- **Name: - Shreya Chandrakant Patil**
- **Internship Domain: - Data Science**
- **Company: - Intrainz Innovations Pvt.Ltd**
- **Major Project: - Online Retail Recommendation System**

INTRODUCTION: -

Key Stages in the Project:

- **Dataset Handling:** Imported the dataset using pandas.read_excel().
- **Data Preprocessing:** Eliminated missing values, filtered necessary records, and maintained data consistency.
- **User-Product Matrix:** Constructed a pivot table to map customer purchase patterns.
- **Memory Optimization:** Converted the matrix into a sparse representation using csr_matrix.
- **Similarity Computation:** Applied cosine similarity to analyze customer behavior and preferences.
- **Recommendation Algorithm:** Developed a function that suggests relevant products based on purchasing trends of similar customers.

This recommendation system enhances the shopping experience by providing personalized product suggestions based on purchase history.

CODE: -

```
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
from scipy.sparse import csr_matrix

# Load dataset
data_path = r"C:\Users\Shreya Patil\Desktop\INTERNSHIPS\intrainz internship
\Data_science_internship\OnlineRetail.xlsx"
data = pd.read_excel(data_path, sheet_name="OnlineRetail")
```

```
# Data Preprocessing
data.dropna(subset=['CustomerID'], inplace=True)
data = data[(data['Quantity'] > 0) & (data['UnitPrice'] > 0)]
data['CustomerID'] = data['CustomerID'].astype(int)

# Generate user-item interaction table
interaction_matrix = data.pivot_table(index='CustomerID',
columns='StockCode', values='Quantity', aggfunc='sum', fill_value=0)

# Convert to sparse representation
sparse_interactions = csr_matrix(interaction_matrix)

# Compute similarity matrix
similarity_scores = cosine_similarity(sparse_interactions)
similarity_df = pd.DataFrame(similarity_scores,
index=interaction_matrix.index, columns=interaction_matrix.index)

# Recommendation function
def suggest_items(user_id, top_n=5):
    if user_id not in similarity_df.index:
        return "Invalid Customer ID."

    similar_users =
similarity_df[user_id].sort_values(ascending=False).index[1:6]
    related_purchases = data[data['CustomerID'].isin(similar_users)]

    suggested_items =
related_purchases['StockCode'].value_counts().index[:top_n].tolist()

    return data[data['StockCode'].isin(suggested_items)][['StockCode',
'Description']].drop_duplicates()

# Sample execution
sample_user = data['CustomerID'].iloc[0]  # Selecting a random user
item_suggestions = suggest_items(sample_user)

print(item_suggestions)
```