```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.neighbors import NearestNeighbors
from sklearn.model_selection import train_test_split

# Load the dataset
file_path = r"C:\Users\rithi\OneDrive\Desktop\intrainz\OnlineRetail.xlsx"
data = pd.read_excel(file_path)

# Data preprocessing
# Drop rows with missing values
data.dropna(inplace=True)

# Convert InvoiceDate to datetime
data['InvoiceDate'] = pd.to_datetime(data['InvoiceDate'])

# Remove duplicates
data.drop_duplicates(inplace=True)

# Generate data descriptions
data_description = data.describe()
print(data_description)

# Use Seaborn for visualization
# Distribution of quantities
plt.figure(figsize=(10, 6))
sns.histplot(data['Quantity'], bins=50, kde=True)
plt.title('Distribution of Quantities')
plt.xlabel('Quantity')
plt.ylabel('Frequency')
```

```
plt.show()


# Top 10 most purchased products
top_products = data['Description'].value_counts().head(10)
plt.figure(figsize=(10, 6))
sns.barplot(x=top_products.values, y=top_products.index, palette='viridis')
plt.title('Top 10 Most Purchased Products')
plt.xlabel('Quantity Purchased')
plt.ylabel('Product Description')
plt.show()


# Create pivot tables
pivot_table = data.pivot_table(index='CustomerID', columns='StockCode', values='Quantity',
fill_value=0)
pivot_table.columns = pivot_table.columns.astype(str)  # Ensure all column names are strings


# Find popular items globally, country-wise, and month-wise
# Globally
global_popular_items = data['StockCode'].value_counts().head(10)
print("Globally Popular Items:")
print(global_popular_items)


# Country-wise
country_popular_items = data.groupby('Country')['StockCode'].apply(lambda x:
x.value_counts().head(1))
print("Country-wise Popular Items:")
print(country_popular_items)


# Month-wise
data['Month'] = data['InvoiceDate'].dt.month
month_popular_items = data.groupby('Month')['StockCode'].apply(lambda x:
x.value_counts().head(1))
```

```
print("Month-wise Popular Items:")
print(month_popular_items)

# Define function to analyze and print recommendations
def recommend_products(customer_id, pivot_table, model_knn, n_recommendations=5):
    if customer_id not in pivot_table.index:
        print(f"Customer ID {customer_id} not found in pivot table.")
        return []

    customer_data = pivot_table.loc[customer_id].values.reshape(1, -1)
    distances, indices = model_knn.kneighbors(customer_data, n_neighbors=n_recommendations + 1)

    recommendations = []
    for i in range(1, len(indices.flatten())):
        idx = indices.flatten()[i]
        if idx < len(pivot_table.columns):
            recommendations.append(pivot_table.columns[idx])
        else:
            print(f"Index {idx} is out of bounds for pivot table with size {len(pivot_table.columns)}.")
            continue  # Skip the out-of-bounds index

    return recommendations

# Fit the KNN model
model_knn = NearestNeighbors(metric='cosine', algorithm='brute')
model_knn.fit(pivot_table)

# Example: Recommend products for a specific customer
customer_id = 17850.0
recommended_products = recommend_products(customer_id, pivot_table, model_knn)
print(f"Products recommended for customer {customer_id}: {recommended_products}")
```

```
# Function to evaluate the recommendation system
def evaluate_model(data, model_knn, n_recommendations=5):
    hits = 0
    total = 0

    # Split the data into train and test sets
    train_data, test_data = train_test_split(data, test_size=0.2, random_state=42)
    pivot_table_train = train_data.pivot_table(index='CustomerID', columns='StockCode',
values='Quantity', fill_value=0)
    pivot_table_test = test_data.pivot_table(index='CustomerID', columns='StockCode',
values='Quantity', fill_value=0)

    # Ensure all column names are strings
    pivot_table_train.columns = pivot_table_train.columns.astype(str)
    pivot_table_test.columns = pivot_table_test.columns.astype(str)

    # Align test pivot table with train pivot table
    pivot_table_test = pivot_table_test.reindex(columns=pivot_table_train.columns, fill_value=0)

    # Refit the model on training data
    model_knn.fit(pivot_table_train)

    for customer_id in pivot_table_test.index:
        if customer_id in pivot_table_train.index:
            test_products = set(pivot_table_test.loc[customer_id].index)
            recommendations = set(recommend_products(customer_id, pivot_table_train, model_knn,
n_recommendations))
            hits += len(test_products.intersection(recommendations))
            total += len(test_products)
    hit_rate = hits / total if total > 0 else 0
    return hit_rate
```

```
# Evaluate the model
hit_rate = evaluate_model(data, model_knn)
print(f"Hit Rate: {hit_rate}")
```