- **Name: Aashi Talwar**
- **Data Science Major Project**
- **Title: Recommendation System**

Online Retail Recommendation System Project

- **1. Project Goal and Dataset**

Goal: I have developed a recommendation system to suggest products to online shoppers, similar to features on popular e-commerce websites. This system aims to improve user experience and potentially drive sales.

Dataset: I have utilized the "Online Retail" dataset from Kaggle for this project. This dataset provides valuable transactional information about an online retail store.

Columns: I have worked with columns such as invoice number, product descriptions, quantities, customer IDs, and countries, each of which played a crucial role in building the recommendation system.

- **2. Data Preprocessing and Exploration**

Data Cleaning: I have cleaned the data by handling missing values in the 'CustomerID' column, removing duplicates, and converting the 'InvoiceDate' to a suitable format for analysis.

Exploratory Data Analysis (EDA): I have explored the dataset to gain insights into product popularity. I have identified globally popular items, country-wise popular items, and month-wise popular items. To visualize these trends, I have used Seaborn and Matplotlib libraries to create bar plots and heatmaps.

- **3. Recommendation System Development**

- **Data Sampling:** I have sampled 20% of the original dataset to ensure faster processing during the model development and testing phase.

- **Feature Engineering:** To build the recommendation system, I have focused on product descriptions.

TF-IDF Vectorization: I have used TF-IDF to convert product descriptions into numerical vectors, which enable comparisons based on word importance.

- **Dimensionality Reduction:** I have applied Truncated SVD to reduce the complexity of the data and improve efficiency.

- **Similarity Calculation (KNN):**

I have implemented the K-Nearest Neighbors algorithm with cosine similarity to identify products similar to a given product. This approach leverages the reduced TF-IDF vectors to find neighbors in the product space.

- **4. User Input and Recommendations**

- **Streamlit Integration:** I have developed an interactive web application using Streamlit to showcase the recommendation system.

User Input: The application allows users to input a product name.

Product Matching: I have implemented fuzzy string matching to ensure accurate product identification, even with minor spelling errors in the user's input.

Recommendation Generation: Once a product is matched, the KNN model retrieves the most similar products based on the pre-calculated similarity matrix.

Displaying Recommendations: The application then displays these recommended products to the user.

- **5. Conclusion and Future Enhancements**

I have successfully built and deployed an online retail recommendation system using Python and Streamlit. For future enhancements, I plan to explore other algorithms, incorporate user preferences, and improve the application's user interface.**

Importing Necessary libraries

In [63]:
```python
import streamlit as st
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD
from sklearn.neighbors import NearestNeighbors
from thefuzz.process import extractOne
```

In [22]:
```
pip install thefuzz
```

```
Collecting thefuzz
  Downloading thefuzz-0.22.1-py3-none-any.whl.metadata (3.9 kB)
Collecting rapidfuzz<4.0.0,>=3.0.0 (from thefuzz)
  Downloading rapidfuzz-3.12.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (12 kB)
Downloading thefuzz-0.22.1-py3-none-any.whl (8.2 kB)
Downloading rapidfuzz-3.12.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.1/3.1 MB 25.7 MB/s eta 0:00:00
Installing collected packages: rapidfuzz, thefuzz
Successfully installed rapidfuzz-3.12.2 thefuzz-0.22.1
```

In [61]:
```
pip install streamlit pandas numpy scikit-learn thefuzz matplotlib seaborn
```

```
Collecting streamlit
  Downloading streamlit-1.43.2-py2.py3-none-any.whl.metadata (8.9 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (2.0.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: thefuzz in /usr/local/lib/python3.11/dist-packages (0.22.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0.13.2)
Requirement already satisfied: altair<6,>=4.0 in /usr/local/lib/python3.11/dist-packages (from streamlit) (5.5.0)
Requirement already satisfied: blinker<2,>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from streamlit) (1.9.0)
Requirement already satisfied: cachetools<6,>=4.0 in /usr/local/lib/python3.11/dist-packages (from streamlit) (5.5.2)
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.11/dist-packages (from streamlit) (8.1.8)
Requirement already satisfied: packaging<25,>=20 in /usr/local/lib/python3.11/dist-packages (from streamlit) (24.2)
Requirement already satisfied: pillow<12,>=7.1.0 in /usr/local/lib/python3.11/dist-packages (from streamlit) (11.1.0)
Requirement already satisfied: protobuf<6,>=3.20 in /usr/local/lib/python3.11/dist-packages (from streamlit) (4.25.6)
Requirement already satisfied: pyarrow>=7.0 in /usr/local/lib/python3.11/dist-packages (from streamlit) (18.1.0)
Requirement already satisfied: requests<3,>=2.27 in /usr/local/lib/python3.11/dist-packages (from streamlit) (2.32.3)
Requirement already satisfied: tenacity<10,>=8.1.0 in /usr/local/lib/python3.11/dist-packages (from streamlit) (9.0.0)
Requirement already satisfied: toml<2,>=0.10.1 in /usr/local/lib/python3.11/dist-packages (from streamlit) (0.10.2)
Requirement already satisfied: typing-extensions<5,>=4.4.0 in /usr/local/lib/python3.11/dist-packages (from streamlit) (4.12.2)
Collecting watchdog<7,>=2.1.5 (from streamlit)
  Downloading watchdog-6.0.0-py3-none-manylinux2014_x86_64.whl.metadata (44 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.3/44.3 kB 2.7 MB/s eta 0:00:00
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in /usr/local/lib/python3.11/dist-packages (from streamlit) (3.1.44)
Collecting pydeck<1,>=0.8.0b4 (from streamlit)
  Downloading pydeck-0.9.1-py2.py3-none-any.whl.metadata (4.1 kB)
Requirement already satisfied: tornado<7,>=6.0.3 in /usr/local/lib/python3.11/dist-packages (from streamlit) (6.4.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.1)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: rapidfuzz<4.0.0,>=3.0.0 in /usr/local/lib/python3.11/dist-packages (from thefuzz) (3.12.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (4.56.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from altair<6,>=4.0->streamlit) (3.1.6)
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.11/dist-packages (from altair<6,>=4.0->streamlit) (4.23.0)
Requirement already satisfied: narwhals>=1.14.2 in /usr/local/lib/python3.11/dist-packages (from altair<6,>=4.0->streamlit) (1.30.0)
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.11/dist-packages (from gitpython!=3.1.19,<4,>=3.0.7->streamlit) (4.0.12)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.27->streamlit) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.27->streamlit) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.27->streamlit) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.27->streamlit) (2025.1.31)
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.11/dist-packages (from gitdb<5,>=4.0.1->gitpython!=3.1.19,<4,>=3.0.7->streamlit) (5.0.2)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->altair<6,>=4.0->streamlit) (3.0.2)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (25.3.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (2024.10.1)
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (0.36.2)
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (0.23.1)
Downloading streamlit-1.43.2-py2.py3-none-any.whl (9.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.7/9.7 MB 48.9 MB/s eta 0:00:00
Downloading pydeck-0.9.1-py2.py3-none-any.whl (6.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.9/6.9 MB 55.6 MB/s eta 0:00:00
Downloading watchdog-6.0.0-py3-none-manylinux2014_x86_64.whl (79 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.1/79.1 kB 4.7 MB/s eta 0:00:00
Installing collected packages: watchdog, pydeck, streamlit
Successfully installed pydeck-0.9.1 streamlit-1.43.2 watchdog-6.0.0
```

Loading the dataset

In [2]:
```python
# Step 1: Load the dataset
file_path = "/OnlineRetail (1).xlsx"  # Update with your file path
df = pd.read_excel(file_path)
df
```

Out[2]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
...	...	...	...	...	...	...	...	...
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	2011-12-09 12:50:00	0.85	12680.0	France
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09 12:50:00	2.10	12680.0	France
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09 12:50:00	4.15	12680.0	France
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09 12:50:00	4.15	12680.0	France
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09 12:50:00	4.95	12680.0	France

541909 rows × 8 columns

In [3]:
```python
# Step 2: Data Cleaning & Description
df.dropna(subset=["CustomerID"], inplace=True)
df.drop_duplicates(inplace=True)
```

In [4]:
```python
# Convert Invoice Date to datetime and extract Month
df["InvoiceDate"] = pd.to_datetime(df["InvoiceDate"])
df["Month"] = df["InvoiceDate"].dt.month
```

Globally Popular Items

In [5]:
```python
# Step 3: Finding Popular Items (Globally, Country-wise, Month-wise)

## Globally Popular Items
popular_items_global = df["Description"].value_counts().head(10)
popular_items_global
```

Out[5]:

Description	count
WHITE HANGING HEART T-LIGHT HOLDER	2058
REGENCY CAKESTAND 3 TIER	1894
JUMBO BAG RED RETROSPOT	1659
PARTY BUNTING	1409
ASSORTED COLOUR BIRD ORNAMENT	1405
LUNCH BAG RED RETROSPOT	1345
SET OF 3 CAKE TINS PANTRY DESIGN	1224
POSTAGE	1196
LUNCH BAG BLACK SKULL.	1099
PACK OF 72 RETROSPOT CAKE CASES	1062

dtype: int64

Country-wise Popular Items

In [6]:

```
## Country-wise Popular Items
popular_items_country = df.groupby("Country")["Description"].value_counts().groupby(level=0).head(3)
popular_items_country
```

Out[6]:

Country	Description	count
Australia	SET OF 3 CAKE TINS PANTRY DESIGN	10
	LUNCH BAG RED RETROSPOT	9
	RED TOADSTOOL LED NIGHT LIGHT	9
Austria	POSTAGE	14
	RETROSPOT TEA SET CERAMIC 11 PC	4
...	...	...
United Kingdom	REGENCY CAKESTAND 3 TIER	1564
	JUMBO BAG RED RETROSPOT	1502
Unspecified	ASSORTED COLOUR BIRD ORNAMENT	3
	SET OF 10 LED DOLLY LIGHTS	3
	12 MESSAGE CARDS WITH ENVELOPES	2

111 rows × 1 columns

dtype: int64

Month-wise Popular Items

In [8]:

```
## Month-wise Popular Items
popular_items_month = df.groupby("Month")["Description"].value_counts().groupby(level=0).head(3)
popular_items_month
```

Out[8]:

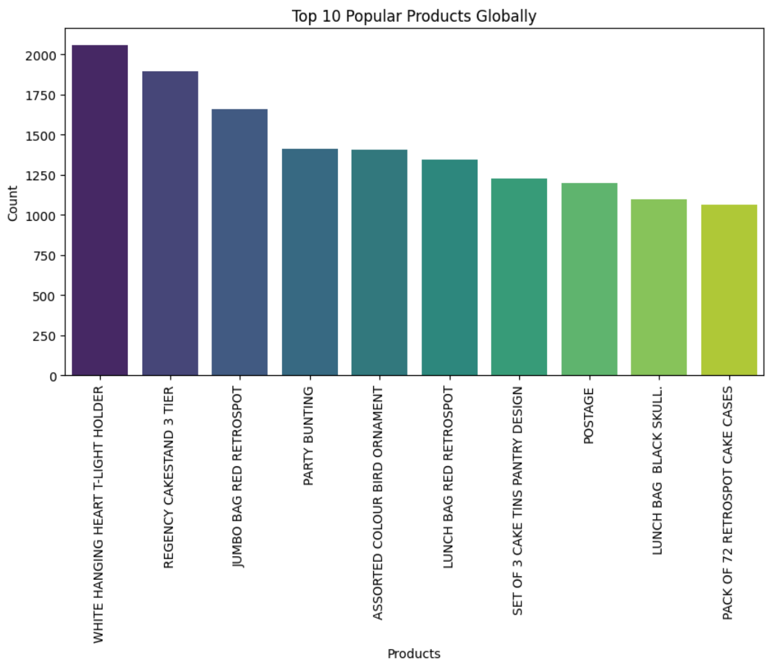
Month	Description	count
1	WHITE HANGING HEART T-LIGHT HOLDER	164
	SET OF 3 CAKE TINS PANTRY DESIGN	137
	REGENCY CAKESTAND 3 TIER	132
2	WHITE HANGING HEART T-LIGHT HOLDER	130
	REGENCY CAKESTAND 3 TIER	129
	SET OF 3 CAKE TINS PANTRY DESIGN	129
3	REGENCY CAKESTAND 3 TIER	197
	WHITE HANGING HEART T-LIGHT HOLDER	174
	SET OF 3 CAKE TINS PANTRY DESIGN	169
4	REGENCY CAKESTAND 3 TIER	173
	PARTY BUNTING	165
	WHITE HANGING HEART T-LIGHT HOLDER	160
5	SPOTTY BUNTING	211
	PARTY BUNTING	210
	WHITE HANGING HEART T-LIGHT HOLDER	201
6	PARTY BUNTING	181
	SPOTTY BUNTING	155
	LUNCH BAG DOILEY PATTERN	137
7	PARTY BUNTING	158
	SPOTTY BUNTING	154
	LUNCH BAG DOILEY PATTERN	147
8	JUMBO BAG RED RETROSPOT	159
	SPOTTY BUNTING	150
	LUNCH BAG RED RETROSPOT	138
9	HOT WATER BOTTLE KEEP CALM	193
	JUMBO BAG RED RETROSPOT	192
	JUMBO BAG VINTAGE DOILY	173
10	PAPER CHAIN KIT 50'S CHRISTMAS	208
	HOT WATER BOTTLE KEEP CALM	182
	JUMBO BAG RED RETROSPOT	177
11	RABBIT NIGHT LIGHT	458
	PAPER CHAIN KIT 50'S CHRISTMAS	356
	HOT WATER BOTTLE KEEP CALM	266
12	WHITE HANGING HEART T-LIGHT HOLDER	266
	PAPER CHAIN KIT 50'S CHRISTMAS	240
	REGENCY CAKESTAND 3 TIER	204

dtype: int64

Globally Popular Items

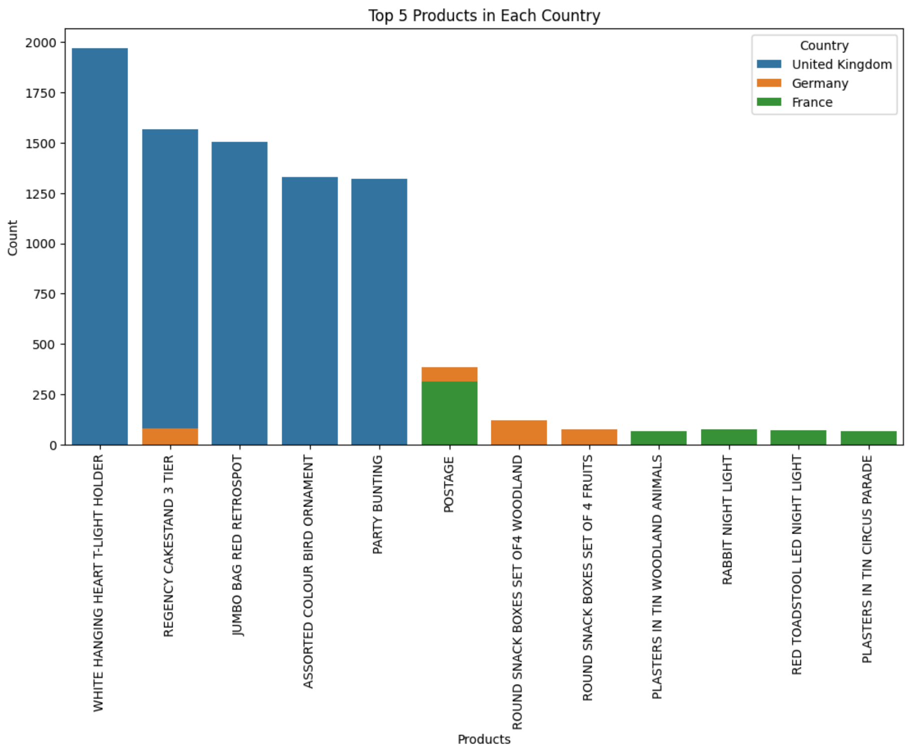
In [9]:
```python
# Step 4: Visualizations (Seaborn & Pivot Tables)

# Globally Popular Items
plt.figure(figsize=(10, 5))
sns.barplot(x=popular_items_global.index, y=popular_items_global.values, palette="viridis")
plt.xticks(rotation=90)
plt.title("Top 10 Popular Products Globally")
plt.xlabel("Products")
plt.ylabel("Count")
plt.show()
```

```
<ipython-input-9-e3252fdb35b9>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=popular_items_global.index, y=popular_items_global.values, palette="viridis")
```



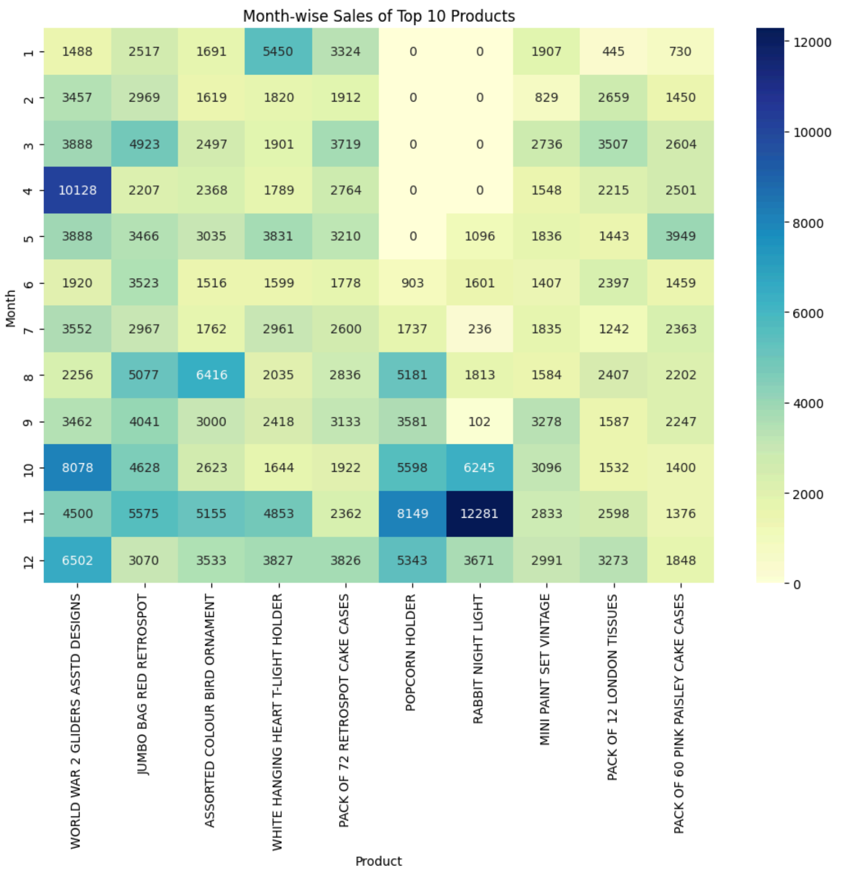
Country-wise Popular Items

In [10]:
```python
# Country-wise Popular Items (Example: UK)
plt.figure(figsize=(12, 6))
top_countries = df["Country"].value_counts().head(3).index
for country in top_countries:
    country_data = df[df["Country"] == country]["Description"].value_counts().head(5)
    sns.barplot(x=country_data.index, y=country_data.values, label=country)

plt.xticks(rotation=90)
plt.title("Top 5 Products in Each Country")
plt.xlabel("Products")
plt.ylabel("Count")
plt.legend(title="Country")
plt.show()
```



Month-wise Popular Items (using Pivot Table and Seaborn heatmap)

In [11]:
```python
# Month-wise Popular Items (using Pivot Table and Seaborn heatmap)
month_wise_sales = pd.pivot_table(df, values="Quantity", index="Month", columns="Description", aggfunc="sum", fill_value=0)
top_products_month = month_wise_sales.sum().sort_values(ascending=False).head(10).index
month_wise_sales_top = month_wise_sales[top_products_month]

plt.figure(figsize=(12, 8))
sns.heatmap(month_wise_sales_top, cmap="YlGnBu", annot=True, fmt="d")
plt.title("Month-wise Sales of Top 10 Products")
plt.xlabel("Product")
plt.ylabel("Month")
plt.show()
```



Month	WORLD WAR 2 GLIDERS ASSTD DESIGNS	JUMBO BAG RED RETROSPOT	ASSORTED COLOUR BIRD ORNAMENT	WHITE HANGING HEART T-LIGHT HOLDER	PACK OF 72 RETROSPOT CAKE CASES	POPCORN HOLDER	RABBIT NIGHT LIGHT	MINI PAINT SET VINTAGE	PACK OF 12 LONDON TISSUES	PACK OF 60 PINK PAISLEY CAKE CASES
1	1488	2517	1691	5450	3324	0	0	1907	445	730
2	3457	2969	1619	1820	1912	0	0	829	2659	1450
3	3888	4923	2497	1901	3719	0	0	2736	3507	2604
4	10128	2207	2368	1789	2764	0	0	1548	2215	2501
5	3888	3466	3035	3831	3210	0	1096	1836	1443	3949
6	1920	3523	1516	1599	1778	903	1601	1407	2397	1459
7	3552	2967	1762	2961	2600	1737	236	1835	1242	2363
8	2256	5077	6416	2035	2836	5181	1813	1584	2407	2202
9	3462	4041	3000	2418	3133	3581	102	3278	1587	2247
10	8078	4628	2623	1644	1922	5598	6245	3096	1532	1400
11	4500	5575	5155	4853	2362	8149	12281	2833	2598	1376
12	6502	3070	3533	3827	3826	5343	3671	2991	3273	1848

Loading and Sampling the Dataset (20% for faster processing)

In [66]:
```python
# Step 1: Load and Sample the Dataset (20% for faster processing)
@st.cache_data
def load_data():
    file_path = "/OnlineRetail (1).xlsx"  # Ensure this file is in the same directory
    df = pd.read_excel(file_path)
    df = df.sample(frac=0.2, random_state=42).reset_index(drop=True)
    return df
df = load_data()
```

```
2025-03-19 15:37:10.638 No runtime found, using MemoryCacheStorageManager
2025-03-19 15:37:10.641 No runtime found, using MemoryCacheStorageManager
2025-03-19 15:37:10.642 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:37:10.646 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:37:10.647 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:37:11.154 Thread 'Thread-10': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:37:11.161 Thread 'Thread-10': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:38:54.646 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:38:54.647 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
```

In [67]:
```python
# Step 2: Data Cleaning & Description
df1.dropna(subset=["CustomerID"], inplace=True)
df1.drop_duplicates(inplace=True)

# Convert Invoice Date to datetime and extract Month
df1["InvoiceDate"] = pd.to_datetime(df1["InvoiceDate"])
df1["Month"] = df1["InvoiceDate"].dt.month
```

Finding Popular Items (Globally, Country-wise, Month-wise)

In [68]:
```python
# Step 3: Finding Popular Items (Globally, Country-wise, Month-wise)

## Globally Popular Items
popular_items_global1 = df1["Description"].value_counts().head(10)
popular_items_global1
```

Out[68]:

Description	count
WHITE HANGING HEART T-LIGHT HOLDER	422
REGENCY CAKESTAND 3 TIER	370
JUMBO BAG RED RETROSPOT	339
ASSORTED COLOUR BIRD ORNAMENT	285
PARTY BUNTING	282
LUNCH BAG RED RETROSPOT	266
POSTAGE	241
LUNCH BAG BLACK SKULL.	232
LUNCH BAG SPACEBOY DESIGN	226
SET OF 3 CAKE TINS PANTRY DESIGN	225

dtype: int64

Country-wise Popular Items

In [69]:
```python
## Country-wise Popular Items
popular_items_country1 = df1.groupby("Country")["Description"].value_counts().groupby(level=0).head(3)
popular_items_country1
```

Out[69]:

		count
Country	**Description**	
Australia	**REGENCY CAKESTAND 3 TIER**	4
	SET OF 6 SOLDIER SKITTLES	4
	BAKING SET 9 PIECE RETROSPOT	3
Austria	**POSTAGE**	4
	BREAD BIN DINER STYLE RED	2
...	**...**	...
United Kingdom	**JUMBO BAG RED RETROSPOT**	308
	REGENCY CAKESTAND 3 TIER	301
Unspecified	**SET OF 10 LED DOLLY LIGHTS**	3
	SET OF 2 WOODEN MARKET CRATES	2
	12 MESSAGE CARDS WITH ENVELOPES	1

111 rows × 1 columns

dtype: int64

Month-wise Popular Items

In [70]:
```python
## Month-wise Popular Items
popular_items_month1 = df1.groupby("Month")["Description"].value_counts().groupby(level=0).head(3)
popular_items_month1
```

Out[70]:

		count
Month	**Description**	
1	**WHITE HANGING HEART T-LIGHT HOLDER**	30
	NATURAL SLATE HEART CHALKBOARD	24
	REGENCY CAKESTAND 3 TIER	24
2	**WHITE HANGING HEART T-LIGHT HOLDER**	31
	SET OF 3 CAKE TINS PANTRY DESIGN	25
	REGENCY CAKESTAND 3 TIER	23
3	**REGENCY CAKESTAND 3 TIER**	36
	SET OF 3 CAKE TINS PANTRY DESIGN	36
	WHITE HANGING HEART T-LIGHT HOLDER	35
4	**REGENCY CAKESTAND 3 TIER**	39
	PARTY BUNTING	29
	PAPER CHAIN KIT EMPIRE	28
5	**WHITE HANGING HEART T-LIGHT HOLDER**	48
	SPOTTY BUNTING	45
	PARTY BUNTING	41
6	**PARTY BUNTING**	34
	REGENCY CAKESTAND 3 TIER	33
	LUNCH BAG DOILEY PATTERN	32
7	**PARTY BUNTING**	35
	LUNCH BAG RED RETROSPOT	33
	JUMBO BAG RED RETROSPOT	30
8	**SPOTTY BUNTING**	43
	LUNCH BAG CARS BLUE	31
	JUMBO BAG RED RETROSPOT	29
9	**JUMBO BAG VINTAGE DOILY**	35
	LUNCH BAG BLACK SKULL.	34
	REGENCY CAKESTAND 3 TIER	33
10	**PAPER CHAIN KIT 50'S CHRISTMAS**	50
	SET OF 20 VINTAGE CHRISTMAS NAPKINS	35
	BAKING SET 9 PIECE RETROSPOT	32
11	**RABBIT NIGHT LIGHT**	89
	PAPER CHAIN KIT 50'S CHRISTMAS	83
	PAPER CHAIN KIT VINTAGE CHRISTMAS	56
12	**WHITE HANGING HEART T-LIGHT HOLDER**	64
	PAPER CHAIN KIT 50'S CHRISTMAS	44
	SCOTTIE DOG HOT WATER BOTTLE	44

dtype: int64

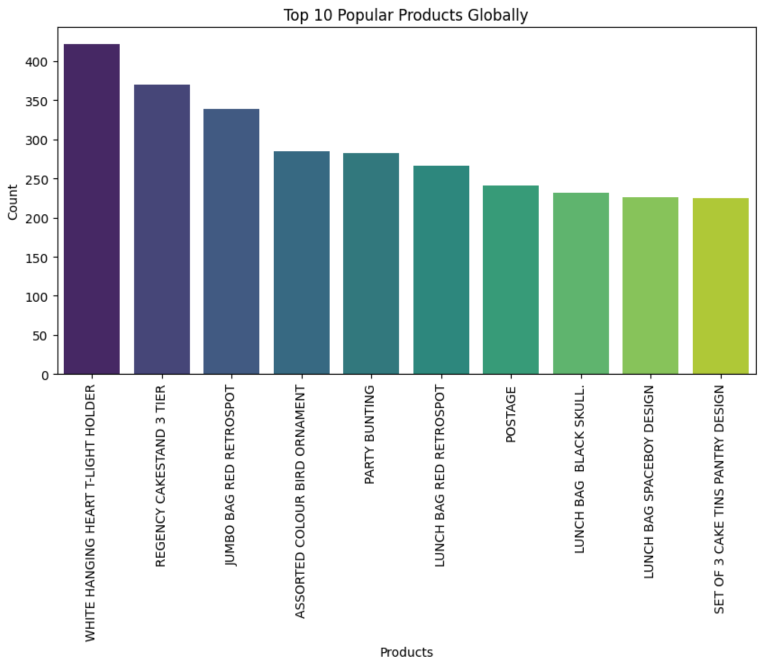
In [72]:
```python
# Step 4: Visualizations in Streamlit
st.title("📊 Retail Product Analysis & Recommendations")

st.subheader("Top 10 Globally Popular Products")
st.bar_chart(popular_items_global1)

st.subheader("Top Products by Country")
selected_country = st.selectbox("Select a country:", df["Country"].unique())
country_data = df[df["Country"] == selected_country]["Description"].value_counts().head(5)
st.bar_chart(country_data)
```

```
2025-03-19 15:40:34.982 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:40:34.984 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:40:34.985 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:40:34.986 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:40:35.005 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:40:35.006 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:40:35.008 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:40:35.009 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:40:35.018 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:40:35.019 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:40:35.022 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:40:35.023 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:40:35.024 Session state does not function when running a script without `streamlit run`
2025-03-19 15:40:35.025 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:40:35.026 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:40:35.073 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:40:35.075 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
```

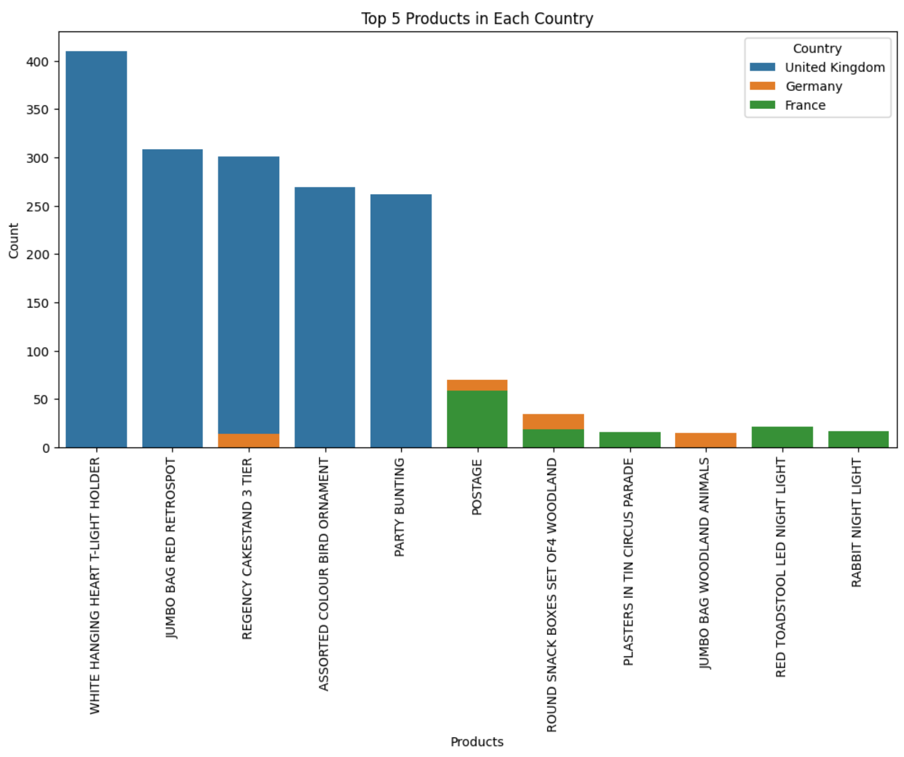
```
Out[72]:  DeltaGenerator()
```

```
In [73]: # Step 4: Visualizations (Seaborn & Pivot Tables)

         # Globally Popular Items
         plt.figure(figsize=(10, 5))
         sns.barplot(x=popular_items_global1.index, y=popular_items_global1.values, palette="viridis")
         plt.xticks(rotation=90)
         plt.title("Top 10 Popular Products Globally")
         plt.xlabel("Products")
         plt.ylabel("Count")
         plt.show()
```

```
<ipython-input-73-84e04606f4f0>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=popular_items_global1.index, y=popular_items_global1.values, palette="viridis")
```



```
In [74]: # Country-wise Popular Items (Example: UK)
         plt.figure(figsize=(12, 6))
         top_countries = df1["Country"].value_counts().head(3).index
         for country in top_countries:
             country_data = df1[df1["Country"] == country]["Description"].value_counts().head(5)
             sns.barplot(x=country_data.index, y=country_data.values, label=country)

         plt.xticks(rotation=90)
         plt.title("Top 5 Products in Each Country")
         plt.xlabel("Products")
         plt.ylabel("Count")
         plt.legend(title="Country")
         plt.show()
```



Recommendation System Optimization

```
In [75]: # Step 5: Recommendation System Optimization
         vectorizer = TfidfVectorizer(stop_words='english')
         item_vectors = vectorizer.fit_transform(df1["Description"].astype(str))
         item_vectors
```

```
Out[75]: <Compressed Sparse Row sparse matrix of dtype 'float64'
                 with 327655 stored elements and shape (81008, 1872)>
```

Dimensionality Reduction

```
In [76]: # Dimensionality Reduction
         from sklearn.feature_extraction.text import TfidfVectorizer
         from sklearn.decomposition import TruncatedSVD
         from sklearn.neighbors import NearestNeighbors
         from thefuzz.process import extractOne
         svd = TruncatedSVD(n_components=50, random_state=42)
         item_vectors_reduced = svd.fit_transform(item_vectors)
         item_vectors_reduced
```

```
Out[76]: array([[ 4.40977066e-02,  4.71744990e-01, -2.02305832e-01, ...,
                 -1.15705880e-01,  2.29977465e-02, -8.00702592e-02],
                [ 3.36426605e-04,  2.75049080e-03, -1.11580261e-03, ...,
                  1.67727626e-03, -1.11917614e-04,  2.04378324e-03],
                [ 1.54894416e-01,  9.94948834e-02,  2.58090801e-01, ...,
                  9.84923119e-02, -8.89558205e-02,  1.10382473e-01],
                ...,
                [ 5.04012440e-01, -1.86398802e-01, -3.06346524e-01, ...,
                  1.68074952e-02, -5.26338651e-02,  2.87048714e-02],
                [ 1.16467227e-02,  3.25981004e-02,  1.52214148e-02, ...,
                 -4.06847415e-02,  4.38877765e-02,  1.87651138e-02],
                [ 1.74738067e-02,  2.81082477e-02,  3.43872947e-02, ...,
                 -3.35994992e-02,  4.66532950e-02,  9.77863796e-03]])
```

KNN Model for Similarity Search

```
In [77]: # KNN Model for Similarity Search
         knn = NearestNeighbors(n_neighbors=6, metric='cosine')
         knn.fit(item_vectors_reduced)
```

```
Out[77]: ▾ NearestNeighbors

         NearestNeighbors(metric='cosine', n_neighbors=6)
```

User Input-based Recommendations

```
In [78]: # Step 6: User Input-based Recommendations
         st.subheader("🔍 Product Recommendation System")
         product_name = st.text_input("Enter a product name:", "")
```

```
2025-03-19 15:42:44.687 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:42:44.689 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:42:44.691 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:42:44.693 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:42:44.694 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:42:44.696 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:42:44.697 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-03-19 15:42:44.698 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
```

```
In [79]: def recommend_products(df):
             """User input-based product recommendation."""
             product_name = input("Enter a product name: ")
             closest_match, score = extractOne(product_name, df["Description"].unique())
             if score < 80:
                 return print("Product not found in dataset")

             # Get the index of the matched product within the vectorized data
             descriptions = df["Description"].astype(str).unique()  # Get unique descriptions
             matched_index = np.where(descriptions == closest_match)[0][0]  # Find index of the matched description

             # Recalculate TF-IDF and reduce dimensions using the unique descriptions
             vectorizer = TfidfVectorizer(stop_words='english')
             item_vectors = vectorizer.fit_transform(descriptions)  # Vectorize unique descriptions
             svd = TruncatedSVD(n_components=50, random_state=42)
             item_vectors_reduced = svd.fit_transform(item_vectors)

             # KNN Model for Similarity Search
             knn = NearestNeighbors(n_neighbors=6, metric='cosine')
             knn.fit(item_vectors_reduced)
```

```
    distances, indices = knn.kneighbors([item_vectors_reduced[matched_index]])

    # Get recommendations from the unique descriptions and their indices
    top_products = [descriptions[i] for i in indices[0][1:6] if i < len(descriptions)]

    print("Recommended products:")
    for product in top_products:
        print(f"- {product}")
```

Predict & Recommend

In [80]:
```
# Step 6: Predict & Recommend
recommend_products(df)
```

```
Enter a product name: PARTY BUNTING
Recommended products:
- PARTY INVITES SPACEMAN
- PARTY INVITES DINOSAURS
- PARTY INVITES FOOTBALL
- PARTY TIME PENCIL ERASERS
- HEN PARTY CORDON BARRIER TAPE
```