -
Avg(date_diff(order_delivered_customer_date ,order_purchase_timestamp ,day)))
as avg_early_delivery_day
from `target.orders` o join `target.customers` c
on o.customer_id = c.customer_id
where order_status = 'delivered'
group by c.customer_state
order by avg_early_delivery_day desc
limit 5
```

Output :-



Row	customer_state	Avg_estimated_day	Avg_actual_day	avg_early_deliver...
1	AC	40.72499999999...	20.63749999999...	20.0875
2	RO	38.38683127572...	18.91358024691...	19.47325102880...
3	AP	45.86567164179...	26.73134328358...	19.13432835820...
4	AM	44.92413793103...	25.98620689655...	18.93793103448...
5	RR	45.63414634146...	28.97560975609...	16.65853658536...

Bar Chart :-

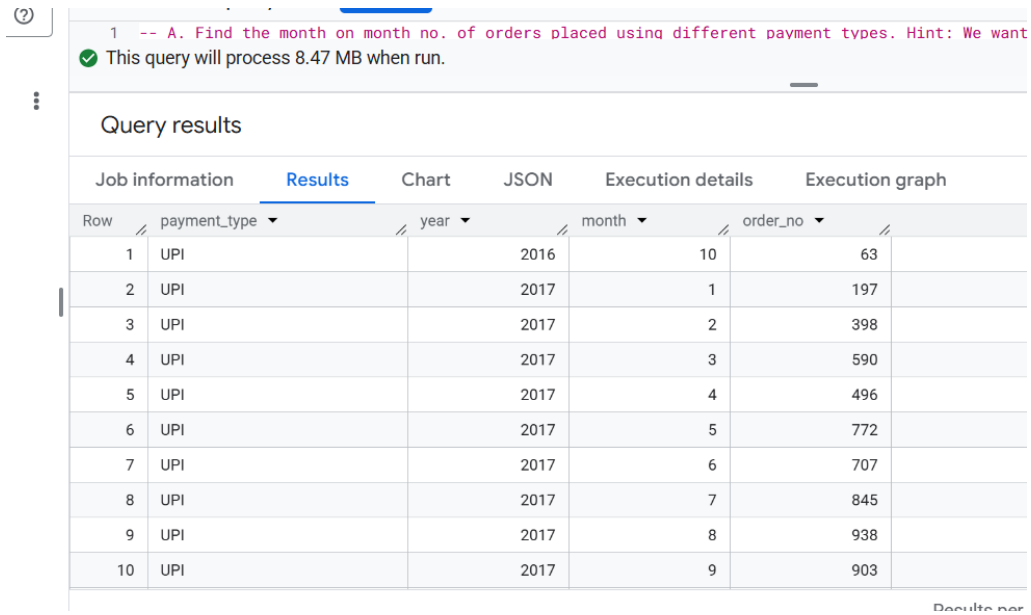


VI. Analysis based on the payments:

A. Find the month on month no. of orders placed using different payment types. Hint: We want you to count the no. of orders placed using different payment methods in each month over the past years.
Query :-

```
select payment_type ,
extract(year FROM order_purchase_timestamp ) as  year,
extract(month FROM order_purchase_timestamp) as month,
count(*) order_no
from `target.orders` o join `target.payments` p
on o.order_id = p.order_id
group by payment_type , year,month
order by payment_type, year, month
```

Output :-



Row	payment_type	year	month	order_no
1	UPI	2016	10	63
2	UPI	2017	1	197
3	UPI	2017	2	398
4	UPI	2017	3	590
5	UPI	2017	4	496
6	UPI	2017	5	772
7	UPI	2017	6	707
8	UPI	2017	7	845
9	UPI	2017	8	938
10	UPI	2017	9	903

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

Query :-

```
select payment_installments ,
count (*) no_order
from `target.payments`
where payment_installments >=1
group by payment_installments
order by payment_installments
```

Output :-

Query results

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

Row	payment_installm...	no_order
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	5	5239
6	6	3920
7	7	1626
8	8	4268
9	9	644
10	10	5328

Insights and Recommendations

Key Insights:

- The dataset spans from September 2016 to October 2018, capturing over 100,000+ real-world e-commerce transactions across Brazil.
- Target has a wide geographic footprint — customers placed orders from hundreds of cities and all major states.
- There's a clear upward trend in the number of orders, especially noticeable during late 2017 and 2018, indicating consistent growth.
- November and December showed high order volumes, suggesting strong seasonality around holidays and year-end.
- Afternoon and evening hours were the most common times for placing orders, aligning with post-work customer behavior.
- States like São Paulo (SP), Rio de Janeiro (RJ), and Minas Gerais (MG) had the highest order volumes and payment totals.
- Average delivery times in many states were faster than estimated, which reflects efficient logistics and customer satisfaction potential.
- Freight costs vary significantly by region — remote areas have higher logistics costs, while states near major sellers have lower averages.
- Credit card is the most frequently used payment method, but boleto and voucher payments are also popular in certain regions.
- A significant portion of customers prefer installment-based payments, especially 2–3 installments, indicating reliance on EMI.

Business Recommendations:

- Increase marketing and inventory during peak shopping months like November and December to maximize revenue during seasonal spikes.
- Enhance delivery infrastructure in slower or higher-freight-cost states by investing in regional fulfillment centers or local partners.
- Promote digital payments like credit cards and vouchers by offering discounts, cashback, or loyalty points to reduce payment friction.
- Introduce faster delivery incentives in high-performing states to maintain competitive edge and set internal delivery benchmarks.
- Highlight installment options clearly during checkout, especially EMI schemes, to encourage purchases from cost-conscious buyers.
- Use customer location, order frequency, and payment behavior to create personalized marketing campaigns and predictive logistics planning.
- Continuously monitor delivery performance by region and benchmark the top 5 fastest states against the slowest to drive operational improvement.
- Focus on freight optimization strategies such as dynamic pricing, warehouse mapping, or shared seller logistics to control costs.

Executive Summary

This case study analyzes the e-commerce operations of Target in Brazil using structured SQL queries on customer, order, payment, delivery, and product datasets.

The analysis covers over 100,000 transactions from September 2016 to October 2018 and focuses on identifying business trends, customer behavior, and operational bottlenecks.

Key findings include a consistent increase in monthly orders, strong seasonal peaks during November–December, and dominant use of digital payments (especially credit cards). Customers in major states like São Paulo and Rio de Janeiro contribute the highest revenue. Delivery times in several regions are faster than expected, showcasing efficient logistics, while other regions show opportunities to improve shipping timelines and reduce freight costs.

The project concludes with actionable recommendations to optimize marketing strategies, payment methods, and delivery performance — all aligned with real customer patterns.