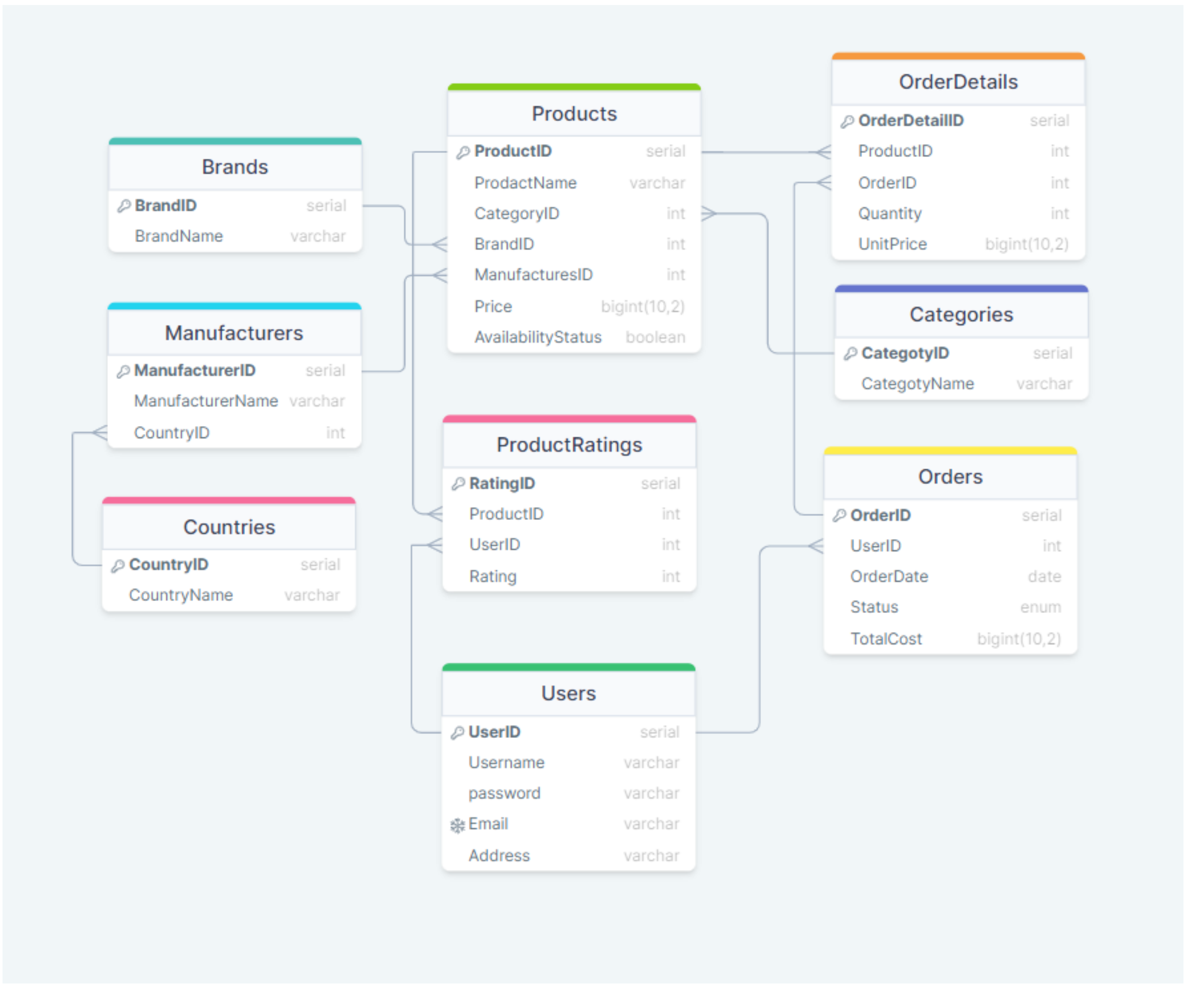
Documentation for Database Schema for online store of medical equipment

This document provides an overview of the database schema, detailing the structure and relationships of the tables within the database. The database consists of nine tables, each serving a specific purpose in managing products, users, orders, and related information.

ER-diagram



Tables Description

<u>Countries of Origin</u>

This table stores the countries from which products originate.

- **Table Name**: `CountriesOfOrigin`
- **Columns**:
    - `CountryID`: An automatically incremented unique identifier for each country. This serves as the primary key.
    - `CountryName`: The name of the country. This must be unique and not null.

Product Categories

This table contains the different categories to which products can belong.

- **Table Name**: `Categories`
- **Columns**:
  - `CategoryID`: An automatically incremented unique identifier for each category. This serves as the primary key.
  - `CategoryName`: The name of the category. This must be unique and not null.

Brands

This table records the brands associated with products.

- **Table Name**: `Brands`
- **Columns**:
  - `BrandID`: An automatically incremented unique identifier for each brand. This serves as the primary key.
  - `BrandName`: The name of the brand. This must be unique and not null.

Manufacturers

This table includes details about the manufacturers of products.

- **Table Name**: `Manufacturers`
- **Columns**:
  - `ManufacturerID`: An automatically incremented unique identifier for each manufacturer. This serves as the primary key.
  - `ManufacturerName`: The name of the manufacturer. This must be unique and not null.
  - `CountryID`: A foreign key referencing the `CountryID` in the `CountriesOfOrigin` table to indicate the manufacturer's country of origin.

Products

This table contains information about the products available.

- **Table Name**: `Products`
- **Columns**:
  - `ProductID`: An automatically incremented unique identifier for each product. This serves as the primary key.
  - `ProductName`: The name of the product. This must be unique and not null.
  - `CategoryID`: A foreign key referencing the `CategoryID` in the `Categories` table to categorize the product.
  - `BrandID`: A foreign key referencing the `BrandID` in the `Brands` table to indicate the product's brand.
  - `ManufacturerID`: A foreign key referencing the `ManufacturerID` in the `Manufacturers` table to indicate the product's manufacturer.
  - `Price`: The price of the product. This is a numeric value with up to two decimal places and cannot be null.
  - `AvailabilityStatus`: A boolean indicating the product's availability status.

    - `unique_product`: A unique constraint ensuring that each combination of `ProductName`, `CategoryID`, `BrandID`, and `ManufacturerID` is unique.

Users

This table stores information about the users of the system.

- **Table Name**: `Users`
- **Columns**:
    - `UserID`: An automatically incremented unique identifier for each user. This serves as the primary key.
    - `username`: The username of the user. This cannot be null.
    - `email`: The email address of the user. This must be unique and not null.
    - `address`: The address of the user.
    - `phone`: The phone number of the user.

Orders

This table records the orders placed by users.

- **Table Name**: `Orders`
- **Columns**:
    - `OrderID`: An automatically incremented unique identifier for each order. This serves as the primary key.
    - `UserID`: A foreign key referencing the `UserID` in the `Users` table to indicate the user who placed the order.
    - `OrderDate`: The date and time when the order was placed. This defaults to the current timestamp.
    - `Status`: The status of the order.
    - `TotalCost`: The total cost of the order. This is a numeric value with up to two decimal places and cannot be null.

Order Details

This table provides details about the products included in each order.

- **Table Name**: `OrderDetails`
- **Columns**:
    - `OrderDetailID`: An automatically incremented unique identifier for each order detail. This serves as the primary key.
    - `OrderID`: A foreign key referencing the `OrderID` in the `Orders` table to indicate which order this detail belongs to.
    - `ProductID`: A foreign key referencing the `ProductID` in the `Products` table to indicate which product is included in the order.
    - `Quantity`: The quantity of the product in the order.
    - `UnitPrice`: The price per unit of the product. This is a numeric value with up to two decimal places and cannot be null.

Product Ratings

This table stores user ratings for products.

- **Table Name**: `ProductRatings`
- **Columns**:
    - `RatingID`: An automatically incremented unique identifier for each rating. This serves as the primary key.
    - `ProductID`: A foreign key referencing the `ProductID` in the `Products` table to indicate which product is being rated.
    - `UserID`: A foreign key referencing the `UserID` in the `Users` table to indicate which user provided the rating.
    - `Rating`: The rating given to the product, which must be an integer between 1 and 5 (inclusive).

Relationships

- **CountriesOfOrigin to Manufacturers**: Each manufacturer is associated with a country of origin. This is enforced through a foreign key constraint on the `CountryID` in the `Manufacturers` table referencing the `CountryID` in the `CountriesOfOrigin` table.
- **Categories to Products**: Each product belongs to a category. This is enforced through a foreign key constraint on the `CategoryID` in the `Products` table referencing the `CategoryID` in the `Categories` table.
- **Brands to Products**: Each product is associated with a brand. This is enforced through a foreign key constraint on the `BrandID` in the `Products` table referencing the `BrandID` in the `Brands` table.
- **Manufacturers to Products**: Each product is manufactured by a manufacturer. This is enforced through a foreign key constraint on the `ManufacturerID` in the `Products` table referencing the `ManufacturerID` in the `Manufacturers` table.
- **Users to Orders**: Each order is placed by a user. This is enforced through a foreign key constraint on the `UserID` in the `Orders` table referencing the `UserID` in the `Users` table.
- **Orders to OrderDetails**: Each order detail is associated with an order. This is enforced through a foreign key constraint on the `OrderID` in the `OrderDetails` table referencing the `OrderID` in the `Orders` table.
- **Products to OrderDetails**: Each order detail includes a product. This is enforced through a foreign key constraint on the `ProductID` in the `OrderDetails` table referencing the `ProductID` in the `Products` table.
- **Products to ProductRatings**: Each rating is associated with a product. This is enforced through a foreign key constraint on the `ProductID` in the `ProductRatings` table referencing the `ProductID` in the `Products` table.
- **Users to ProductRatings**: Each rating is provided by a user. This is enforced through a foreign key constraint on the `UserID` in the `ProductRatings` table referencing the `UserID` in the `Users` table.

Documentation for the first ETL (Extract, Transform, Load) Process

Overview

This ETL script (ETL1_csv.sql) is designed to import order data from CSV files (order_details.csv, orders.csv, product_ratings.csv) into a database, transforming and loading it into the appropriate tables in the `orders` schema. The ETL process involves several steps to ensure data is accurately imported and related correctly within the database.

Temporary Tables

All temporary tables are created in the orders schema to stage data before loading it into the permanent tables.

TempOrders

- **Purpose**: Stores raw order data.
- **Columns**: `OrderID, OrderDate, FullName, Email, Address, Phone, Status`.

TempOrderDetails

- **Purpose**: Stores raw order detail data.
- **Columns**: `OrderDetailsID, OrderID, ProductName, Category, Brand, Manufacturer, CountryOfOrigin, Price, Quantity, AvailabilityStatus`.

TempProductRatings

- **Purpose**: Stores raw product rating data.
- **Columns**: `ProductName, UserFullName, Rating.`

Data Import

Data is imported from CSV files into the temporary tables using the `COPY` command. This process ensures that data from external sources is staged and can be validated before being moved into the main database tables.

Data Transformation and Loading

Categories

Inserts distinct product categories from `TempOrderDetails` into the `Categories` table, avoiding duplicates with conflict handling.

Brands

Inserts distinct product brands from `TempOrderDetails` into the `Brands` table, avoiding duplicates with conflict handling.

Countries of Origin

Inserts distinct countries of origin from `TempOrderDetails` into the `CountriesOfOrigin` table, avoiding duplicates with conflict handling.

Manufacturers

Inserts distinct manufacturers and their associated countries into the `Manufacturers` table by joining with `CountriesOfOrigin`.

Users

Inserts distinct user data from `TempOrders` into the `Users` table, avoiding duplicates with conflict handling.

Products

Inserts distinct products from `TempOrderDetails` into the `Products` table by joining with `Categories`, `Brands`, and `Manufacturers`, ensuring no duplicate product entries.

Orders

Inserts order data from `TempOrders` into the `Orders` table by joining with `Users`, setting an initial `TotalCost` of zero.

Order Details

Inserts order detail data from `TempOrderDetails` into the `OrderDetails` table by joining with `Orders` and `Products`.

Update TotalCost in Orders

Updates the `TotalCost` field in the `Orders` table by calculating the sum of `UnitPrice` from `OrderDetails` for each order.

Product Ratings

Inserts product rating data from `TempProductRatings` into the `ProductRatings` table by joining with `Products` and `Users`, ensuring ratings are within the valid range (1 to 5).

Documentation for Data Warehouse Schema

Overview

This documentation describes the structure and purpose of the data warehouse schema named `datawarehouse (data_warehouse.sql`. It includes dimension tables, fact tables, triggers for Slowly Changing Dimension Type 2 (SCD2) implementation, and several views for reporting and analysis.

Dimension Tables

DimCategories

- **Purpose**: Stores product categories.
- **Columns**:
  - CategoryID: Primary key.
  - CategoryName: Name of the category, not null.

DimDate

- **Purpose**: Stores date information for time-based analysis.
- **Columns**:
  - DateID: Primary key.
  - Date: The date value, unique and not null.
  - Year: Year part of the date.
  - Month: Month part of the date.
  - Day: Day part of the date.
  - Quarter: Quarter part of the date.

DimUsers

- **Purpose**: Stores user information.
- **Columns**:
  - UserID: Primary key.
  - Username: Name of the user, not null.
  - Email: Email address of the user.
  - Address: Address of the user.
  - Phone: Phone number of the user.

DimCountries

- **Purpose**: Stores country information.
- **Columns**:
  - CountryID: Primary key.
  - CountryName: Name of the country, not null.

DimManufacturers

- **Purpose**: Stores manufacturer information.
- **Columns**:
  - ManufacturerID: Primary key.
  - ManufacturerName: Name of the manufacturer, not null.
  - CountryID: Foreign key referencing DimCountries(CountryID).

DimBrands

- **Purpose**: Stores brand information.
- **Columns**:
  - BrandID: Primary key.
  - BrandName: Name of the brand, not null.

DimProducts

- **Purpose**: Stores product information.
- **Columns**:
    - ProductID: Primary key.
    - ProductName: Name of the product, not null.
    - CategoryID: Foreign key referencing DimCategories(CategoryID).
    - BrandID: Foreign key referencing DimBrands(BrandID).
    - ManufacturerID: Foreign key referencing DimManufacturers(ManufacturerID).
    - Price: Price of the product.
    - AvailabilityStatus: Availability status of the product.
    - StartDate: Start date for SCD2 implementation, default is current date.
    - EndDate: End date for SCD2 implementation, default is '9999-12-31'.
    - IsCurrent: Boolean flag for current record, default is true.
    - Unique constraint on ProductID, BrandID, CategoryID, and ManufacturerID.

Fact Tables

Sales_Fact

- **Purpose**: Stores sales data.
- **Columns**:
    - SalesID: Primary key.
    - OrderID: Foreign key referencing Orders(OrderID).
    - ProductID: Foreign key referencing DimProducts(ProductID).
    - UserID: Foreign key referencing DimUsers(UserID).
    - DateID: Foreign key referencing DimDate(DateID).
    - Quantity: Quantity of products sold.
    - UnitPrice: Price per unit.
    - TotalCost: Total cost of the sale.

ProductRatings_Fact

- **Purpose**: Stores product ratings.
- **Columns**:
    - RatingID: Primary key.
    - ProductID: Foreign key referencing DimProducts(ProductID).
    - UserID: Foreign key referencing DimUsers(UserID).
    - BrandID: Foreign key referencing DimBrands(BrandID).
    - Rating: Rating value, with a check constraint to ensure values are between 1 and 5.

SCD2 Implementation

A trigger function and trigger are created to manage Slowly Changing Dimension Type 2 (SCD2) for DimProducts.

- **Function**: dim_products_update_trigger
    - **Purpose**: Handles updates to DimProducts by closing the current record and creating a new record with updated values.
- **Trigger**: dim_products_update

    - **Purpose**: Fires after an update on DimProducts to invoke the SCD2 trigger function.

Views

sales_trend

- **Purpose**: Analyzes total sales over time.
- **Columns**:
    - Date: The date of sales.
    - TotalSales: Sum of total sales on each date.

sales_by_user

- **Purpose**: Analyzes sales by user.
- **Columns**:
    - UserID: User identifier.
    - Username: User name.
    - NumberOfSales: Count of sales made by the user.
    - TotalSales: Sum of total sales by the user.

sales_by_country_of_origin

- **Purpose**: Analyzes sales by the country of origin of products.
- **Columns**:
    - CountryName: Name of the country.
    - TotalSales: Sum of total sales for products from each country.

sales_by_category

- **Purpose**: Analyzes sales by product category.
- **Columns**:
    - CategoryName: Name of the category.
    - TotalSales: Sum of total sales for each category.

sales_by_brand

- **Purpose**: Analyzes sales by product brand.
- **Columns**:
    - BrandName: Name of the brand.
    - TotalSales: Sum of total sales for each brand.

average_rating_by_product

- **Purpose**: Analyzes average rating of products.
- **Columns**:
    - ProductName: Name of the product.
    - AverageRating: Average rating of the product.

average_rating_by_brand

- **Purpose**: Analyzes average rating of brands.

- **Columns**:
    - BrandName: Name of the brand.
    - AverageRating: Average rating of the brand.

Documentation for the second ETL Process

Overview

This ETL (ETL2_data_warehouse.sql) process is designed to transfer data from the operational database into the datawarehouse schema. The ETL process involves several steps to ensure data is accurately imported into the data warehouse tables.

Load Data into Dimension Tables

DimCategories

- **Action**: Insert distinct categories from the Categories table into datawarehouse.DimCategories.
- **Conflict Handling**: Ignores the insertion if a CategoryID already exists.

DimCountries

- **Action**: Insert distinct countries from the CountriesOfOrigin table into datawarehouse.DimCountries.
- **Conflict Handling**: Ignores the insertion if a CountryID already exists.

DimBrands

- **Action**: Insert distinct brands from the Brands table into datawarehouse.DimBrands.
- **Conflict Handling**: Ignores the insertion if a BrandID already exists.

DimManufacturers

- **Action**: Insert distinct manufacturers from the Manufacturers table into datawarehouse.DimManufacturers.
- **Conflict Handling**: Ignores the insertion if a ManufacturerID already exists.

DimUsers

- **Action**: Insert distinct users from the Users table into datawarehouse.DimUsers.
- **Conflict Handling**: Ignores the insertion if a UserID already exists.

DimDate

- **Action**: Insert distinct dates from the Orders table into datawarehouse.DimDate.
- **Transformation**: Extracts year, month, day, and quarter from OrderDate.
- **Conflict Handling**: Ignores the insertion if a Date already exists.

DimProducts

- **Action**: Insert distinct products from the Products table into datawarehouse.DimProducts.
- **Conflict Handling**: Ignores the insertion if a ProductID already exists.

Load Data into Fact Tables

Sales_Fact

- **Action**: Insert sales data from OrderDetails and Orders into datawarehouse.Sales_Fact.
- **Transformation**: Joins OrderDetails with Orders to get UserID and TotalCost, and joins with DimDate to get DateID.

ProductRatings_Fact

- **Action**: Insert product ratings from ProductRatings into datawarehouse.ProductRatings_Fact.
- **Transformation**: Joins ProductRatings with Products to get BrandID.

Documentation for Power BI Report

Overview

This documentation describes the process of creating a Power BI report based on data loaded into the `datawarehouse` schema. The report includes dashboards built from dimension tables, fact tables, and views. The goal of the report is to provide comprehensive analysis and visualizations of sales and product ratings data.

Data Sources

The following tables and views were loaded into Power BI from the `datawarehouse` schema:

Dimension Tables

- DimCategories
- DimCountries
- DimBrands
- DimManufacturers
- DimUsers
- DimDate
- DimProducts

Fact Tables

- Sales_Fact
- ProductRatings_Fact

Views

- sales_trend
- sales_by_user
- sales_by_country_of_origin
- sales_by_category
- sales_by_brand
- average_rating_by_product
- average_rating_by_brand

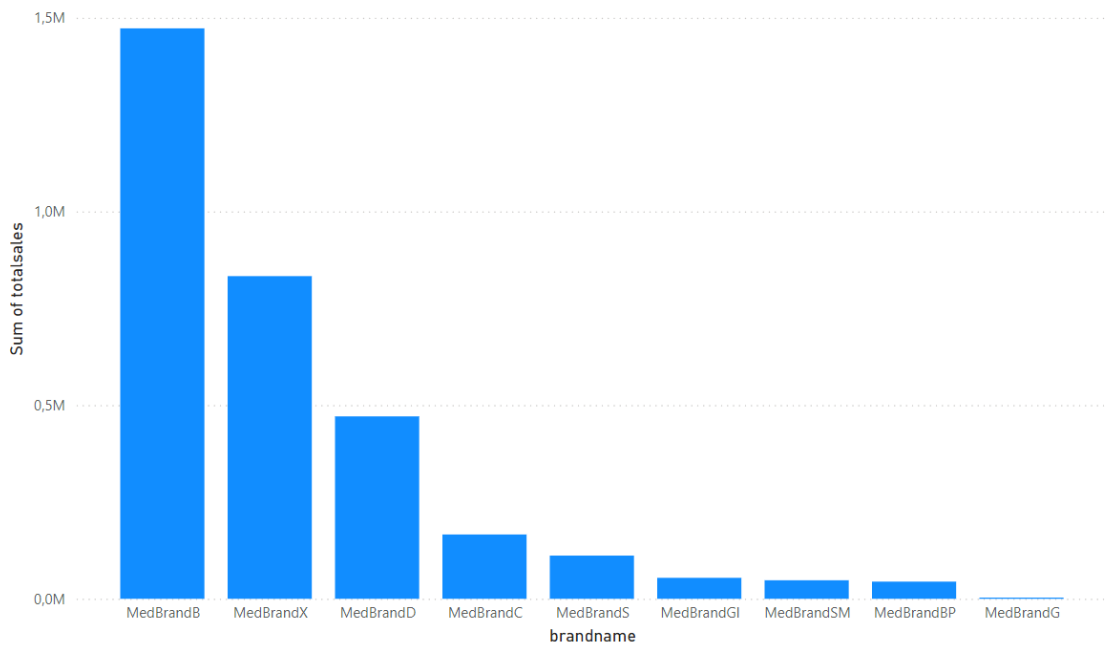
Dashboards

The Power BI report includes the following dashboards, each providing specific insights derived from the views in the `datawarehouse` schema:

Page Sales analysis (Filters: date, brandname, username, categoryname)

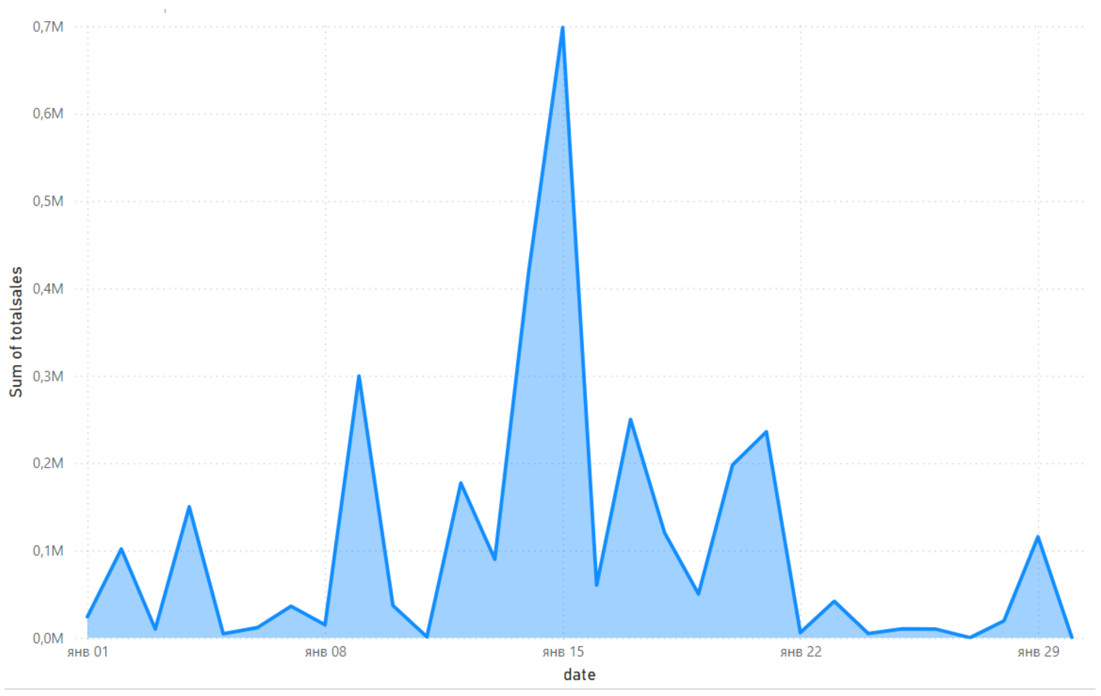
1. Sales by Brand Dashboard

- **Data Source**: `sales_by_brand`
- **Visualizations**: Column chart showing total sales by brand.
- **Insights**: Shows the performance of different brands in terms of sales.



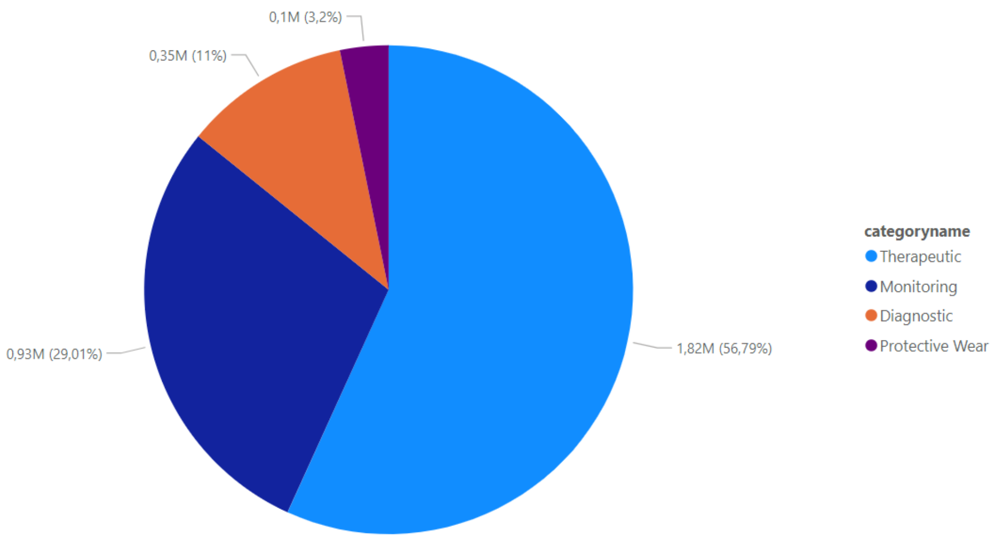
2. Sales Trend Dashboard

- **Data Source**: `sales_trend`
- **Visualizations**: Area chart showing total sales over time.
- **Insights**: Trends in sales over different periods (daily, monthly, yearly).



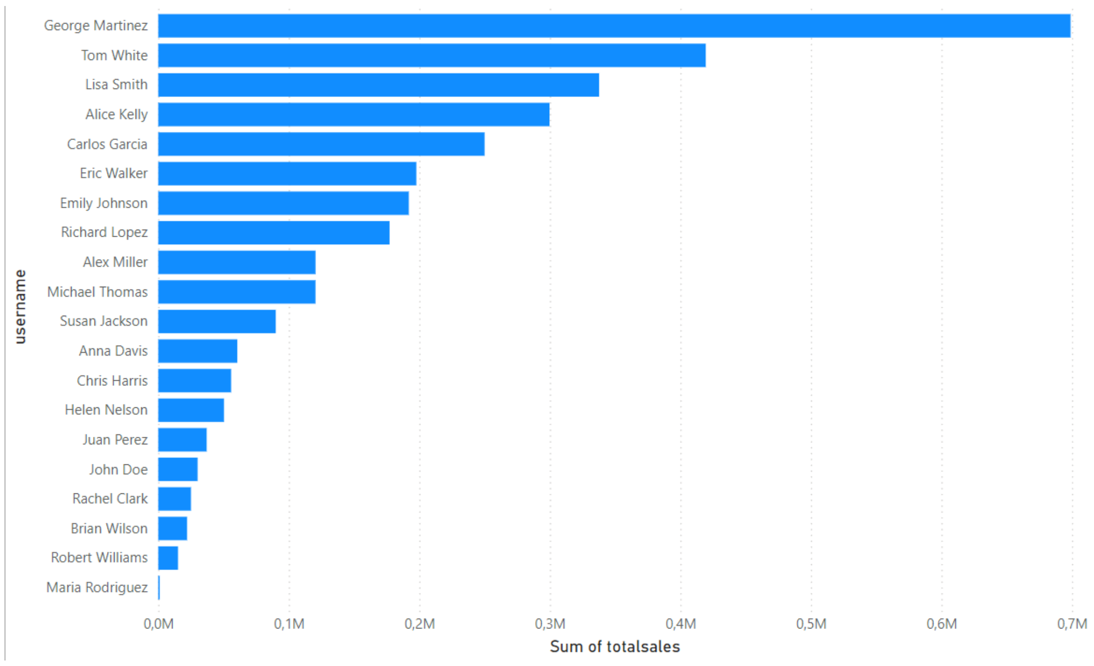
3. Sales by Category Dashboard

- **Data Source**: `sales_by_category`
- **Visualizations**: Pie chart showing the distribution of sales by category.
- **Insights**: Provides an overview of sales distribution across different product categories.



4. Sales by User Dashboard

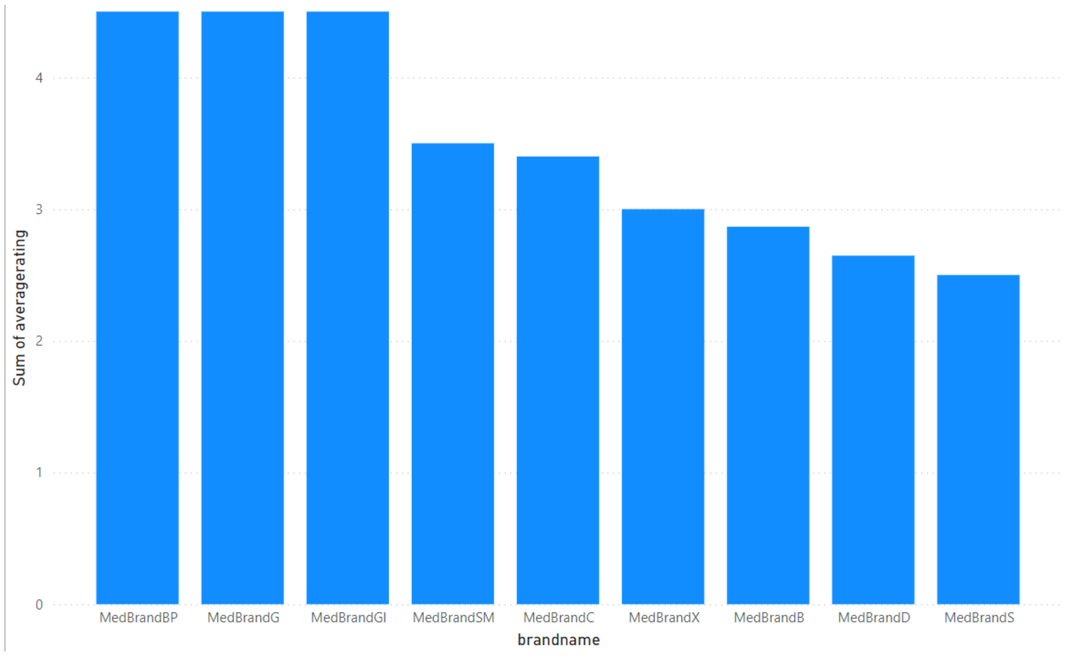
- **Data Source**: `sales_by_user`
- **Visualizations**: Bar chart showing the total sales amount per user.
- **Insights**: Identifies top users by sales volume and revenue contribution.



Page Rating analysis (Filters: brandname, productname)

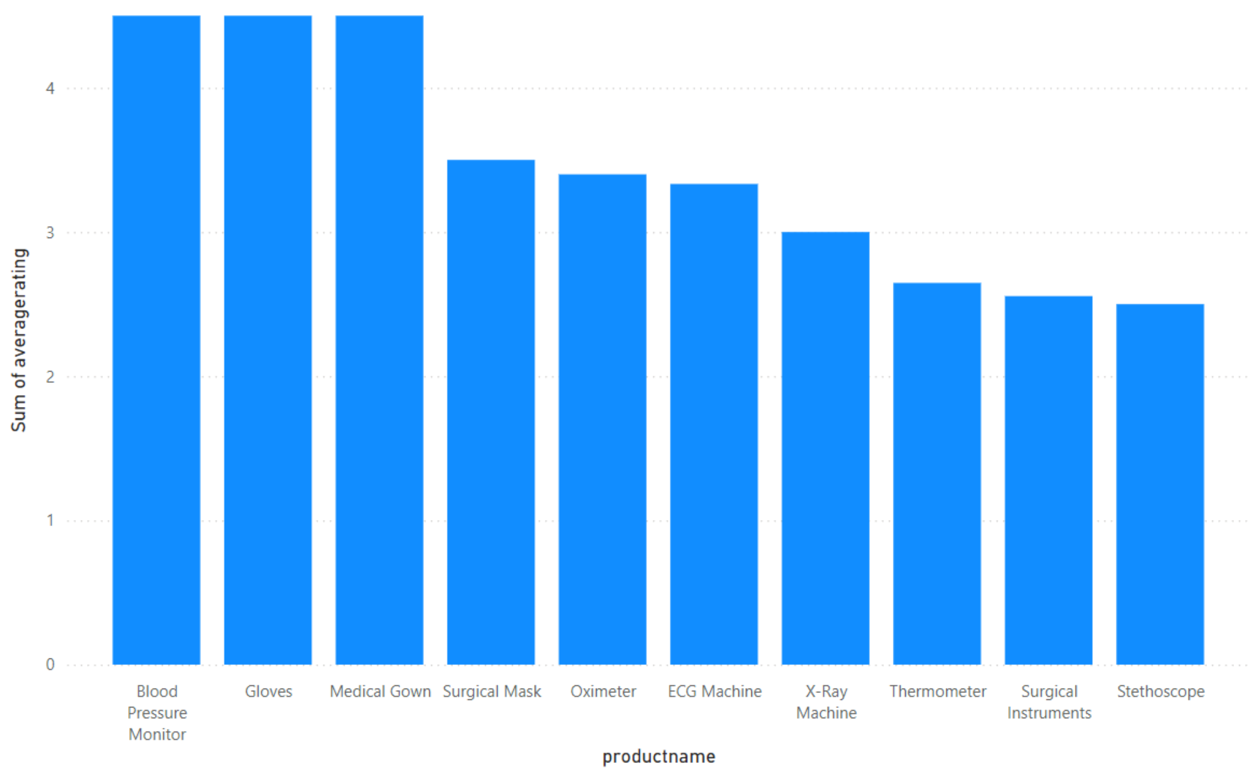
5. Average Rating by Brand Dashboard

- **Data Source**: `average_rating_by_brand`
- **Visualizations**: Column chart showing the average rating for each brand.
- **Insights**: Highlights the overall rating performance of different brands.



6. Average Rating by Product Dashboard

- **Data Source**: `average_rating_by_product`
- **Visualizations**: Column chart showing the average rating for each product.
- **Insights**: Identifies the highest and lowest rated products.


Sum of averagerating
0
1
2
3
4
Blood Pressure Monitor
Gloves
Medical Gown
Surgical Mask
Oximeter
ECG Machine
X-Ray Machine
Thermometer
Surgical Instruments
Stethoscope
productname