```
# -*- coding: utf-8 -*-
"""Alaiba_Nawaz_Day4,5_W3.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1-xGUwLroQBLLRjSwYpo4UwsVi7HMolyy
"""

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_csv("/content/country_vaccinations.csv")

data.info()

data.head(10)

data.describe()

#handeling missing vals
data.isna().sum()
data.dropna()

data.dropna(subset=['date'], inplace=True)
data['date'] = pd.to_datetime(data['date'])
data.sort_values(by='date', inplace=True)

country_vaccination_totals =
data.groupby('country')['total_vaccinations'].sum().reset_index()
sorted_countries =
country_vaccination_totals.sort_values(by='total_vaccinations',
ascending=False)

# Get the top 10 countries with the highest vaccinations
top_10_countries = sorted_countries.head(10)

# Plot the data using a bar chart
```

```
plt.figure(figsize=(12, 8))
plt.bar(top_10_countries['country'],
top_10_countries['total_vaccinations'])
plt.xlabel('Country')
plt.ylabel('Total Vaccinationa')
plt.title('Top 10 Countries with Highest Vaccinations')
plt.xticks(rotation=45)

# Show the plot
plt.tight_layout()
plt.show()

# Plot the time series for each country or region
plt.figure(figsize=(12, 8))
sns.lineplot(data=data, x='date', y='total_vaccinations', hue='country')
plt.xlabel('Date')
plt.ylabel('Total Vaccinations')
plt.title('Vaccination Progress over Time by Country/Region')
plt.xticks(rotation=45)

# Show the plot
plt.tight_layout()
plt.show()



data=data.groupby(['vaccines','country'])['total_vaccinations'].max()
data=data.sort_values(ascending=False)[0:10]
plt.figure(figsize=(14, 14))
plt.pie(data, labels=data.keys(), autopct='%1.1f%%')
plt.title('Distribution of Vaccination Types')
plt.axis('equal')
plt.show()

"""Task2"""

import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
import spacy
```

```
# Sample data for restaurants
data = {
    'restaurant_name': ['Pf-Changs', 'Pizzeria', 'LCY', 'Sushi World',
'Nandos'],
    'address': ['12-C MM_Alam', '90 Saint_Lu', '888 DHA_2', '190 Avenue',
'209 Skardu'],
    'cuisine_type': ['Chinese', 'Italian', 'American', 'Mexico',
'Pakistani'],
    'price_range': ['$1000', '$500', '$5000', '$500', '$3000'],
}
restaurant_data = pd.DataFrame(data)

# Create a combined feature for recommendation using relevant columns
restaurant_data['features'] = restaurant_data['cuisine_type'] + ' ' +
restaurant_data['price_range'] + ' ' + restaurant_data['address']

# Vectorizing the combined features
vectorizer = TfidfVectorizer(stop_words='english')
tfidf_matrix =
vectorizer.fit_transform(restaurant_data['features'].fillna(''))

# Computing similarity scores
cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)

# Returns top N Recommendations based on user preferences (By default 2)
def content_based_recommendation(user_preferences, restaurant_data,
topN=2):
    user_tfidf_vector = vectorizer.transform([user_preferences])
    cosine_similarities_user = linear_kernel(user_tfidf_vector,
tfidf_matrix)
    similar_indices = cosine_similarities_user[0].argsort()[:-topN-1:-1]
    recommendations_content_based = restaurant_data.iloc[similar_indices]
    return recommendations_content_based

def naturalLanguageUnderstandingModule(userInput, restaurant_data,
backendVisible=False):

    nlp = spacy.load("en_core_web_sm")
    doc = nlp(userInput)
```

```
    # Initialize variables to store extracted information
    location = None
    budget = None
    cuisine_preferences = []

    # Extract information using POS tagging
    for token in doc:
        if 'GPE' in token.ent_type_:
            location = token.text
        elif token.pos_ == 'NUM':
            budget = float(token.text)
        elif token.pos_ == 'ADJ':
            cuisine_preferences.append(token.text)

    # Convert cuisine preferences to lowercase and remove duplicates
    cuisine_preferences = list(set([preference.lower() for preference in
cuisine_preferences]))

    result = {
        'location': location,
        'budget': budget,
        'cuisine': ' '.join(cuisine_preferences)
    }

    user_preferences = f"{result['cuisine']} cuisine and
{result['budget']} price in {result['location']}"

    # Content-Based Filtering Recommendations based on user preferences
    recommendations = content_based_recommendation(user_preferences,
restaurant_data)

    if backendVisible:
        print("User Preferences: ", result)
        print("Content-Based Recommendations: ")
        print(recommendations)

    return result, recommendations

# Main function to run the chatbot
```

```
def run_chatbot():
    print("Welcome to Restaurant Recommendation Chatbot!")
    userInput = input("Please enter your preferences: ")

    # Get NLP module result and recommendations
    nlp_result, recommendations =
naturalLanguageUnderstandingModule(userInput, restaurant_data)

    # Display recommendations
    print("\nRecommended Restaurants:")
    print(recommendations[['restaurant_name', 'address', 'cuisine_type',
'price_range']])

# Run the chatbot
run_chatbot()
```