Executed SQL queries for inventory analysis of Beverage store

This project is a data-driven approach using intermediate-level SQL queries to explore the inventory dynamics of a beverage store prioritizing products based on revenue contribution and sales volume.

DATA -The data source for this project is <https://www.kaggle.com/datasets>. The data is cleaned using EXCEL.

TOOL USED- POSTGRE SQL /PgADMIN4

**OVERVIEW-**

A beverage store aims to identify the products that drive the highest profitability and contribute significantly to overall revenue streams. It wants to allocate resources effectively, streamline product portfolios, and tailor marketing strategies to focus on high-performing products.

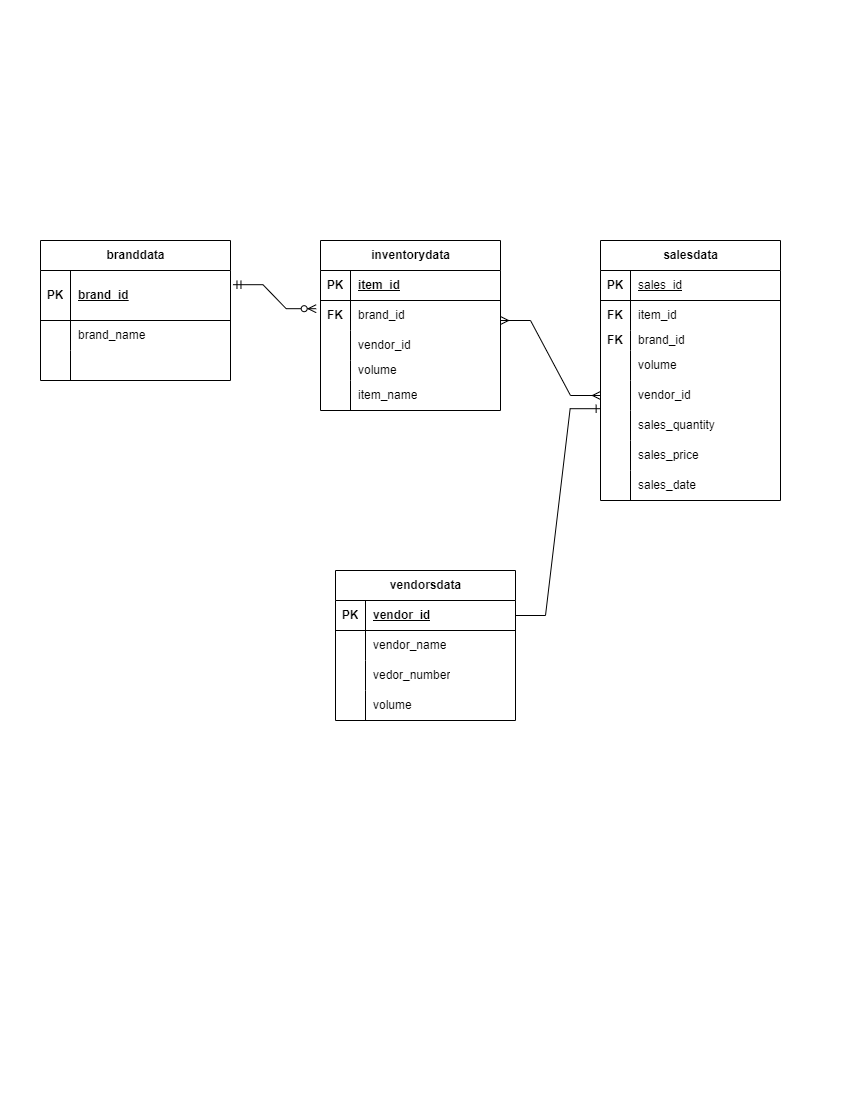
The primary objective is to optimize inventory management through a data-driven approach. Using database and leveraging SQL queries to explore inventory dynamics, prioritize products based on revenue contribution and sales volume, and analyze brand performance.

**PROBLEM STATEMENT –** To identify and analyze products contribution to revenue.

**LEARNING -**Identifying the products performance and their contribution to revenue for inventory management.

The detailed explanation of each query along with its purpose and learnings is explained at the end of each query.

SCHEMA



Database of this project is named as inventory\_database, Which is created in POSTGRE SQL .

Explanation of 4 Tables created in database:

1. branddata Table:

Columns:

brand\_id: Unique identifier for each brand (Primary Key).

brand\_name: Name of the brand.

Usage:

This table stores information about different brands. It serves as a reference table for associating brands with inventory items and sales data. It helps in analyzing inventory and sales performance based on brand categories.

2. vendorsdata Table:

Columns:

vendor\_id: Unique identifier for each vendor (Primary Key).

vendor\_name: Name of the vendor.

vendor\_number: Vendor's contact number.

volume: Total volume associated with the vendor.

Usage: It provides details about vendors' contact information and their associated volumes.

3. inventorydata Table:

Columns:

item\_id: Unique identifier for each inventory item (Primary Key).

brand\_id: Foreign key referencing the branddata table, indicating the brand to which the item belongs.

volume: Total volume of the item in inventory.

item\_name: Name of the inventory item.

4. salesdata Table:

Columns:

sales\_id: Unique identifier for each sales transaction (Primary Key).

item\_id: Foreign key referencing the inventorydata table, specifying the sold item.

brand\_id: Foreign key referencing the branddata table, indicating the brand of the sold item.

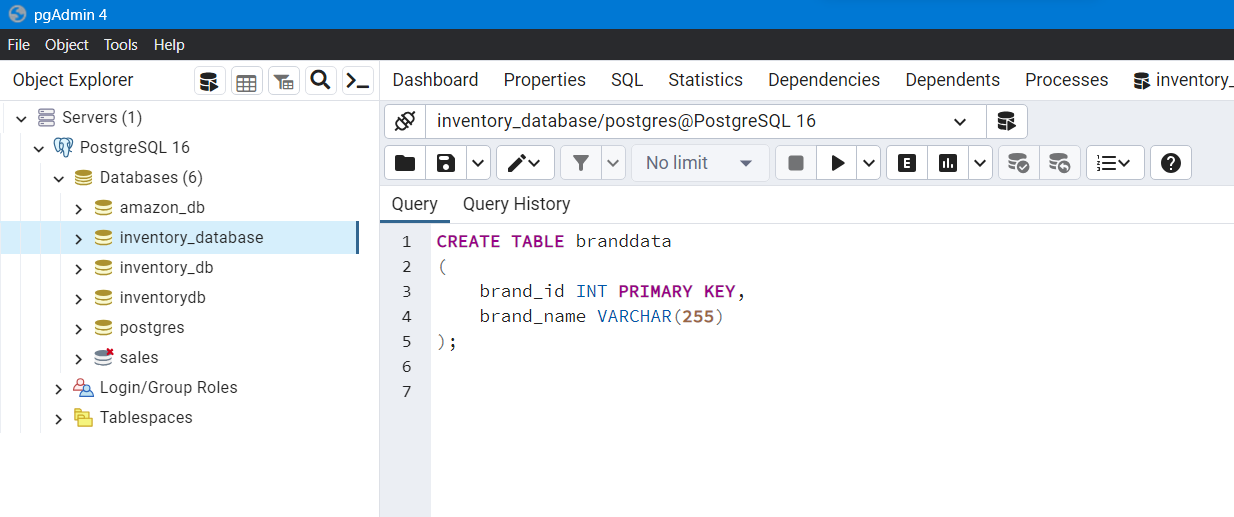
volume: Total volume of the sold item.

sales\_quantity: Quantity of items sold in the transaction.

sales\_price: Price per item in the transaction.

sales\_date: Date of the sales transaction

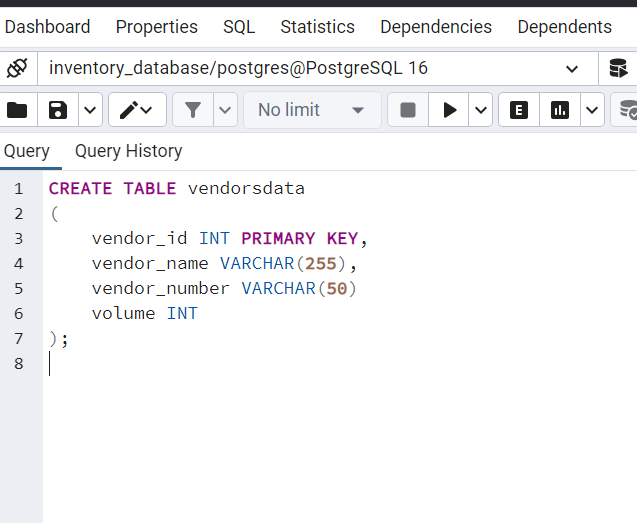
CREATING A TABLE branddata in database inventory\_database using the query in POSTGRE SQL



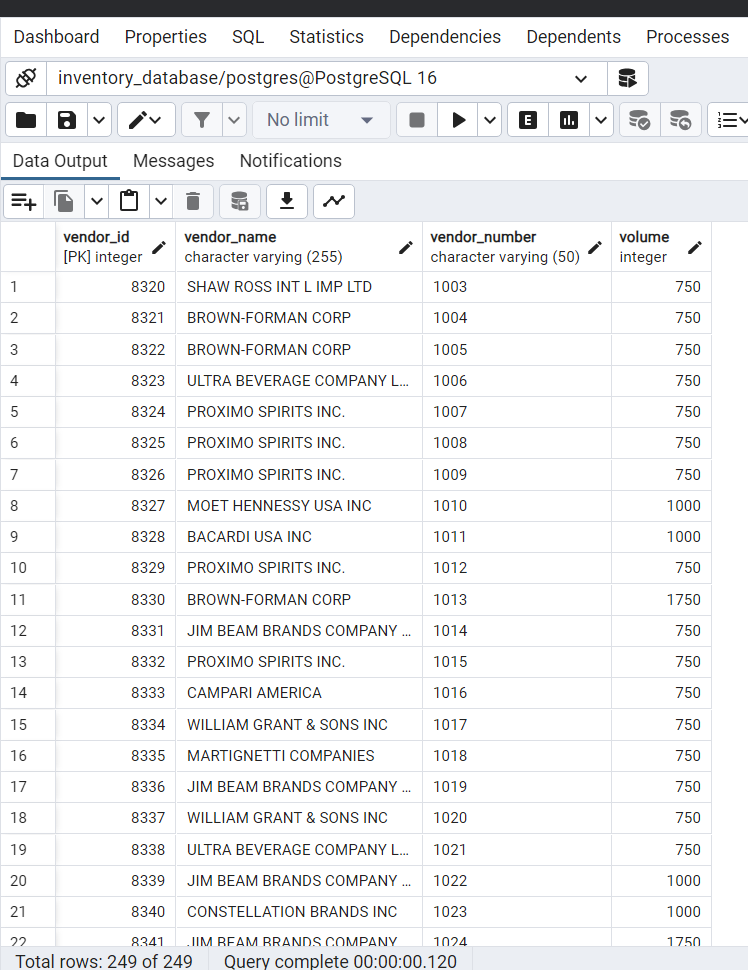
Result after the data is imported in table branddata



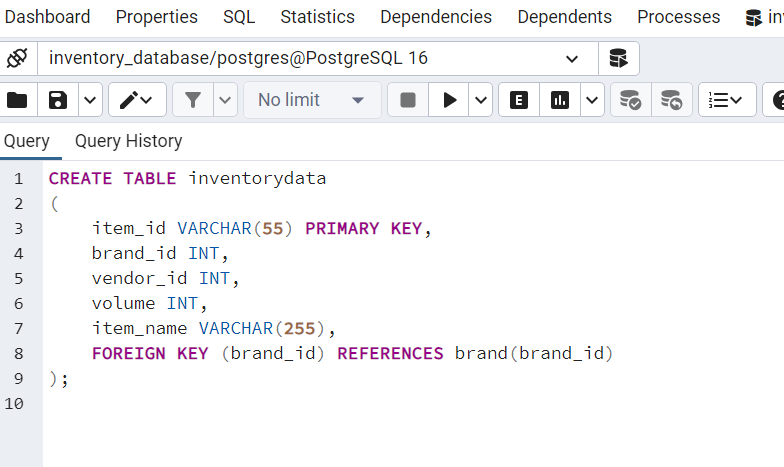
CREATING A TABLE vendorsdata in inventory\_database using the query in POSTGRE SQL



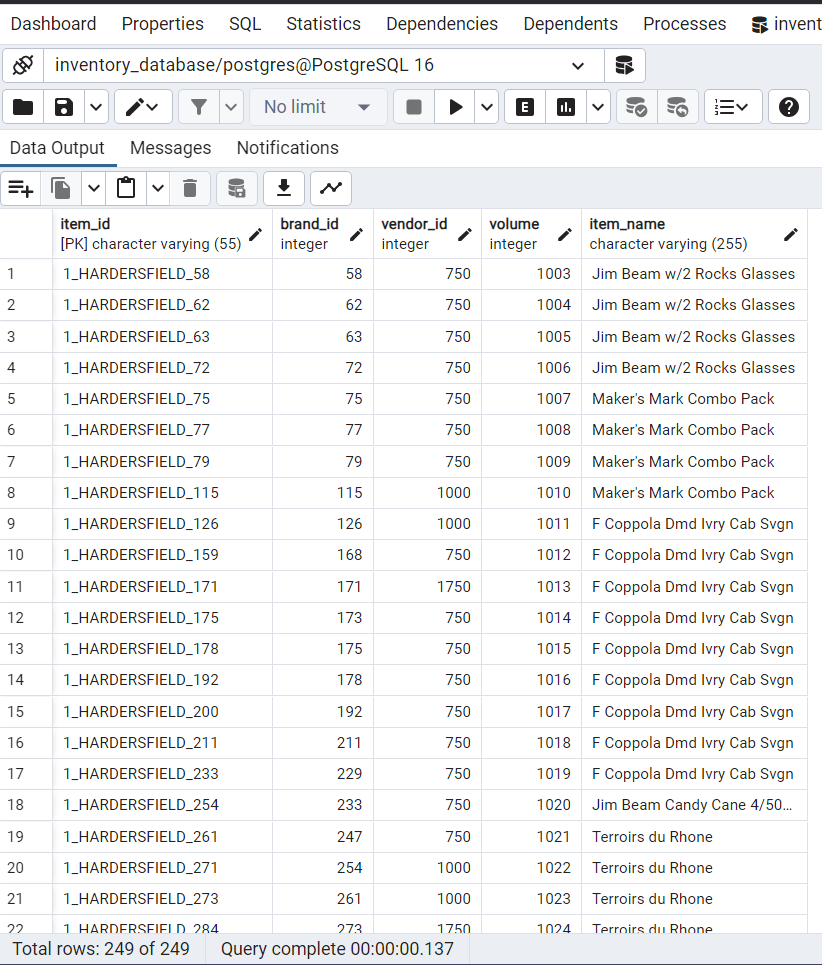
Result after data is imported in table vendorsdata

=

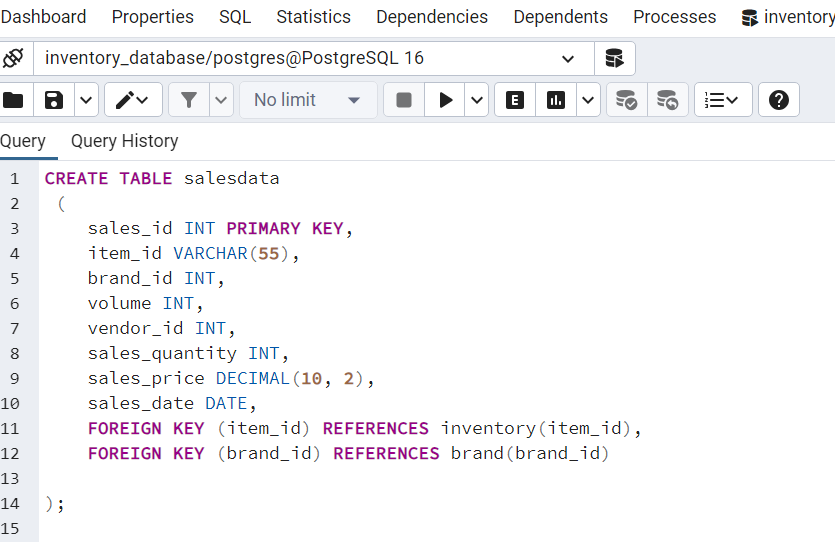
CREATING A TABLE inventorydata in inventory\_database using the query in POSTGRE SQL



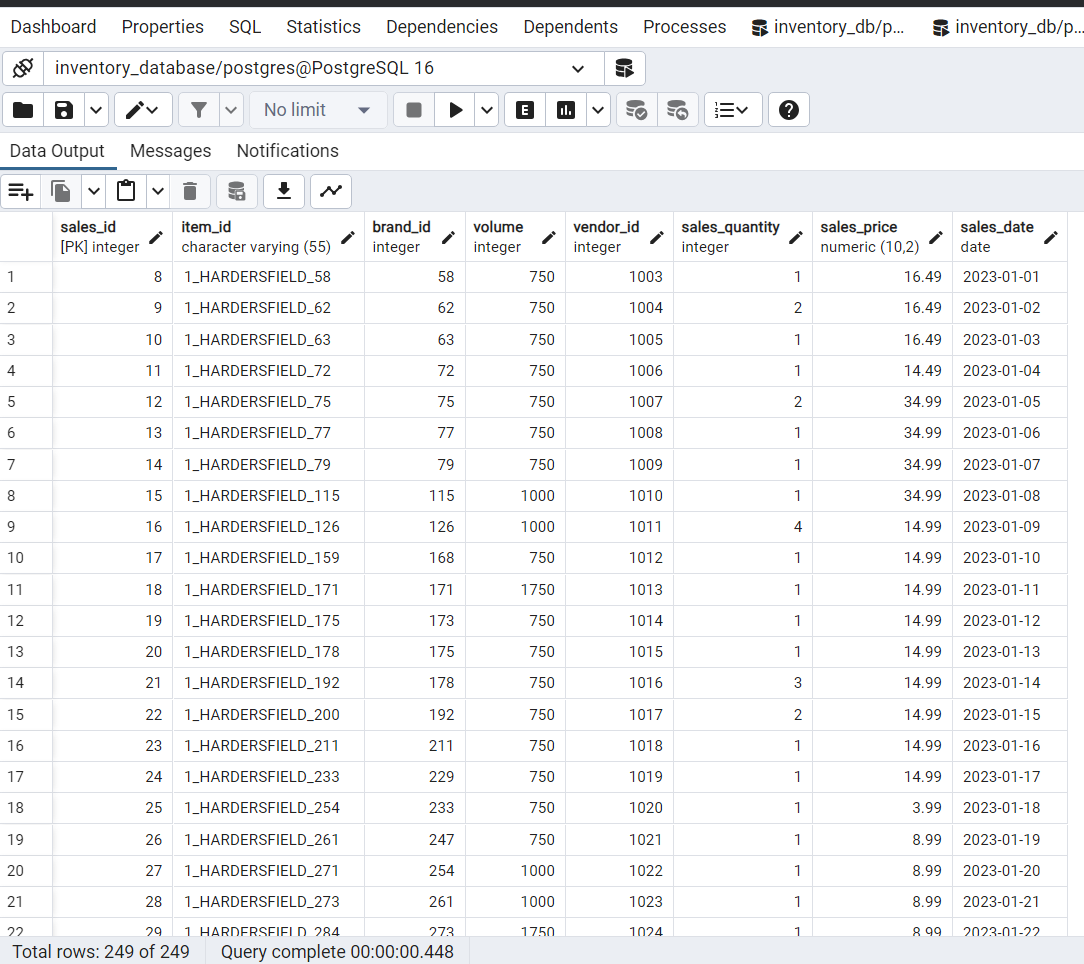
Result after data is imported in table inventorydata



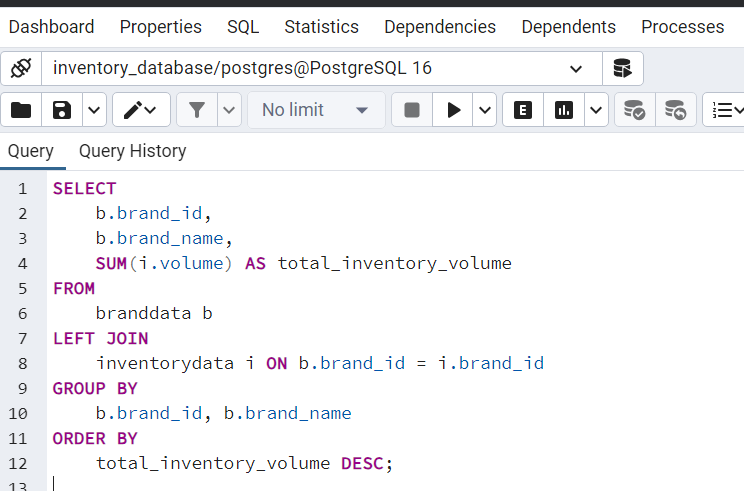
CREATING A TABLE salesdata in inventory\_database using the query in POSTGRE SQL



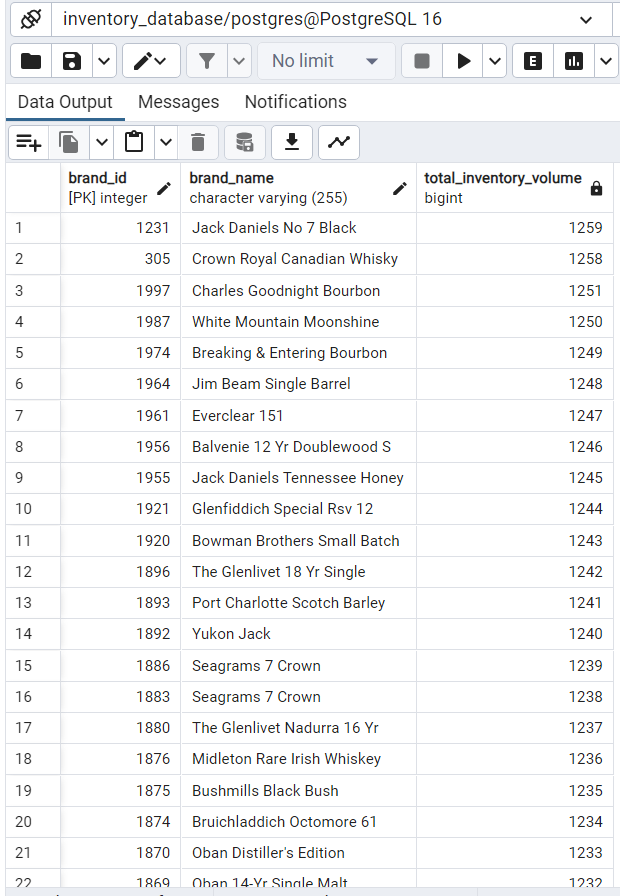
Result after data is imported in table salesdata



1.QUERY TO FIND TOTAL VOLUME OF INVENTORY BY BRAND



Result Of Query Total Volume Of Inventory By Brand



SELECT columns to be included in the query result:

b.brand\_id, b.brand\_name

SUM(i.volume) AS total\_inventory\_volume: It calculates the total inventory volume for each brand by summing up the volumes of all associated inventory items. The result is aliased as total\_inventory\_volume.

LEFT JOIN inventorydata i ON b.brand\_id = i.brand\_id: It performs a LEFT JOIN between the branddata table (b) and the inventorydata table (i) based on the brand\_id column. This join ensures that all records from the branddata table are included in the result set, even if there are no matching records in the inventorydata table.

The GROUP BY clause groups the result set by brand\_id and brand\_name:

b.brand\_id, b.brand\_name: It groups the results by the unique identifier and name of each brand.

The ORDER BY clause sorts the result set based on the total inventory volume in descending order:

total\_inventory\_volume DESC: It orders the result set by the calculated total inventory volume in descending order, ensuring that brands with the highest inventory volumes appear first.

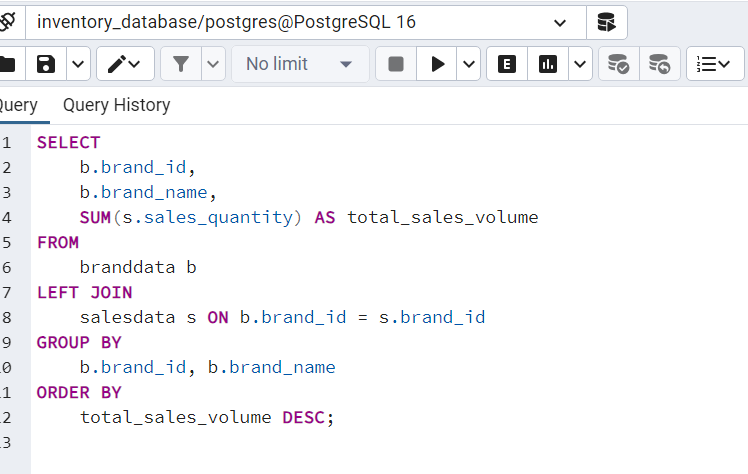
**Problem:**

Understanding the overall inventory volume for each brand helps in assessing the distribution of inventory across different brands.

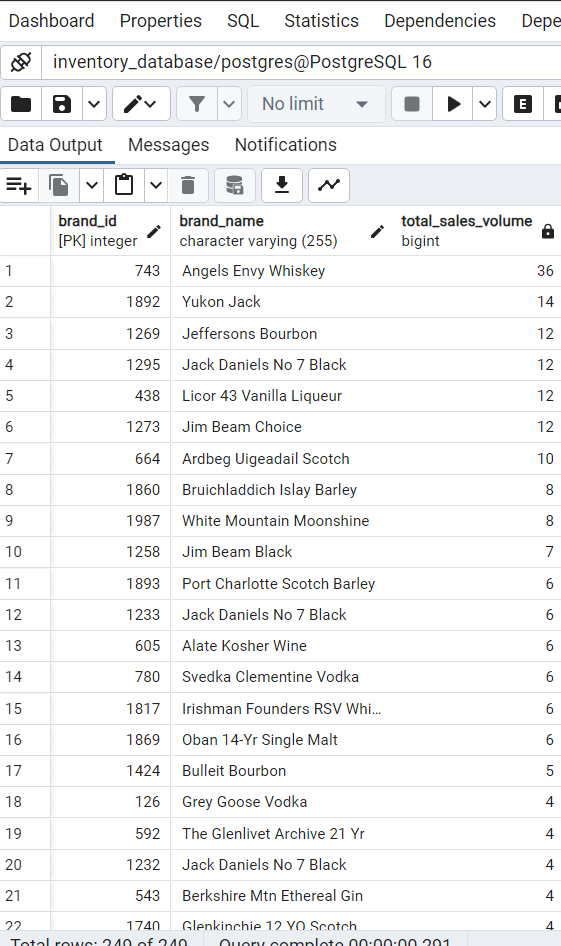
**Solution:**

This query calculates the total volume of inventory items for each brand. It provides a comprehensive overview of inventory distribution, aiding in inventory planning and management.

2.TOTAL SALES VOLUME BY BRAND:



Results Of Query Total Sales Volume By Brand:



The SELECT statement specifies brand\_id and brand\_name columns from the branddata table, as well as the calculated sum of sales\_quantity column as total\_sales\_volume.

The FROM clause specifies the tables from which data will be retrieved.

It references the branddata table as b, indicating that it will be used as the primary table for data retrieval.

The LEFT JOIN clause is used to join the branddata table (b) with the salesdata table (s) based on the brand\_id column.

This ensures that all records from the branddata table are included in the result set, regardless of whether there are matching records in the salesdata table.

The ON clause specifies the condition for joining the two tables.

It matches rows from the branddata table (b) with rows from the salesdata table (s) where the brand\_id values are equal.

The GROUP BY clause is used to group the result set by one or more columns.

In this query, it groups the result set by brand\_id and brand\_name, as we want to calculate the total sales volume for each brand.

The SUM function calculates the total sum of the sales\_quantity column for each group of brands.

It calculates the total sales volume for each brand.

The ORDER BY clause is used to sort the result set based on specified columns.

In this query, it sorts the result set by total\_sales\_volume in descending order, so that brands with the highest total sales volume appear first in the result set.

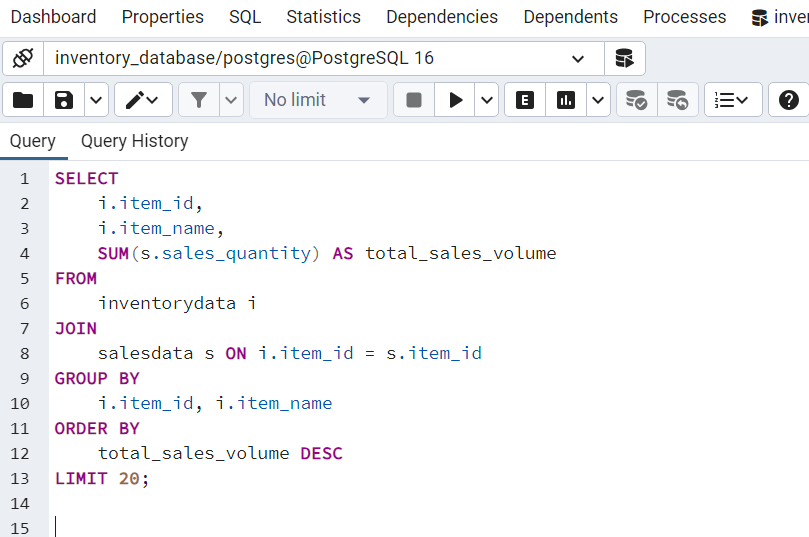
**Problem:**

Analyzing sales volume by brand is crucial for identifying the performance of different brands in terms of sales.

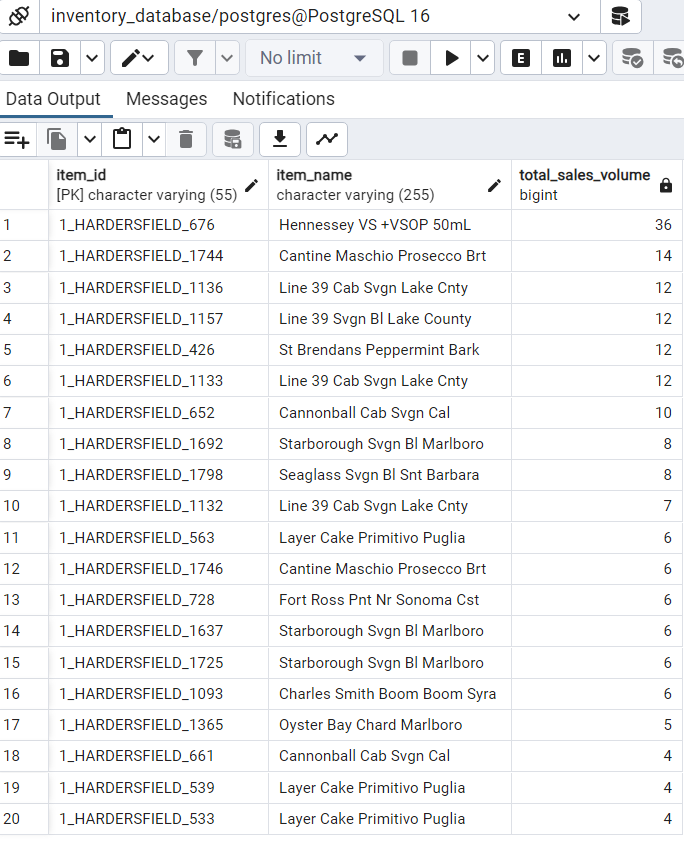
**Solution:**

This query computes the total sales volume for each brand. It enables stakeholders to assess the popularity and demand for products from different brands, facilitating strategic decision-making

3.TOP SELLING ITEMS BY VOLUME



Result Of Query Top Selling Items By Volume



The SELECT statement specifies the columns item\_id, item\_name, and the total sales quantity (total\_sales\_volume) in the result set.

The FROM clause specifies the tables from which data will be retrieved. Here, it references two tables: inventorydata (aliased as i) and salesdata (aliased as s).

The JOIN clause is used to join the inventorydata table with the salesdata table based on the item\_id column. This ensures that only records with matching item\_id values from both tables are included in the result set.

The ON clause specifies the condition for joining the two tables. It matches rows from the inventorydata table (i) with rows from the salesdata table (s) where the item\_id values are equal.

The GROUP BY clause is used to group the result set by item\_id and item\_name. This means that the calculation of the total sales volume will be done for each unique combination of item\_id and item\_name.

Within the SELECT statement, the SUM function is used to calculate the total sales quantity (sales\_quantity) for each item. This aggregation function calculates the sum of sales quantity for each group of items.

The ORDER BY clause sorts the result set by total\_sales\_volume in descending order. This means that items with the highest total sales volume will appear first in the result set.

The LIMIT clause is used to restrict the number of rows returned by the query. In this case, it limits the result set to the top 20 items with the highest total sales volume.

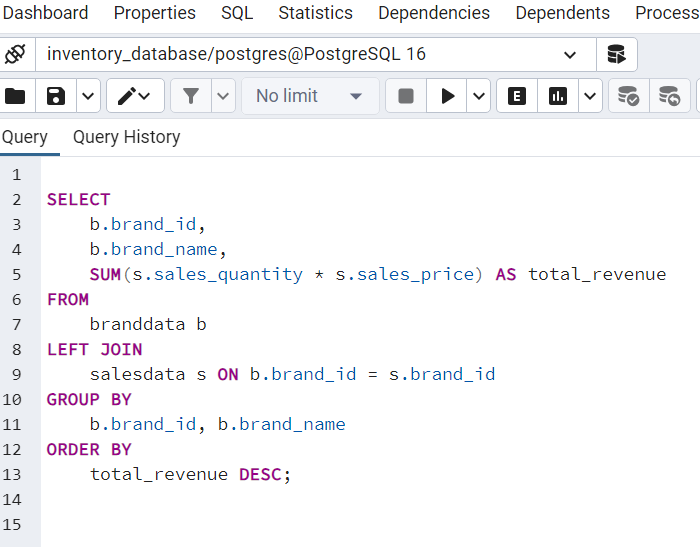
**Problem:**

Identifying the top-selling items based on volume helps in recognizing high-demand products.

**Solution:**

This query identifies the items with the highest sales volume. It provides insights into consumer preferences and allows businesses to focus on promoting and managing inventory for these popular products.

4.TOTAL REVENUE BY BRAND:



Result Of Query Total Revenue By Brand:



The SELECT statement specifies the columns brand\_id, brand\_name, and the minimum inventory volume (lowest\_inventory\_volume) in the result.

The FROM clause specifies the tables from which data will be retrieved. It references the branddata table (aliased as b) and the inventorydata table (aliased as i).

The JOIN clause is used to join the branddata table with the inventorydata table based on the brand\_id column. This ensures that only records with matching brand\_id values from both tables are included in the result set.

The ON clause specifies the condition for joining the two tables. It matches rows from the branddata table (b) with rows from the inventorydata table (i) where the brand\_id values are equal.

The GROUP BY clause is used to group the result set by brand\_id and brand\_name. This means that the calculation of the minimum inventory volume will be done for each unique combination of brand\_id and brand\_name.

Within the SELECT statement, the MIN function is used to calculate the minimum inventory volume (volume) for each brand. This function finds the smallest inventory volume among the grouped records.

The ORDER BY clause sorts the result set by lowest\_inventory\_volume in ascending order. This means that brands with the lowest inventory volume will appear first in the result set.

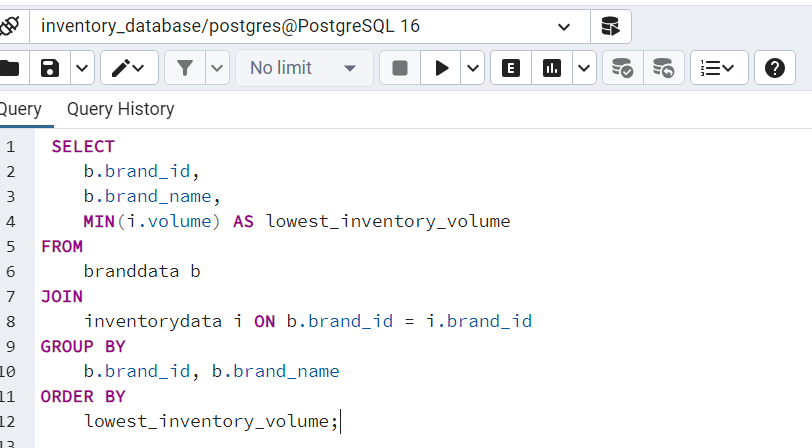
Problem:

Analyzing revenue by brand assists in evaluating the financial performance of different brands.

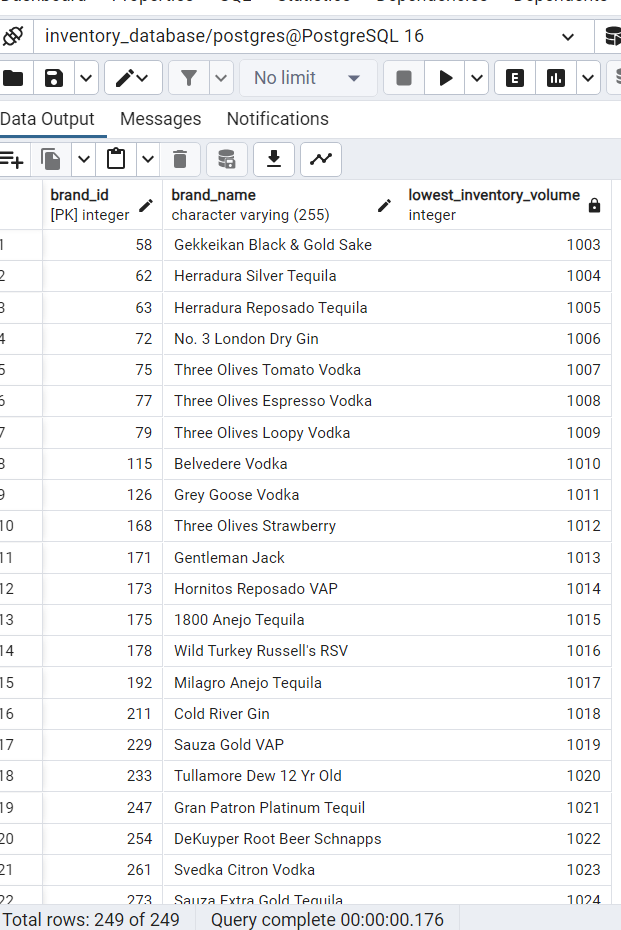
Solution:

This query calculates the revenue generated by each brand. It helps in understanding the contribution of each brand to the overall revenue, aiding in resource allocation and brand management strategies.

5.FIND THE BRAND WITH THE LOWEST INVENTORY VOLUME



Result Of Query Brand With The Lowest Inventory Volume



The SELECT statement specifies the columns brand\_id, brand\_name, and the minimum inventory volume as lowest\_inventory\_volume in the result set.

The FROM clause specifies the tables from which data will be retrieved. It references two tables: branddata (aliased as b) and inventorydata (aliased as i).

The JOIN clause is used to join the branddata table with the inventorydata table based on the brand\_id column. This ensures that only records with matching brand\_id values from both tables are included in the result set.

The ON clause specifies the condition for joining the two tables. It matches rows from the branddata table (b) with rows from the inventorydata table (i) where the brand\_id values are equal.

The GROUP BY clause is used to group the result set by brand\_id and brand\_name. This means that the calculation of the minimum inventory volume will be done for each unique combination of brand\_id and brand\_name.

Within the SELECT statement, the MIN function is used to calculate the minimum inventory volume (volume) for each brand. This function finds the smallest inventory volume among the grouped records.

The ORDER BY clause sorts the result set by lowest\_inventory\_volume in ascending order. This means that the brand with the lowest inventory volume will appear first in the result set.

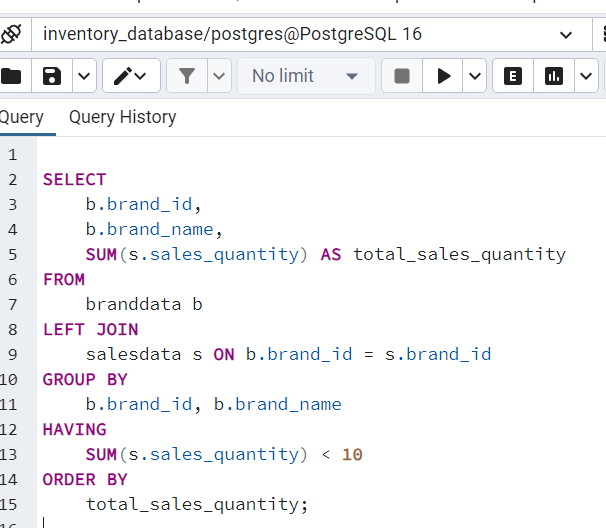
Problem:

Identifying brands with the lowest inventory volume is essential for inventory optimization and avoiding stockouts.

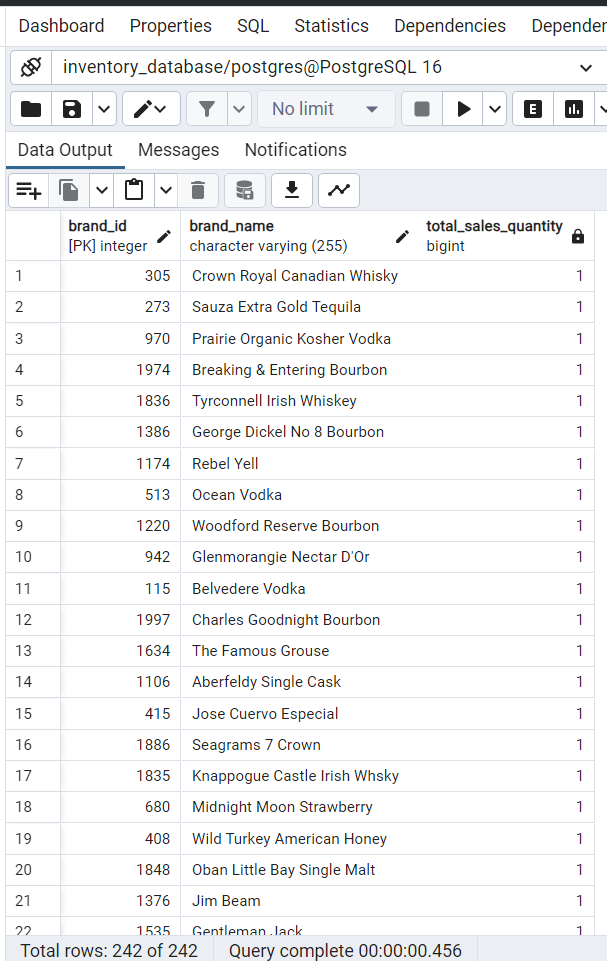
Solution:

This query identifies the brand with the lowest inventory volume. It enables businesses to prioritize restocking efforts for brands with low inventory levels, ensuring continuous availability of products to meet customer demand.

6. TO IDENTIFY LOW IN STOCK PRODUCTS



Result Of Query To Identify Low In Stock Products



To identify low-performing brands based on low sales quantity

The SELECT statement specifies brand\_id, brand\_name, and the sum of sales\_quantity as total\_sales\_quantity.

The FROM clause specifies the primary table from which data will be retrieved, which is the brand table (aliased as b).

The LEFT JOIN clause is used to join the brand table with the sales table based on the brand\_id column. This ensures that all brands are included in the result set, even if they have no corresponding sales records.

The ON clause specifies the condition for joining the two tables. It matches rows from the brand table (b) with rows from the sales table (s) where the brand\_id values are equal.

The GROUP BY clause is used to group the result set by brand\_id and brand\_name. This means that the sum of sales quantity will be calculated for each unique combination of brand\_id and brand\_name.

The HAVING clause filters the grouped results based on a condition. In this case, it filters out brands with a total sales quantity less than 10.

The ORDER BY clause sorts the result set by total\_sales\_quantity in ascending order. This means that brands with the lowest total sales quantity will appear first in the result set

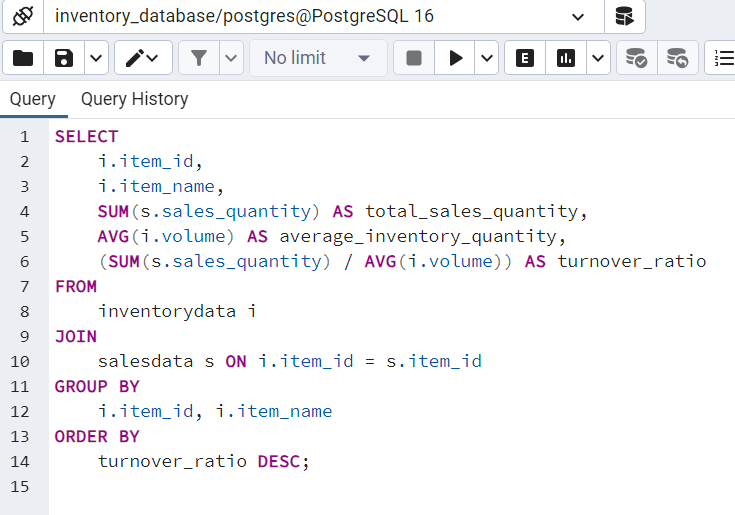
Problem:

Recognizing low in-stock products helps in preventing stockouts and ensuring adequate inventory levels.

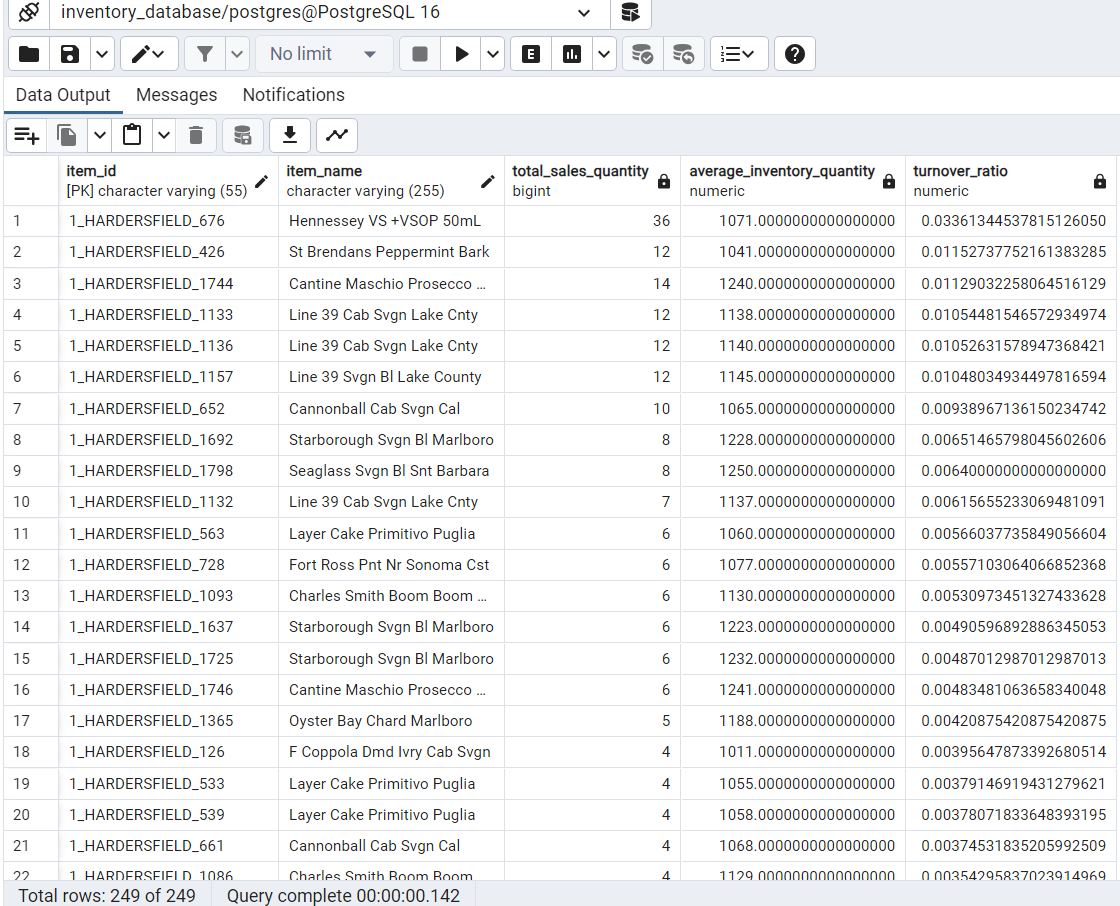
Solution:

This query identifies products that have low inventory levels. It allows businesses to replenish inventory for these products timely, minimizing the risk of lost sales due to stockouts.

7. TO CALCULATE THE PRODUCT TURNOVER RATIO FOR INVENTORY ANALYSIS



Result Of Query To Calculate The Product Turnover Ratio For Inventory Analysis



The SELECT statement specifies the columns item\_id, item\_name, the total sales quantity (total\_sales\_quantity), the average inventory quantity (average\_inventory\_quantity), and the turnover ratio.

The FROM clause specifies the primary tables from which data will be retrieved, which are the inventorydata table (aliased as i) and the salesdata table (aliased as s).

The JOIN clause is used to join the inventorydata table with the salesdata table based on the item\_id column. This ensures that only records with matching item\_id values from both tables are included in the result set.

The ON clause specifies the condition for joining the two tables. It matches rows from the inventorydata table (i) with rows from the salesdata table (s) where the item\_id values are equal.

The GROUP BY clause is used to group the result set by item\_id and item\_name. This means that the calculations of total sales quantity and average inventory quantity will be done for each unique combination of item\_id and item\_name.

Within the SELECT statement, the SUM function is used to calculate the total sales quantity (sales\_quantity) for each item, and the AVG function is used to calculate the average inventory quantity (volume) for each item.

The turnover ratio is calculated as the ratio of total sales quantity to average inventory quantity for each item. This is done by dividing the total sales quantity by the average inventory quantity.

The ORDER BY clause sorts the result set by turnover ratio in descending order. This means that items with the highest turnover ratios will appear first in the result set.

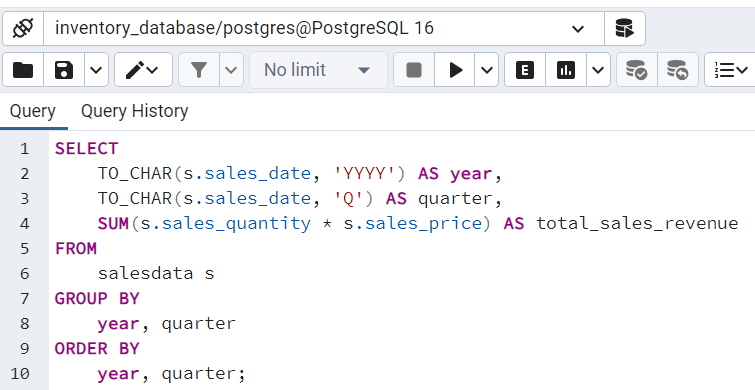
Problem:

Calculating the product turnover ratio provides insights into inventory efficiency and helps in inventory management decision-making.

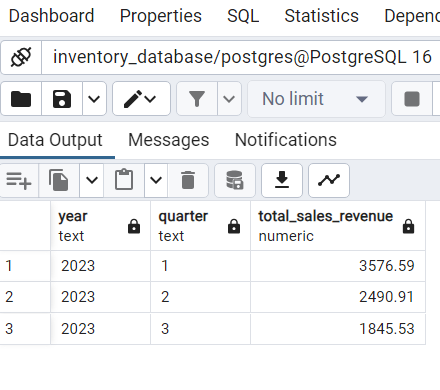
Solution:

This query computes the turnover ratio for products, which is the ratio of sales to average inventory. It assists in assessing how quickly products are sold and restocked, aiding in optimizing inventory levels and reducing carrying cost.

8.FIND THE TOTAL SALES REVENUE FOR EACH QUARTER:



Query Of Result Total Sales Revenue For Each Quarter



The SELECT statement specifies the columns to be included in the result set. It selects the year (year), quarter (quarter), and the total sales revenue (total\_sales\_revenue) calculated as the sum of sales quantity multiplied by sales price.

The FROM clause specifies the primary table from which data will be retrieved, which is the salesdata table (aliased as s).

Grouping Data by Year and Quarter:

CHAR function is used to extract the year and quarter information from the sales\_date column and format it accordingly.

The GROUP BY clause is used to group the result set by year and quarter. This means that the total sales revenue will be calculated for each unique combination of year and quarter.

Aggregation Function:

Within the SELECT statement, the SUM function is used to calculate the total sales revenue (sales\_quantity \* sales\_price) for each quarter of each year. This function calculates the sum of sales revenue for each group of year and quarter.

The ORDER BY clause sorts the result set by year and quarter in ascending order. This ensures that the results are organized chronologically by year and quarter.

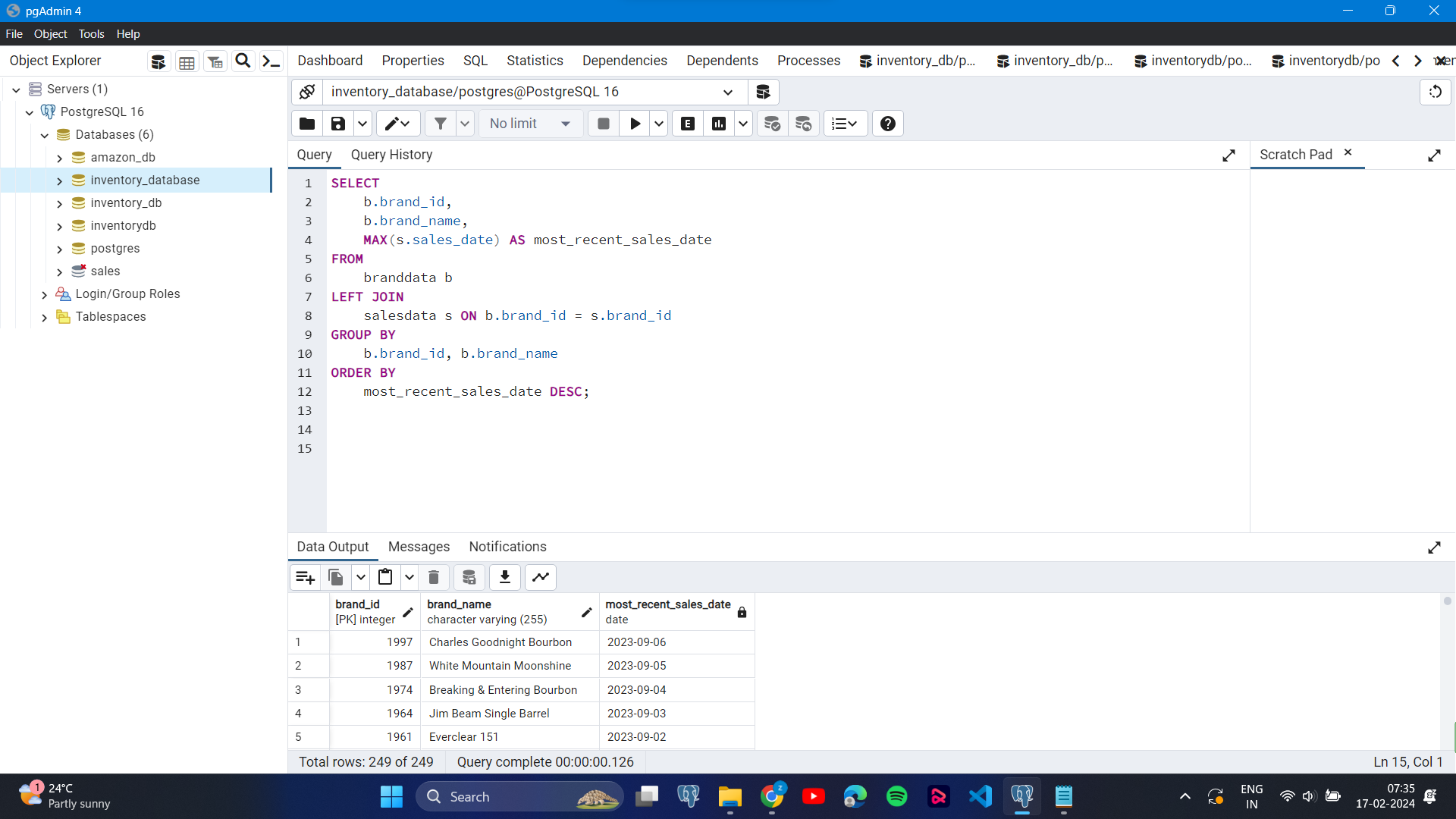
Problem:

Analyzing sales revenue by quarter helps in tracking sales performance over time and identifying seasonal trends.

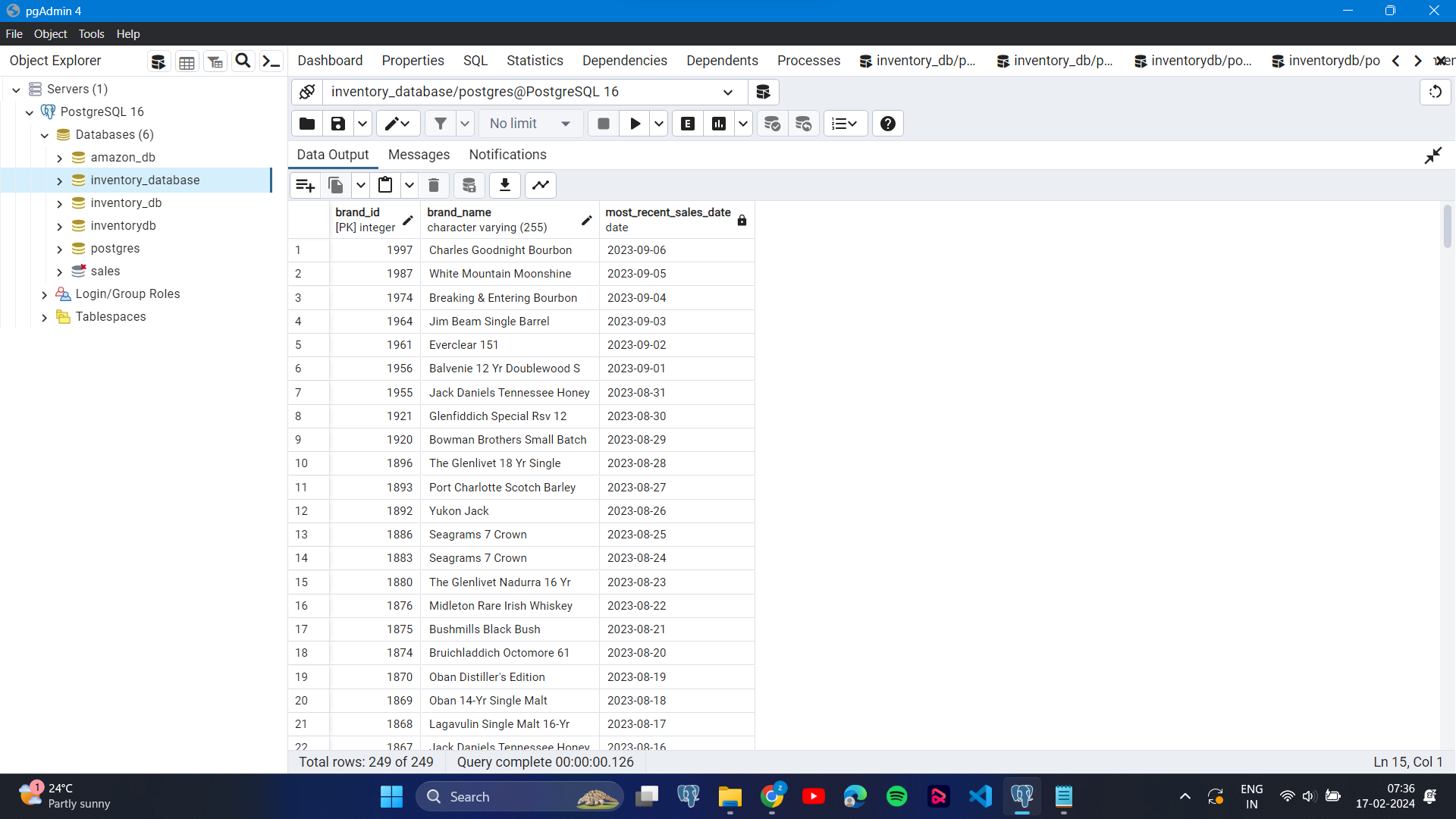
Solution:

This query calculates the total sales revenue for each quarter. It allows businesses to understand revenue patterns throughout the year, facilitating sales forecasting and strategic planning.

9. FIND THE MOST RECENT SALES DATE FOR EACH BRAND:



Result Of Query Most Recent Sales Date For Each Products



The SELECT statement specifies the columns to be included in the result set. It selects brand\_id, brand\_name, and the maximum sales date (most\_recent\_sales\_date).

The FROM clause specifies the primary table from which data will be retrieved, which is the branddata table (aliased as b).

The LEFT JOIN clause is used to join the branddata table with the salesdata table based on the brand\_id column. This ensures that all brands are included in the result set, even if they have no corresponding sales records.

The ON clause specifies the condition for joining the two tables. It matches rows from the branddata table (b) with rows from the salesdata table (s) where the brand\_id values are equal.

The GROUP BY clause is used to group the result set by brand\_id and brand\_name. This means that the maximum sales date will be calculated for each unique combination of brand\_id and brand\_name.

Within the SELECT statement, the MAX function is used to calculate the maximum sales date (sales\_date) for each brand. This function finds the most recent sales date among the grouped records.

The ORDER BY clause sorts the result set by most\_recent\_sales\_date in descending order. This means that brands with the most recent sales dates will appear first in the result set.

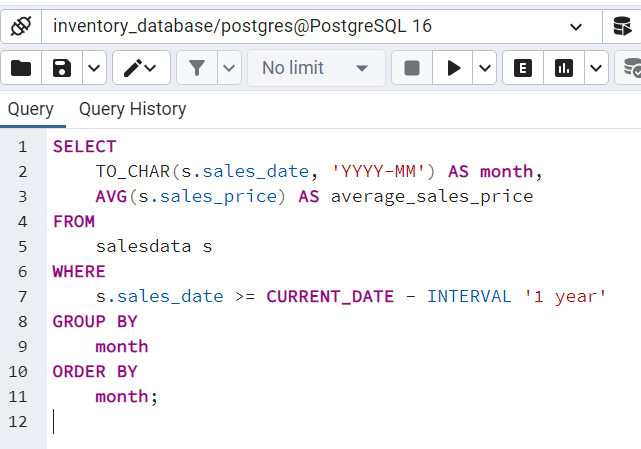
Problem:

Identifying the most recent sales date for each brand helps in monitoring sales activity and assessing brand performance.

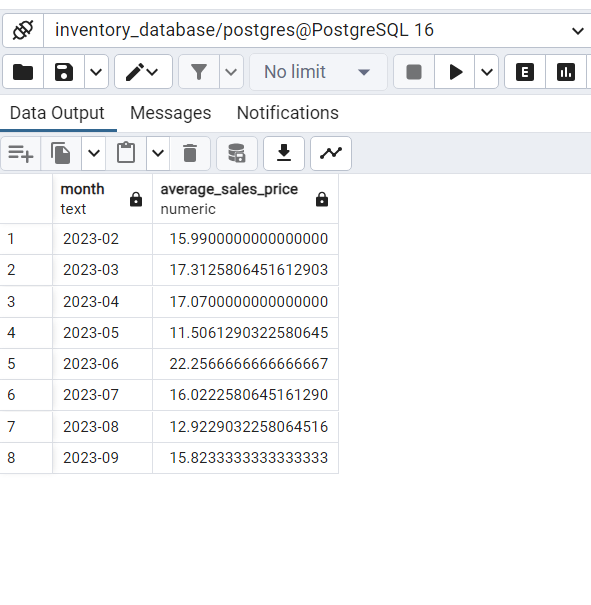
Solution:

This query retrieves the most recent sales date for each brand. It enables businesses to stay updated on sales activities and track the timeliness of sales for each brand.

10.FIND THE AVERAGE SALES PRICE FOR EACH MONTH IN THE LAST YEAR



Results of the query the average sales price for each month in the last year



It selects two columns: month, which represents the year and month formatted as 'YYYY-MM', and average\_sales\_price, which represents the average sales price for each month.

It specifies the primary table from which data will be retrieved, which is the salesdata table (aliased as s).

It filters the rows based on the condition that the sales date (sales\_date) is greater than or equal to the date one year ago from the current date (CURRENT\_DATE - INTERVAL '1 year'). This restricts the data to the past year.

It groups the result set by the formatted sales date (month). This groups the sales data into monthly intervals based on the year and month.

Within the SELECT statement, the AVG function is used to calculate the average sales price (sales\_price) for each month. This function computes the average value of sales prices for each monthly interval.

It orders the result set by month in ascending order. This ensures that the results are arranged chronologically by month.

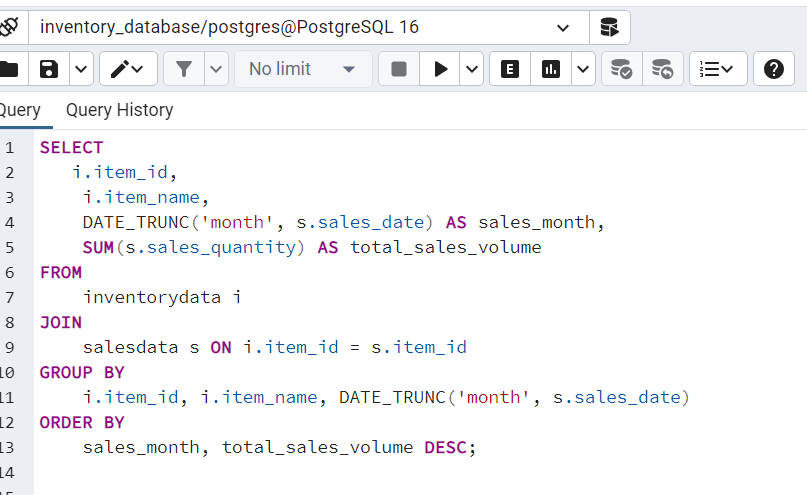
Problem:

Calculating the average sales price for each month helps in understanding pricing trends and assessing pricing strategies' effectiveness.

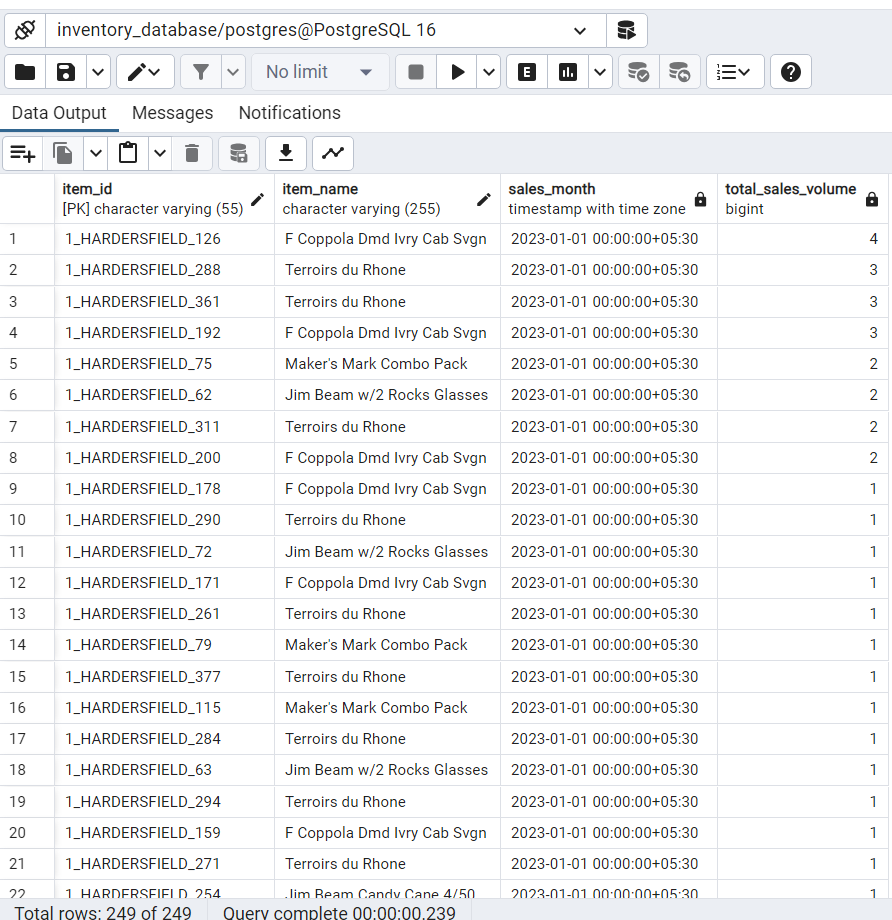
Solution:

This query computes the average sales price for each month in the last year. It provides insights into fluctuations in pricing over time, allowing businesses to adjust pricing strategies accordingly.

11.TOTAL SALES VOLUME BY PRODUCT AND MONTH:



Result Of Query Total Sales Volume By Product And Month:



selects four columns: item\_id, item\_name, sales\_month, and total\_sales\_volume.

It specifies the primary tables from which data will be retrieved, which are the inventorydata table (aliased as i) and the salesdata table (aliased as s).

It joins the inventorydata and salesdata tables based on the common item\_id column. This ensures that only records with matching item\_id values from both tables are included in the result set.

It groups the result set by item\_id, item\_name, and the truncated sales date (sales\_month). This means that the total sales volume will be calculated for each unique combination of item, item name, and month.

Within the SELECT statement, the SUM function is used to calculate the total sales volume (sales\_quantity) for each item on a monthly basis. This function computes the sum of sales quantities for each group of item, item name, and month.

The DATE\_TRUNC function truncates the sales date (sales\_date) to the beginning of the month. This ensures that sales are aggregated on a monthly basis.

It orders the result set first by sales\_month in ascending order and then by total\_sales\_volume in descending order within each month. This means that the results are sorted chronologically by month and within each month, items with the highest sales volume appear first.

Problem:

Analyzing sales volume by product and month helps in understanding sales patterns and identifying product performance over time.

Solution:

This query calculates the total sales volume for each product by month. It enables businesses to track sales trends for individual products, facilitating inventory planning and marketing strategies.

These SQL queries provide valuable insights into various aspects of inventory management, enabling businesses to optimize inventory levels, enhance sales performance, and make informed strategic decisions.