**Project Context: Expanding Amazon's Success from Brazil to India**

**Background**

Amazon has established itself as a global leader in e-commerce, successfully penetrating markets in the U.S., Europe, and Asia. In Brazil, Amazon has played a pivotal role in connecting small and medium businesses with millions of customers, strengthening its market presence. Given the demographic and economic similarities between Brazil and India—both having large, diverse consumer bases—Amazon sees an opportunity to replicate its success in India.

**Objective**

As a data analyst for Amazon India, our role is to analyse Amazon Brazil’s data to uncover key trends, customer behaviours, and preferences. The insights derived from this analysis will help Amazon India:

* Understand purchasing patterns and consumer behaviour in Brazil.
* Identify strategies that contributed to Amazon’s success in Brazil.
* Determine which trends and business strategies can be adapted for the Indian market.
* Enhance the overall customer experience and optimize business opportunities in India.

This project aims to leverage data-driven decision-making to strengthen Amazon India’s market position, replicating and localizing the success seen in Brazil.

**Schema**

The database schema consists of six interconnected tables that provide insights into Amazon Brazil’s e-commerce operations. Each table serves a distinct purpose:

* **Customer :** Stores customer information, including customer id, customer unique id, customer zip code prefix.
* **Orders:** Captures essential order details, such as, customer id, order status, purchase timestamp, approved date, deliver carrier date, deliver customer date, and estimated delivered date, to track the entire lifecycle.
* **Order items:**  Provides a breakdown of each order by listing individual items along with price, seller, product, and shipping information.
* **Product:** Contains product specifications such as category, size, and weight, which are crucial for product-level analysis.
* **Seller:** Holds seller information, including unique seller IDs and seller zip code , allowing performance evaluation across different regions.
* **Payment:** Logs transaction details, including payment method and amount, helping analyse customer payment preferences and financial performance.

These tables are interconnected through unique identifiers such as customer id, order id, product id, and seller id, facilitating seamless data retrieval for analytical purposes.

**Analyses - I**

**Problem Statement 1**

To simplify its financial reports, Amazon India needs to standardize payment values

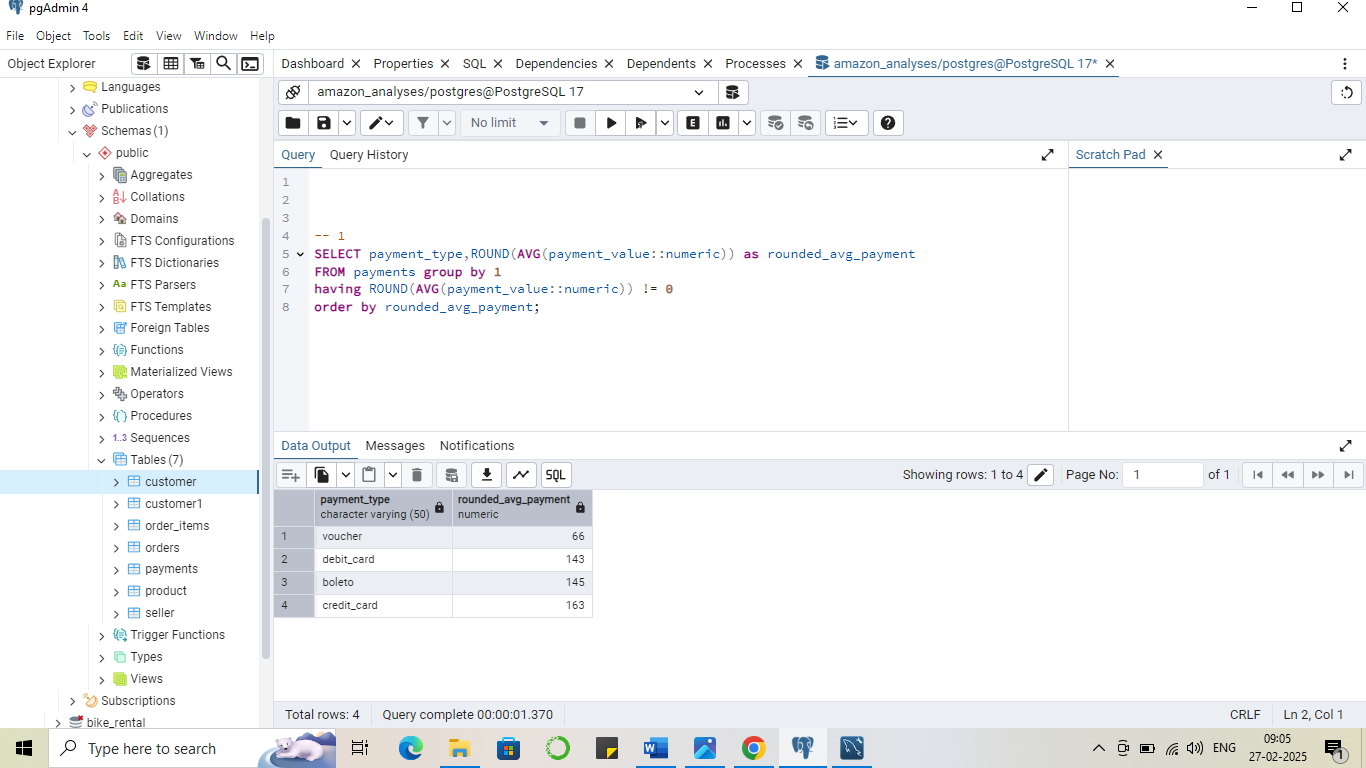
**Approach;**

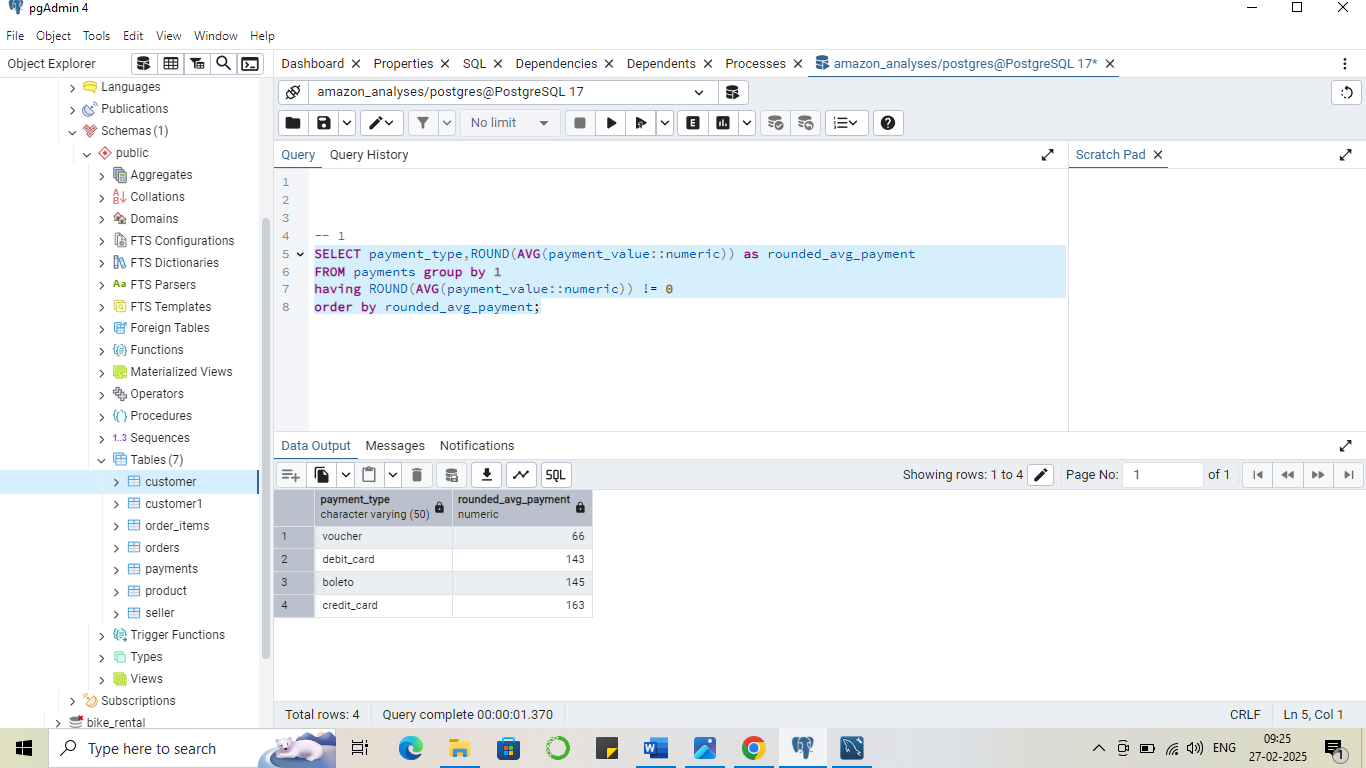
**Identifying Relevant Tables**

* The Payments table contains transaction details.
* The payment**\_**type column identifies the type of payment used.
* The payment**\_**value column stores the amount paid per transaction.

**Defining Query Logic**

* Group the data by payment type using GROUPBY.
* Calculate the average payment value using AVG(payment\_value::numeric).
* Round the calculated average for better readability using ROUND().
* Exclude zero-value payments using HAVING ROUND(AVG(payment\_value::numeric)) != 0.
* Sort the results by the rounded average payment value using ORDER BY.

**Output**



**Recommendations**

* 1. Prioritize Credit and Boleto Payments:-
* Credit cards (₹163) and boleto (₹145) have the highest average transaction value.
* Amazon India should ensure smooth processing for these methods and consider offering EMI options or cashback for credit card users.
  1. Enhance Debit Card Transactions
* The average debit card payment (₹143) is slightly lower than credit cards.
* Simplifying debit card transactions and offering discounts on debit payments could encourage more usage.
  1. Encourage Voucher Usage
* Vouchers have the lowest average transaction value (₹66), suggesting they are used for smaller purchases.
* Amazon India can introduce higher-value vouchers, loyalty rewards**,** or bundle discounts to increase their usage and transaction size.
  1. Adapt Boleto Strategy for India
* Boleto (₹145) is a popular offline payment method in Brazil.
* A similar strategy in India could involve cash-based digital payments like UPI, Paytm Postpaid, or Buy Now Pay Later (BNPL) services to cater to cash-preferred customers.

**Problem Statement 2**

To refine its payment strategy, Amazon India wants to know the distribution of orders by payment type.

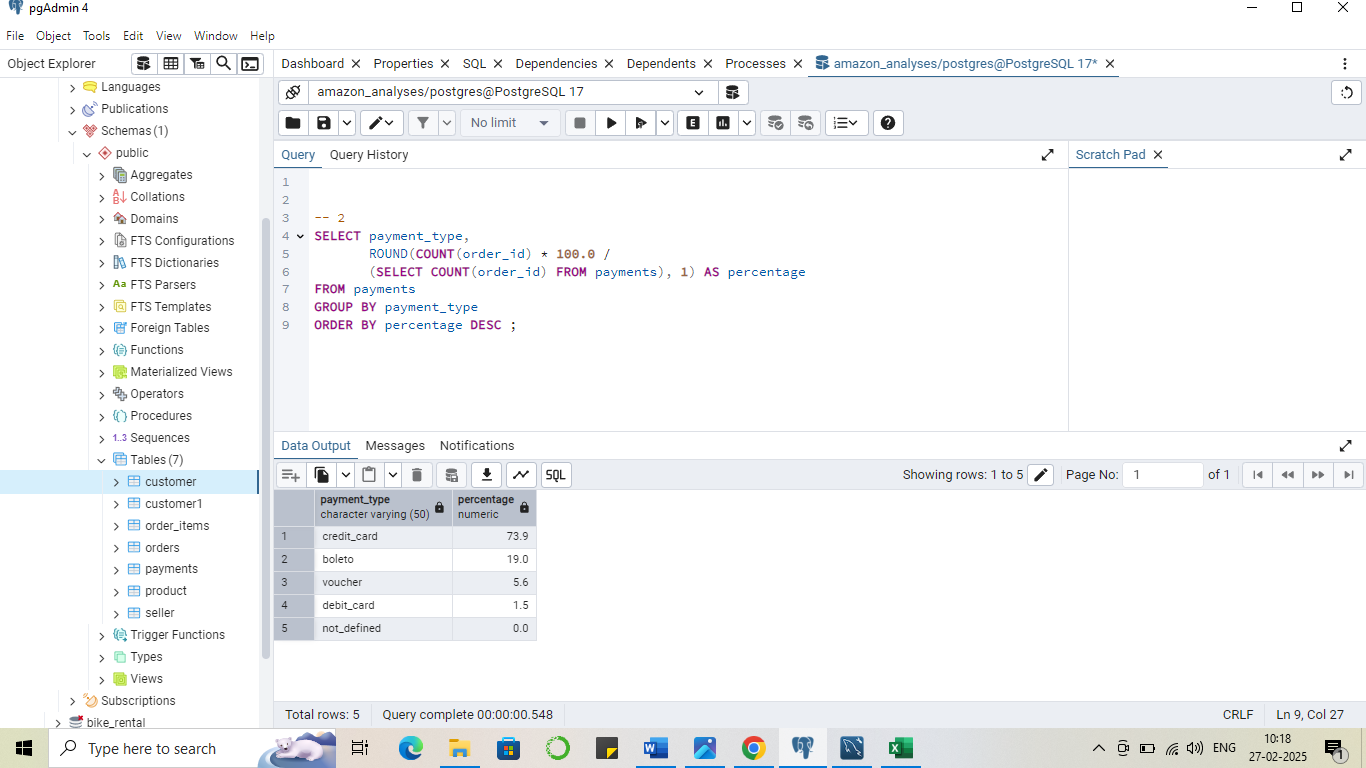
**Approach;**

**Identifying Relevant Tables**

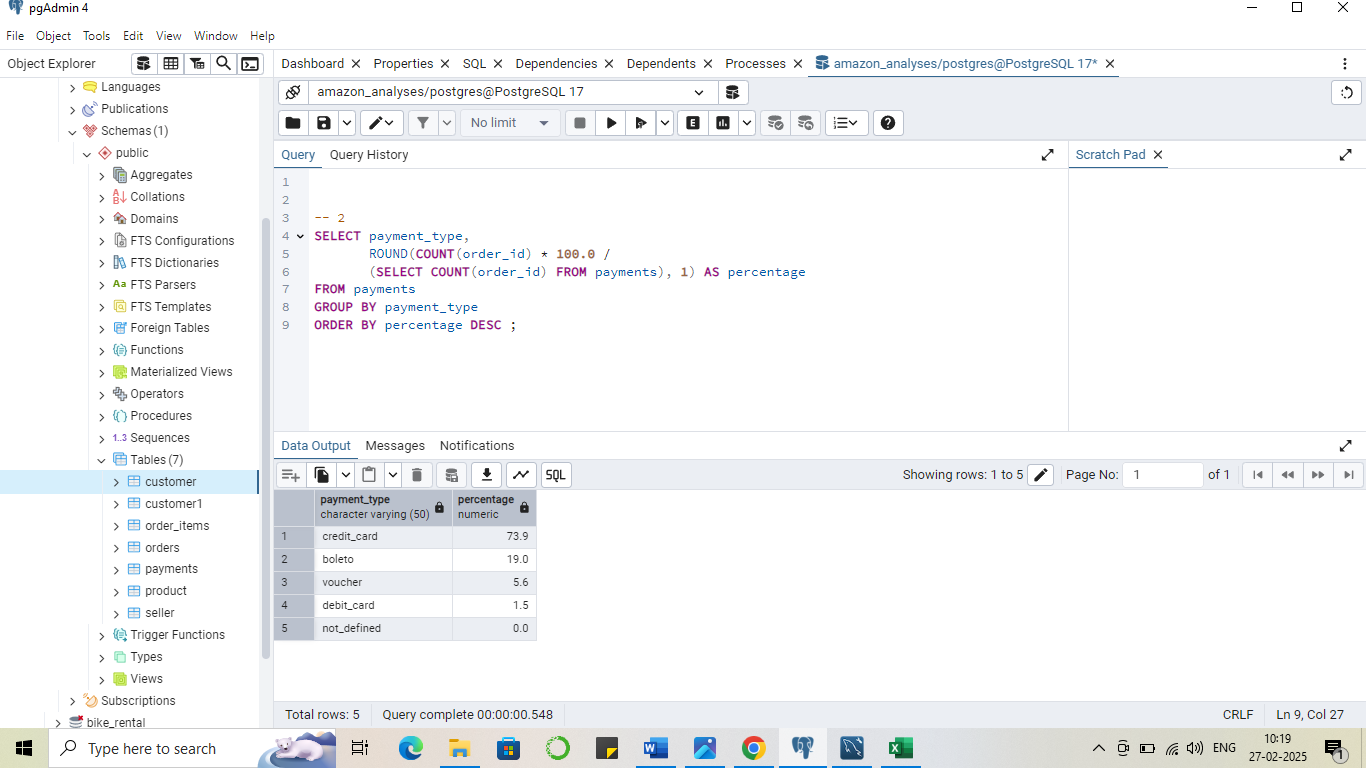
* The payments table contains transaction details related to orders.
* The payment\_type column specifies the type of payment used.
* The order\_id column helps count the number of orders for each payment type.

**Defining Query Logic**

* Group the data by payment\_type to analyze the distribution of orders for each payment method.
* Calculate the total number of orders for each payment type using COUNT(order\_id).
* Determine the percentage contribution of each payment type by dividing its count by the total number of orders, then multiplying by 100.
* Use ROUND(..., 1) to format the percentage with one decimal place for better readability.
* Sort the results in descending order by percentage using ORDER BY percentage DESC to highlight the most frequently used payment methods.



**Output**



**Recommendations**

1. Focus on Credit Card Optimization

* With 73.9% of orders made via credit cards, ensuring a seamless and secure payment experience for credit card users is essential.
* Offering EMI options, cashback, or reward points can further enhance customer retention and encourage higher spending.

1. Enhance Support for Boleto-like Alternatives

* 19% of transactions use boleto, a popular offline payment method in Brazil.
* Amazon India can adopt a similar approach by promoting UPI, Pay on Delivery (PoD), or Buy Now Pay Later (BNPL) services to cater to customers preferring non-card payments.

1. Increase Voucher Usage Through Incentives

* Vouchers contribute 5.6% of transactions, indicating moderate adoption.
* Expanding voucher-based promotions, loyalty rewards, or cashback on future purchases can drive more engagement.

1. Encourage Debit Card Transactions

* Debit cards account for only 1.5% of payments, suggesting limited usage.
* Introducing exclusive debit card discounts, bank tie-ups, or simplified OTP-based authentication may improve adoption.

**Problem Statement 3**

Amazon India seeks to create targeted promotions for products within specific price ranges.

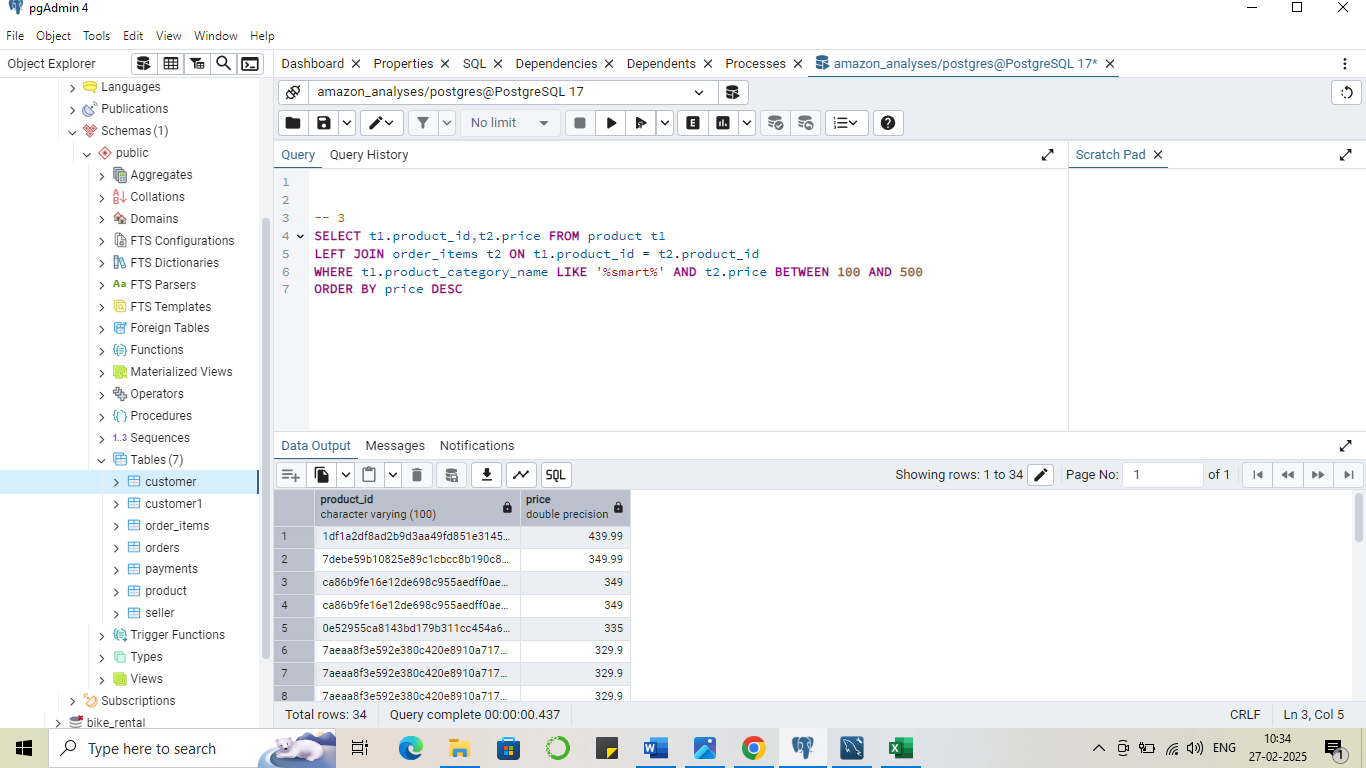
**Approach;**

**Identifying Relevant Tables**

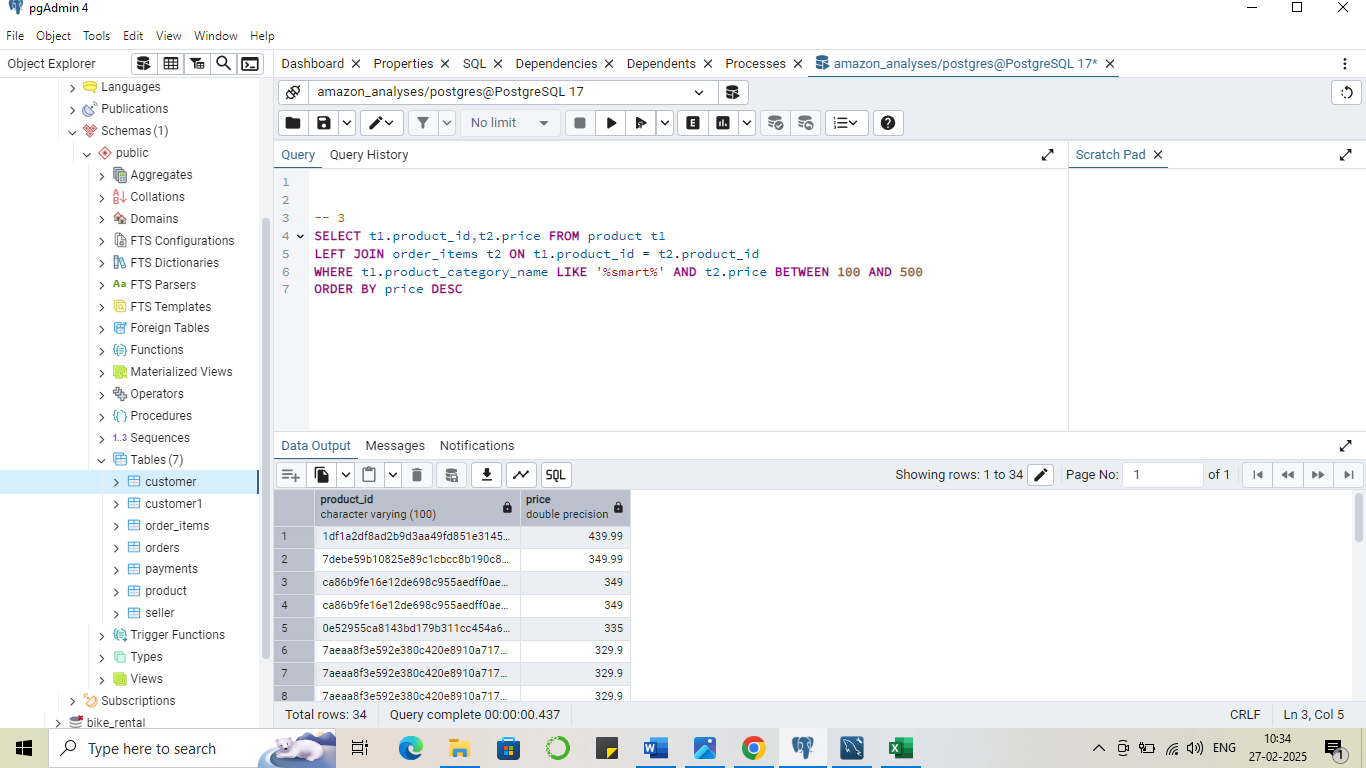
* The product table contains product details, including product\_id and product\_category\_name.
* The order\_items table provides pricing information for each product.
* The product\_id column is used to join both tables.

**Defining Query Logic**

* Perform a **LEFT JOIN** between product and order\_items on product\_id to retrieve product pricing details.
* Filter results where the product\_category\_name contains the word "smart" using LIKE '%smart%' to target relevant products.
* Include only products with a price between 100 and 500 using the BETWEEN clause to focus on specific price ranges.
* Sort results in descending order of price using ORDER BY price DESC to highlight higher-priced products first.



**Output**



**Recommandation**

1. Segment Products into Promotional Price Brackets(Premium Segment, Mid-Range, Budget-Friendly)
2. Create Bundle Deals Based on Popular Price Points(Budget products and mid values products to get higher order value)
3. Optimize Discounts for Price-Sensitive Customers(price 199 instead of 200 or deal of the day)

**Problem Statement 4**

To identify seasonal sales patterns, Amazon India needs to focus on the most successful months.

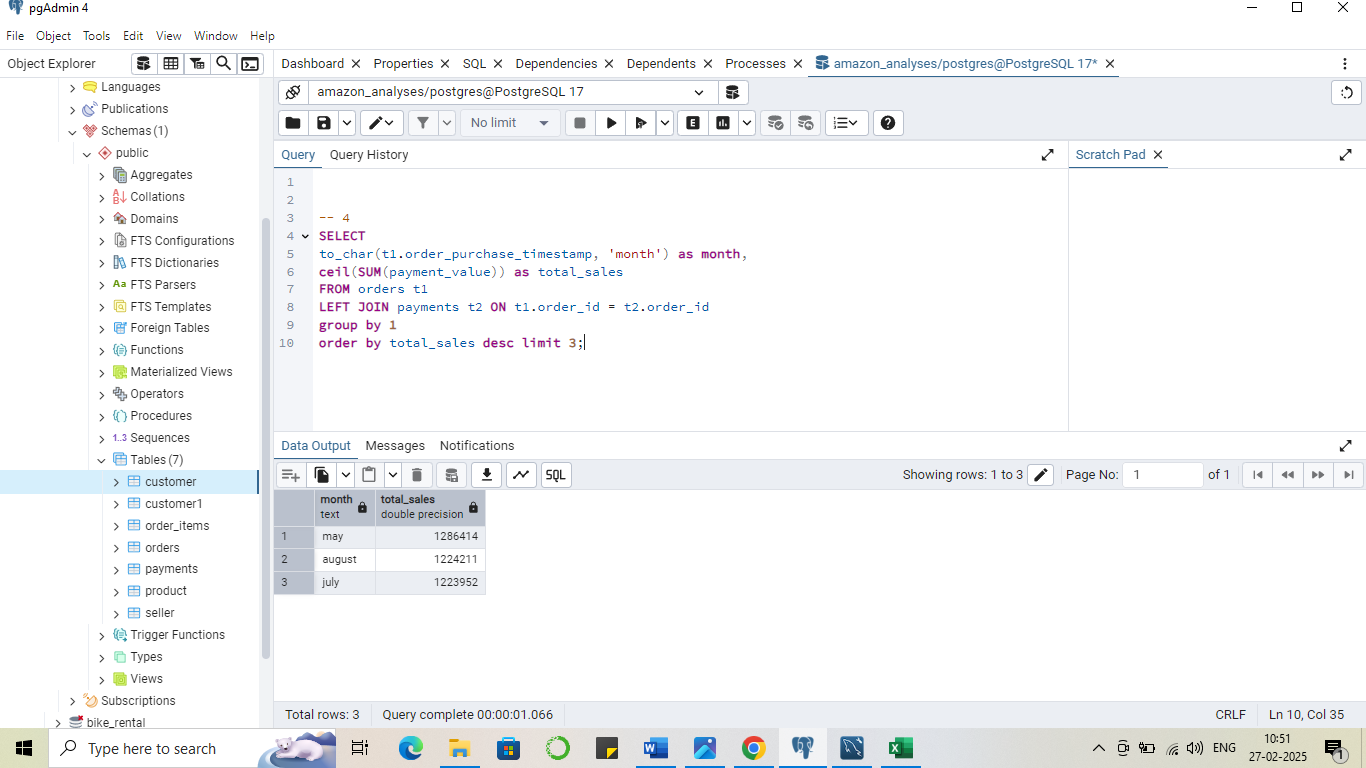
**Approach;**

**Identifying Relevant Tables**

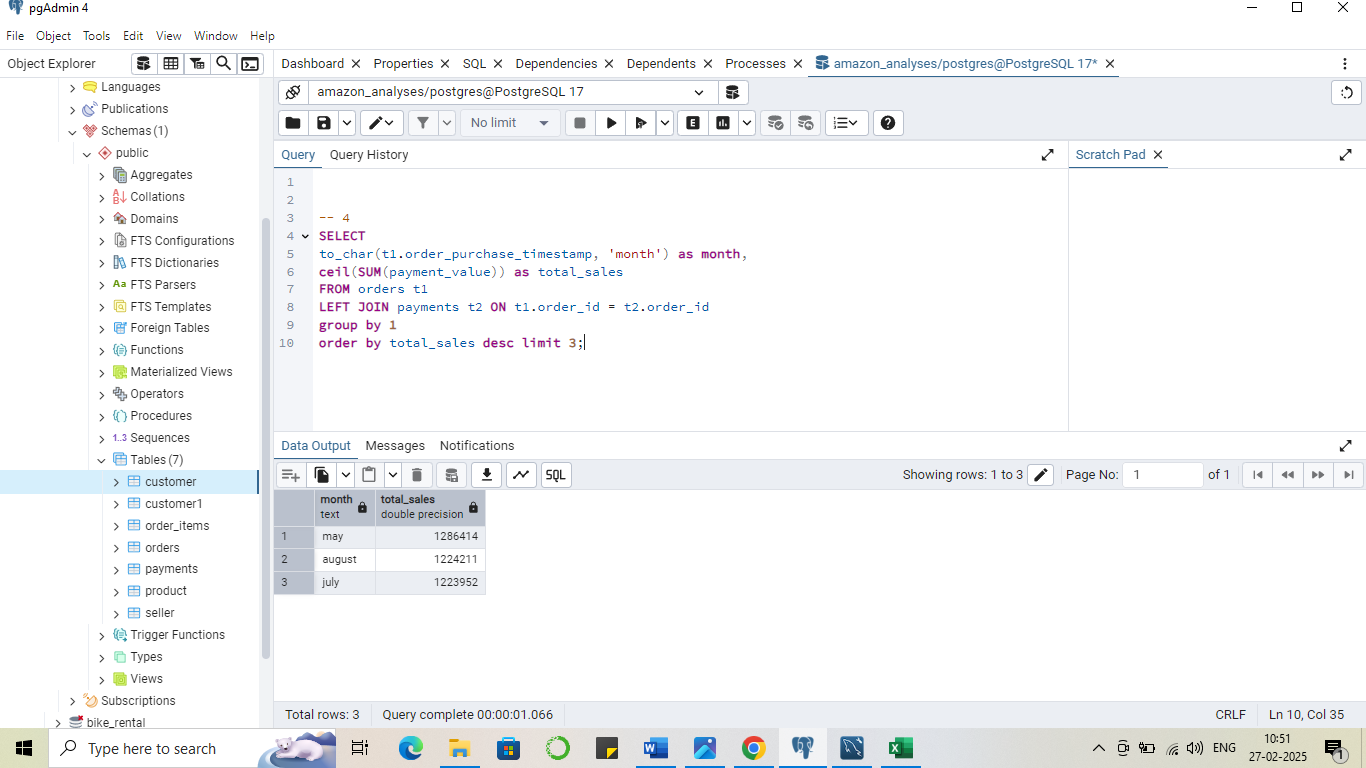
1. The orders table provides order timestamps to extract the month of purchase.
2. The payments table contains payment values, which represent total sales.

**Define Query Logic:**

1. Extract the month from order\_purchase\_timestamp using to\_char().
2. Calculate the total sales per month by summing payment\_value
3. Use ceil() to round up the sales value.
4. Group by month to aggregate sales data.
5. Order results in descending order to identify the top-performing months.
6. Limit the output to top 3 months based on sales.



**Output**



**Recommendations:**

1. Peak Season Promotions;- May, August, and July show the highest sales. Amazon India should plan major promotional campaigns, discounts, and marketing efforts during these months.
2. Inventory Management – Ensure sufficient stock availability before these peak months to meet increased demand and prevent stockouts.

**Problem Statement 5**

Amazon India is interested in product categories with significant price variations.

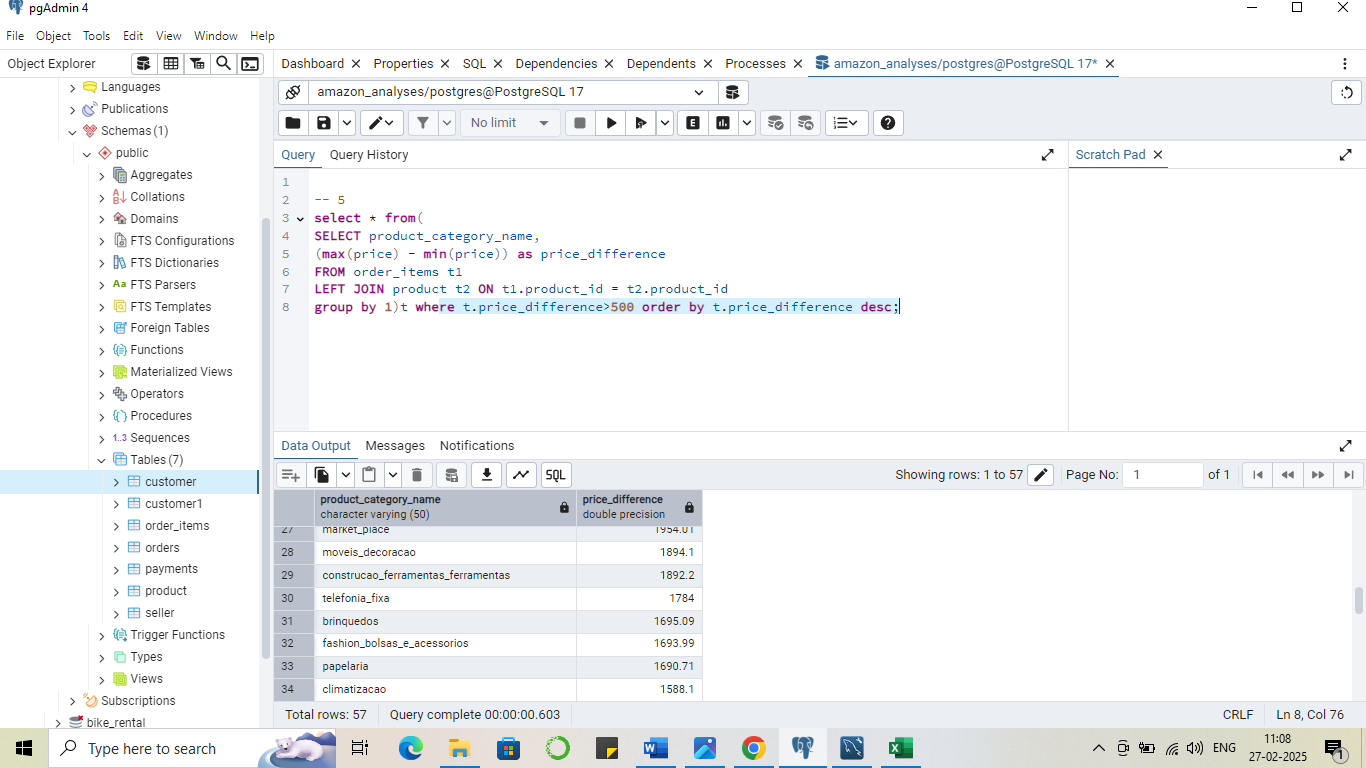
**Approach;**

**Identifying Relevant Tables**

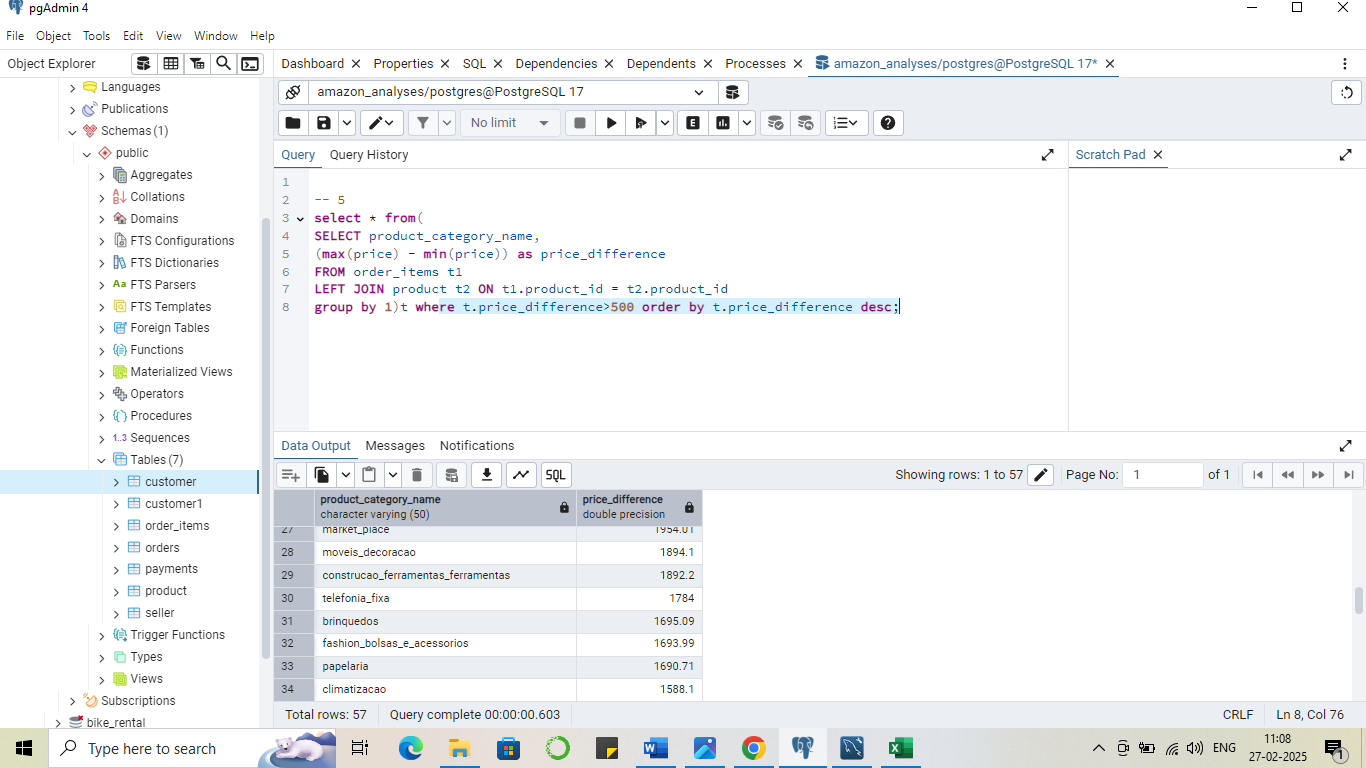
* The order\_items table contains product prices for each order.
* The product table provides product category names to analyze price variations across categories.

**Define Query Logic:**

* Use a LEFT JOIN to link order\_items with product based on product\_id
* Group by product\_category\_name to calculate price variations within each category.
* Compute price difference as max(price) - min(price).
* Filter categories where the price difference is greater than 500 to identify significant variations.
* Order the results in descending order based on price\_difference to highlight the most variable categories.



**Output**



1. Target Promotions for High-Variation Categories: bundel pricing or give discount on premium products.
2. Segment Products for Better Marketing: like budget, mid and premium
3. Run market segmentation analysis on premium vs. budget product performance. For furtur decision

**Problem Statement 6**

To enhance the customer experience, Amazon India wants to find which payment types have the most consistent transaction amounts.

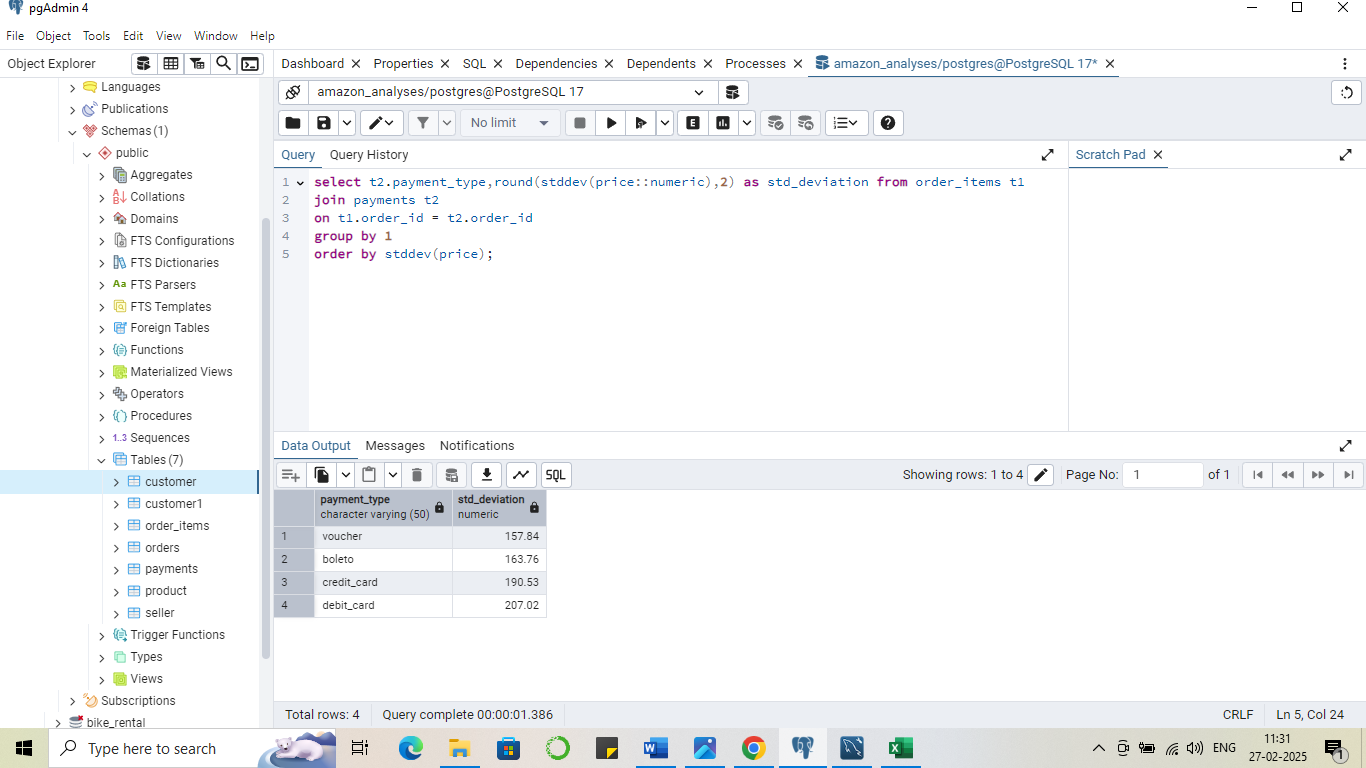
**Approach;**

**Identifying Relevant Tables**

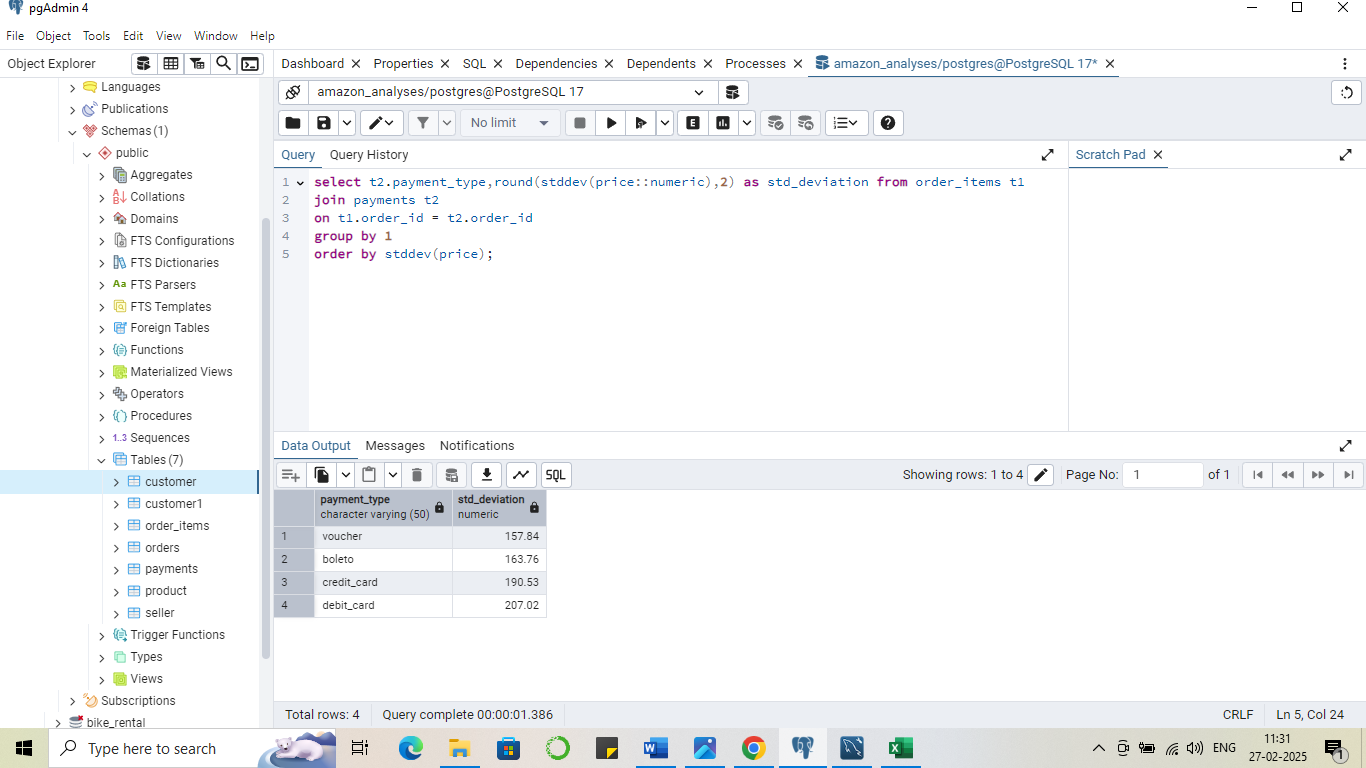
* order\_items table contains product prices for each order
* payments table contains payment details including payment type

**Defining Query logic**

* Join order\_items with payments using order\_id to associate payment types with transaction amounts
* Use the stddev(price::numeric) function to calculate the standard deviation of transaction amounts for each payment type
* Group by payment\_type to compute the standard deviation for each category
* Sort the results in ascending order of standard deviation to identify the most consistent payment types



**Output**



**Recommendations**

1. Promote Voucher Payments;- as voucher have the lower standard deviation and I should be encourage voucher-based promotions can help maintain stable transaction values.
2. Optimize Boleto Transactions;- as have slightly higher stddev so we can provide some discounts to make this consistent
3. Monitor Credit and Debit Card Transactions;- As they have the higher variance so we can analyses it and try to get some patterns which will help in making optimised pricing strategies.

**Problem Statement 7**

Amazon India wants to identify products that may have incomplete name in order to fix it from their end

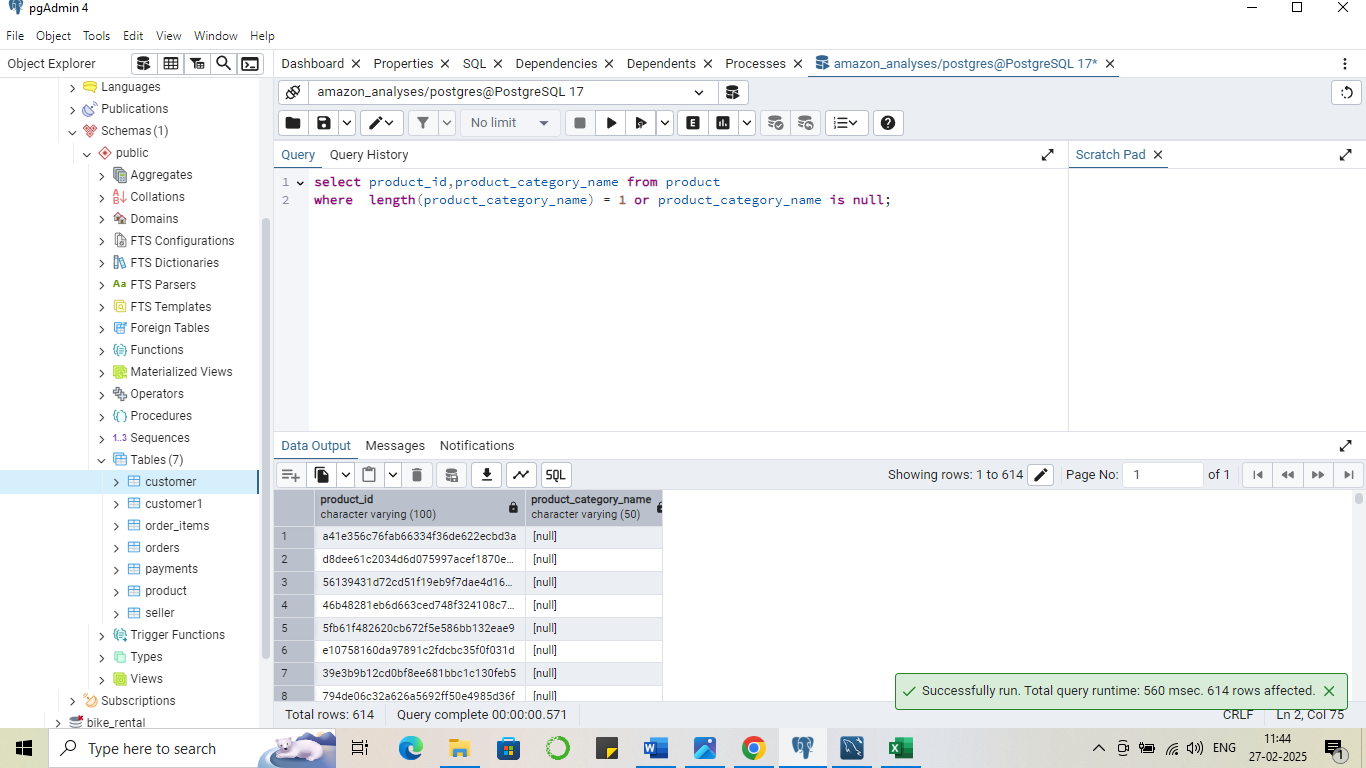
**Approach;**

**Identifying Relevant Tables**

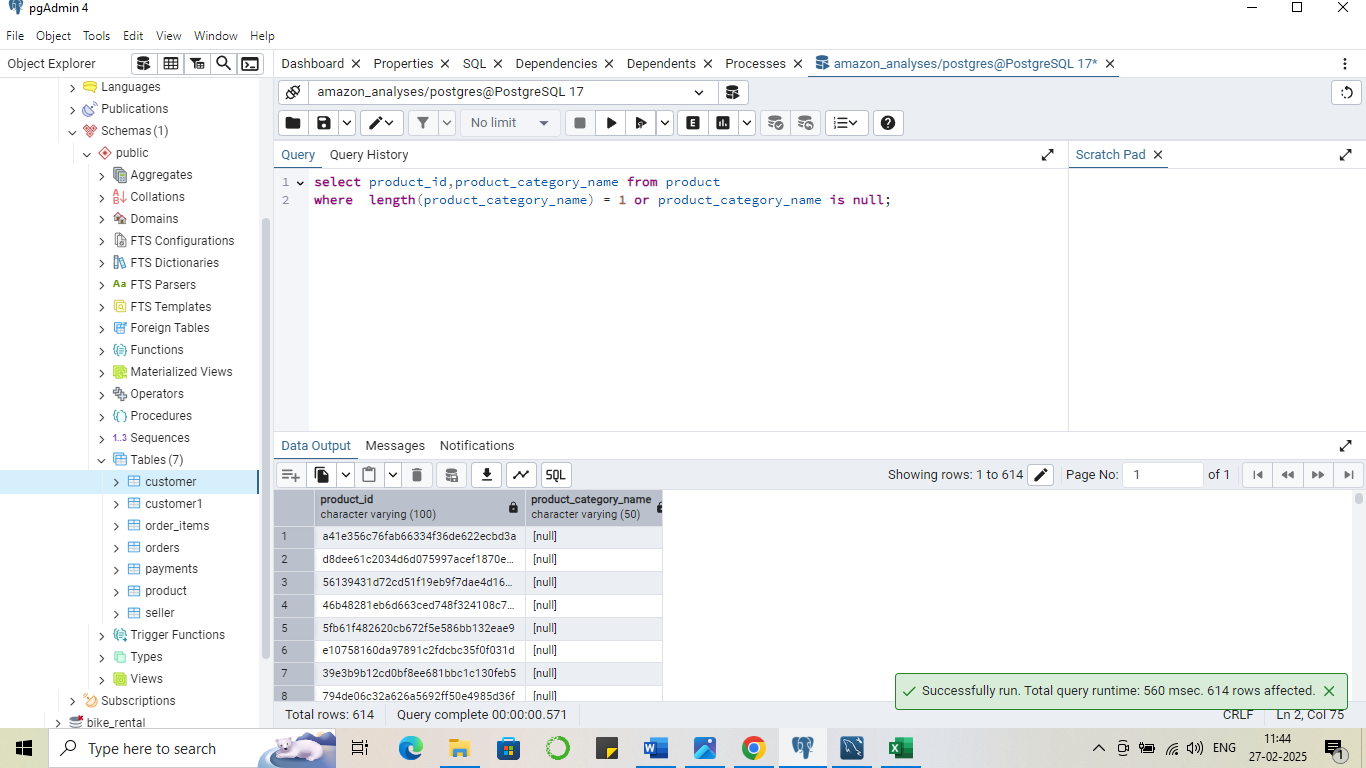
The product table is used since it contains product details, including product\_id and product\_category\_name

**Defining Query Logic**

* The length(product\_category\_name) = 1 condition helps identify categories with a single character, which may indicate incomplete or incorrect data.
* The product\_category\_name IS NULL condition detects missing values in the product category column.
* Filtering these values helps Amazon India locate and correct incomplete product names.



**Output**



**Analyses - II**

**Problem Statement 2.1**

Amazon India wants to understand which payment types are most popular across different order value segments (e.g., low, medium, high).

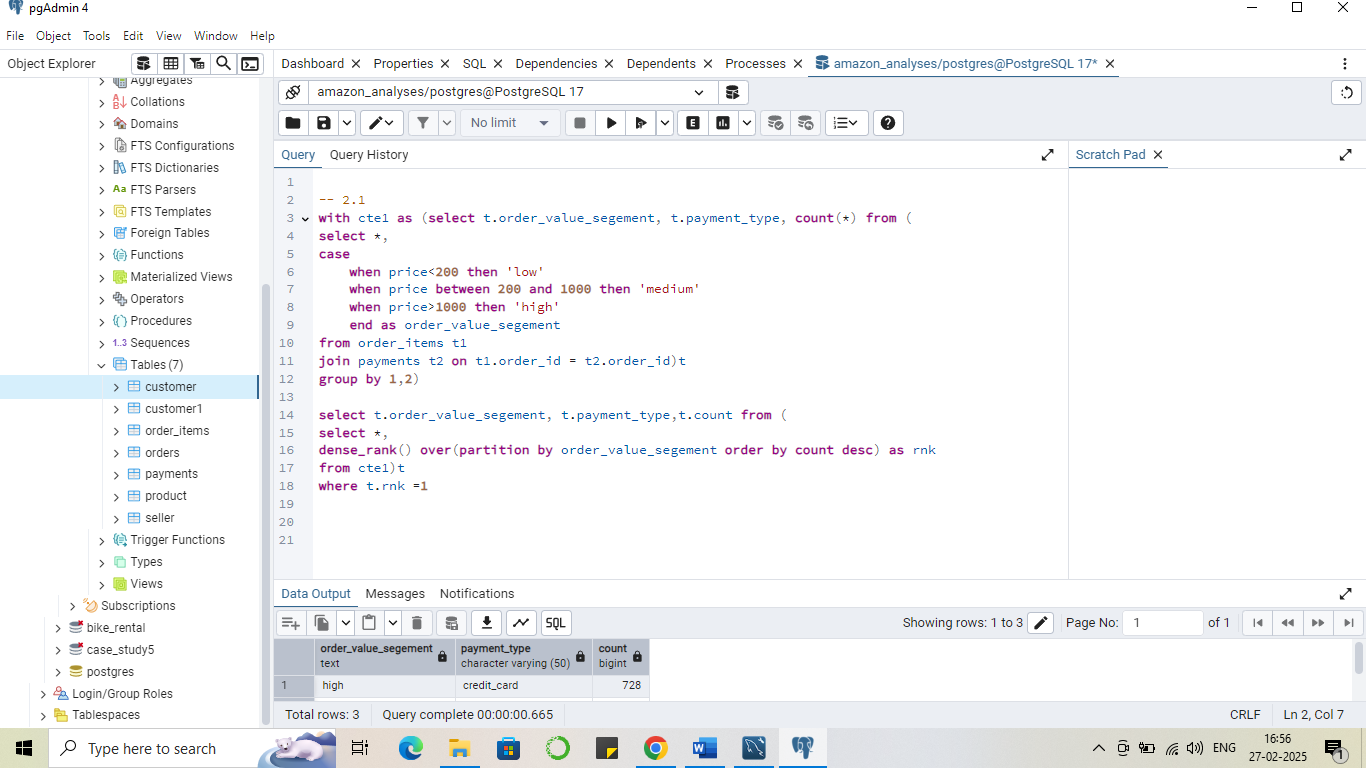
**Approach;**

**Identifying Relevant Tables**

* Payments table used here as it contain the informations about the payment option used by customers.
* Then use order items table to get values of different orders and segment them into low, medium, and high based on there values

**Defining Query Logic**

* At the first we will use case statement in the order items table to label all orders (low, medum and high) based on there order values .
* Then join this table with the payment table on order id column to get which payment option used for different orders.
* Then I use sub- query where I use count aggregation function based on the segment type and paymnet type for that used group by clause.
* Then use cte to make query look clear and then used dense rank to get the most frequent used payment type out of all by getting only 1st rank of all.
* Used partition by segment order by the count values as desc and then used condition to get the 1st one.



**Output**



**Recommendation**

1. Encourage Other Payment Methods;- So people could use other method as well

* Provide discounts, cashback, or reward points for payments made via debit cards, digital wallets, or UPI to balance payment method distribution.

1. Optimize Credit Card Offers

* Partner with banks to offer exclusive credit card deals, such as zero-interest EMIs or instant discounts on high-value purchases.

**Problem Statement 2.2**

Amazon India wants to analyse the price range and average price for each product category.

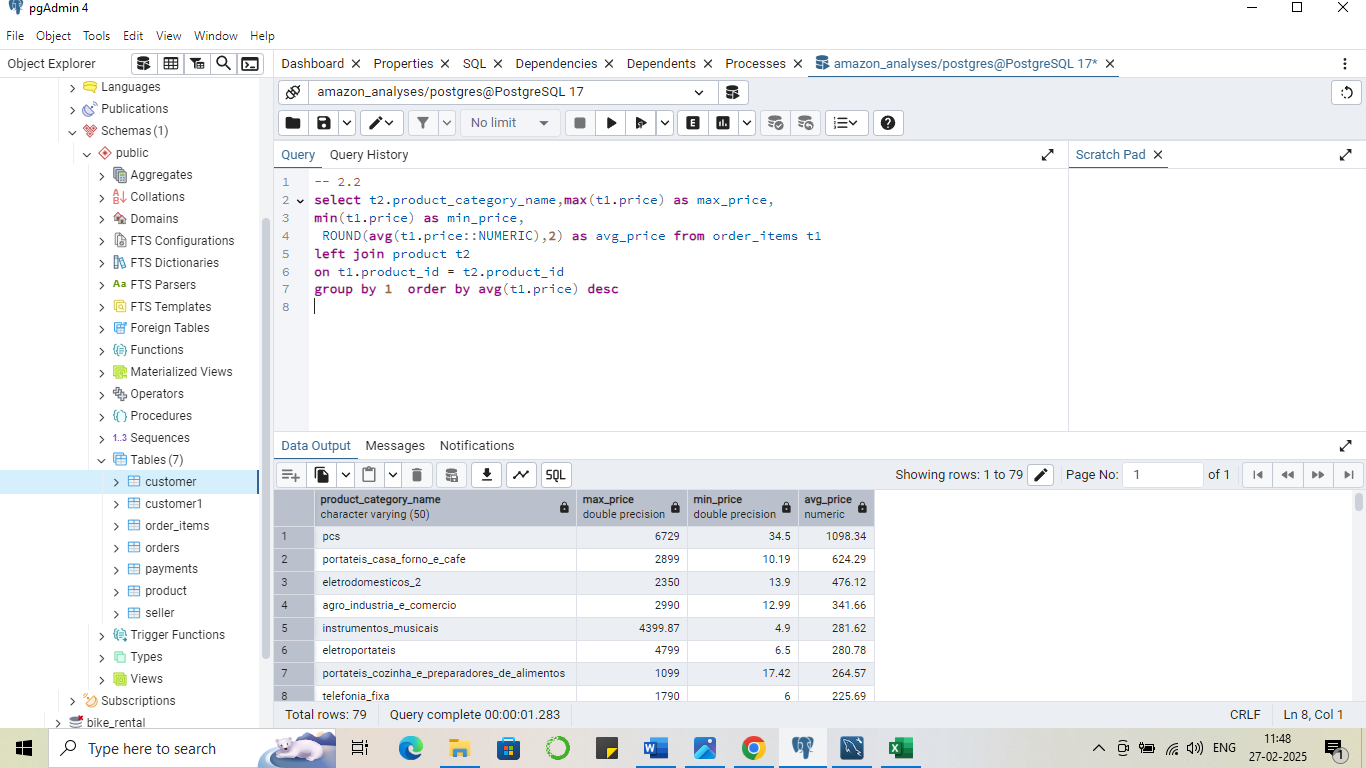
**Approach;**

**Identifying Relevant Tables**

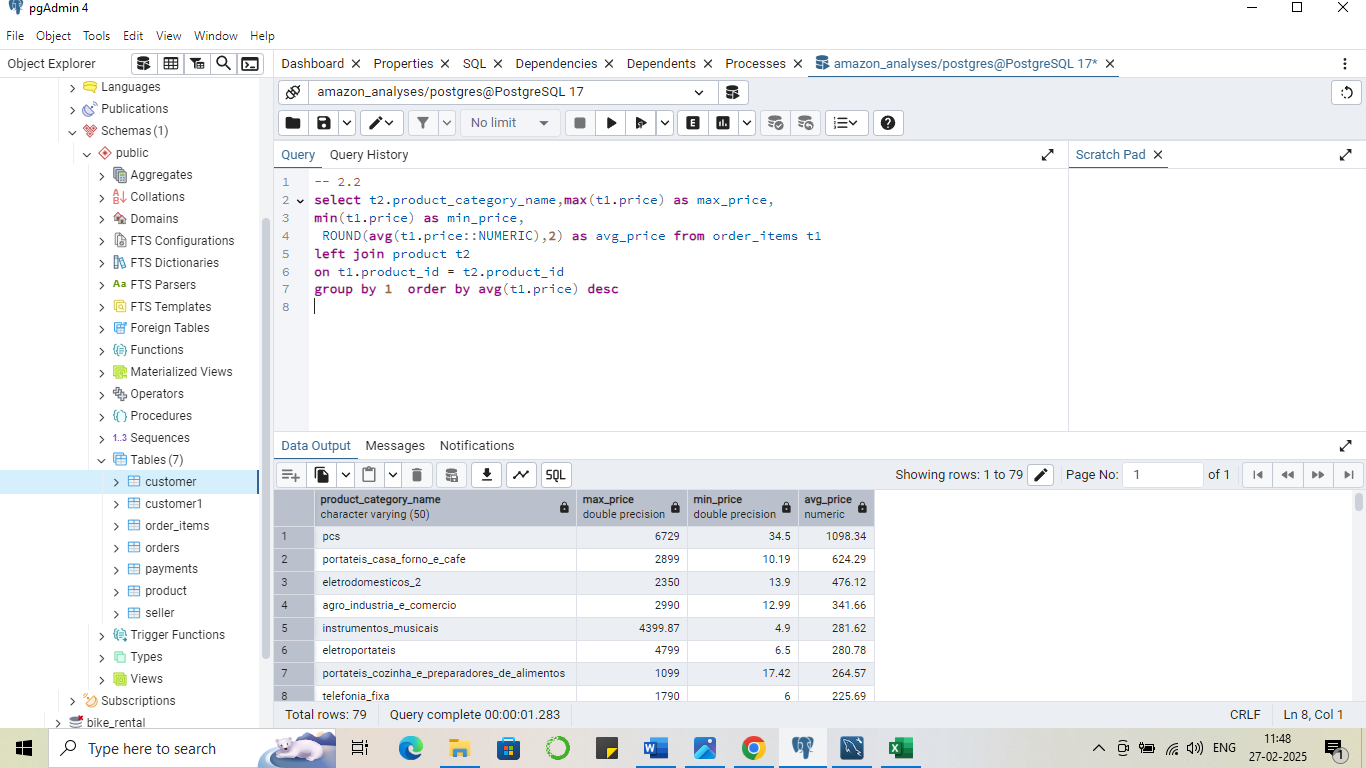
* The order\_items table contains price details for each product.
* The product table provides category information for each product.
* The product\_id column is the key to joining both tables.

**Defining Query Logic**

* Perform a LEFT JOIN between order\_items and product on product\_id to get category names.
* Group the data by product\_category\_name to analyze pricing at the category level.  
  Calculate maximum price using MAX(price) and minimum price using MIN(price) for each category.
* Compute the average price using AVG(price::NUMERIC), rounding it to two decimal places for clarity.
* Order the results by average price in descending order using ORDER BY avg\_price DESC to highlight high-value categories first.



**Output**



**Recommendations**

1. Optimize Pricing Strategy Based on Category Demand

* High-Average Price Categories (e.g., "pcs", "portateis\_casa\_forno\_e\_cafe") - Offer seasonal discounts or installment payment options to increase conversions.
* Low-Average Price Categories (e.g., "flores", "fraldas\_higiene")- Implement bundling strategies.

**Problem Statement 3**

Amazon India wants to identify the customers who have placed multiple orders over time.

* The orders table contain the information of the order and the customer who placed that order

**Defining Query Logic**

* First select distinct customer from the orders table
* Then used group by to get the count of distinct order placed by each customer
* And then using having clause used condition that where there is count greater then 1 for final output.



**Output**

and more..

**Recommendation**

1. Implement a Loyalty & Rewards Program
2. Personalized Marketing & Retargeting;- Notify customers about restocks or upgrades of previously bought products. Or recommendation passed on past order.
3. Exclusive Deals for High-Value Customers- Personalised discounts

**Problem Statement 4**

Amazon India wants to categorize customers into different types ('New – order qty. = 1' ;  'Returning' –order qty. 2 to 4;  'Loyal' – order qty. >4) based on their purchase history.

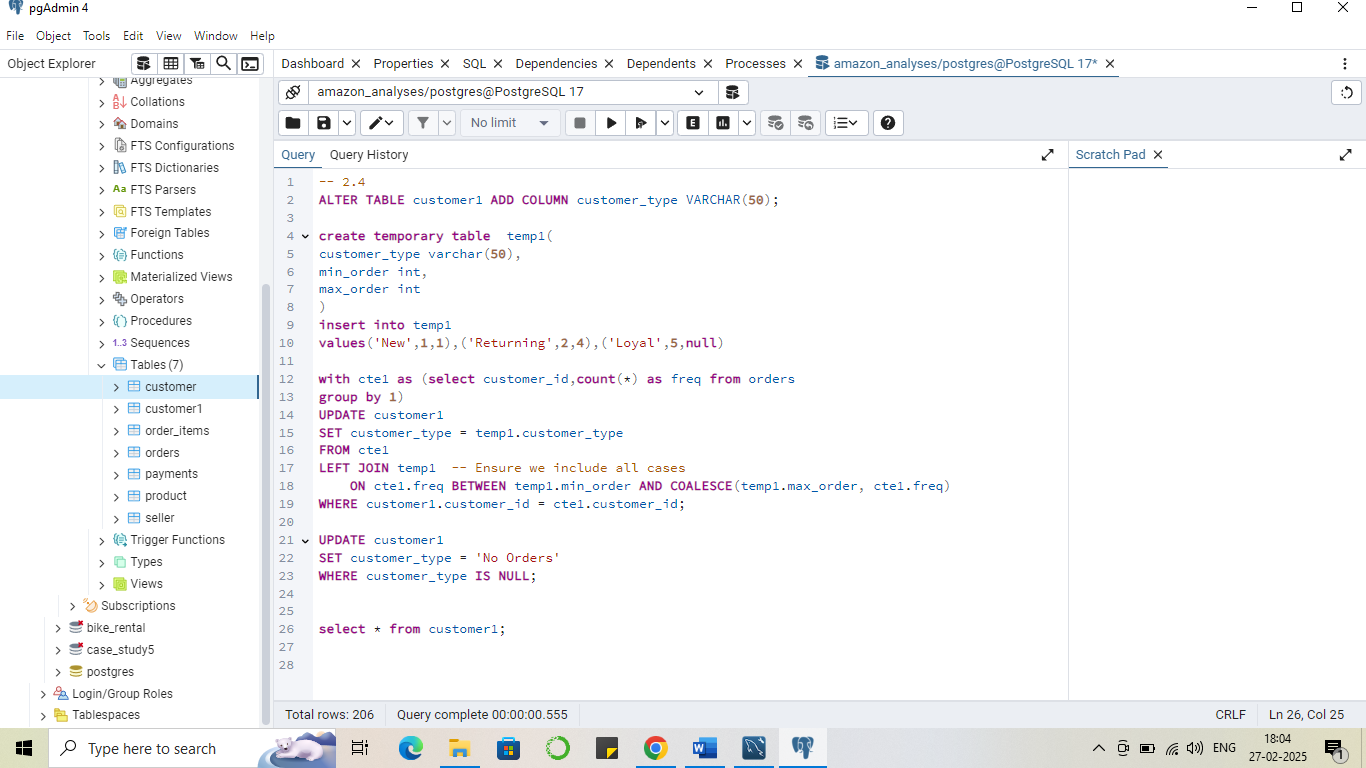
**Approach;**

**Identifying Relevant Tables**

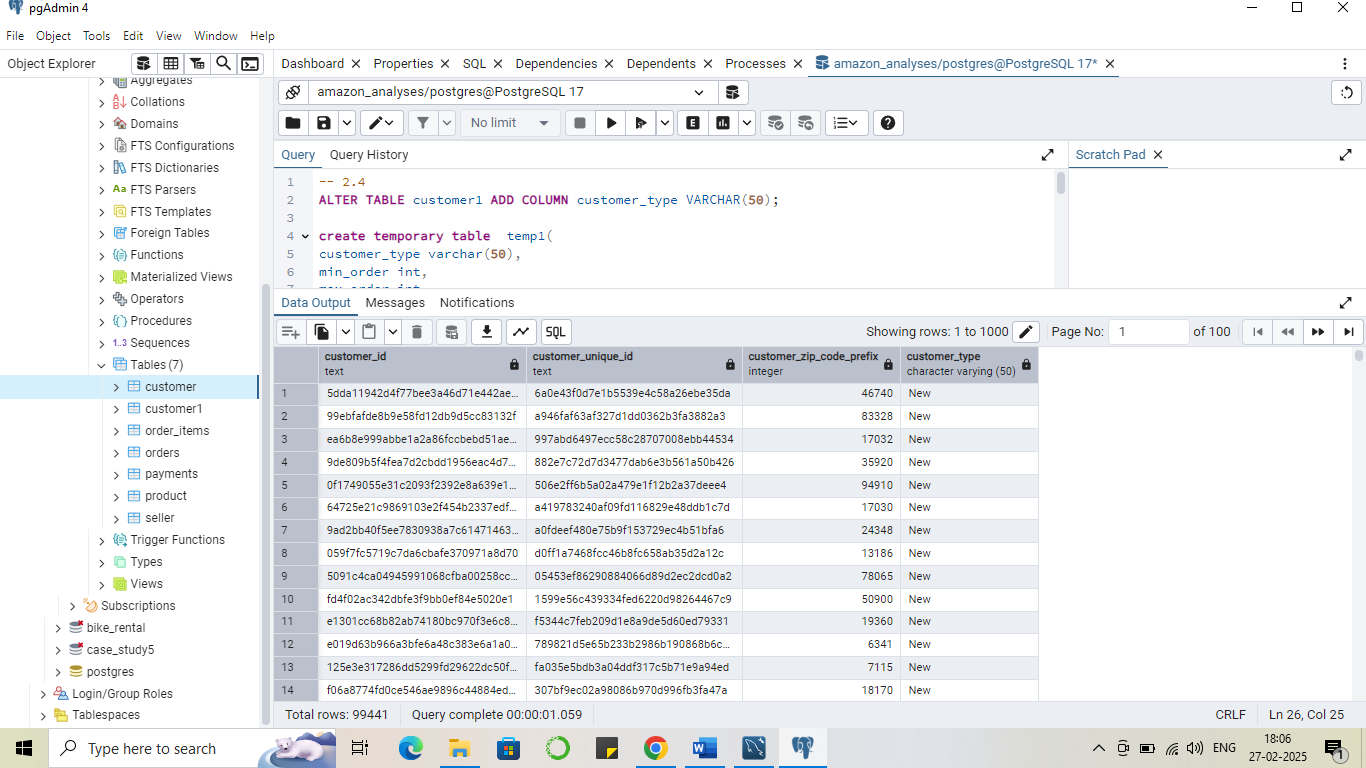
* Customer table as this provide information about their id and frequency of there order

**Defining Query Logic**

* First created a temporary table to get the min and max order frequency to get customer type.
* Then inserted the values in that temporary table.
* Created a cte where I stored the customer\_ id and there order frequency
* Before doing update I alter and add a column name customer type column in the table.
* Now I use update command where I set cus. type to the values in the temporary table based on the condition(frequency of order).
* For that I have to join this update command with temporary table to get the condition values.
* Then there are some customer who never ordered so I update them as no ordered.



**Output**

Table got updated   
and more..

**Problem Statement 5**

Amazon India wants to know which product categories generate the most revenue.

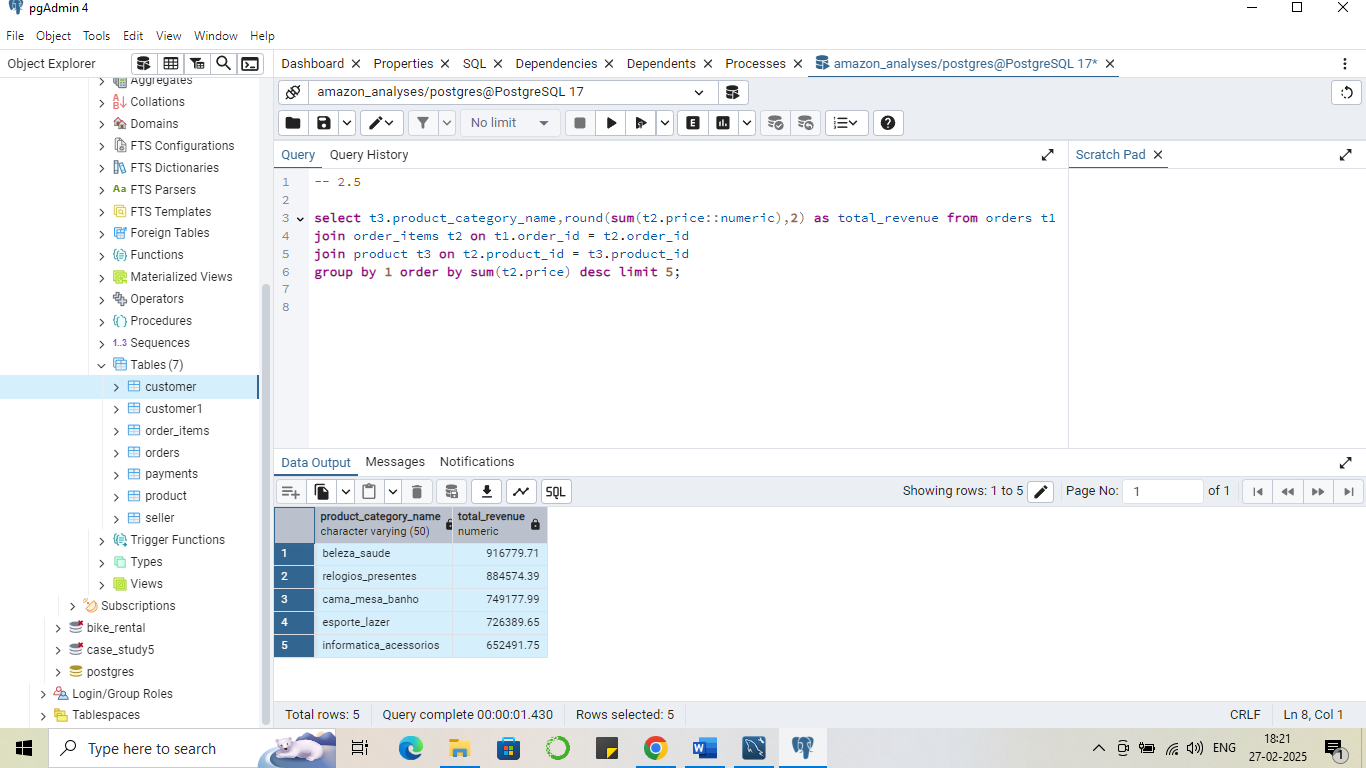
**Approach;**

**Identifying Relevant Tables**

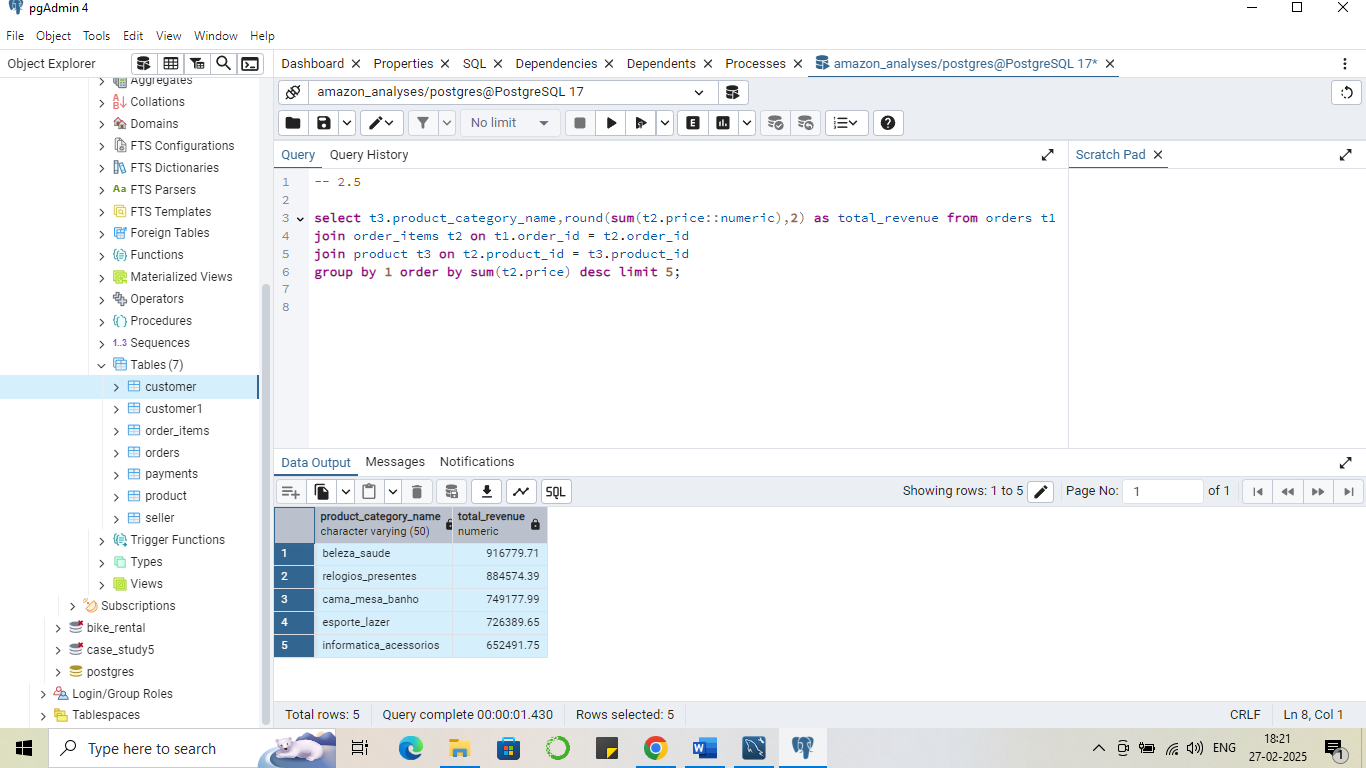
* Order table used to get the information about the order and there product id
* Then order items table help to get the price of the order which help in revenue finding
* And then use product table to get product category name

**Defining Query Logic**

* First take order table to get the information of product via order id
* Then join this with order item to get the price of order which will help to get the revenue
* Also join product table on product id to get the category name
* At last we use group by on category and sum the price into 2 decimal and extract only these two variable
* Then order by the sum values in desc and take top 5 categories



**Output**



**Analyses – III**

**Problem Statement 1**

The marketing team wants to compare the total sales between different seasons.

**Approach;**

**Identifying Relevant Tables**

* Order table used to get the order table which would help to get season
* And order item table used to get price of order

**Defining Query Logic**

* First I get the month name from the column ‘order\_purchase\_timestamp’ in orders table
* Along with this join the table with order items to get price of different orders
* Then use case statement and month names and uses sub-query to get the season out of it and store it in a cte for clear view.
* Then use group by to get sum values of price based on different seasons



Output



**Problem Statement 2**

The inventory team is interested in identifying products that have sales volumes above the overall average.

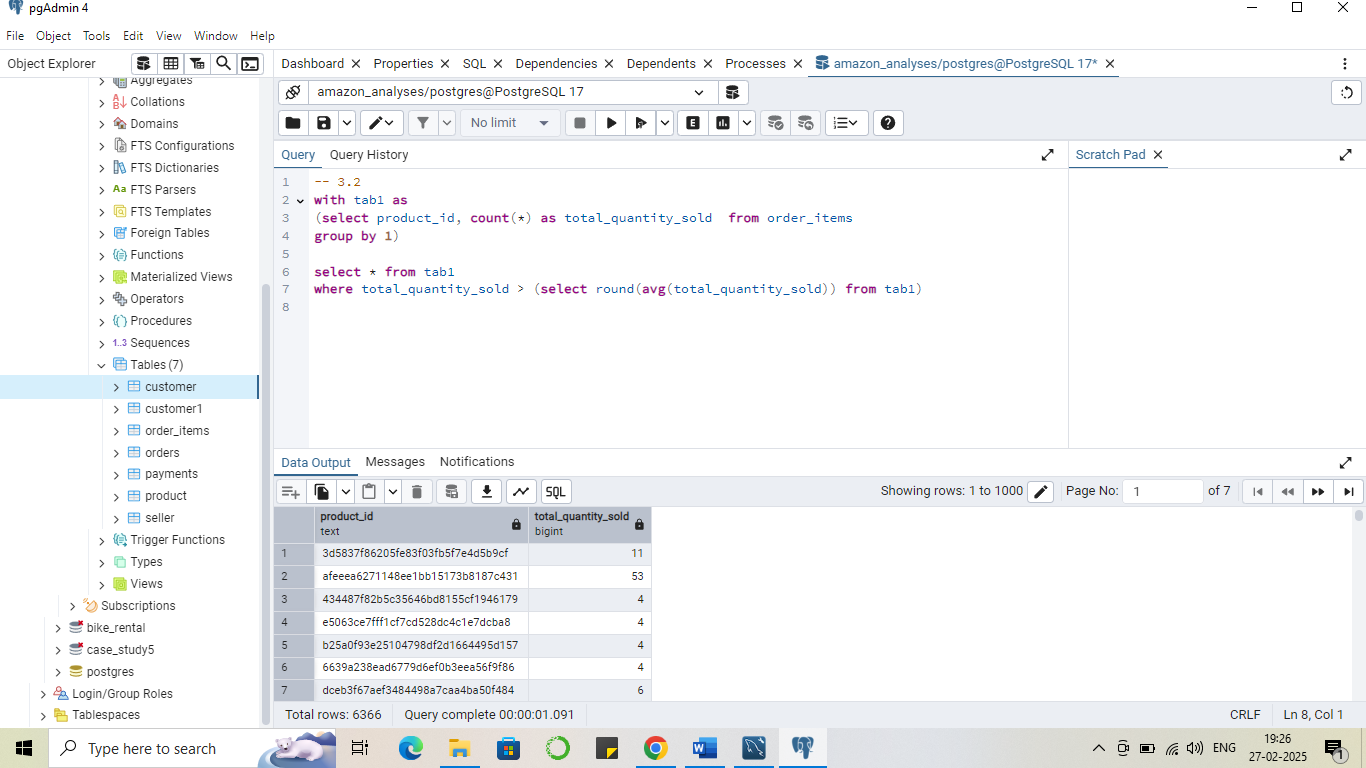
**Approach;**

**Identifying Relevant Tables**

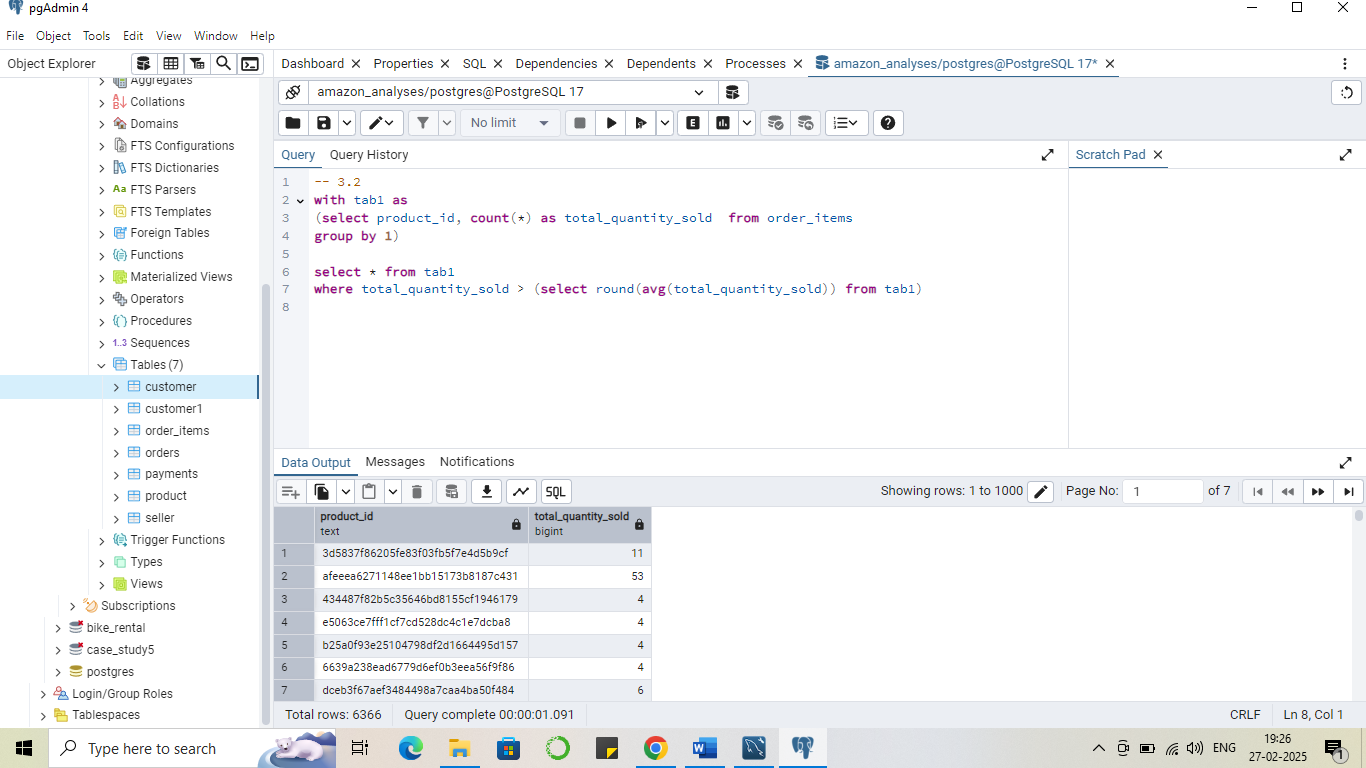
* Order items table used to get the product id and the count of product sold

**Defining Query Logic**

* First we use order items table to get the product id and count of product sold using group by here
* Put the query into cte
* Then use cte data and filter it using where clause to get product sales greater then then the avg of all product unit for that used subquery



Output

more..

**Recommendations**

1. Can manager a inventory according to the higher selling product
2. Can use bundles or use discounting method for the lower sales one

**Problem Statement 3**

To understand seasonal sales patterns, the finance team is analysing the monthly revenue trends over the past year (year 2018)

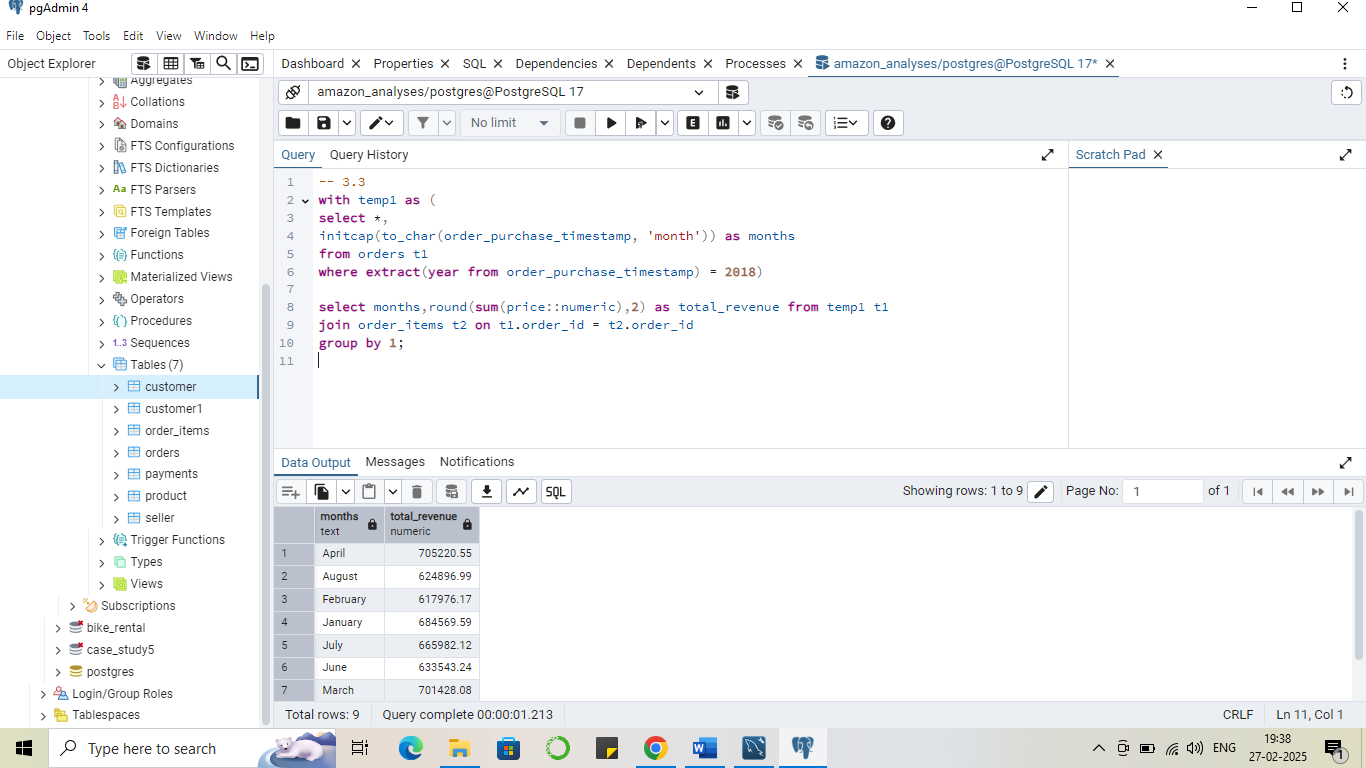
**Approach;**

**Identifying Relevant Tables**

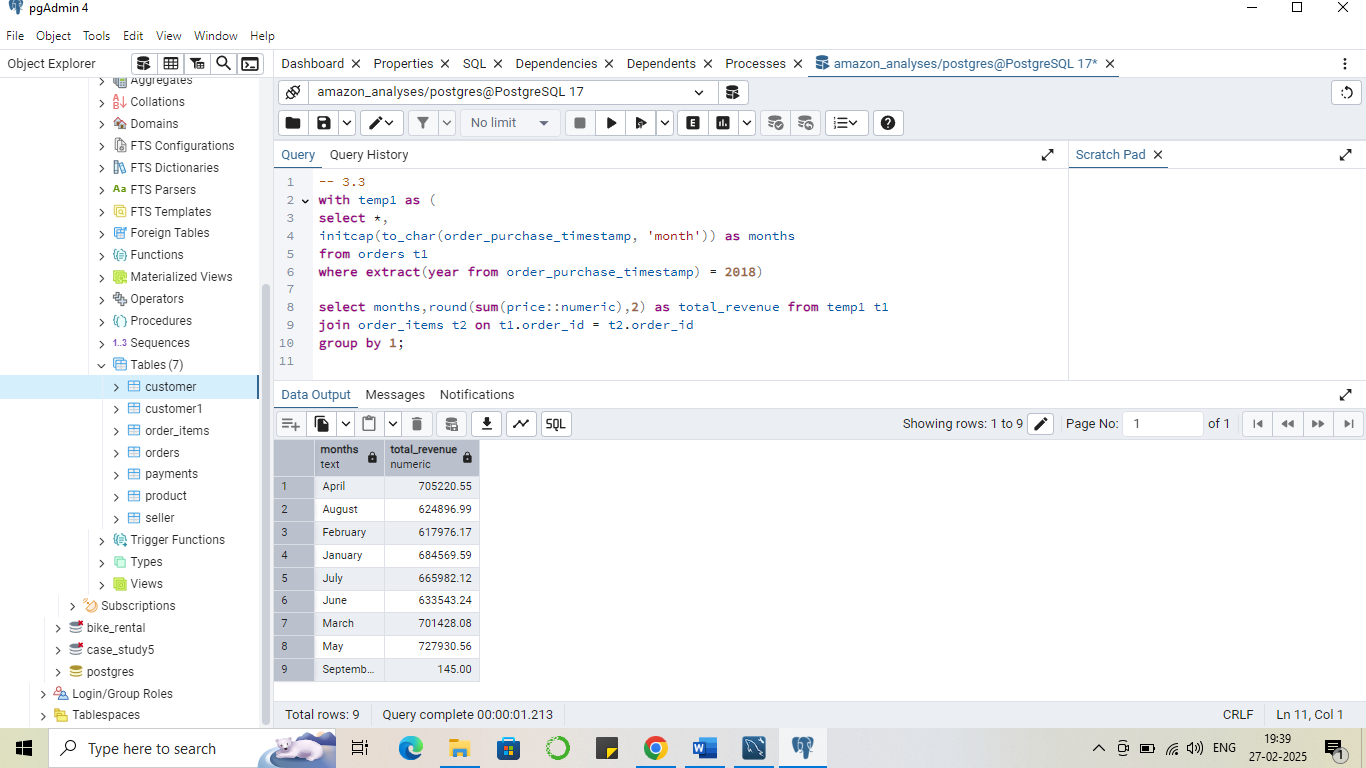
* Order table use to get the month from the date of order
* Order items table use to get the price of different orders

**Defining Query Logic**

* Use order table from the column ‘order\_purchase\_timestamp’ we can get month name and also used where condition to get into data where year is 2018
* Put that into cte
* Then group by based on month to get month by revenue using sum function



Output



**Recommendation**

1. Strengthen Sales During Peak Months- Offer flash sales or exclusive deals to maximize revenue during high-demand periods.
2. Investigate the September Revenue Drop
3. Leverage Seasonal Trends for Better Inventory & Marketing

**Problem Statement 4**

A loyalty program is being designed  for Amazon India.

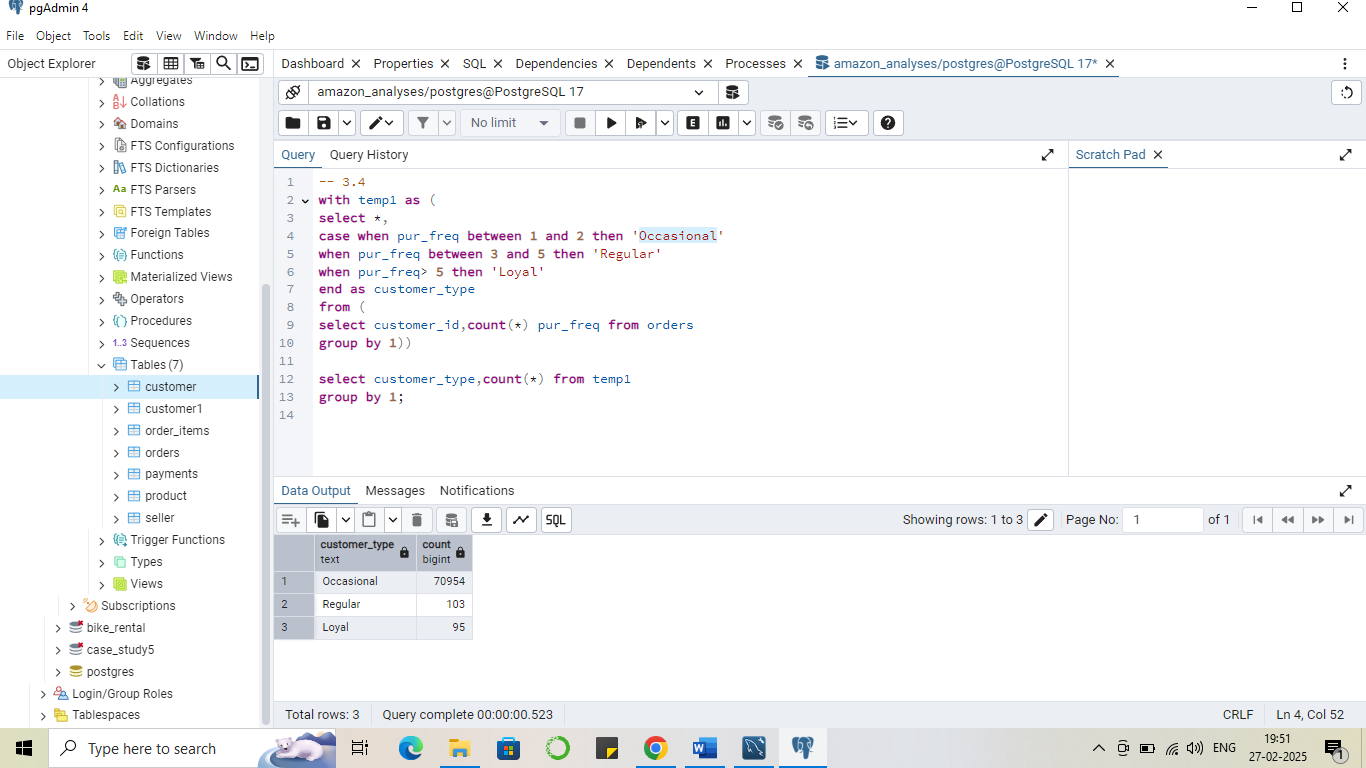
**Approach;**

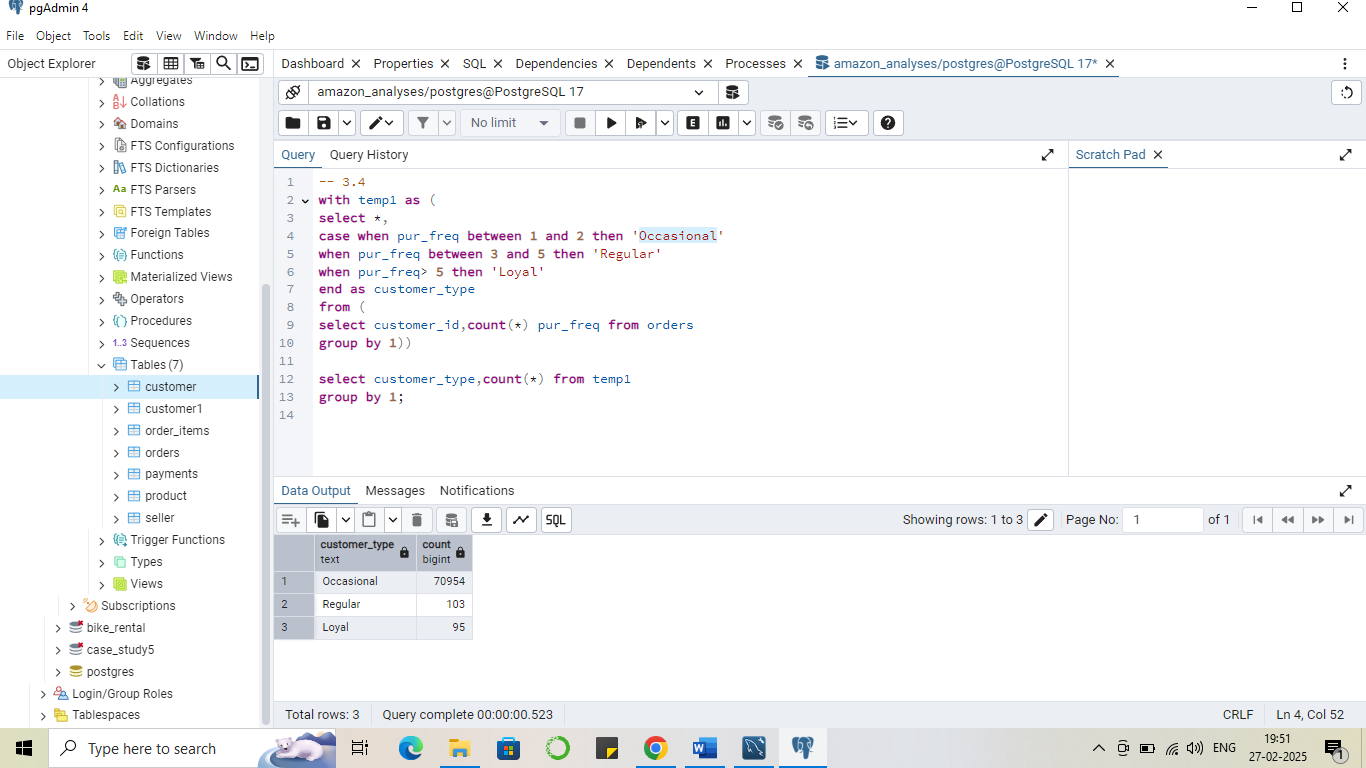
**Identifying Relevant Tables**

* Use orders table to get customers and there purchase frequency

**Defining Query Logic**

* First I take the orders table where I group the table based on customer and there purchase frequency
* Then used case statement to categorise the customers based on purchase frequency like Occasional, regular and loyal
* Then store query into cte
* Then used group by to group data based on customer type and count there values



Output 

**Problem Statement 5**

Amazon wants to identify high-value customers to target for an exclusive rewards program.

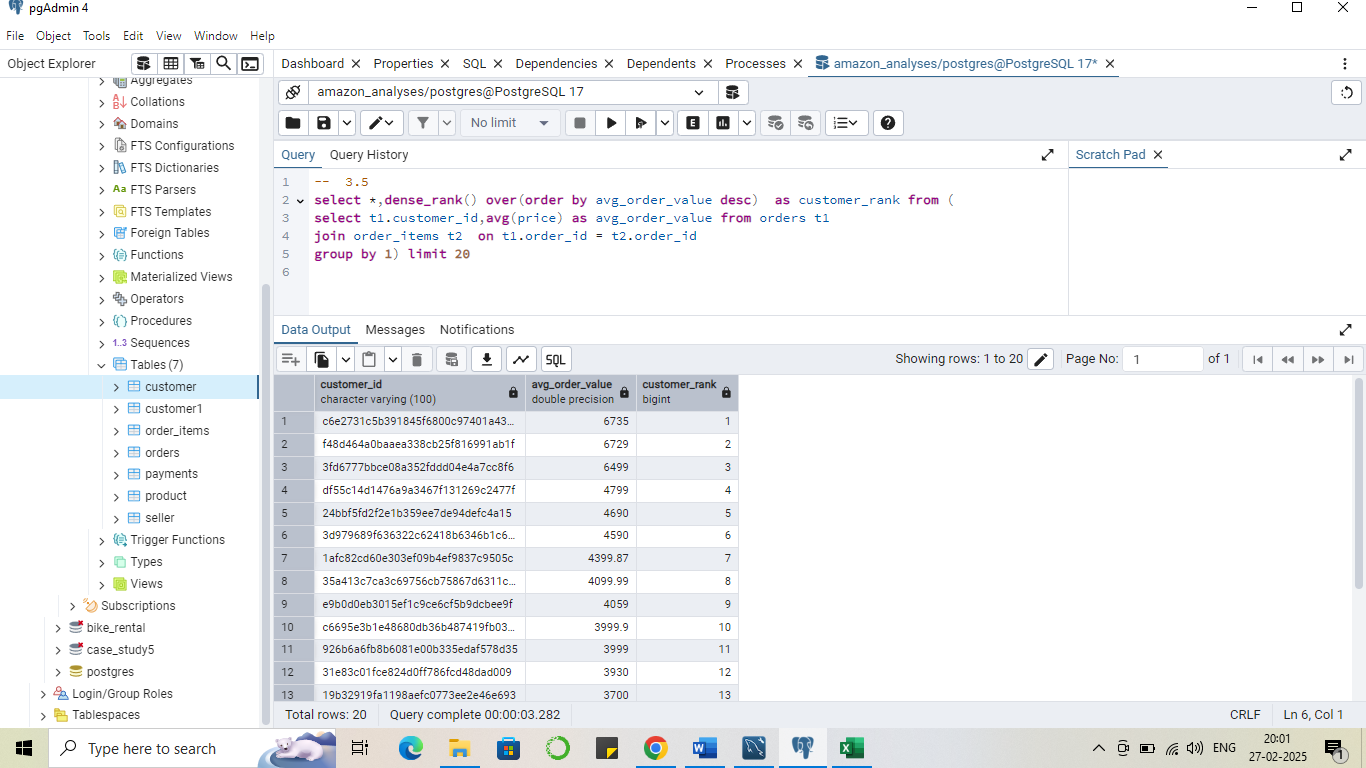
**Approach;**

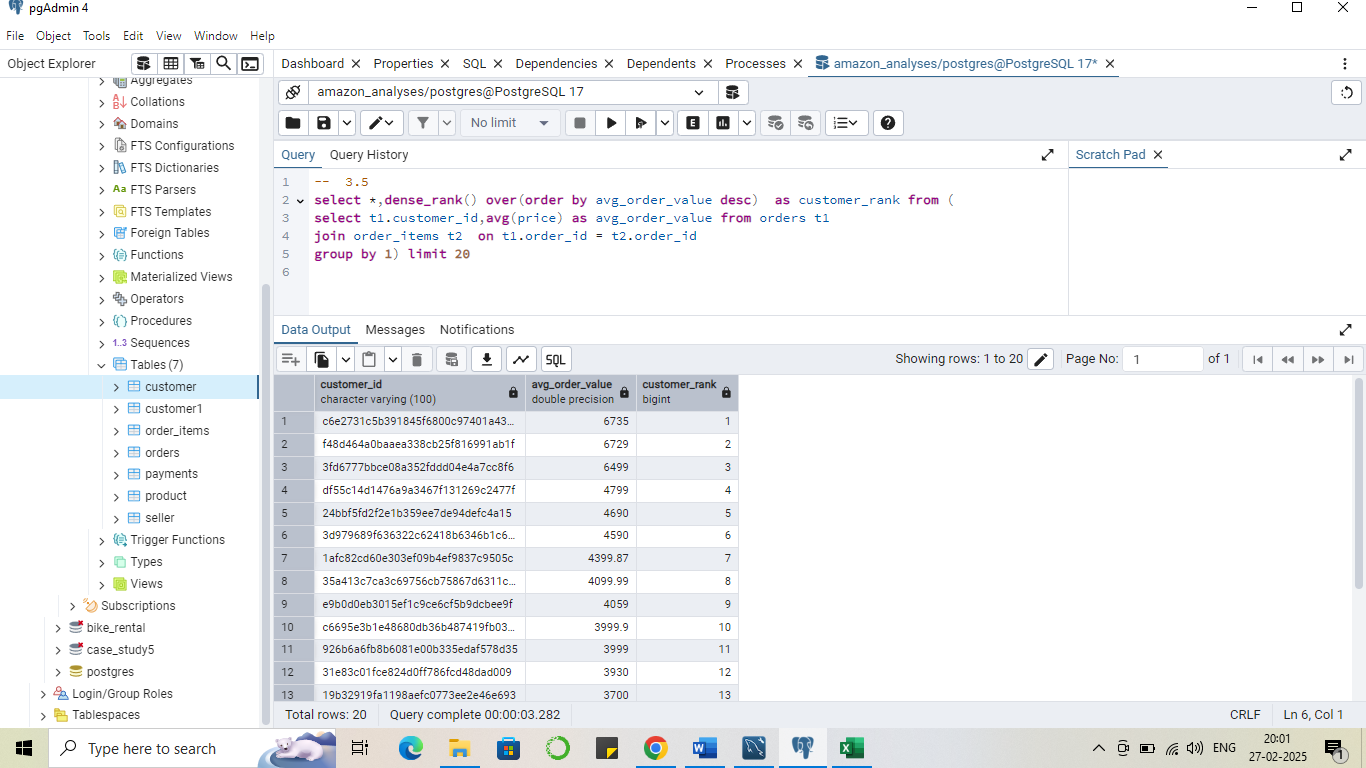
**Identifying Relevant Tables**

* Used order table to get the customer id
* Used order items table to get the price of all orders

**Defining Query Logic**

* Use order table to get the customer id
* Then use order items to get price of all orders
* Use group by to get the average of price of all orders
* Then using subquery get the dense rank function to rank the customer in desc based on the average order price
* And take the top 20 one



more..

**Recommendations**

1. Implement a Loyalty & Rewards Program
2. Personalized Marketing & Retargeting;- Notify customers about restocks or upgrades of previously bought products. Or recommendation passed on past order.
3. Exclusive Deals for High-Value Customers- Personalised discounts

**Problem Statement 6**

Amazon wants to analyze sales growth trends for its key products over their lifecycle.

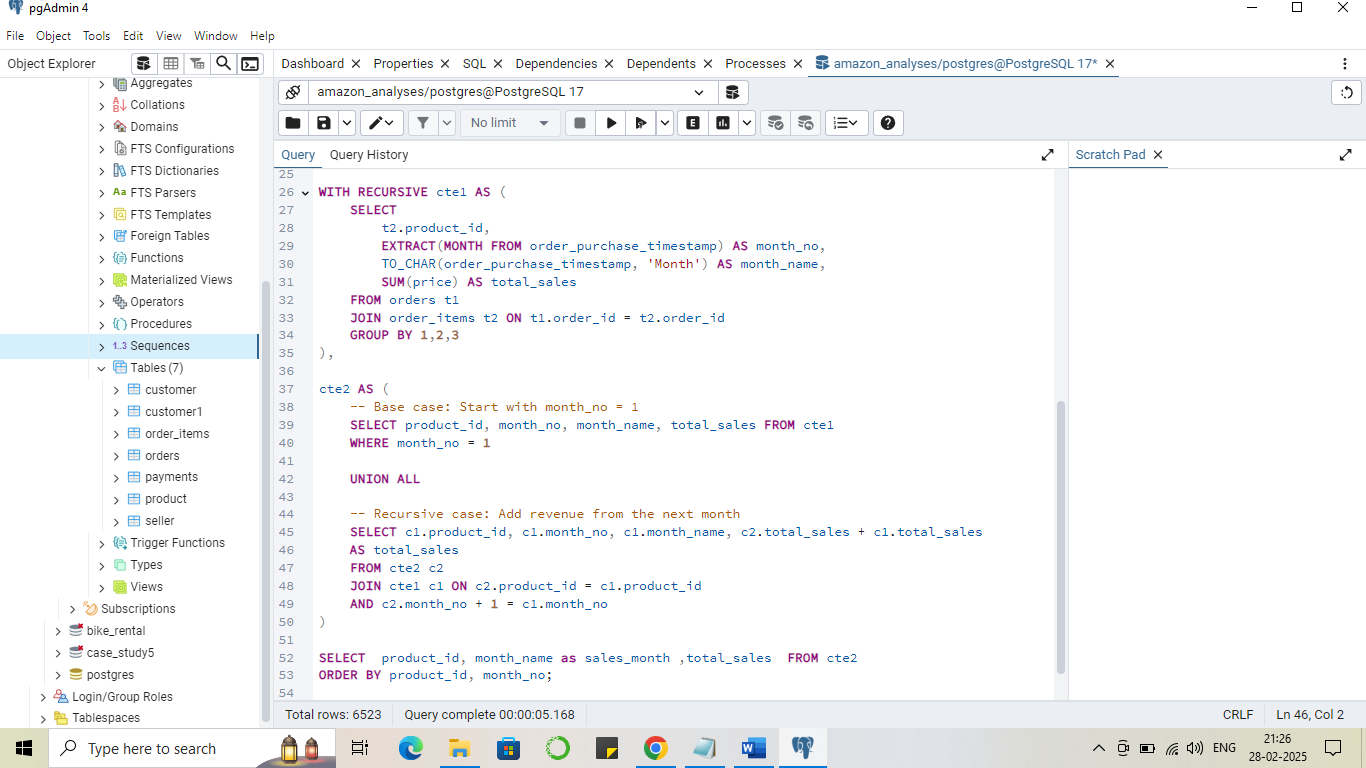
**Approach;**

**Identifying Relevant Tables**

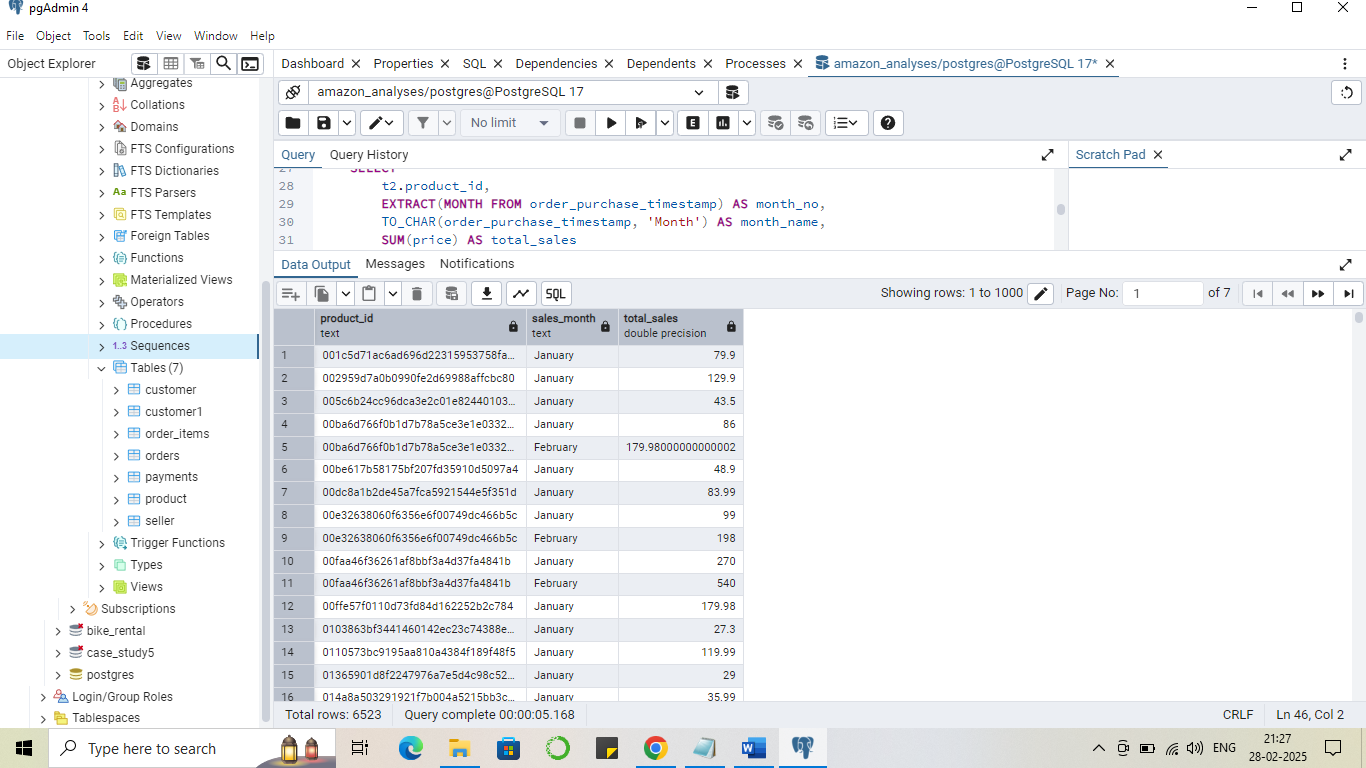
* Order table to get the order date month from date
* Then use order items table to get price of different orders
* Use product table to get the product id

**Defining Query Logic**

* Compute Monthly Revenue per Product (cte1)
* Recursive CTE to Calculate Cumulative Sales (cte2)
* The recursion ensures that each month’s revenue is added to the previous month’s total.
* Retrieve Final Cumulative Sales Data



**Output (sample)**



**Problem Statement 7**

To understand how different payment methods affect monthly sales growth, Amazon wants to compute the total sales for each payment method and calculate the month-over-month growth rate for the past year (year 2018).

**Approach;**

**Identifying Relevant Tables**

* Orders table top get the month of order
* Order item table to get the sales price
* Payment table to get the payment type for various orders

**Defining Query Logic**

* Firstly from the orders table use column ‘order\_purchase\_timestamp’ to get the month of order
* Then join order item table to get the price of different orders
* And then join payment table to get payment type for various orders
* Then I filter my data of 2018 using where clause
* Then I group my data based on payment type and month number and sum the price value
* Then stored it into cte
* Then I have to get the percentage change from previous month payment type wise so I use window function lag with partition by payment type and order by month to get that
* Formula used = ((current month – previous month)/ previous month)\*100.0



Output

