Online Retail Recommendation System

A Project Report submitted to Plasmid Innovation Pvt. Ltd. As part of my Internship Program

Raja Sree Samala
rajasrisamala2004@gmail.com

Abstract

This report details the development of a popularity-based recommendation system for Plasmid Innovation Pvt. Ltd., designed to provide data-driven insights into product sales trends. The primary goal was to create a functional system capable of identifying and recommending top-selling products from historical sales data. The project utilized the "Online Retail" dataset from the UCI Machine Learning Repository, which contains transactional records from a UK-based online retailer.

The methodology centered on a popularity-based approach, implemented by analyzing the total quantity of items sold. After extensive data preprocessing—including cleaning missing values, handling cancelled orders, and feature engineering—the system was built to generate recommendations at three distinct levels: globally, for specific countries, and for individual months.

The system successfully generated ranked lists and visualizations of top-selling items, effectively demonstrating its capabilities. Key findings revealed significant variations in product popularity across different geographical regions and clear evidence of seasonal purchasing trends. For example, analysis showed distinct consumer preferences in countries like France and Germany compared to the global average, and a surge in gift-related items during the pre-holiday season.

In conclusion, this project delivers a valuable and practical tool that provides actionable insights for inventory management, targeted marketing campaigns, and strategic business decisions. While non-personalized, it serves as a robust foundational system for understanding broad customer purchasing behavior.

Contents

1. Introduction

1.1 Background

The advent of the internet has revolutionized commerce, with online shopping becoming an integral part of daily life for millions worldwide. E-commerce platforms offer unparalleled convenience and a vast selection of products. To navigate this extensive inventory and enhance user experience, these platforms frequently employ recommendation systems. These systems analyze user behavior and product data to suggest items that a customer is likely to find relevant and appealing, thereby aiding in product discovery and often influencing purchasing decisions.

1.2 Problem Statement

While sophisticated personalized recommendation systems exist, there is often a foundational need for understanding general product popularity trends within an online retail environment. This project addresses the task of developing a robust popularity-based recommendation system for Plasmid Innovation Pvt. Ltd. The system aims to identify and suggest products based on their overall sales performance, as well as their popularity within specific contexts such as geographical regions and time periods.

1.3 Objectives

The primary objectives of this project are as follows:

- To preprocess and clean the provided online retail dataset to ensure data quality and sustainability for analysis.
- To implement a mechanism to identify globally popular products based on sales quantity.
- To develop functionality to determine popular products specific to different countries.
- To create a method for finding popular products on a month-by-month basis, allowing for the observation of seasonal trends.
- To structure these findings into a functional recommendation system that can output lists of top-selling items based on the selected criteria (global, country-wise, month-wise).

1.4 Scope

The scope of this project is focused on the development and demonstration of a popularity-based recommendation system using historical sales data. Recommendations are generated based on the total quantity of items sold. The system will provide recommendations at three levels: global, country-specific, and month-specific.

This project does *not* extend to implementing more advanced personalized recommendation techniques such as collaborative filtering or content-based filtering. Furthermore, the system does

not incorporate real-time recommendation capabilities or user interface development, focusing instead on the backend logic and data analysis for popularity-driven suggestions.

1.5 Report Structure

This report is organized as follows:

- **Section 2:** Dataset Description provides an overview of the online retail dataset used for this project, including details about its source and features.
- **Section 3:** Methodology and System Architecture outlines the data preprocessing steps undertaken, explains the logic behind the different types of popularity-based recommendations, and details the tools and libraries utilized.
- **Section 4:** Implementation and Results presents the outcomes of the recommendation system, showcasing examples of globally popular items, country-specific recommendations, and month-wise popular products, supported by visualizations.
- **Section 5:** Conclusion and Future work summarizes the project's achievements, discusses its limitations, and suggests potential avenues for future enhancements.
- Finally, the Appendix contains the complete python code developed for the project.

2. Dataset Description

This section provides a comprehensive overview of the dataset used for this project, detailing its source, structure, and key features. The dataset forms the foundation for building the popularity-based recommendation system.

2.1 Source of data

The dataset used is the "Online Retail" dataset, sourced from the UCI Machine Learning Repository and widely available on platforms like Kaggle. It contains transactional data for a UK-based and registered non-store online retail company. The company mainly sells unique all-occasion gifts, and many of its customers are wholesalers. The data covers transactions occurring between December 1, 2010, and December 9, 2011.

2.2 Overview

The dataset contains historical transaction data from the online retailer over the specified one-year period. It is structured such that each row represents a single line item within a transaction, meaning an invoice with multiple products will have multiple corresponding rows in the data. In its raw form, the dataset consists of 541,909 rows and 8 columns.

2.3 Columns/Features

The dataset is comprised of the following eight features:

Feature Name	**Data Type**	**Description**
InvoiceNo	Alphanumeric/Object	A 6-digit integral number uniquely assigned to each transaction. If this code starts with the letter 'C', it indicates a cancellation.
StockCode	Alphanumeric/Object	A 5-digit integral number uniquely assigned to each distinct product.
Description	Text/Object	The name of the product.
Quantity	Numeric (Integer)	The quantity of each product per transaction. Negative values indicate returns.
InvoiceDate	Date/Time/Object	The date and time when each transaction was generated.
UnitPrice	Numeric (Float)	The price of a single unit of the product in British pounds (£).
CustomerID	Numeric (Integer)	A 5-digit integral number uniquely assigned to each customer.
Country	Text/Object	The name of the country where each customer resides.

2.4 Initial Statistics

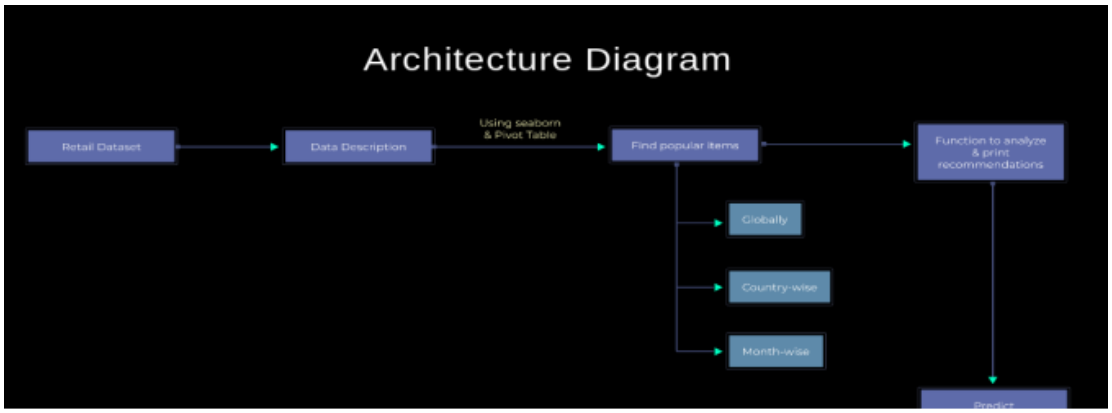
An initial exploration of the raw data was conducted using the Pandas library in Python. The output from df.info() provided a summary of the data types and non-null counts, highlighting key areas for preprocessing.

df.info() Output Summary:

```
Dataset Info:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 541909 entries, 0 to 541908

Data columns (total 8 columns):

 #   Column       Non-Null Count   Dtype

---  ------       --------------   -----
```

```
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  object
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

3.Methodology/System Architecture



3.1 Data Preprocessing

Before analysis, the raw dataset was subjected to a series of cleaning and transformation steps to ensure data quality and integrity.

- Handling Missing values: Rows with a missing CustomerID were removed. This was done because customer identity is crucial for attributing transactions, and without it, the entry lacks key contextual information. Rows with a missing Description were also dropped as the product name is essential for a human-readable recommendation.
- Handling Erroneous Data:

- Cancelled Orders: Transactions with an InvoiceNo starting with the letter 'C' were filtered out as they represent cancellations, not purchases.
- Negative/Zero Quantities: Rows with a Quantity of zero or less were removed, as only positive-quantity purchases are relevant for a popularity-based system.
- Invalid Prices: Items with a UnitPrice of zero were excluded to remove transactions that were likely not standard sales (e.g., promotional items, data errors).
- Non-product Items: Stock codes and descriptions related to non-product charges like 'POSTAGE', 'CARRIAGE', or 'Manual' were identified and removed to ensure that the recommendations consist only of actual retail products.
- Data Type conversion:
- The InvoiceDate column was converted from a string format to a datetime object. This is a critical step that enables time-based analysis, such as filtering by month and year.
- CustomerID was converted to an integer type for consistency.
- Feature Engineering: To support the recommendation logic, new features were created from existing data.
- Year and Month: These were extracted from the InvoiceDate column to allow for easy grouping of data by specific months.
- TotalPrice: This was calculated by multiplying Quantity by unitprice to understand the monetary value of each transaction line, though it was not directly used for the popularity logic in this project.

3.2 Recommendation Logic

The core of the recommendation system is based on item popularity, measured by the total quantity sold. This logic was implemented at three different levels using the Pandas library's powerful groupby() and sorting functionalities.

- *Global popularity:* To find the most popular items across all transactions, the dataset was grouped by StockCode and Description. The Quantity for each group was then summed up, and the results were sorted in descending order. The items at the top of this sorted list are the global best-sellers.
- *Country-wise Popularity:* To generate country-specific recommendations, the dataset was first filtered to include transactions from only the target country. Then, the same global popularity logic (grouping by product, summing quantities, and sorting) was applied to this smaller, country-specific subset of data.

- *Month-wise Popularity:* Similarly, for month-specific recommendations, the dataset was filtered by a specific Year and Month. The popularity logic was then applied to this temporal subset to identify the best-selling items for that particular period.

3.3 Tools and Libraries used

The project was developed entirely in Python 3. The following key libraries were instrumental in its implementation:

- *Pandas:* Used for all core data manipulation tasks, including loading the data, cleaning, preprocessing, and performing the grouping and aggregation logic for finding popular items.
- *Numpy:* Provided support for numerical operations and handling data structures.
- *Matplotlib & Seaborn:* Used for creating data visualizations, such as the bar charts that display the top recommended products, making the results easy to interpret.

4. Implementation and Results

This section presents the results generated by the recommendation system. The outputs for global, country-wise, and month-wise recommendations are displayed and discussed, demonstrating the system's functionality and the insights derived from the data.

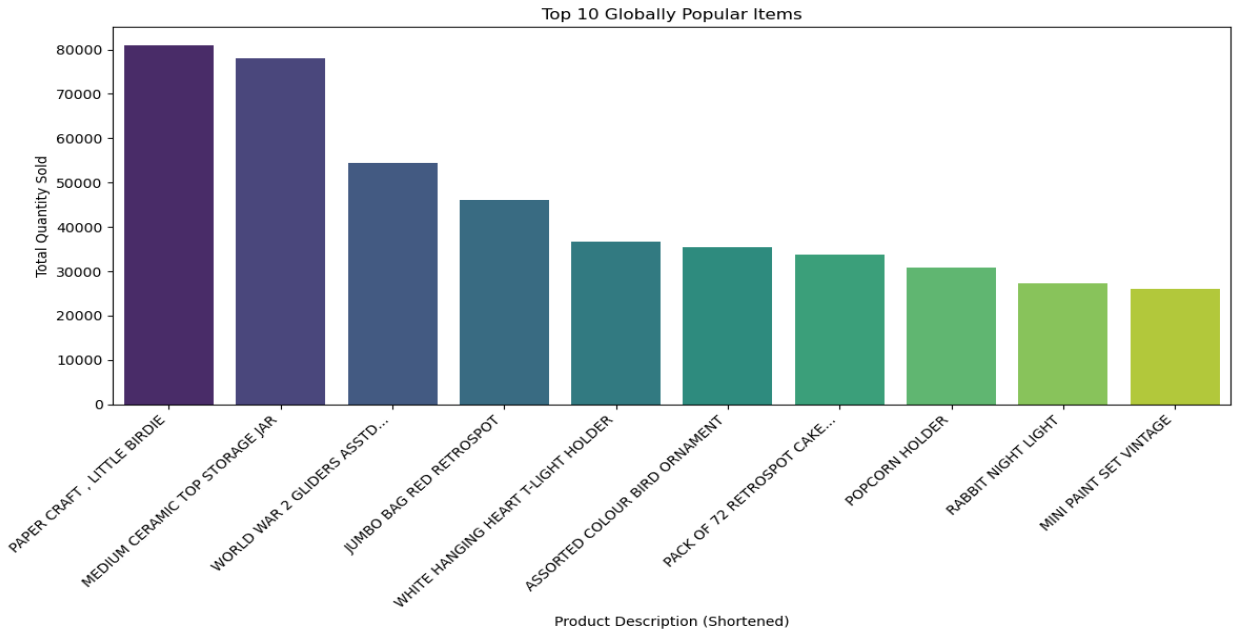
4.1 Global Recommendations

The system was first used to identify the most popular products across the entire dataset, providing a baseline understanding of top-selling items. The get_global_recommendations() function was called to retrieve the top 5 products by total quantity sold.

```
Top 10 Globally Popular Items:

StockCode         Description                Quantity

23843      PAPER CRAFT, LITTLE BIRDIE           80995

23166      MEDIUM CERAMIC TOP STORAGE JAR        77916

84077      WORLD WAR 2 GLIDERS ASSTD DESIGNS     54415

85099B     JUMBO BAG RED RETROSPOT               46181
```

```
85123A      WHITE HANGING HEART T-LIGHT HOLDER     36725

84879       ASSORTED COLOUR BIRD ORNAMENT          35362

21212       PACK OF 72 RETROSPOT CAKE CASES        33693

22197       POPCORN HOLDER                         30931

23084       RABBIT NIGHT LIGHT                     27202

22492       MINI PAINT SET VINTAGE                 26076

Name: Quantity, dtype: int64
```

The corresponding visualization for these top items is shown in the bar chart below.



Interpretation:

The global results indicate that items like "PAPER CRAFT, LITTLE BIRDIE" and "MEDIUM CERAMIC TOP STORAGE JAR" are the most popular by volume. These items are generally low-cost, craft-related, or general-purpose household goods. The high quantities sold suggest that many of the customers may be wholesalers or retailers purchasing stock, which aligns with the dataset's description. These global recommendations are useful for high-level inventory management and general marketing campaigns.

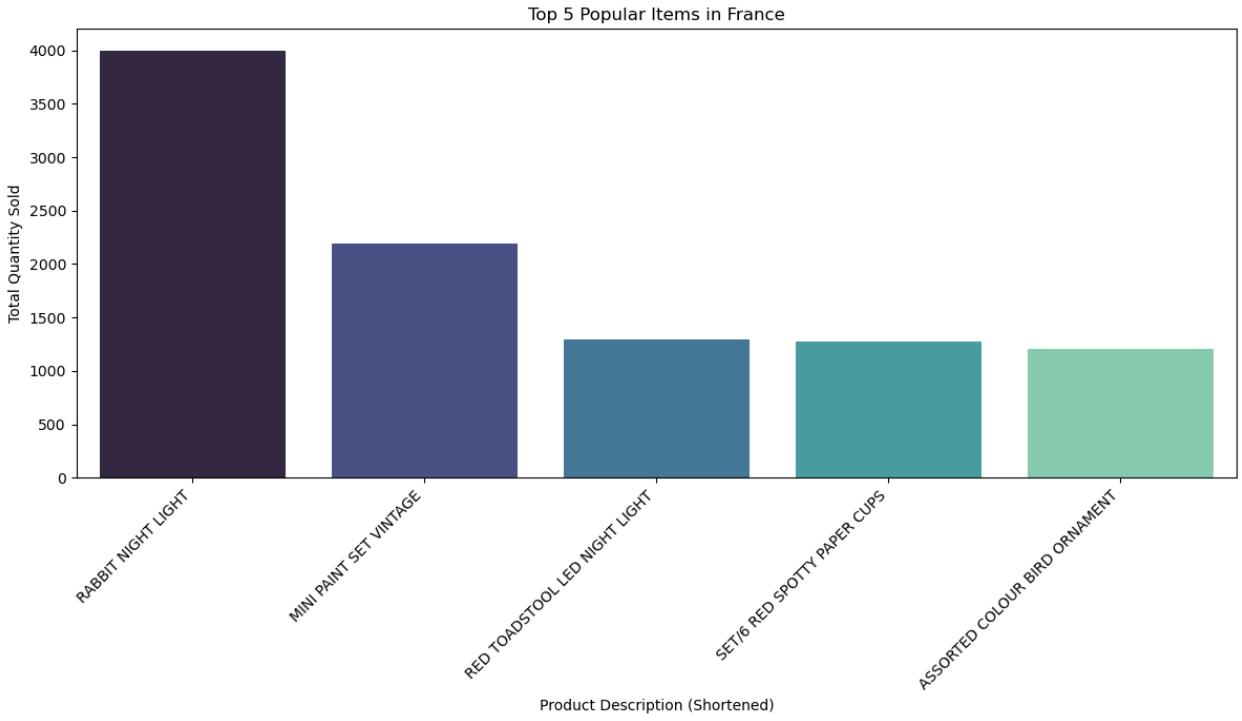
4.2 Country-wise Recommendations

To demonstrate more targeted recommendations, the logic was applied to specific countries. This shows how product popularity can differ across geographical regions. The country 'France' was chosen as an example.

France:

The following results show the top 5 most popular items in France.

```
Top 5 Popular items in France:

StockCode              Description       Quantity
6637                RABBIT NIGHT LIGHT       4000
6254            MINI PAINT SET VINTAGE       2196
5913     RED TOADSTOOL LED NIGHT LIGHT       1291
5708       SET/6 RED SPOTTY PAPER CUPS       1272
7053     ASSORTED COLOUR BIRD ORNAMENT       1204
```



Interpretation:

The country-wise analysis for France reveals a strong preference for decorative and novelty items, showcasing a market with distinct tastes. While ‘RABIT NIGHT LIGHT’ may indicate a high volume of orders, the top products themselves tell a clear story. French customers strongly favor items such as the " MINI PAINT SET VINTAGE" and the " RED TOADSTOOL LED NIGHT LIGHT" series, which appeared in multiple colors within the top sellers. This purchasing behavior points towards a focus on home decor, particularly for children's rooms or as gifts. This demonstrates the value of country-specific recommendations, as the practical items popular in Germany or the craft supplies popular globally would not resonate as strongly in the French market. This information is vital for curating product selections and marketing campaigns specifically for French consumer aesthetics.

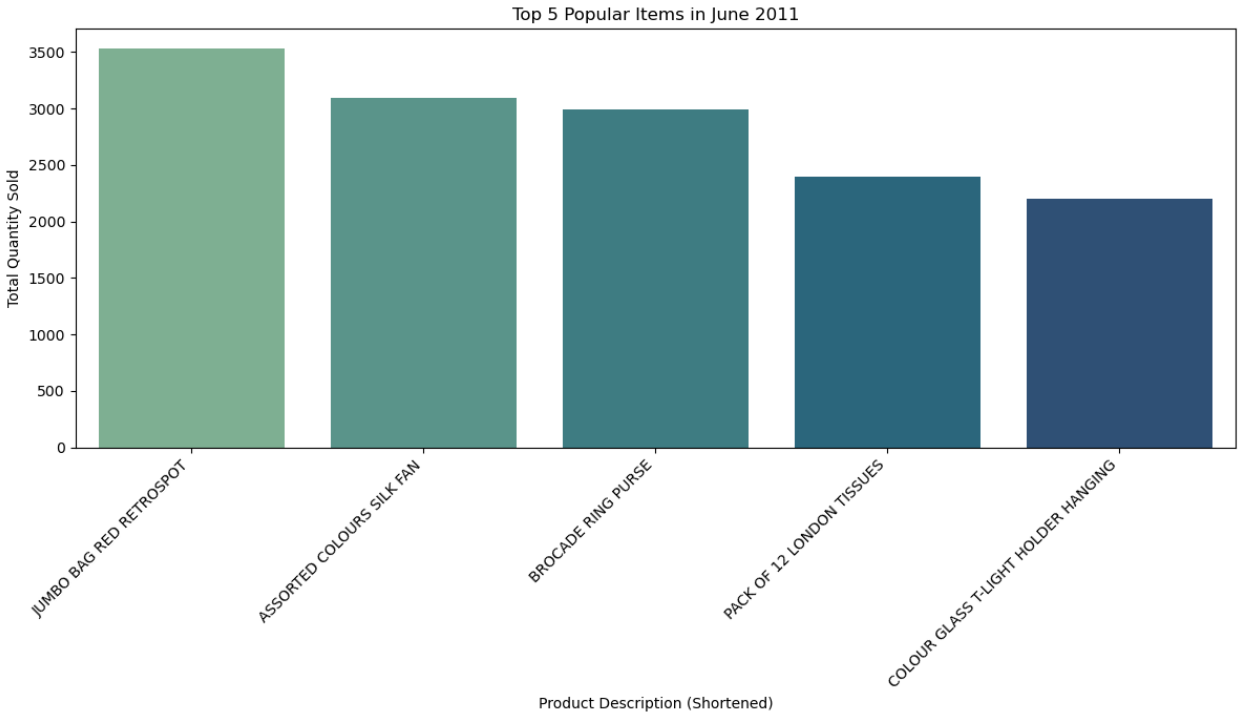
4.3 Month-Wise Recommendations

To analyze seasonal demand, the recommendation logic was applied to specific months. A comparison between a mid-year month (June 2011) and a peak holiday shopping month (November 2011) highlights these trends.

June 2011:

```
Top 5 Popular Items in June 2011:
 StockCode             Description            Quantity
15629              JUMBO BAG RED RETROSPOT      3529
13411            ASSORTED COLOURS SILK FAN      3096
13454                   BROCADE RING PURSE      2988
14543            PACK OF 12 LONDON TISSUES      2397
15499  COLOUR GLASS T-LIGHT HOLDER HANGING      2204
```

Top 5 Popular Items in June 2011 by Quantity Sold

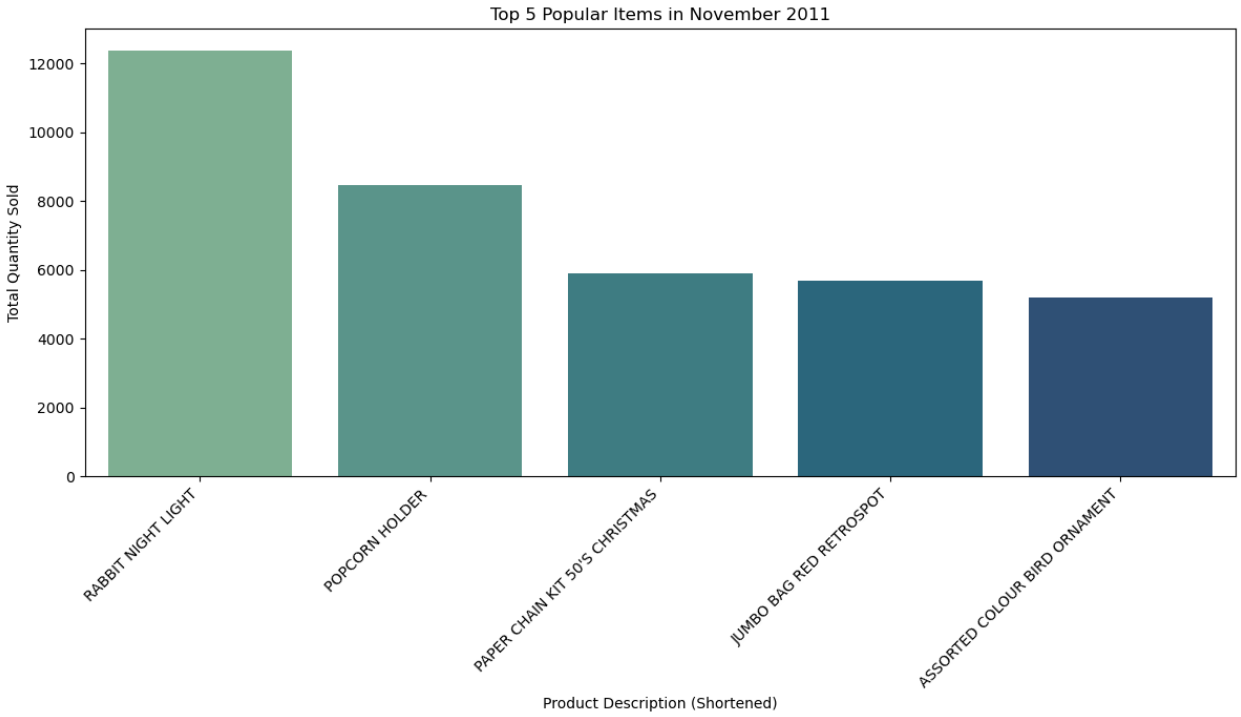


November 2011:

Top 5 Popular Items in November 2011:

StockCode	Description	Quantity
27364	RABBIT NIGHT LIGHT	12393
26620	POPCORN HOLDER	8458
26531	PAPER CHAIN KIT 50'S CHRISTMAS	5919
28310	JUMBO BAG RED RETROSPOT	5678
28207	ASSORTED COLOUR BIRD ORNAMENT	5190

Top 5 Popular Items in November 2011 by Quantity Sold



Interpretation:

The month-wise results clearly illustrate seasonal purchasing behavior. The top items in November 2011, such as, 'RABBIT NIGHT LIGHT', 'PAPER CHAIN KIT 50'S CHRISTMAS' are strongly indicative of preparation for the Christmas holiday season. In contrast, the popular items from June 2011 reflect more general, non-seasonal purchasing. This analysis is extremely valuable for inventory planning, allowing a business to stock up on seasonally popular items ahead of peak demand and to promote different products at different times of the year.

5.Conclusion and Future work

This section summarizes the project's achievements and findings, acknowledges its limitations, and proposes potential directions for future development to enhance its capabilities.

5.1 Conclusion

This project successfully achieved its objective of developing a functional popularity-based recommendation system for an online retail dataset. By methodically preprocessing and analyzing

historical transaction data, a system was built that can generate valuable insights into product popularity at different scales.

The main achievements of this project include:

- The successful cleaning and preparation of a large, real-world transactional dataset, making it suitable for analysis.
- The implementation of a core recommendation logic to identify top-selling products based on quantity sold.
- The development of three distinct recommendation functionalities:
  1. **Global Recommendations:** Providing a high-level view of the most popular items across the entire customer base
  2. **Country-Wise Recommendations:** Offering targeted insights into regional purchasing preferences.
  3. **Month-Wise Recommendations:** Revealing seasonal trends and demand fluctuations over time.

The system effectively demonstrates that even a straightforward, non-personalized approach can provide actionable business intelligence for marketing, inventory management, and strategic planning. The project serves as a foundational step in leveraging data to understand customer behavior and product performance

5.2 Limitations

While the implemented system is effective for its intended purpose, it is important to acknowledge its inherent limitations:

- **No Personalization:** The system recommends the same popular items to every user within a given segment (e.g., everyone in France gets the same recommendations). It does not adapt to individual user tastes or purchase history.
- **Popularity Bias:** The system has a natural tendency to promote items that are already popular, potentially creating a feedback loop where popular items become even more popular. This can lead to a lack of diversity in recommendations and makes it difficult for new or niche products to gain visibility.
- **Cold-Start Problem:** The system cannot recommend new products. A new item will have zero sales history, and therefore, it will never appear in a popularity-based list until it has accumulated a significant number of purchases.

5.3 Future work

To address the limitations and build a more sophisticated and effective recommendation engine, the following enhancements are proposed for future work:

- **Implement Collaborative Filtering:** This would be the most significant upgrade. By analyzing the user-item interaction matrix, this technique could provide personalized

recommendations based on the preferences of similar users ("Users who bought this also bought..." or "Users like you also liked..."). This would directly address the lack of personalization.

- **Develop a Content-Based Filtering System:** By analyzing the textual content of the Description field, a content-based model could recommend products with similar attributes. For example, if a user buys a "RED SPOTTED MUG", it could recommend a "BLUE SPOTTED MUG" or a "RED SPOTTED BOWL". This would help with product discovery and can work well even for new items if their descriptions are available.
- **Create a Hybrid Model:** A hybrid approach that combines the current popularity-based model with collaborative and/or content-based filtering could offer the best of all worlds. Popular items could be used as a fallback for new users (solving the user cold-start problem), while personalized recommendations are served to users with an established history.
- **Develop a User Interface (UI):** Building a simple graphical user interface or a web-based API would make the system interactive. This would allow a user (e.g., a marketing manager) to easily select a country or month and instantly receive a list of recommended products without needing to run the Python script.

6. Appendix: Python Code

1. Importing Libraries and Dataset loading and initial description

```
>import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime

dataset_path = 'Online_Retail.csv'

try:
    df_retail = pd.read_csv(dataset_path, encoding='latin1')
    print(f"Successfully loaded dataset from: {dataset_path}")
except FileNotFoundError:
    print(f"ERROR: Dataset file '{dataset_path}' not found.")
    print("Please ensure the file is in the correct location or update the 'dataset_path'
variable in the script.")
    df_retail = None

Successfully loaded dataset from: Online_Retail.csv

> df_retail
```

```
if df_retail is not None:
    print("Dataset Shape:", df_retail.shape)
    print("\nFirst 5 rows of the dataset:")
    print(df_retail.head())

    print("\nDataset Info:")
    df_retail.info()

    print("\nDescriptive Statistics:")
    print(df_retail.describe(include='all'))
else:
    print("Dataset could not be loaded. Please check the file path and try again.")
```

2. Data preprocessing and EDA

```
if df_retail is not None:
```

2.1 Handling Missing Values

```
    print("Missing values before cleaning:")

    print(df_retail.isnull().sum())


    # Drop rows where CustomerID is missing, as it's often important for user-based analysis

    # For purely popularity-based on StockCode, this might be optional, but good practice.

    df_retail.dropna(subset=['CustomerID'], inplace=True)


    # Fill missing Descriptions with 'UNKNOWN' or drop them.

    # For popularity, Description is key for human readability.

    df_retail.dropna(subset=['Description'], inplace=True) # Drop rows with missing description


    print("\nMissing values after CustomerID & Description handling:")

    print(df_retail.isnull().sum())

print("Dataset Shape after handling missing values:", df_retail.shape)
```

2.2 Data Cleaning

```
# - Remove cancelled transactions (InvoiceNo starts with 'C').
# - Remove items with negative or zero quantity.
# - Remove items with zero or negative unit price.
```

```
    # - Remove rows where StockCode might be non-product related (e.g., 'POST', 'D', 'M',
'BANK CHARGES', 'CRUK')df_retail['InvoiceNo'] = df_retail['InvoiceNo'].astype(str)
df_retail = df_retail[~df_retail['InvoiceNo'].str.startswith('C', na=False)]

    # Remove rows with non-positive Quantity
df_retail = df_retail[df_retail['Quantity'] > 0]

    # Remove rows with non-positive UnitPrice (or very low, e.g., < 0.001)
df_retail = df_retail[df_retail['UnitPrice'] > 0.001]

    # Remove common non-product stock codes (this list can be expanded based on EDA)
non_product_codes = ['POST', 'D', 'M', 'BANK CHARGES', 'CRUK', 'S', 'AMAZONFEE', 'DOT',
'TEST001', 'TEST002', 'ADJUST', 'ADJUST2']
df_retail = df_retail[~df_retail['StockCode'].astype(str).isin(non_product_codes)]
    # Also, some descriptions might indicate non-product items
df_retail =
df_retail[~df_retail['Description'].astype(str).str.contains('POSTAGE|CARRIAGE|Manual|Disc
ount|AMAZON FEE', case=False, na=False)]

print("\nDataset Shape after cleaning (cancellations, quantity, price, non-product
codes):", df_retail.shape)
```

Dataset Shape after cleaning (cancellations, quantity, price, non-product codes): (396196, 8)

2.3 Data Type Conversion & Feature Engineering

```
df_retail['InvoiceDate'] = pd.to_datetime(df_retail['InvoiceDate'])

    # Extract Month and Year
df_retail['Year'] = df_retail['InvoiceDate'].dt.year
df_retail['Month'] = df_retail['InvoiceDate'].dt.month
df_retail['MonthName'] = df_retail['InvoiceDate'].dt.strftime('%B') # For better display

    # Create TotalPrice column
df_retail['TotalPrice'] = df_retail['Quantity'] * df_retail['UnitPrice']

print("\nData with new features (Year, Month, TotalPrice):")
print(df_retail[['InvoiceDate', 'Year', 'Month', 'MonthName', 'TotalPrice']].head())
```

3. Architecture Implementation: Finding Popular Items

3.1 Globally Popular Items

```
globally_popular_items = df_retail.groupby(['StockCode',
'Description'])['Quantity'].sum().sort_values(ascending=False)

print("Top 10 Globally Popular Items:")
print(globally_popular_items.head(10))
```

```
# visualizing using seaborn
plt.figure(figsize=(12, 7)) # Adjusted size
top_n_global = 10
    # Taking only the first few words of description for cleaner plot labels
global_desc_short = [' '.join(desc.split()[:5]) + ('...' if len(desc.split()) > 5 else
'') for desc in
globally_popular_items.head(top_n_global).index.get_level_values('Description')]

sns.barplot(x=global_desc_short,
y=globally_popular_items.head(top_n_global).values,
               palette='viridis')
plt.xlabel("Product Description (Shortened)")
plt.ylabel("Total Quantity Sold")
plt.title(f"Top {top_n_global} Globally Popular Items")
plt.xticks(rotation=45, ha='right') # Adjusted rotation
plt.tight_layout()
plt.show()
```

3.2 country-wise popular items

```
country_popular_items = df_retail.groupby(['Country', 'StockCode',
'Description'])['Quantity'].sum().reset_index()
country_popular_items = country_popular_items.sort_values(['Country', 'Quantity'],
ascending=[True, False])

country_example = 'France' # Example country
print(f"\nAvailable countries: {df_retail['Country'].nunique()}")
print(f"Top 5 countries by sales volume:
{df_retail.groupby('Country')['Quantity'].sum().nlargest(5).index.tolist()}")

if country_example in country_popular_items['Country'].unique():
            top_items_country = country_popular_items[country_popular_items['Country']
== country_example].head(10)
print(f"\nTop 10 Popular Items in {country_example}:")
print(top_items_country[['Description', 'Quantity']])

#visualization
plt.figure(figsize=(12, 7)) # Adjusted size
 top_n_country = 5
 country_desc_short = [' '.join(desc.split()[:5]) + ('...' if len(desc.split()) > 5
else '') for desc in top_items_country.head(top_n_country)['Description']]

 sns.barplot(x=country_desc_short,
                    y=top_items_country.head(top_n_country)['Quantity'],
                    palette='mako')
 plt.xlabel("Product Description (Shortened)")
 plt.ylabel("Total Quantity Sold")
 plt.title(f"Top {top_n_country} Popular Items in {country_example}")
 plt.xticks(rotation=45, ha='right') # Adjusted rotation
 plt.tight_layout()
 plt.show()
```

3.3 Month-wise popular items

```
month_popular_items = df_retail.groupby(['Year', 'Month', 'MonthName', 'StockCode',
'Description'])['Quantity'].sum().reset_index()
month_popular_items = month_popular_items.sort_values(['Year', 'Month', 'Quantity'],
ascending=[True, True, False])

    # Example: Top items in a specific month and year
    # Let's pick a month with likely high sales, e.g., November 2011 (pre-Christmas)
year_example = 2011
month_example_num = 11 # November
month_example_name = datetime(2000, month_example_num, 1).strftime('%B')


filtered_month_data = month_popular_items[
        (month_popular_items['Year'] == year_example) & (month_popular_items['Month']
== month_example_num)
    ]

if not filtered_month_data.empty:
         top_items_month_year = filtered_month_data.head(10)
print(f"\nTop 10 Popular Items in {month_example_name} {year_example}:")
print(top_items_month_year[['Description', 'Quantity']])
```

```
>month_popular_items = df_retail.groupby(['Year', 'Month', 'MonthName', 'StockCode',
'Description'])['Quantity'].sum().reset_index()
```

```
month_popular_items = month_popular_items.sort_values(['Year', 'Month', 'Quantity'],
ascending=[True, True, False])

    # Example: Top items in a specific month and year
    # Let's pick a month with likely high sales, e.g., June 2011
year_example = 2011
month_example_num = 6 # June
month_example_name = datetime(2000, month_example_num, 1).strftime('%B')


filtered_month_data = month_popular_items[
        (month_popular_items['Year'] == year_example) & (month_popular_items['Month']
== month_example_num)
    ]

if not filtered_month_data.empty:
         top_items_month_year = filtered_month_data.head(10)
print(f"\nTop 10 Popular Items in {month_example_name} {year_example}:")
print(top_items_month_year[['Description', 'Quantity']])


#visualization
plt.figure(figsize=(12, 7)) # Adjusted size
top_n_month = 5
```

```
month_desc_short = [' '.join(desc.split()[:5]) + ('...' if len(desc.split()) > 5 else
'') for desc in top_items_month_year.head(top_n_month)['Description']]

sns.barplot(x=month_desc_short,
                    y=top_items_month_year.head(top_n_month)['Quantity'],
                    palette='crest')
plt.xlabel("Product Description (Shortened)")
plt.ylabel("Total Quantity Sold")
plt.title(f"Top {top_n_month} Popular Items in {month_example_name} {year_example}")
plt.xticks(rotation=45, ha='right') # Adjusted rotation
plt.tight_layout()
plt.show()

#visualization
plt.figure(figsize=(12, 7)) # Adjusted size
top_n_month = 5
month_desc_short = [' '.join(desc.split()[:5]) + ('...' if len(desc.split()) > 5 else
'') for desc in top_items_month_year.head(top_n_month)['Description']]

sns.barplot(x=month_desc_short,
                    y=top_items_month_year.head(top_n_month)['Quantity'],
                    palette='crest')
plt.xlabel("Product Description (Shortened)")
plt.ylabel("Total Quantity Sold")
plt.title(f"Top {top_n_month} Popular Items in {month_example_name} {year_example}")
plt.xticks(rotation=45, ha='right') # Adjusted rotation
plt.tight_layout()
plt.show()
```

4. Function to Analyze and print recommendations (predict)

```
def get_global_recommendations(data, top_n=5):
        """Returns the top N globally popular items."""
        if data is None or data.empty:
            return "Dataset not loaded or is empty."
        popular_items = data.groupby(['StockCode',
'Description'])['Quantity'].sum().sort_values(ascending=False)
        return popular_items.head(top_n)
def get_country_recommendations(data, country, top_n=5):
        """Returns the top N popular items for a specific country."""
        if data is None or data.empty:
            return "Dataset not loaded or is empty."
        country_data = data[data['Country'] == country]
        if country_data.empty:
            return pd.Series(dtype='float64', name=f"No data available for country:
{country}") # Return empty Series for consistency
        popular_items = country_data.groupby(['StockCode',
'Description'])['Quantity'].sum().sort_values(ascending=False)
        return popular_items.head(top_n)
def get_month_recommendations(data, year, month_num, top_n=5): # Changed to month_num

        #Returns the top N popular items for a specific year and month number."""
```

```
        if data is None or data.empty:
            return "Dataset not loaded or is empty."

        month_name = datetime(2000, month_num, 1).strftime('%B')
        month_data = data[(data['Year'] == year) & (data['Month'] == month_num)]
        if month_data.empty:
            return pd.Series(dtype='float64', name=f"No data available for
{month_name} {year}") # Return empty Series
        popular_items = month_data.groupby(['StockCode',
'Description'])['Quantity'].sum().sort_values(ascending=False)
        return popular_items.head(top_n)
```

5. Demonstration of Recommendation System

```
if df_retail is not None and not df_retail.empty:
        print("--- Global Recommendations ---")
        global_recs = get_global_recommendations(df_retail, top_n=5)
        print(global_recs)
        print("-" * 30)

        target_country = 'Germany' # A country likely to have data
        print(f"\n--- {target_country} Recommendations ---")
        country_recs = get_country_recommendations(df_retail, country=target_country,
top_n=5)
        if isinstance(country_recs, pd.Series) and not country_recs.empty:
            print(country_recs)
        else:
            print(f"No recommendations for {target_country} (or data empty).")
        print("-" * 30)

        target_year = 2011
        target_month = 11 # November
        month_name_display = datetime(2000, target_month, 1).strftime('%B')
        print(f"\n--- {month_name_display} {target_year} Recommendations ---")
        month_recs = get_month_recommendations(df_retail, year=target_year,
month_num=target_month, top_n=5)
        if isinstance(month_recs, pd.Series) and not month_recs.empty:
            print(month_recs)
        else:
            print(f"No recommendations for {month_name_display} {target_year} (or data
empty).")
        print("-" * 30)

        # Example for a country that might not exist or have little data
        target_country_no_data = 'Neverland'
        print(f"\n--- {target_country_no_data} Recommendations ---")
        country_recs_no_data = get_country_recommendations(df_retail,
country=target_country_no_data, top_n=5)
        if isinstance(country_recs_no_data, pd.Series) and not
country_recs_no_data.empty:
            print(country_recs_no_data)
        else: # This branch will likely be taken if 'Neverland' is not in the data
```

```
            print(f"No recommendations for {target_country_no_data} (or data empty).")
        print("-" * 30)
else:
        print("Skipping recommendation demonstration as dataset was not loaded or is
empty after processing.")
```