CODE

```
#Installing required packages
!pip install numpy
!pip install pandas
pip install scikit-surprise
pip install matplotlib
pip install -U scikit-learn
pip install nltk
pip install scipy

#Importing required packages
import pandas as pd
meds = pd.read_csv("meds_info.csv")
meds
meds['Uses'].value_counts()
meds.loc[meds['Uses'].str.contains('Bacterial'),'Uses'] = 'Bacterial Infection'
meds
meds.loc[meds['Uses'].str.contains('Bacterial','bacterial'),'Uses'] = 'Bacterial
Infection'
meds.loc[meds['Uses'].str.contains('Hypertension','hypertension'),'Uses'] =
'Hypertension'
meds.loc[meds['Uses'].str.contains('Pain relief'),'Uses'] = 'Pain Relief'
meds.loc[meds['Uses'].str.contains('Cough','cough'),'Uses'] = 'Cough'
meds.loc[meds['Uses'].str.contains('Cancer','cancer'),'Uses'] = 'Cancer'
meds.loc[meds['Uses'].str.contains('Dandruff','dandruff'),'Uses'] = 'Dandruff'
meds.loc[meds['Uses'].str.contains('Diabetes','diabetes'),'Uses'] = 'Diabetes'
meds.loc[meds['Uses'].str.contains('diabetes'),'Uses'] = 'Diabetes'
```

```
meds
meds['Uses'].str.contains('Diabetes').value_counts()
meds.loc[10000]
meds['Uses'].value_counts()
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
meds
meds.shape
meds.duplicated().sum()
meds[meds['Medicine Name']=='Avil Injection']
meds.drop_duplicates(inplace=True)
meds.duplicated().sum()
meds.shape
meds['Uses']
#Removing unnecessary words
def remove_unnecessary_words(sentence):
   unnecessary_words = ['treatment','of','and','condition',]
   words = sentence.split()
   cleaned_words = [word for word in words if word.lower() not in
unnecessary_words]
   cleaned_sentence = ' '.join(cleaned_words)
   return cleaned_sentence
meds['cleaned_text'] = meds['Uses'].apply(remove_unnecessary_words)
print(meds['cleaned_text'])
def split_words(sentence): #for words like 'AnxietyTreatment'
   return ' '.join(re.findall(r'[a-zA-Z][^A-Z]*', sentence))
import re
meds['cleaned_text'] = meds['cleaned_text'].apply(split_words)
```

```
meds['cleaned_text']
meds['new_uses'] = meds['cleaned_text'].apply(remove_unnecessary_words)
print(meds['new_uses'])
changed_rows = meds[meds['cleaned_text'] != meds['new_uses']]
print("Changed Rows:")
print(changed_rows)
num_changes = len(changed_rows)
print(f"\nNumber of Changes: {num_changes}")
meds[meds['Manufacturer'].duplicated()].sort_values(by='Manufacturer',
ascending = False)
meds
meds['club'] = meds['new_uses'] + meds['Manufacturer']
meds
meds['club'] = meds['club'].apply(split_words)
meds
meds2 = meds[['Medicine Name','club']]
meds2
meds2['club'] = meds2['club'].apply(lambda x:x.lower())
meds2
#Count Vectorizer
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features = 11900, stop_words = 'english')
cv.fit_transform(meds2['club']).toarray().shape
vectors = cv.fit_transform(meds['club']).toarray()
vectors[0]
len(cv.get_feature_names_out())
#Natural Language Processing
import nltk
```

```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
def stem(text):
   y = []
   for i in text.split():
      y.append(ps.stem(i))
   return " ".join(y)
meds2['club'] = meds2['club'].apply(stem)
#Cosine Similarity
from sklearn.metrics.pairwise import cosine_similarity
cosine_similarity(vectors)
cosine_similarity(vectors).shape
sim = cosine_similarity(vectors)
sim[38]
sim[0].shape
sorted(list(enumerate(sim[0])), reverse = True, key = lambda x:x[1])[1:6]
meds.rename(columns={'Medicine Name': 'Medicine_Name'}, inplace=True)
meds2.rename(columns={'Medicine Name': 'Medicine_Name'}, inplace=True)
med_index = meds2[meds2['Medicine_Name'] == 'Aerodil-LS
Expectorant'].index[0]
distance = sim[med_index]
med_list = sorted(list(enumerate(distance)), reverse = True, key = lambda
x:x[1])[1:6]
med_index
distance
med_list
meds
meds.sort_values(by='Manufacturer', ascending = False)
```

```
#Recommendation Function
def recommendation(medicine):
   med_index = meds2[meds2['Medicine_Name'] == medicine].index[0]
   distance = sim[med_index]
   med_list = sorted(list(enumerate(distance)), reverse = True, key = lambda
x:x[1])[1:6]
   for i in med_list:
      print(meds2.iloc[i[0]].Medicine_Name)
recommendation('Ascoril LS Syrup')
meds[meds['Medicine_Name'] == 'Ascoril LS Drops']
meds[meds['Medicine_Name'] == 'Mucaryl LS Syrup']
#Final Working Function
def recommendation(medicine):
   u = meds[meds['Medicine_Name'] == medicine]['Uses'].values
   use = ''.join(map(str, u[0]))
   print(medicine,"is used for",use)
   med_index = meds2[meds2['Medicine_Name'] == medicine].index[0]
   distance = sim[med_index]
   med_list = sorted(list(enumerate(distance)), reverse = True, key = lambda
x:x[1])[1:6]
   print("*****Recommended for you*****")
   for i in med_list:
      print(meds2.iloc[i[0]].Medicine_Name)
      print("Used for:",meds.iloc[i[0]].Uses)
#User Input
user_input = input("Enter the Medicine")
recommendation(user_input)
```