In [3]:
```
import pandas as pd
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity
```

In [4]:
```
df = pd.read_csv("Online Retail Recommendations.csv")
```

In [6]:
```
df = pd.read_csv("Online Retail Recommendations.csv")
print(df.columns)
```

```
Index(['StockCode', 'Quantity', 'UnitPrice', 'CustomerID'], dtype='object')
```

In [7]:
```
df = df.rename(columns={
    "CustomerID": "UserID",
    "StockCode": "ProductID"
})
```

In [8]:
```
df.dropna(subset=["UserID", "ProductID", "Quantity", "UnitPrice"], inplace=True)
df["UserID"] = df["UserID"].astype(str)
df["ProductID"] = df["ProductID"].astype(str)
df["Rating"] = df["Quantity"] * df["UnitPrice"]
```

In [9]:
```
user_item_matrix = df.pivot_table(index='UserID', columns='ProductID', values='Rating').fillna(0)
```

In [10]:
```
item_similarity = cosine_similarity(user_item_matrix.T)
item_similarity_df = pd.DataFrame(item_similarity,
                                  index=user_item_matrix.columns,
                                  columns=user_item_matrix.columns)
```

In [11]:
```
#Define recommendation function
def recommend_products(user_id, top_n=5):
    if user_id not in user_item_matrix.index:
        print("User not found.")
        return []

    user_ratings = user_item_matrix.loc[user_id]
    rated_items = user_ratings[user_ratings > 0].index.tolist()

    scores = {}
    for item in rated_items:
        similar_items = item_similarity_df[item].drop(labels=rated_items)
        for new_item, score in similar_items.items():
            scores[new_item] = scores.get(new_item, 0) + score

    recommended_items = sorted(scores.items(), key=lambda x: x[1], reverse=True)[:top_n]
    return [item for item, score in recommended_items]
```

In [12]:
```
user_ids = user_item_matrix.index.tolist()
print("Available User IDs:", user_ids[:5])
```

```
Available User IDs: ['12347.0', '12348.0', '12349.0', '12352.0', '12354.0']
```

In [13]:
```
sample_user = user_ids[0]
print(f"\n Top recommendations for User {sample_user}:")
print(recommend_products(sample_user, top_n=5))
```

```
 Top recommendations for User 12347.0:
['21124', '23007', '23057', '22078', '23020']
```

In [15]:
```
#top 5 recommendations for first 5 users
print("🔍 Top 5 product recommendations for sample users:\n")

for user in user_ids[:5]:
    recommendations = recommend_products(user, top_n=5)
    print(f"User {user}: {recommendations}")
```

```
🔍 Top 5 product recommendations for sample users:

User 12347.0: ['21124', '23007', '23057', '22078', '23020']
User 12348.0: ['84821', '23502', '23394', '21875', '22735']
User 12349.0: ['16216', '21882', '22059', '20992', '21159']
User 12352.0: ['16169P', '20832', '21615', '21624', '21709']
User 12354.0: ['35471D', '22904', '23356', '47599A', '22457']
```

In [14]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt

# Create a test set by masking some known ratings
test_df = df.copy()
test_df = test_df.sample(frac=0.1, random_state=42)

# Actual ratings
actual = []
predicted = []

for index, row in test_df.iterrows():
    user = row["UserID"]
    product = row["ProductID"]
    true_rating = row["Rating"]

    if user in user_item_matrix.index and product in item_similarity_df.columns:
        user_vector = user_item_matrix.loc[user]
        similar_scores = item_similarity_df[product]

        # Weighted average prediction
        numerator = np.dot(user_vector, similar_scores)
        denominator = similar_scores.sum()
        if denominator != 0:
            predicted_rating = numerator / denominator
            actual.append(true_rating)
            predicted.append(predicted_rating)

# Compute RMSE
rmse = sqrt(mean_squared_error(actual, predicted))
print(f" RMSE of recommendation model: {rmse:.2f}")
```

```
 RMSE of recommendation model: 25.74
```