```
!pip install pandas numpy scikit-learn openpyxl seaborn matplotlib surprise
```

```
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 (1.24.4)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: openpyxl in /usr/local/lib/python3.11/dist-packages (3.1.5)
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0.13.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: surprise in /usr/local/lib/python3.11/dist-packages (0.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.15.2)
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: et-xmlfile in /usr/local/lib/python3.11/dist-packages (from openpyxl) (2.0.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.2)
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) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.3)
Requirement already satisfied: scikit-surprise in /usr/local/lib/python3.11/dist-packages (from surprise) (1.1.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
```

```
!pip install numpy==1.24.4
```

```
Requirement already satisfied: numpy==1.24.4 in /usr/local/lib/python3.11/dist-packages (1.24.4)
```

Load Dataset

```
from google.colab import files
import pandas as pd

# Upload your Excel file
uploaded = files.upload()

# Read the file
file_path = next(iter(uploaded))
df = pd.read_excel(file_path)

# Preview
df.head()
```

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving NM DATASET .xlsx to NM DATASET .xlsx
```

	User_ID	Item_ID	Category	Rating	Timestamp	Price	Platform	Location
0	User_913	Item_52	Movies	2.0	5/15/2023	369.55	Web	Africa
1	User_3457	Item_66	Electronics	1.4	8/19/2023	255.15	Web	Africa
2	User_1629	Item_1467	Sports	2.7	3/27/2024	296.69	Web	Europe
3	User_3463	Item_697	Movies	1.6	2023-03-12 00:00:00	55.59	Tablet	North America

Content-Based Recommender

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

# Combine content features
df['combined_features'] = df['Category'].astype(str) + ' ' + df['Item_ID'].astype(str)

# TF-IDF vectorizer
tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(df['combined_features'])

# Cosine similarity matrix
cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)
```

```python
# Index map for items
indices = pd.Series(df.index, index=df['Item_ID']).drop_duplicates()

# Function to get content-based recommendations
def content_based_recommend(item_id, num_recommendations=10):
    if item_id not in indices:
        return f"Item_ID '{item_id}' not found."
    idx = indices[item_id]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:num_recommendations+1]
    item_indices = [i[0] for i in sim_scores]
    return df[['Item_ID', 'Category']].iloc[item_indices]
```

Collaborative Filtering (SVD)

```python
from surprise import SVD, Dataset, Reader
from surprise.model_selection import train_test_split
from surprise.accuracy import rmse

# Use Surprise to prepare dataset
reader = Reader(rating_scale=(0.5, 5.0))
data = Dataset.load_from_df(df[['User_ID', 'Item_ID', 'Rating']], reader)

# Split into training and testing
trainset, testset = train_test_split(data, test_size=0.25, random_state=42)

# Train SVD model
model = SVD()
model.fit(trainset)

# Test RMSE
predictions = model.test(testset)
rmse(predictions)
from surprise import SVD, Dataset, Reader
from surprise.model_selection import train_test_split
from surprise.accuracy import rmse

# Use Surprise to prepare dataset
reader = Reader(rating_scale=(0.5, 5.0))
data = Dataset.load_from_df(df[['User_ID', 'Item_ID', 'Rating']], reader)

# Split into training and testing
trainset, testset = train_test_split(data, test_size=0.25, random_state=42)

# Train SVD model
model = SVD()
model.fit(trainset)

# Test RMSE
predictions = model.test(testset)
rmse(predictions)
# Predict function
def predict_rating(user_id, item_id):
    return model.predict(user_id, item_id).est
```

```
RMSE: 1.1383
RMSE: 1.1397
```

Hybrid Recommender

```python
def hybrid_recommend(user_id, item_id, top_n=10, weight_cb=0.5, weight_cf=0.5):
    if item_id not in indices:
        return f"Item_ID '{item_id}' not found."

    idx = indices[item_id]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:top_n*2+1]

    hybrid_scores = []
    for i, score in sim_scores:
        candidate_id = df['Item_ID'].iloc[i]
        cb_score = score
        cf_score = predict_rating(user_id, candidate_id)
        final_score = (weight_cb * cb_score) + (weight_cf * (cf_score / 5))
        hybrid_scores.append((candidate_id, final_score))
    #The following two lines were indented too far, they should align with the for loop
    top_recommendations = sorted(hybrid_scores, key=lambda x: x[1], reverse=True)[:top_n]
    return pd.DataFrame(top_recommendations, columns=['Recommended Item_ID', 'Score']) #Fixed the indentation
```

Import Libraries

```
# Content-based
print("Content-Based Recommendations:")
print(content_based_recommend('Item_52'))

# Predict individual rating
print("Collaborative Prediction for User_913 & Item_52:")
print(predict_rating('User_913', 'Item_52'))

# Content-based
print("Content-Based Recommendations:")
print(content_based_recommend('Item_52'))

# Predict individual rating
print("Collaborative Prediction for User_913 & Item_52:")
print(predict_rating('User_913', 'Item_52'))

# Hybrid
print("Hybrid Recommendations:")
print(hybrid_recommend('User_913', 'Item_52'))
```

```
Content-Based Recommendations:
       Item_ID Category
5    Item_1131   Movies
279  Item_1620   Movies
7     Item_779   Movies
99   Item_1662   Movies
107  Item_1411   Movies
134  Item_1414   Movies
144  Item_1378   Movies
187   Item_906   Movies
225   Item_672   Movies
237   Item_135   Movies
Collaborative Prediction for User_913 & Item_52:
2.5958427855643276
Content-Based Recommendations:
       Item_ID Category
5    Item_1131   Movies
279  Item_1620   Movies
7     Item_779   Movies
99   Item_1662   Movies
107  Item_1411   Movies
134  Item_1414   Movies
144  Item_1378   Movies
187   Item_906   Movies
225   Item_672   Movies
237   Item_135   Movies
Collaborative Prediction for User_913 & Item_52:
2.5958427855643276
Hybrid Recommendations:
  Recommended Item_ID     Score
0           Item_1131  0.407848
1           Item_1411  0.403989
2            Item_779  0.403735
3           Item_1783  0.399646
4            Item_778  0.394232
5           Item_1085  0.392347
6            Item_286  0.391028
7           Item_1620  0.389312
8           Item_1662  0.384713
9           Item_1378  0.383638
```