**“Data Preparation”**

**“Task 1: Preparing Data for the Recommender System”**

**“Selected Task”:**

1. **“Build a recommender system that recommends books to read for every user based on their personal tastes and previous book ratings.”**

**“Code:”**

import pandas as pd

import numpy as np

from scipy.sparse import csr\_matrix

from sklearn.neighbors import NearestNeighbors

r\_df = pd.read\_csv('Ratings.csv', delimiter=';', on\_bad\_lines='skip')

if 'User-ID' not in r\_df.columns:

if len(r\_df.columns) == 3:

r\_df.columns = ['User-ID', 'ISBN', 'Rating']

else:

print("Unexpected columns found:", r\_df.columns)

raise ValueError("Please check your Ratings.csv file format.")

r\_df['User-ID'] = r\_df['User-ID'].astype(str)

r\_df['ISBN'] = r\_df['ISBN'].astype(str)

r\_df['Rating'] = pd.to\_numeric(r\_df['Rating'], errors='coerce')

r\_df.dropna(subset=['Rating'], inplace=True)

top\_u = r\_df['User-ID'].value\_counts().nlargest(500).index

top\_b = r\_df['ISBN'].value\_counts().nlargest(500).index

r\_df = r\_df[r\_df['User-ID'].isin(top\_u) & r\_df['ISBN'].isin(top\_b)]

r\_clean = r\_df.groupby(['User-ID', 'ISBN']).agg({'Rating': 'mean'}).reset\_index()

u\_mapng = {usr: idx for idx, usr in enumerate(r\_clean['User-ID'].unique())}

b\_mapng = {bk: idx for idx, bk in enumerate(r\_clean['ISBN'].unique())}

rev\_u\_map = {idx: usr for usr, idx in u\_mapng.items()}

rev\_b\_map = {idx: bk for bk, idx in b\_mapng.items()}

r\_clean['u\_idx'] = r\_clean['User-ID'].map(u\_mapng)

r\_clean['b\_idx'] = r\_clean['ISBN'].map(b\_mapng)

spar\_matx = csr\_matrix((r\_clean['Rating'], (r\_clean['u\_idx'], r\_clean['b\_idx'])))

nn\_modl = NearestNeighbors(metric='cosine', algorithm='brute', n\_neighbors=11, n\_jobs=-1)

nn\_modl.fit(spar\_matx)

dist, indx = nn\_modl.kneighbors(spar\_matx)

recs = []

for u\_idx in range(spar\_matx.shape[0]):

sim\_u = indx[u\_idx][1:]

sim\_s = 1 - dist[u\_idx][1:]

sim\_u\_ratings = spar\_matx[sim\_u]

u\_ratings = spar\_matx[u\_idx].toarray().flatten()

read\_b = set(np.where(u\_ratings > 0)[0])

cand\_bks = set(sim\_u\_ratings.nonzero()[1]) - read\_b

w\_scores = {}

for b\_idx in cand\_bks:

rat\_sim = sim\_u\_ratings[:, b\_idx].toarray().flatten()

valid\_idx = rat\_sim > 0

if np.any(valid\_idx):

w\_sum = np.dot(sim\_s[valid\_idx], rat\_sim[valid\_idx])

w\_total = np.sum(sim\_s[valid\_idx])

if w\_total > 0:

w\_avg = w\_sum / w\_total

w\_scores[b\_idx] = w\_avg

top\_5\_bks = sorted(w\_scores.items(), key=lambda x: x[1], reverse=True)[:5]

for b\_idx, sc in top\_5\_bks:

recs.append({

'User\_ID': rev\_u\_map[u\_idx],

'Book\_ID': rev\_b\_map[b\_idx],

'Recommendation\_Score': round(sc, 2)

})

recs\_df = pd.DataFrame(recs)

b\_df = pd.read\_csv('Booksv.csv', delimiter=';', on\_bad\_lines='skip')

b\_df.columns = b\_df.columns.str.strip()

if 'ISBN' not in b\_df.columns:

for col in b\_df.columns:

if col.lower() == 'isbn':

b\_df.rename(columns={col: 'ISBN'}, inplace=True)

break

if 'Book-Title' not in b\_df.columns:

for col in b\_df.columns:

if col.lower() in ['book-title', 'title', 'booktitle']:

b\_df.rename(columns={col: 'Book-Title'}, inplace=True)

break

b\_df['ISBN'] = b\_df['ISBN'].astype(str)

recs\_df['Book\_ID'] = recs\_df['Book\_ID'].astype(str)

merge\_df = recs\_df.merge(b\_df, left\_on='Book\_ID', right\_on='ISBN', how='left')

merge\_df.rename(columns={'Book-Title': 'Book\_Title'}, inplace=True)

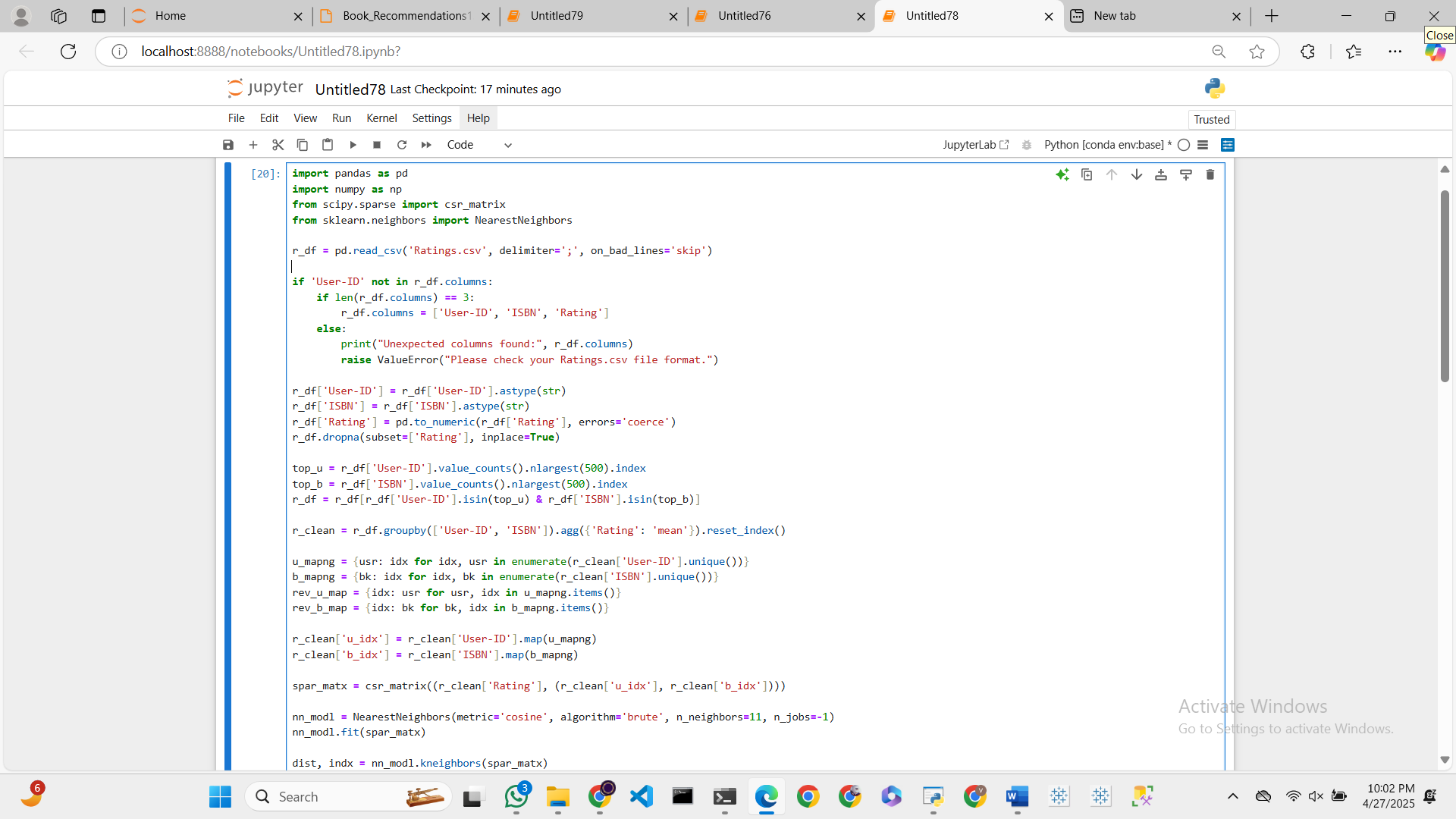
final\_df = merge\_df[['User\_ID', 'Book\_ID', 'Book\_Title', 'Recommendation\_Score']]

final\_df.to\_csv('Book\_Recommendations1.csv', index=False)

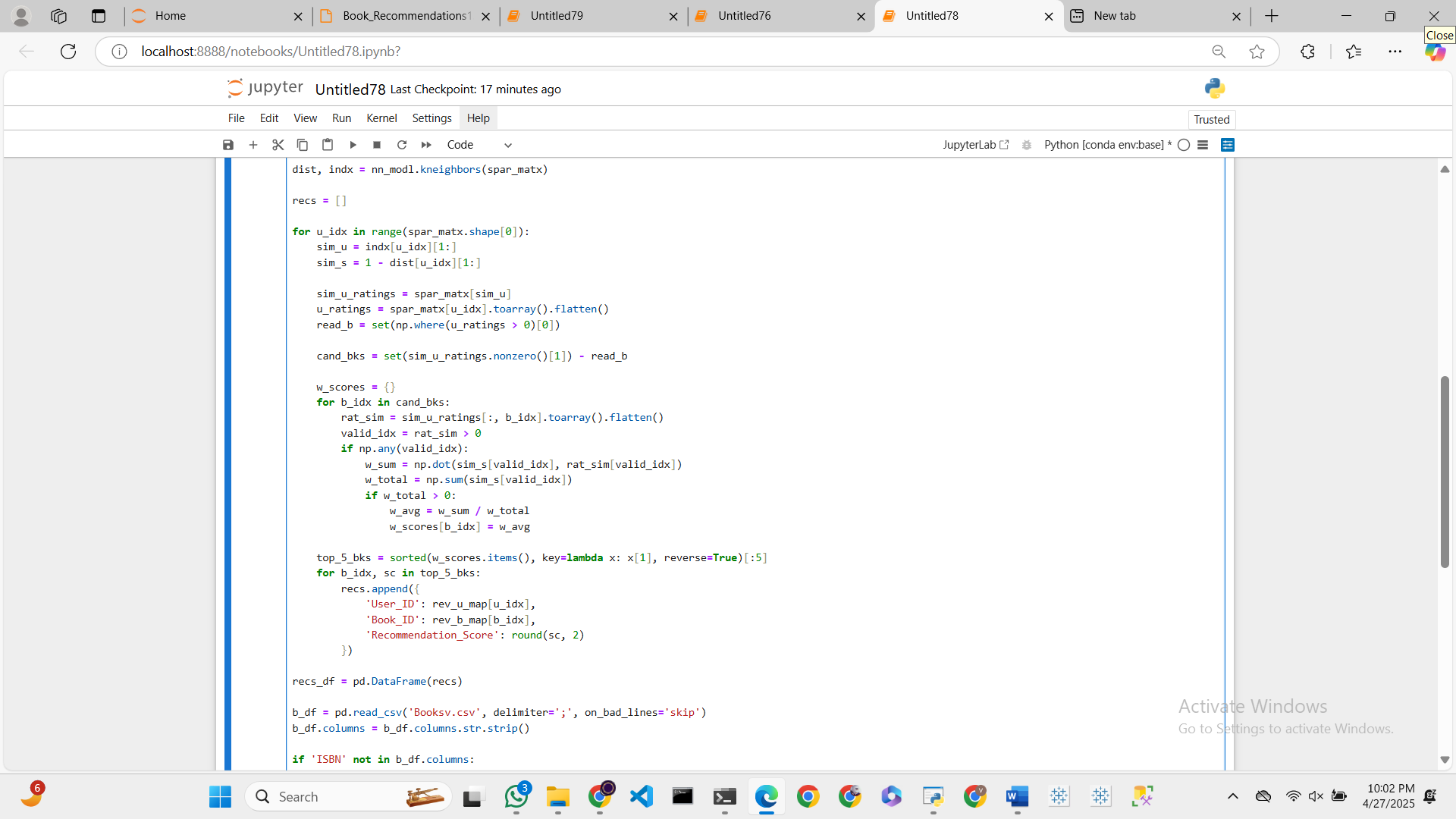
print("Recommendations saved to 'Book\_Recommendations1.csv'")

**Code Screenshot:**

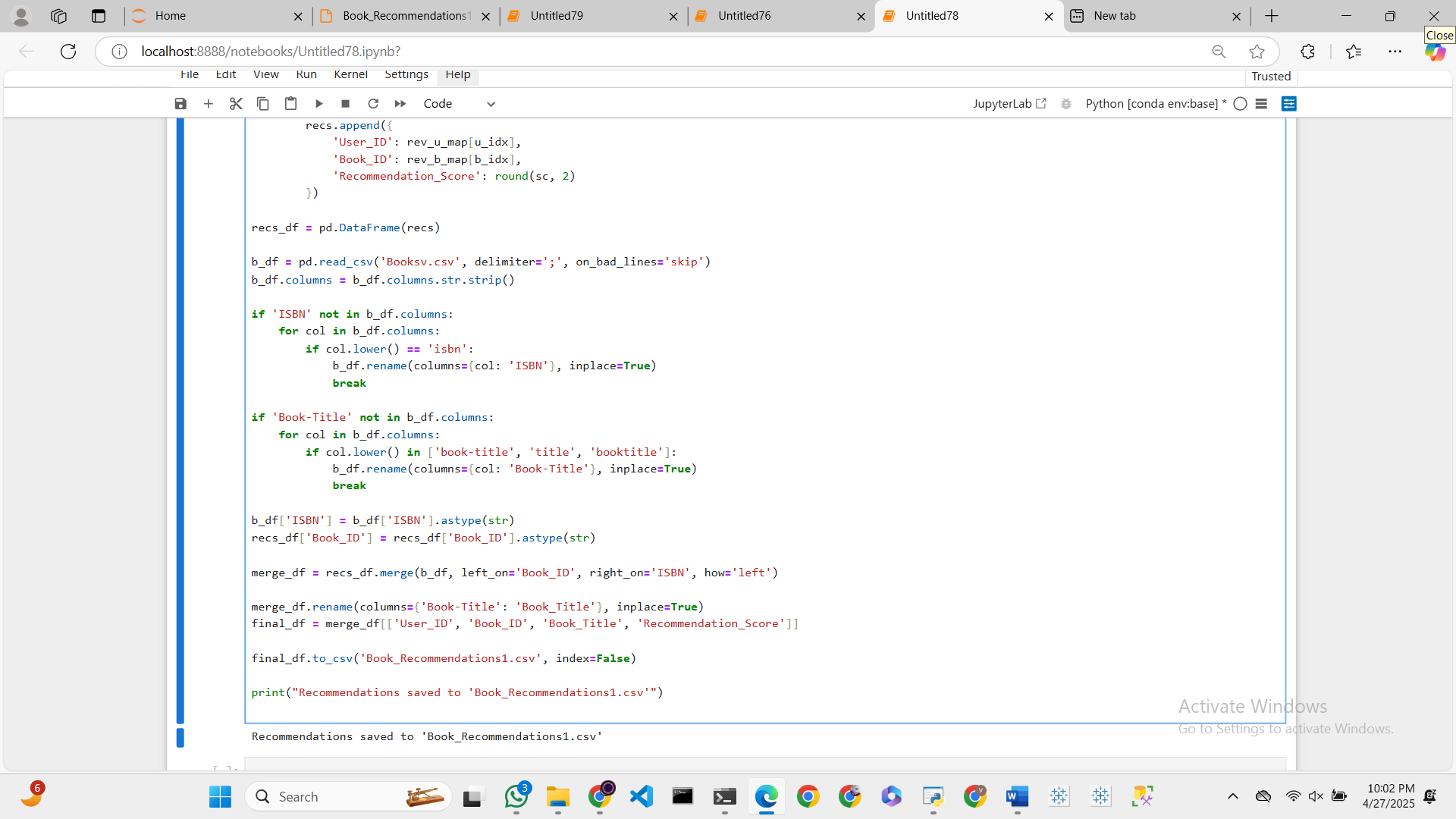
**1)**

****

**2)**

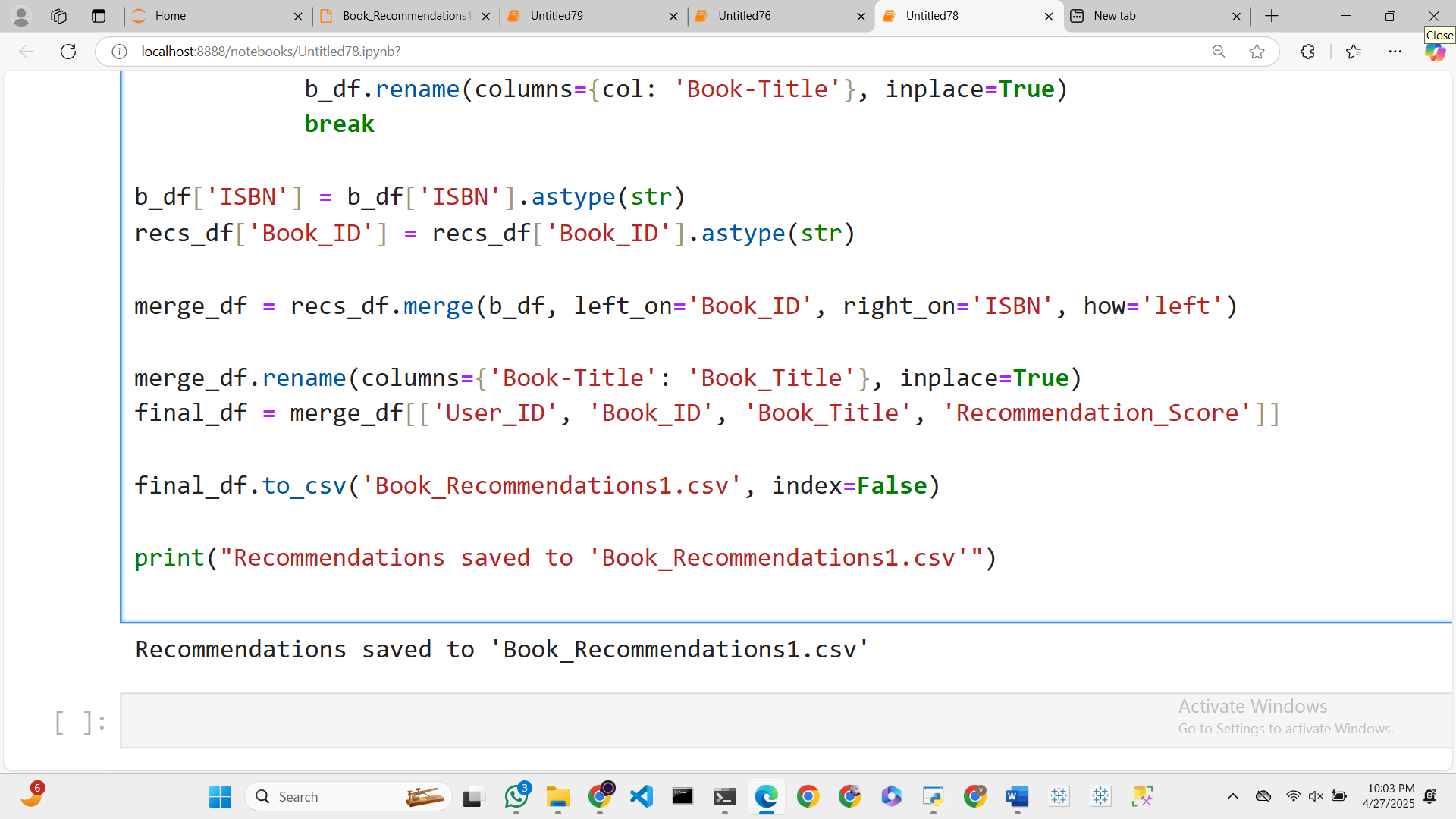
****

**3)**

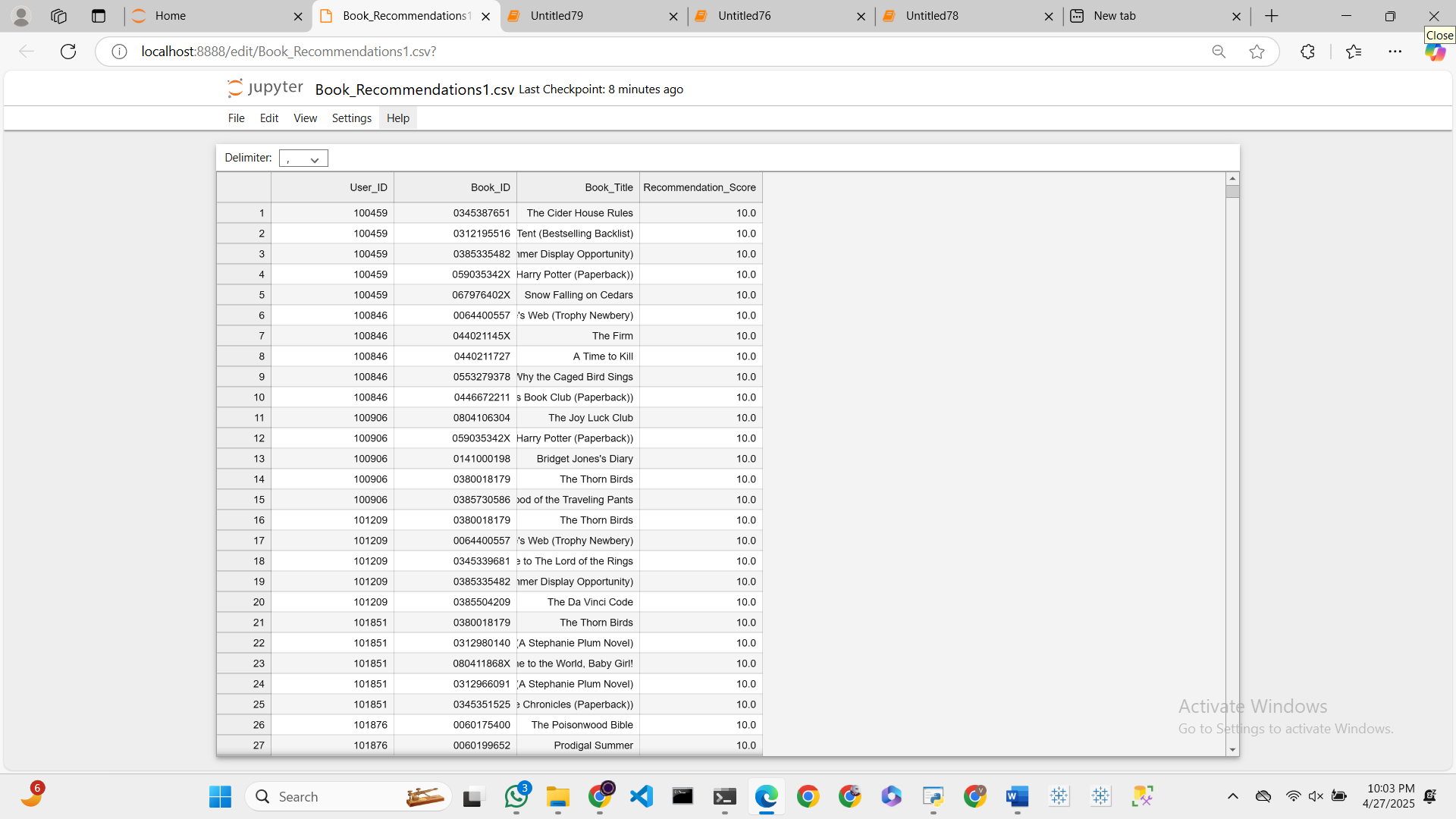
****

**Output:**

**1)**

****

**2)**

****

**“EXPLANATION:”**

This recommender system script takes a user-book ratings dataset and provides personalized book recommendations through collaborative filtering. First, the data is preprocessed by loading the ratings file, correcting column names if needed, removing invalid rows, and truncating the dataset to the top 500 most active users and top 500 most rated books. This Truncating improves performance while keeping the most relevant interactions for important recommendations. Then users and books are given distinct numerical indices, and a sparse matrix is created in which rows represent users, columns represent books, and values represent their corresponding ratings. Sparse matrices are employed since they are effective when handling high-dimensional data with a lot of missing values. The model subsequently calculates user similarity through the Nearest Neighbors algorithm using cosine similarity, comparing the similarity of rating patterns among users but using rating tendencies rather than actual values. The top 10 similar users per user are identified through this. In generating recommendations, the system identifies books rated by these similar users but not yet read by the target user. The score for each candidate book is calculated as a weighted average of ratings of similar users, where the weights are determined from user similarity. The 5 books with highest scores are selected as recommendations per user. Finally, the recommendations are enriched by joining book titles from a separate dataset on ISBNs. The final output has the user ID, book ID, book title, and recommendation score and stores them to a CSV file, hence the results are already prepared and can be used or submitted.