Submitted To:	**Sir Jamal Abdul Ahad**
Submitted By:	**Areeba Khan**
Roll No:	**10172**
Date:	**30th Oct, 2024**
Subject:	**Natural Language Processing**

Question #1:

Choose any corpus of your choice of at least 200 MBs of any domain in NLP and perform the following tasks:

• Text Preprocessing (Text Cleaning, Stemming / Lemmatization)

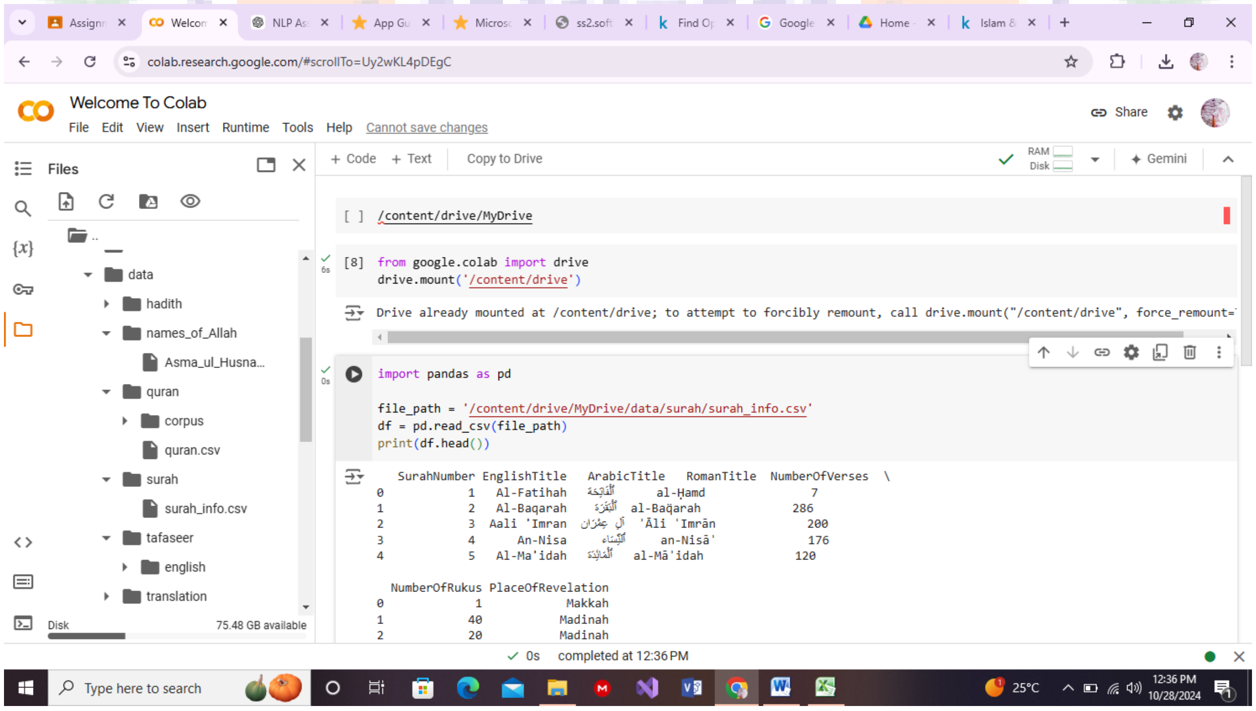
• Word Embedding (using an algorithm like Word2Vec, Glove, FastText)

• Encoding Techniques (Bag of Words, One – Hot)

• Parts of Speech tagging.

Answer:

For this assignment, I chose the **Islam and AI Dataset** from **Kaggle,** as it offers a substantial corpus that combines discussions on religious perspectives and advancements in artificial intelligence, making it rich in context and language. This dataset will allow for comprehensive Natural Language Processing (NLP) analysis across different tasks. Firstly, text preprocessing, including text cleaning and lemmatization, will prepare the dataset by standardizing terms and removing extraneous characters, ensuring cleaner data for further analysis.



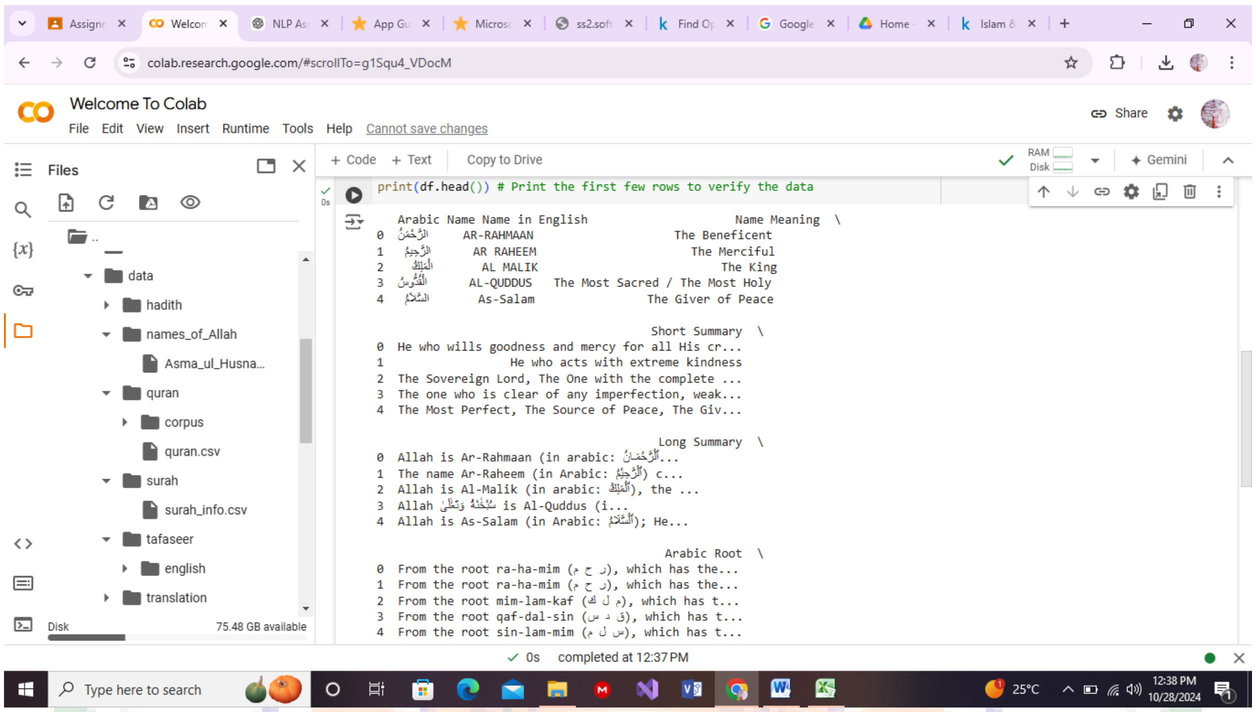
```
[ ] /content/drive/MyDrive
```

```
[8] from google.colab import drive
    drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=
```

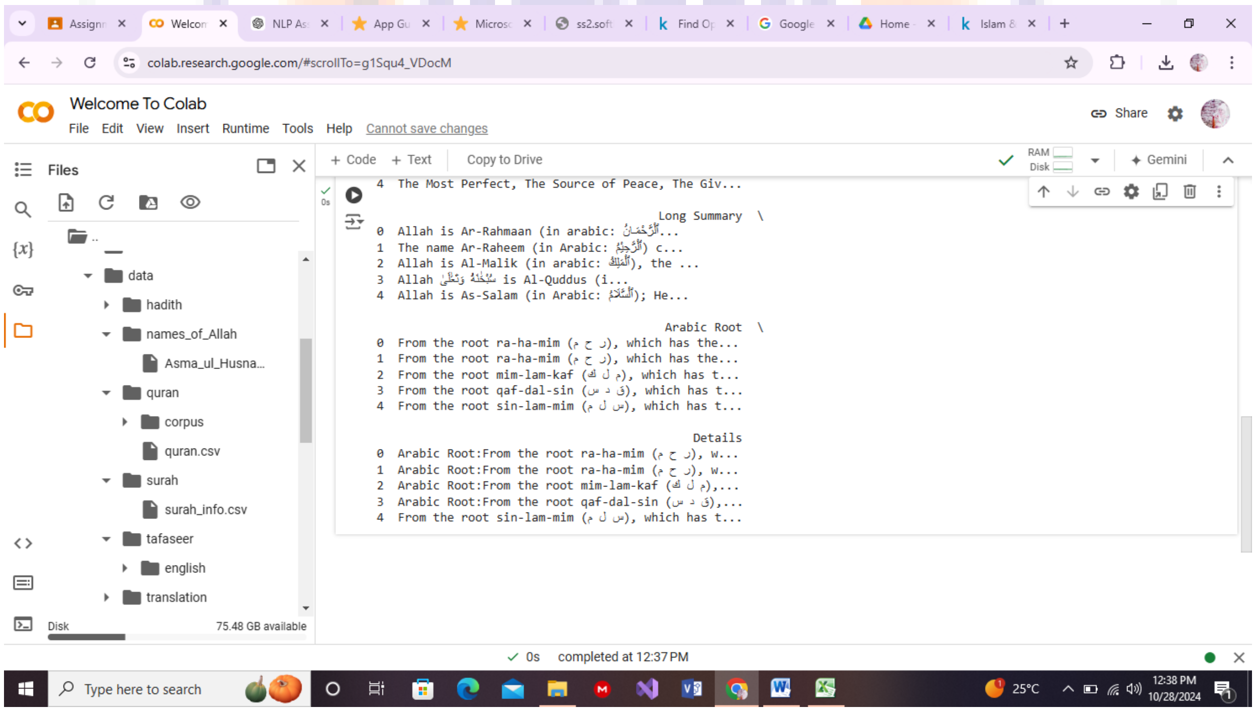
```
import pandas as pd

file_path = '/content/drive/MyDrive/data/surah/surah_info.csv'
df = pd.read_csv(file_path)
print(df.head())
```

```
   SurahNumber EnglishTitle   ArabicTitle   RomanTitle  NumberOfVerses  \
0            1    Al-Fatihah     ٱلْفَاتِحَة       al-Ḥamd               7
1            2    Al-Baqarah      ٱلْبَقَرَة   al-Baqarah             286
2            3  Aali 'Imran   آل عِمْرَان   'Āli 'Imrān             200
3            4       An-Nisa      ٱلنِّسَاء      an-Nisā'             176
4            5    Al-Ma'idah     ٱلْمَائِدَة   al-Mā'idah             120

  NumberOfRukus PlaceOfRevelation
0             1            Makkah
1            40           Madinah
2            20           Madinah
```



```
print(df.head()) # Print the first few rows to verify the data
```

```
  Arabic Name Name in English                      Name Meaning  \
0   الرَّحْمَنُ     AR-RAHMAAN                    The Beneficent
1    الرَّحِيمُ      AR RAHEEM                      The Merciful
2     الْمَلِكُ       AL MALIK                          The King
3   الْقُدُّوسُ       AL-QUDDUS  The Most Sacred / The Most Holy
4    السَّلاَمُ        As-Salam                The Giver of Peace

                                       Short Summary  \
0  He who wills goodness and mercy for all His cr...
1                       He who acts with extreme kindness
2  The Sovereign Lord, The One with the complete ...
3  The one who is clear of any imperfection, weak...
4  The Most Perfect, The Source of Peace, The Giv...

                                        Long Summary  \
0  Allah is Ar-Rahmaan (in arabic: ٱلرَّحْمَـٰنُ...
1  The name Ar-Raheem (in Arabic: ٱلرَّحِيْمُ) c...
2  Allah is Al-Malik (in arabic: ٱلْمَلِكُ), the ...
3  Allah سُبْحَانَهُ وَتَعَالَىٰ is Al-Quddus (i...
4  Allah is As-Salam (in Arabic: ٱلسَّلَامُ); He...

                                         Arabic Root  \
0  From the root ra-ha-mim (ر ح م), which has the...
1  From the root ra-ha-mim (ر ح م), which has the...
2  From the root mim-lam-kaf (م ل ك), which has t...
3  From the root qaf-dal-sin (ق د س), which has t...
4  From the root sin-lam-mim (س ل م), which has t...
```



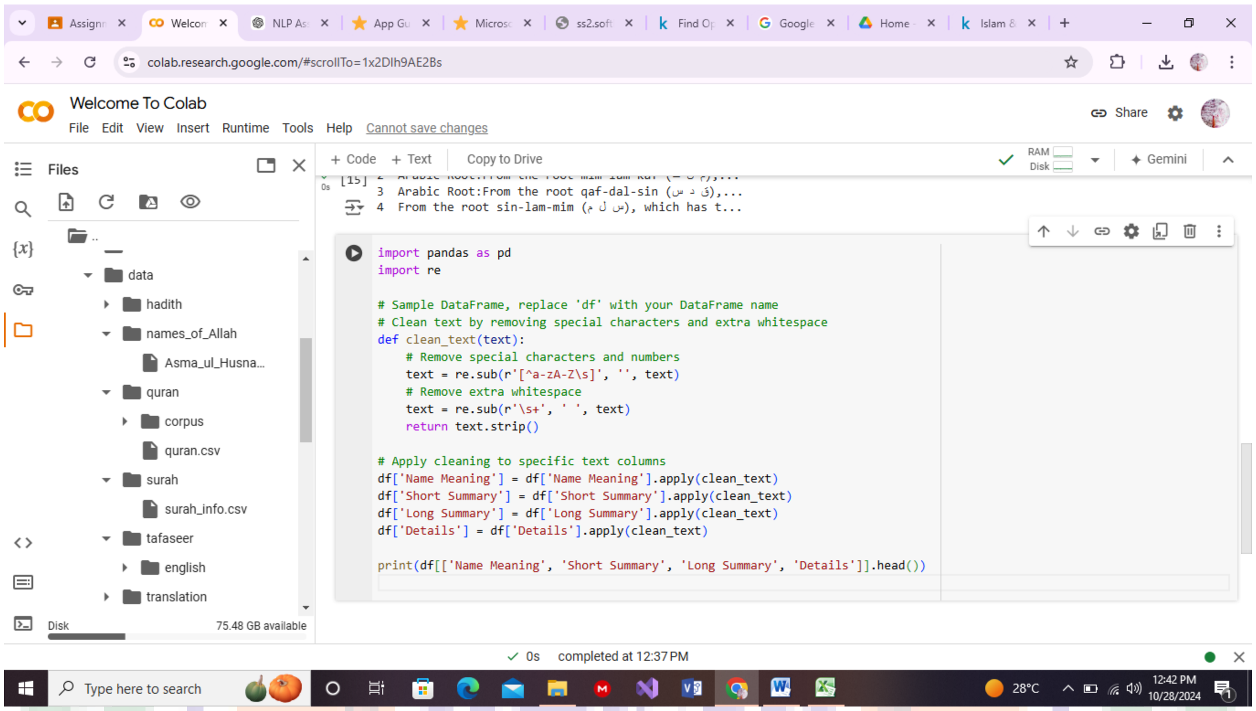
```
4  The Most Perfect, The Source of Peace, The Giv...

                                        Long Summary  \
0  Allah is Ar-Rahmaan (in arabic: ٱلرَّحْمَـٰنُ...
1  The name Ar-Raheem (in Arabic: ٱلرَّحِيْمُ) c...
2  Allah is Al-Malik (in arabic: ٱلْمَلِكُ), the ...
3  Allah سُبْحَانَهُ وَتَعَالَىٰ is Al-Quddus (i...
4  Allah is As-Salam (in Arabic: ٱلسَّلَامُ); He...

                                         Arabic Root  \
0  From the root ra-ha-mim (ر ح م), which has the...
1  From the root ra-ha-mim (ر ح م), which has the...
2  From the root mim-lam-kaf (م ل ك), which has t...
3  From the root qaf-dal-sin (ق د س), which has t...
4  From the root sin-lam-mim (س ل م), which has t...
```

```
                                             Details
0  Arabic Root:From the root ra-ha-mim (ر ح م), w...
1  Arabic Root:From the root ra-ha-mim (ر ح م), w...
2  Arabic Root:From the root mim-lam-kaf (م ل ك),...
3  Arabic Root:From the root qaf-dal-sin (ق د س),...
4  From the root sin-lam-mim (س ل م), which has t...
```

Text Preprocessing (Text Cleaning, Stemming / Lemmatization):

Text cleaning:

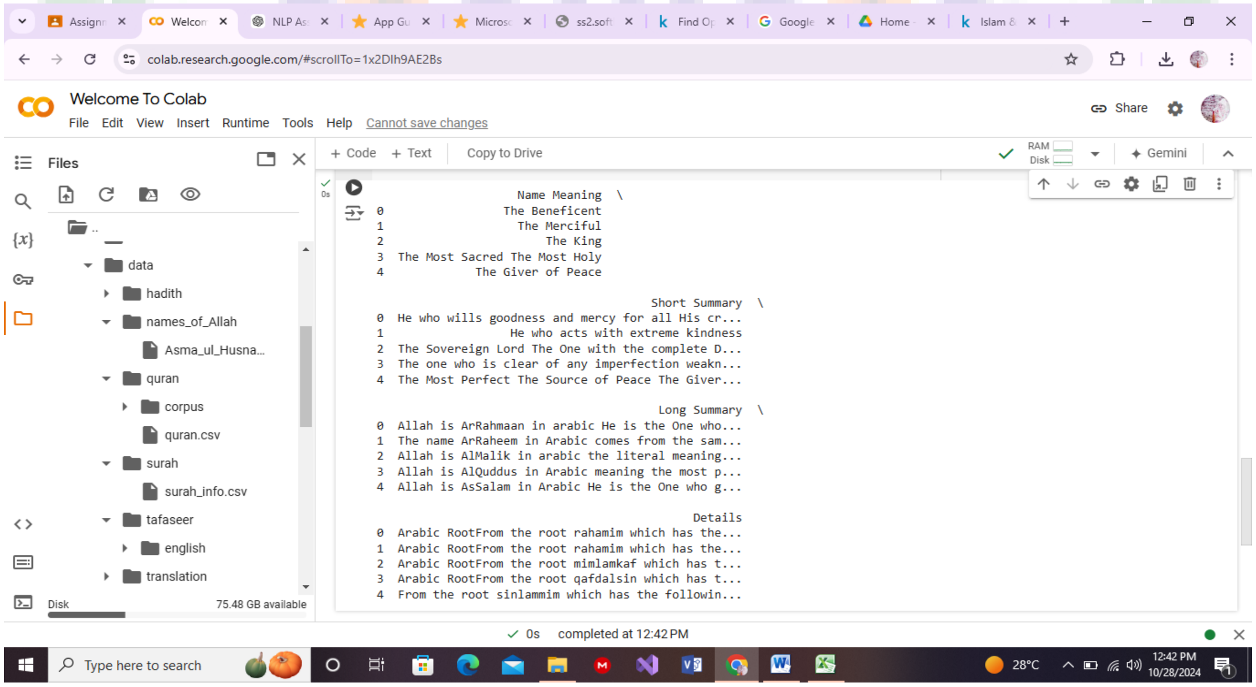
Code:



```
3  Arabic Root:From the root qaf-dal-sin (ق د س),...
4  From the root sin-lam-mim (س ل م), which has t...
```

```python
import pandas as pd
import re

# Sample DataFrame, replace 'df' with your DataFrame name
# Clean text by removing special characters and extra whitespace
def clean_text(text):
    # Remove special characters and numbers
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    # Remove extra whitespace
    text = re.sub(r'\s+', ' ', text)
    return text.strip()

# Apply cleaning to specific text columns
df['Name Meaning'] = df['Name Meaning'].apply(clean_text)
df['Short Summary'] = df['Short Summary'].apply(clean_text)
df['Long Summary'] = df['Long Summary'].apply(clean_text)
df['Details'] = df['Details'].apply(clean_text)

print(df[['Name Meaning', 'Short Summary', 'Long Summary', 'Details']].head())
```

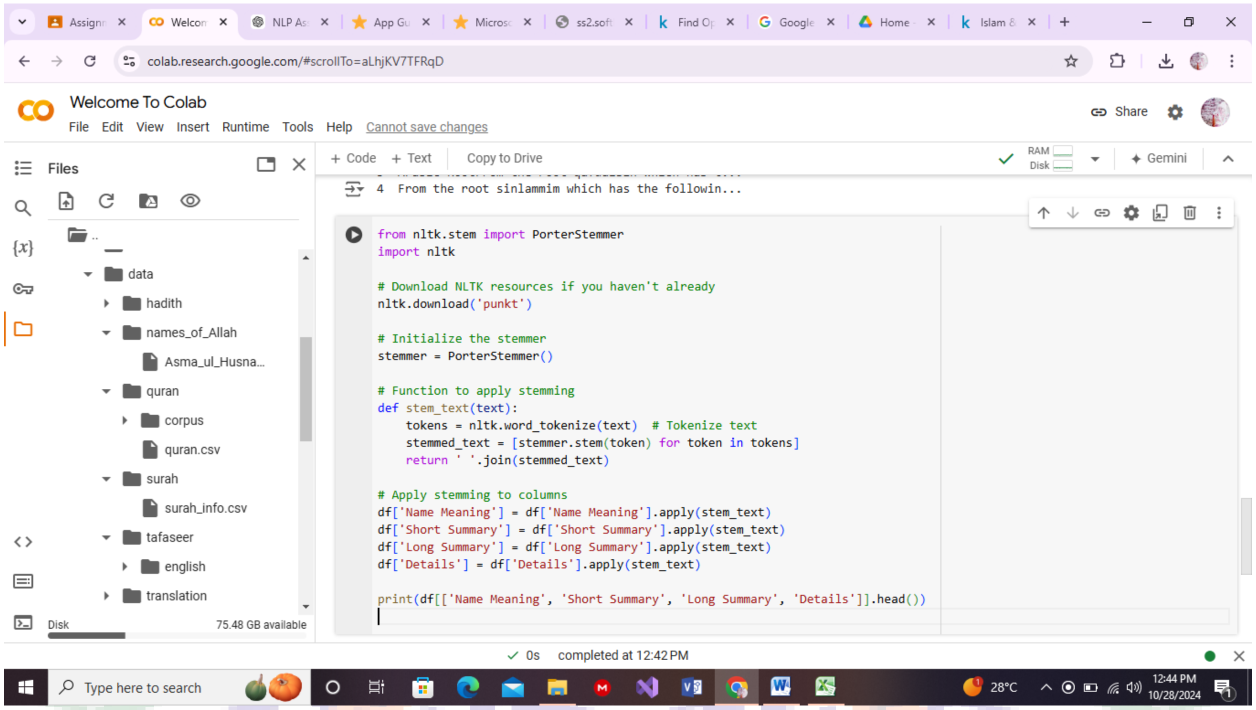
Output:



```
                  Name Meaning  \
0              The Beneficent
1                 The Merciful
2                     The King
3  The Most Sacred The Most Holy
4            The Giver of Peace

                                       Short Summary  \
0  He who wills goodness and mercy for all His cr...
1                      He who acts with extreme kindness
2  The Sovereign Lord The One with the complete D...
3  The one who is clear of any imperfection weakn...
4  The Most Perfect The Source of Peace The Giver...

                                        Long Summary  \
0  Allah is ArRahmaan in arabic He is the One who...
1  The name ArRaheem in Arabic comes from the sam...
2  Allah is AlMalik in arabic the literal meaning...
3  Allah is AlQuddus in Arabic meaning the most p...
4  Allah is AsSalam in Arabic He is the One who g...

                                             Details
0  Arabic RootFrom the root rahamim which has the...
1  Arabic RootFrom the root rahamim which has the...
2  Arabic RootFrom the root mimlamkaf which has t...
3  Arabic RootFrom the root qafdalsin which has t...
4  From the root sinlammim which has the followin...
```

Stemming:

Code:

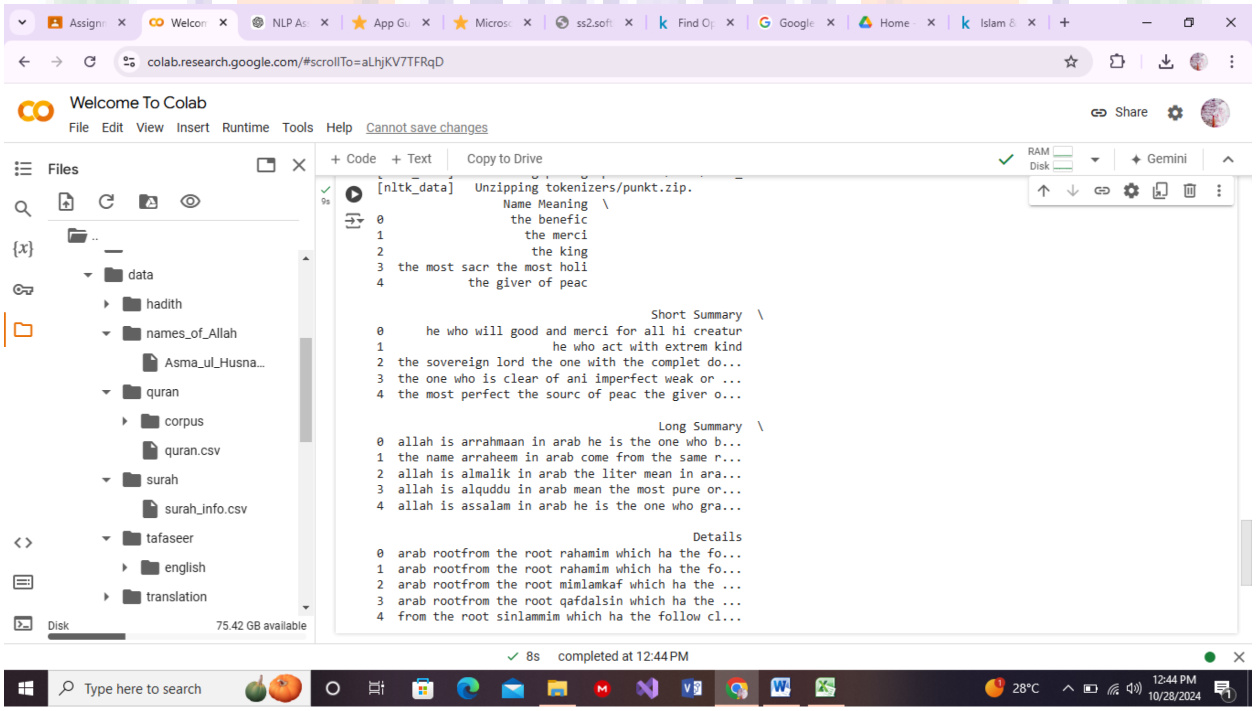


```
4  From the root sinlammim which has the followin...
```

```python
from nltk.stem import PorterStemmer
import nltk

# Download NLTK resources if you haven't already
nltk.download('punkt')

# Initialize the stemmer
stemmer = PorterStemmer()

# Function to apply stemming
def stem_text(text):
    tokens = nltk.word_tokenize(text)  # Tokenize text
    stemmed_text = [stemmer.stem(token) for token in tokens]
    return ' '.join(stemmed_text)

# Apply stemming to columns
df['Name Meaning'] = df['Name Meaning'].apply(stem_text)
df['Short Summary'] = df['Short Summary'].apply(stem_text)
df['Long Summary'] = df['Long Summary'].apply(stem_text)
df['Details'] = df['Details'].apply(stem_text)

print(df[['Name Meaning', 'Short Summary', 'Long Summary', 'Details']].head())
```

✓ 0s   completed at 12:42 PM

Output:

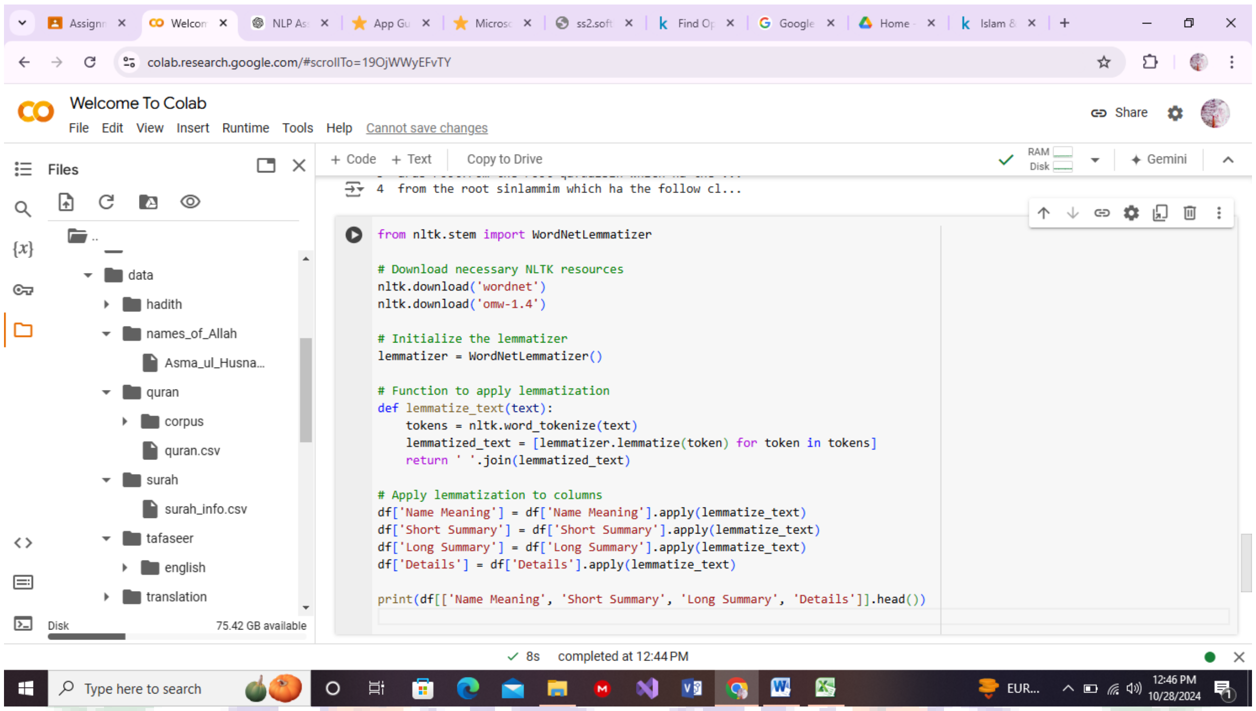


```
[nltk_data]   Unzipping tokenizers/punkt.zip.
                    Name Meaning  \
0                    the benefic
1                      the merci
2                       the king
3  the most sacr the most holi
4              the giver of peac

                                      Short Summary  \
0      he who will good and merci for all hi creatur
1                            he who act with extrem kind
2  the sovereign lord the one with the complet do...
3  the one who is clear of ani imperfect weak or ...
4  the most perfect the sourc of peac the giver o...

                                       Long Summary  \
0  allah is arrahmaan in arab he is the one who b...
1  the name arraheem in arab come from the same r...
2  allah is almalik in arab the liter mean in ara...
3  allah is alquddu in arab mean the most pure or...
4  allah is assalam in arab he is the one who gra...

                                            Details
0  arab rootfrom the root rahamim which ha the fo...
1  arab rootfrom the root rahamim which ha the fo...
2  arab rootfrom the root mimlamkaf which ha the ...
3  arab rootfrom the root qafdalsin which ha the ...
4  from the root sinlammim which ha the follow cl...
```

✓ 8s   completed at 12:44 PM

Lemmatization:

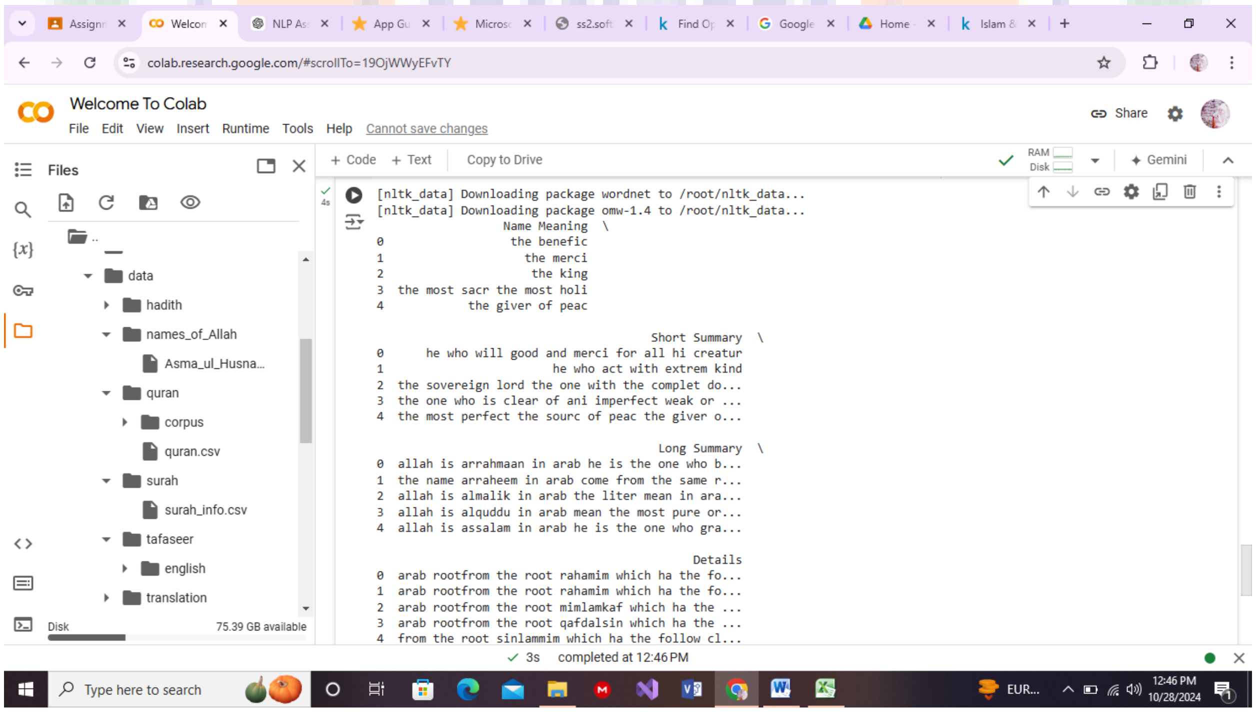
Code:



```
4  from the root sinlammim which ha the follow cl...
```

```python
from nltk.stem import WordNetLemmatizer

# Download necessary NLTK resources
nltk.download('wordnet')
nltk.download('omw-1.4')

# Initialize the lemmatizer
lemmatizer = WordNetLemmatizer()

# Function to apply lemmatization
def lemmatize_text(text):
    tokens = nltk.word_tokenize(text)
    lemmatized_text = [lemmatizer.lemmatize(token) for token in tokens]
    return ' '.join(lemmatized_text)

# Apply lemmatization to columns
df['Name Meaning'] = df['Name Meaning'].apply(lemmatize_text)
df['Short Summary'] = df['Short Summary'].apply(lemmatize_text)
df['Long Summary'] = df['Long Summary'].apply(lemmatize_text)
df['Details'] = df['Details'].apply(lemmatize_text)

print(df[['Name Meaning', 'Short Summary', 'Long Summary', 'Details']].head())
```

Output:



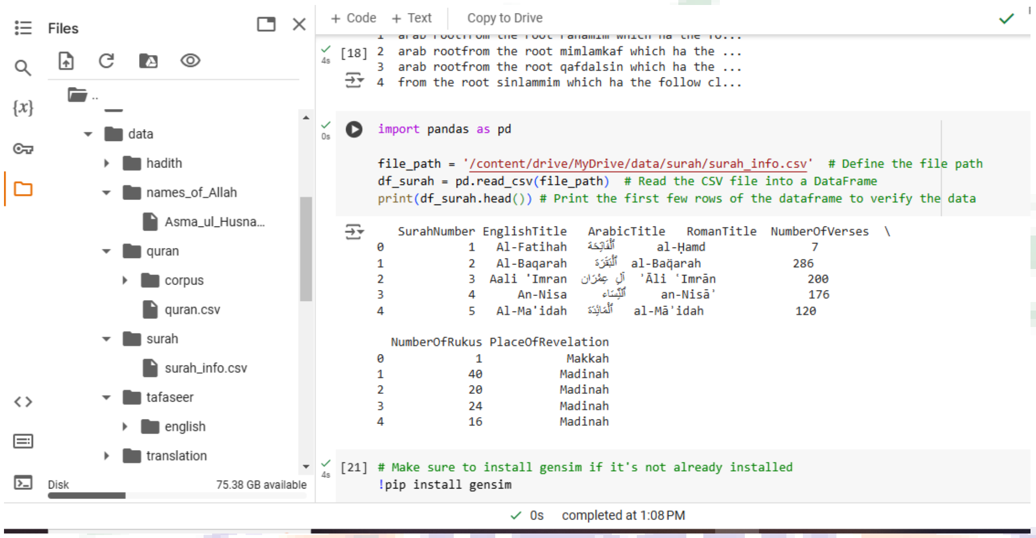
```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
                    Name Meaning  \
0                    the benefic
1                      the merci
2                       the king
3  the most sacr the most holi
4               the giver of peac

                                       Short Summary  \
0       he who will good and merci for all hi creatur
1                           he who act with extrem kind
2  the sovereign lord the one with the complet do...
3  the one who is clear of ani imperfect weak or ...
4  the most perfect the sourc of peac the giver o...

                                        Long Summary  \
0  allah is arrahmaan in arab he is the one who b...
1  the name arraheem in arab come from the same r...
2  allah is almalik in arab the liter mean in ara...
3  allah is alquddu in arab mean the most pure or...
4  allah is assalam in arab he is the one who gra...

                                             Details
0  arab rootfrom the root rahamim which ha the fo...
1  arab rootfrom the root rahamim which ha the fo...
2  arab rootfrom the root mimlamkaf which ha the ...
3  arab rootfrom the root qafdalsin which ha the ...
4  from the root sinlammim which ha the follow cl...
```

Word Embedding (using an algorithm like Word2Vec, Glove, FastText):

To implement word embedding on this dataset . I'll use the **Word2Vec** algorithm, which creates word vectors by examining co-occurrence within a certain context.

Text Preprocessing:

We'll clean and tokenize the text to prepare it for Word2Vec.



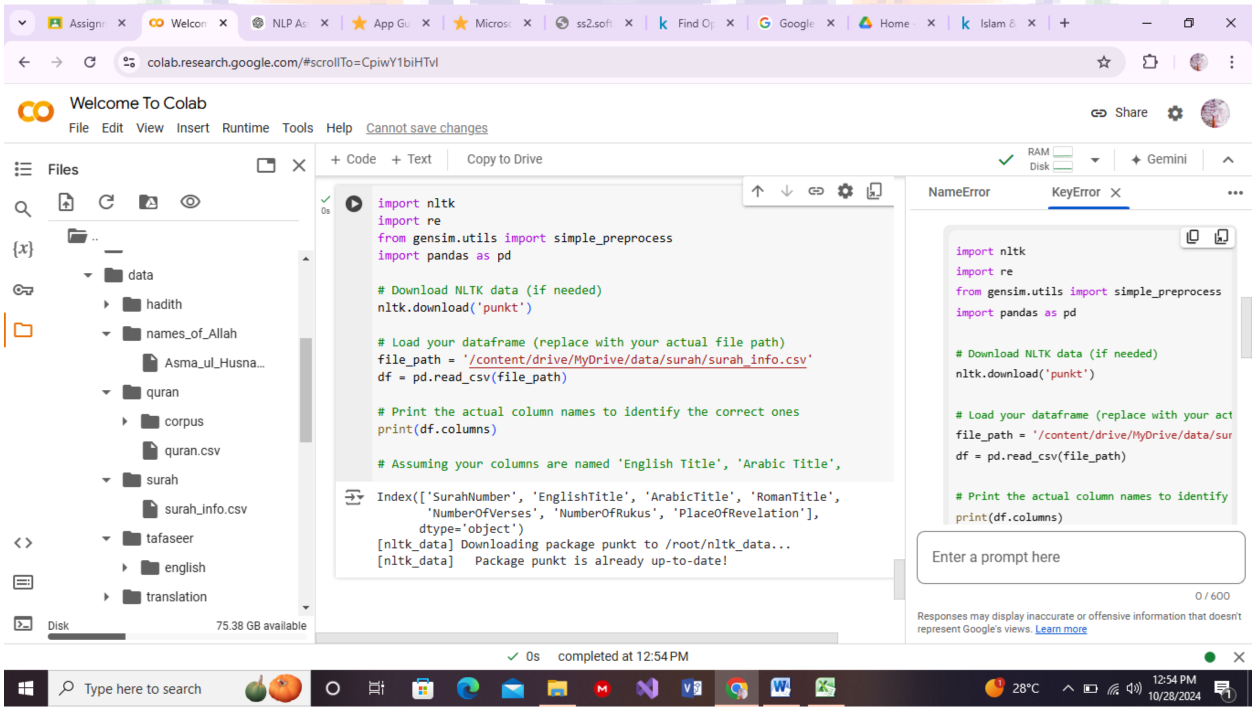
```
[18] 2  arab rootfrom the root mimlamkaf which ha the ...
     3  arab rootfrom the root qafdalsin which ha the ...
     4  from the root sinlammim which ha the follow cl...
```

```python
import pandas as pd

file_path = '/content/drive/MyDrive/data/surah/surah_info.csv'  # Define the file path
df_surah = pd.read_csv(file_path)  # Read the CSV file into a DataFrame
print(df_surah.head()) # Print the first few rows of the dataframe to verify the data
```

```
   SurahNumber EnglishTitle   ArabicTitle   RomanTitle  NumberOfVerses  \
0            1   Al-Fatihah      ٱلْفَاتِحَة      al-Ḥamd               7
1            2   Al-Baqarah      ٱلْبَقَرَة   al-Baqarah             286
2            3  Aali 'Imran   آلِ عِمْرَان  ʾĀli ʿImrān              200
3            4      An-Nisa        ٱلنِّسَاء      an-Nisāʾ              176
4            5   Al-Ma'idah      ٱلْمَائِدَة   al-Māʾidah             120

  NumberOfRukus PlaceOfRevelation
0             1            Makkah
1            40           Madinah
2            20           Madinah
3            24           Madinah
4            16           Madinah
```

```
[21] # Make sure to install gensim if it's not already installed
     !pip install gensim
```

✓ 0s completed at 1:08 PM



Welcome To Colab

colab.research.google.com/#scrollTo=CpiwY1biHTvl

```python
import nltk
import re
from gensim.utils import simple_preprocess
import pandas as pd

# Download NLTK data (if needed)
nltk.download('punkt')

# Load your dataframe (replace with your actual file path)
file_path = '/content/drive/MyDrive/data/surah/surah_info.csv'
df = pd.read_csv(file_path)

# Print the actual column names to identify the correct ones
print(df.columns)

# Assuming your columns are named 'English Title', 'Arabic Title',
```

```
Index(['SurahNumber', 'EnglishTitle', 'ArabicTitle', 'RomanTitle',
       'NumberOfVerses', 'NumberOfRukus', 'PlaceOfRevelation'],
      dtype='object')
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

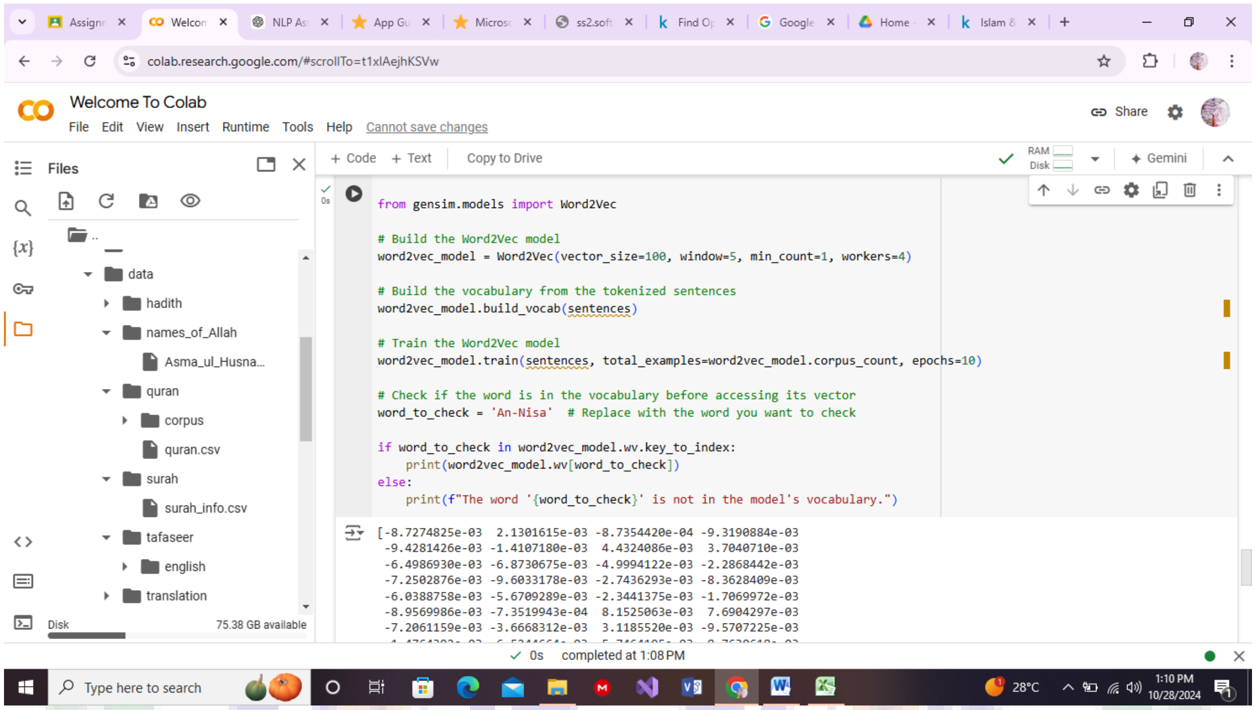
```python
import nltk
import re
from gensim.utils import simple_preprocess
import pandas as pd

# Download NLTK data (if needed)
nltk.download('punkt')

# Load your dataframe (replace with your act
file_path = '/content/drive/MyDrive/data/sur
df = pd.read_csv(file_path)

# Print the actual column names to identify
print(df.columns)
```

Responses may display inaccurate or offensive information that doesn't represent Google's views. Learn more

✓ 0s completed at 12:54 PM

Word2Vec Training:

Code:

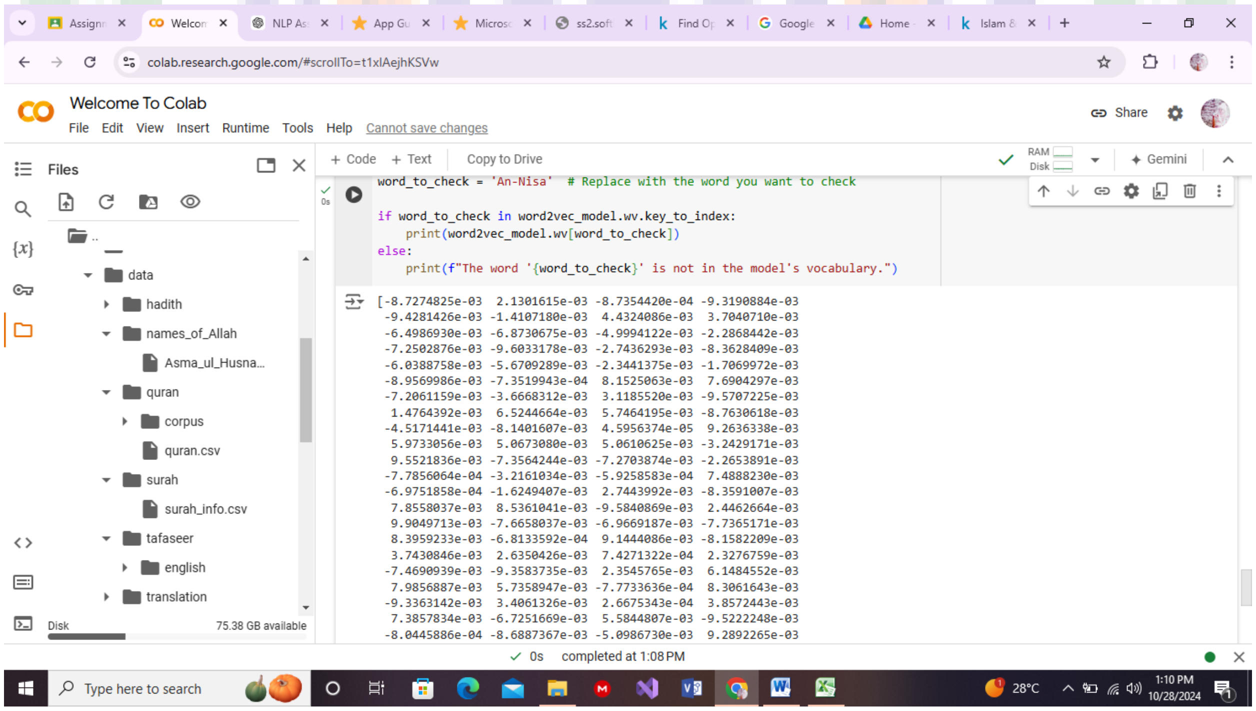


```python
from gensim.models import Word2Vec

# Build the Word2Vec model
word2vec_model = Word2Vec(vector_size=100, window=5, min_count=1, workers=4)

# Build the vocabulary from the tokenized sentences
word2vec_model.build_vocab(sentences)

# Train the Word2Vec model
word2vec_model.train(sentences, total_examples=word2vec_model.corpus_count, epochs=10)

# Check if the word is in the vocabulary before accessing its vector
word_to_check = 'An-Nisa'  # Replace with the word you want to check

if word_to_check in word2vec_model.wv.key_to_index:
    print(word2vec_model.wv[word_to_check])
else:
    print(f"The word '{word_to_check}' is not in the model's vocabulary.")
```

```
[-8.7274825e-03  2.1301615e-03 -8.7354420e-04 -9.3190884e-03
 -9.4281426e-03 -1.4107180e-03  4.4324086e-03  3.7040710e-03
 -6.4986930e-03 -6.8730675e-03 -4.9994122e-03 -2.2868442e-03
 -7.2502876e-03 -9.6033178e-03 -2.7436293e-03 -8.3628409e-03
 -6.0388758e-03 -5.6709289e-03 -2.3441375e-03 -1.7069972e-03
 -8.9569986e-03 -7.3519943e-04  8.1525063e-03  7.6904297e-03
 -7.2061159e-03 -3.6668312e-03  3.1185520e-03 -9.5707225e-03
```

✓ 0s completed at 1:08 PM

Output:



```python
word_to_check = 'An-Nisa'  # Replace with the word you want to check

if word_to_check in word2vec_model.wv.key_to_index:
    print(word2vec_model.wv[word_to_check])
else:
    print(f"The word '{word_to_check}' is not in the model's vocabulary.")
```

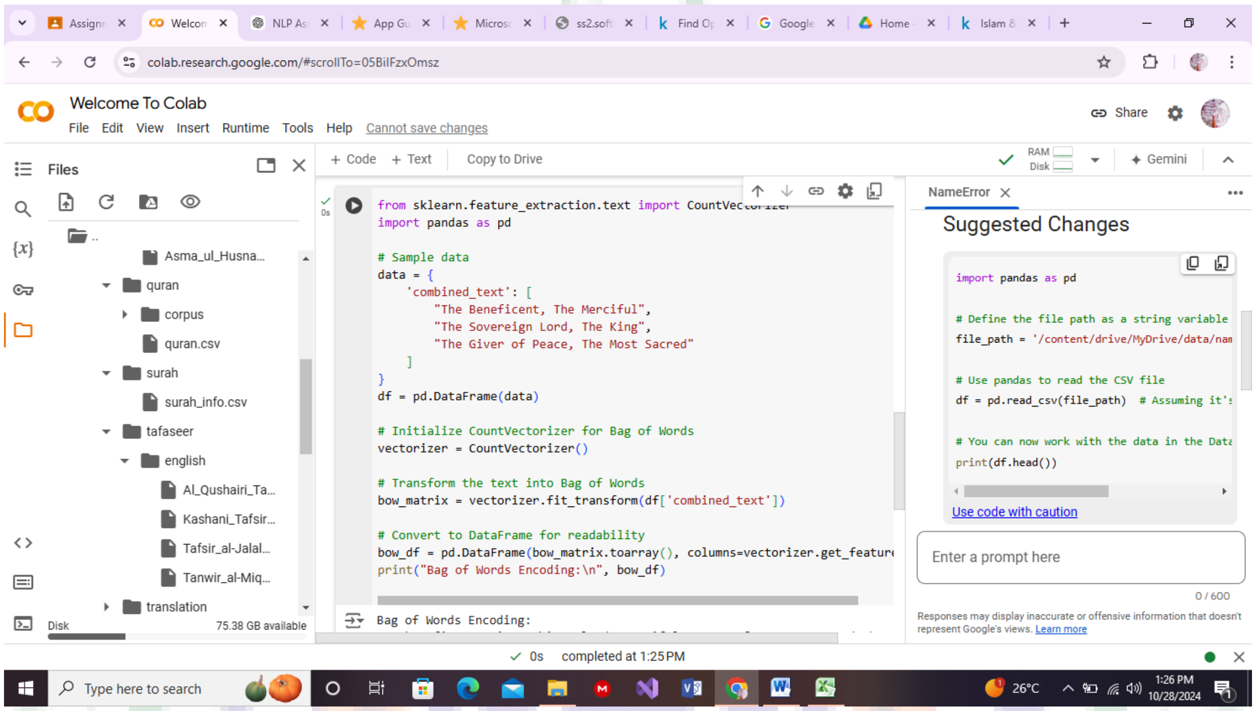
```
[-8.7274825e-03  2.1301615e-03 -8.7354420e-04 -9.3190884e-03
 -9.4281426e-03 -1.4107180e-03  4.4324086e-03  3.7040710e-03
 -6.4986930e-03 -6.8730675e-03 -4.9994122e-03 -2.2868442e-03
 -7.2502876e-03 -9.6033178e-03 -2.7436293e-03 -8.3628409e-03
 -6.0388758e-03 -5.6709289e-03 -2.3441375e-03 -1.7069972e-03
 -8.9569986e-03 -7.3519943e-04  8.1525063e-03  7.6904297e-03
 -7.2061159e-03 -3.6668312e-03  3.1185520e-03 -9.5707225e-03
  1.4764392e-03  6.5244664e-03  5.7464195e-03 -8.7630618e-03
 -4.5171441e-03 -8.1401607e-03  4.5956374e-05  9.2636338e-03
  5.9733056e-03  5.0673080e-03  5.0610625e-03 -3.2429171e-03
  9.5521836e-03 -7.3564244e-03 -7.2703874e-03 -2.2653891e-03
 -7.7856064e-04 -3.2161034e-03 -5.9258583e-04  7.4888230e-03
 -6.9751858e-04 -1.6249407e-03  2.7443992e-03 -8.3591007e-03
  7.8558037e-03  8.5361041e-03 -9.5840869e-03  2.4462664e-03
  9.9049713e-03 -7.6658037e-03 -6.9669187e-03 -7.7365171e-03
  8.3959233e-03 -6.8133592e-04  9.1444086e-03 -8.1582209e-03
  3.7430846e-03  2.6350426e-03  7.4271322e-04  2.3276759e-03
 -7.4690939e-03 -9.3583735e-03  2.3545765e-03  6.1484552e-03
  7.9856887e-03  5.7358947e-03 -7.7733636e-04  8.3061643e-03
 -9.3363142e-03  3.4061326e-03  2.6675343e-04  3.8572443e-03
  7.3857834e-03 -6.7251669e-03  5.5844807e-03 -9.5222248e-03
 -8.0445886e-04 -8.6887367e-03 -5.0986730e-03  9.2892265e-03
```

✓ 0s completed at 1:08 PM

Encoding Techniques (Bag of Words, One – Hot):

Bag Of Words:

Code:



```
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd

# Sample data
data = {
    'combined_text': [
        "The Beneficent, The Merciful",
        "The Sovereign Lord, The King",
        "The Giver of Peace, The Most Sacred"
    ]
}
df = pd.DataFrame(data)

# Initialize CountVectorizer for Bag of Words
vectorizer = CountVectorizer()

# Transform the text into Bag of Words
bow_matrix = vectorizer.fit_transform(df['combined_text'])

# Convert to DataFrame for readability
bow_df = pd.DataFrame(bow_matrix.toarray(), columns=vectorizer.get_feature
print("Bag of Words Encoding:\n", bow_df)
```

```
Bag of Words Encoding:
```

NameError

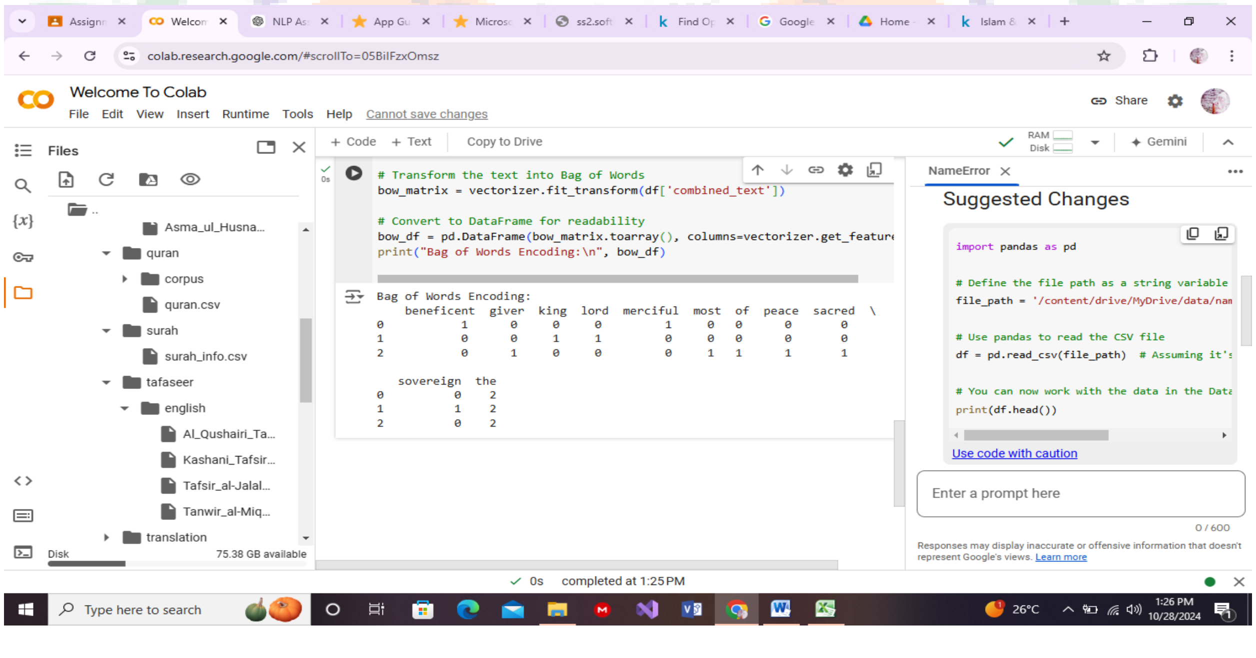
Suggested Changes

```
import pandas as pd

# Define the file path as a string variable
file_path = '/content/drive/MyDrive/data/nam

# Use pandas to read the CSV file
df = pd.read_csv(file_path)  # Assuming it's

# You can now work with the data in the Data
print(df.head())
```

Use code with caution

completed at 1:25 PM

Output:



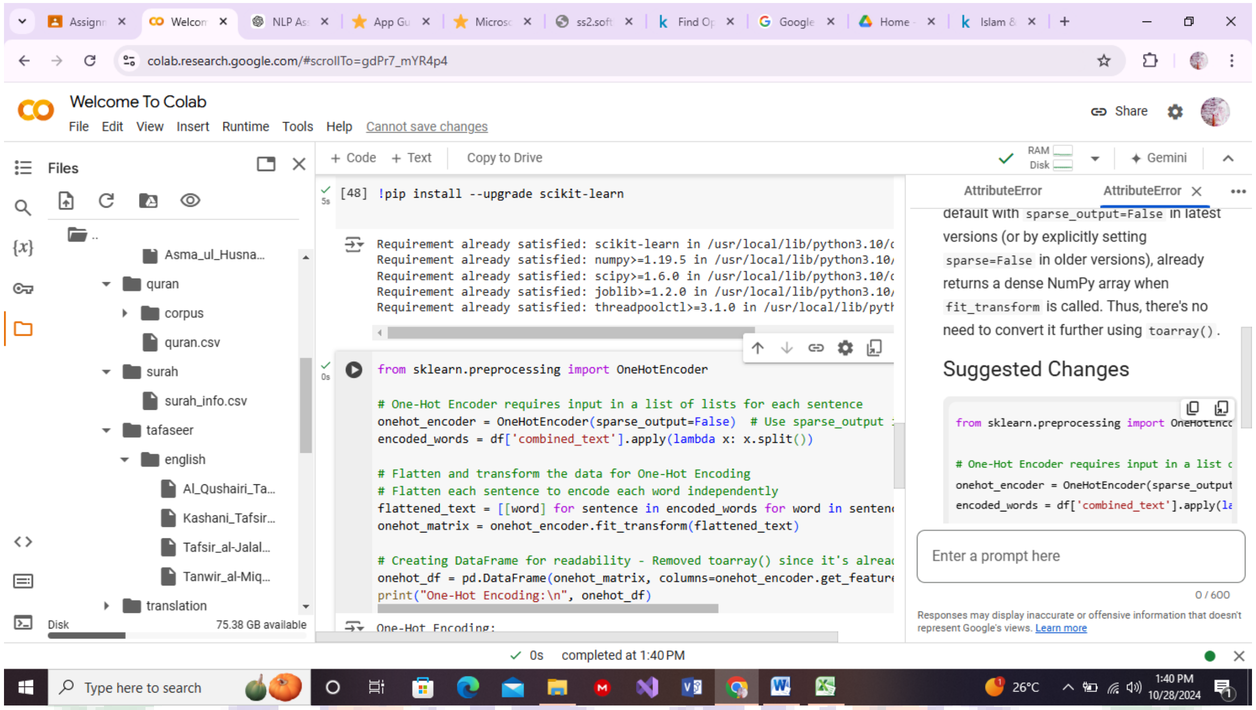
```
# Transform the text into Bag of Words
bow_matrix = vectorizer.fit_transform(df['combined_text'])

# Convert to DataFrame for readability
bow_df = pd.DataFrame(bow_matrix.toarray(), columns=vectorizer.get_feature
print("Bag of Words Encoding:\n", bow_df)
```

```
Bag of Words Encoding:
    beneficent  giver  king  lord  merciful  most  of  peace  sacred  \
0           1      0     0     0         1     0   0      0       0
1           0      0     1     1         0     0   0      0       0
2           0      1     0     0         0     1   1      1       1

   sovereign  the
0          0    2
1          1    2
2          0    2
```

completed at 1:25 PM

One – Hot:

Coding:



```
[48] !pip install --upgrade scikit-learn

Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/c
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.10/
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/c
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/pyth
```

```python
from sklearn.preprocessing import OneHotEncoder

# One-Hot Encoder requires input in a list of lists for each sentence
onehot_encoder = OneHotEncoder(sparse_output=False)  # Use sparse_output i
encoded_words = df['combined_text'].apply(lambda x: x.split())

# Flatten and transform the data for One-Hot Encoding
# Flatten each sentence to encode each word independently
flattened_text = [[word] for sentence in encoded_words for word in senten
onehot_matrix = onehot_encoder.fit_transform(flattened_text)

# Creating DataFrame for readability - Removed toarray() since it's alread
onehot_df = pd.DataFrame(onehot_matrix, columns=onehot_encoder.get_feature
print("One-Hot Encoding:\n", onehot_df)
```

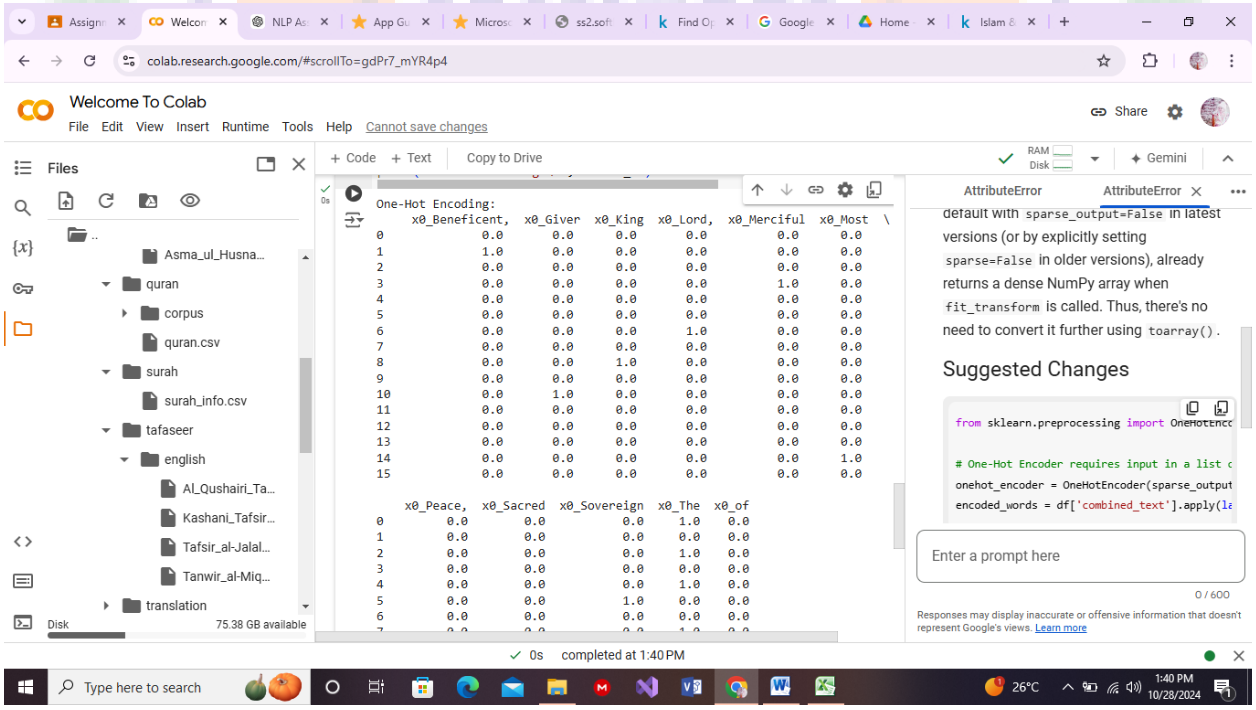
default with `sparse_output=False` in latest versions (or by explicitly setting `sparse=False` in older versions), already returns a dense NumPy array when `fit_transform` is called. Thus, there's no need to convert it further using `toarray()`.

Suggested Changes

```python
from sklearn.preprocessing import OneHotEnc

# One-Hot Encoder requires input in a list o
onehot_encoder = OneHotEncoder(sparse_output
encoded_words = df['combined_text'].apply(la
```

Responses may display inaccurate or offensive information that doesn't represent Google's views. Learn more

Output:



```
One-Hot Encoding:
     x0_Beneficent,  x0_Giver  x0_King  x0_Lord,  x0_Merciful  x0_Most  \
0              0.0       0.0      0.0       0.0          0.0      0.0
1              1.0       0.0      0.0       0.0          0.0      0.0
2              0.0       0.0      0.0       0.0          0.0      0.0
3              0.0       0.0      0.0       0.0          1.0      0.0
4              0.0       0.0      0.0       0.0          0.0      0.0
5              0.0       0.0      0.0       0.0          0.0      0.0
6              0.0       0.0      0.0       1.0          0.0      0.0
7              0.0       0.0      0.0       0.0          0.0      0.0
8              0.0       0.0      1.0       0.0          0.0      0.0
9              0.0       0.0      0.0       0.0          0.0      0.0
10             0.0       1.0      0.0       0.0          0.0      0.0
11             0.0       0.0      0.0       0.0          0.0      0.0
12             0.0       0.0      0.0       0.0          0.0      0.0
13             0.0       0.0      0.0       0.0          0.0      0.0
14             0.0       0.0      0.0       0.0          0.0      1.0
15             0.0       0.0      0.0       0.0          0.0      0.0

    x0_Peace,  x0_Sacred  x0_Sovereign  x0_The  x0_of
0         0.0        0.0           0.0     1.0    0.0
1         0.0        0.0           0.0     0.0    0.0
2         0.0        0.0           0.0     1.0    0.0
3         0.0        0.0           0.0     0.0    0.0
4         0.0        0.0           0.0     1.0    0.0
5         0.0        0.0           1.0     0.0    0.0
6         0.0        0.0           0.0     0.0    0.0
```

default with `sparse_output=False` in latest versions (or by explicitly setting `sparse=False` in older versions), already returns a dense NumPy array when `fit_transform` is called. Thus, there's no need to convert it further using `toarray()`.

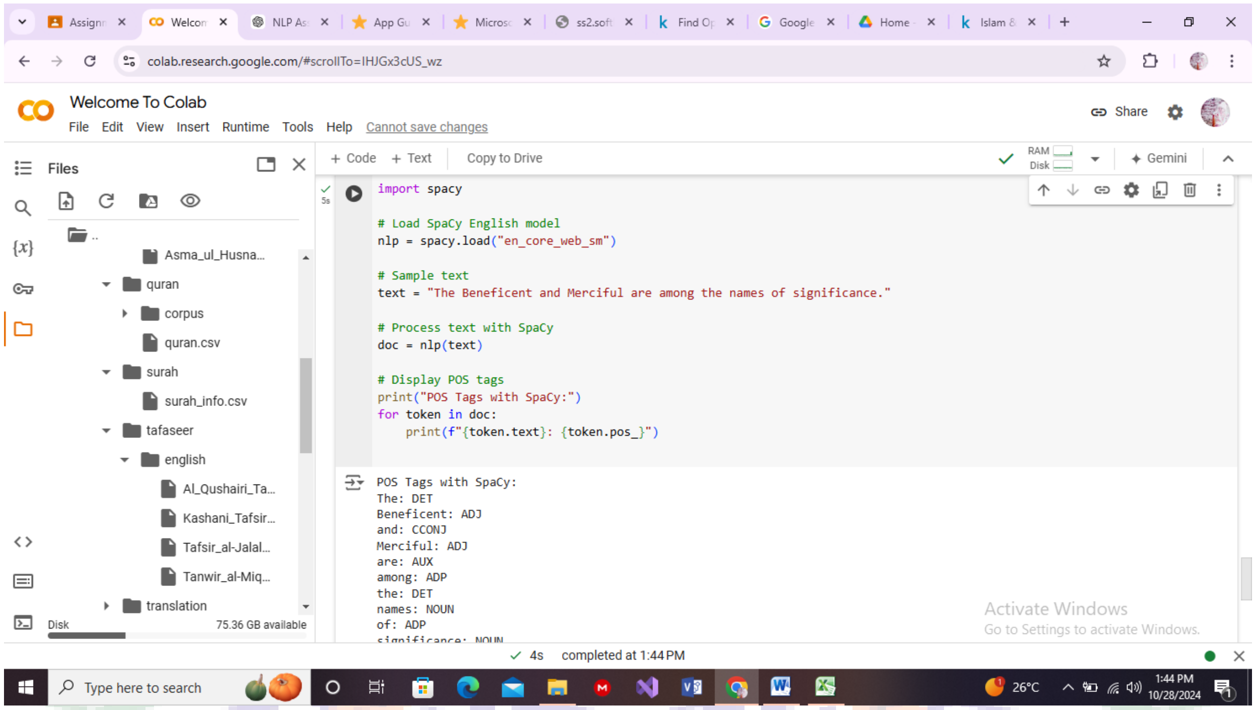
Suggested Changes

```python
from sklearn.preprocessing import OneHotEnc

# One-Hot Encoder requires input in a list o
onehot_encoder = OneHotEncoder(sparse_output
encoded_words = df['combined_text'].apply(la
```

Responses may display inaccurate or offensive information that doesn't represent Google's views. Learn more

Parts of Speech tagging:

Code + Output:


Welcome To Colab
File Edit View Insert Runtime Tools Help Cannot save changes
Files
Copy to Drive
Activate Windows
Go to Settings to activate Windows.
completed at 1:44 PM

```python
import spacy

# Load SpaCy English model
nlp = spacy.load("en_core_web_sm")

# Sample text
text = "The Beneficent and Merciful are among the names of significance."

# Process text with SpaCy
doc = nlp(text)

# Display POS tags
print("POS Tags with SpaCy:")
for token in doc:
    print(f"{token.text}: {token.pos_}")
```

```
POS Tags with SpaCy:
The: DET
Beneficent: ADJ
and: CCONJ
Merciful: ADJ
are: AUX
among: ADP
the: DET
names: NOUN
of: ADP
```

<u>Question #2:</u>

Perform any two of the basic NLP tasks listed below.

• Sentiment Analysis (using VADER, TextBlob)

• Named Entity Recognition (NER)

• Text Classification (Naive Bayes, Logistic Regression, SVM)

• Language Models (N-grams, Markov Chains)

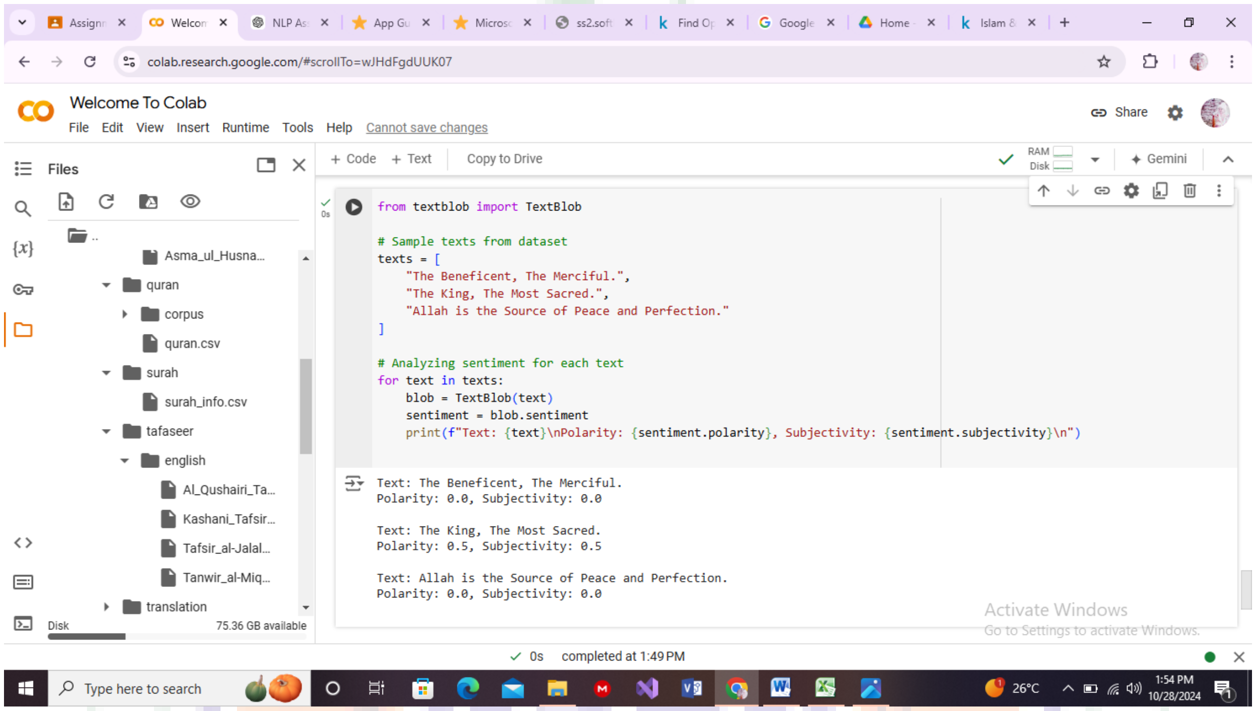
• Topic Modeling (LDA, Latent Semantic Analysis)

<u>Answer:</u>

1. Sentiment Analysis (Using TextBlob):

Sentiment analysis will evaluate the emotional tone in the texts. Although religious and AI content might have more neutral sentiments, it's still insightful for polarity analysis.

Code + Output:



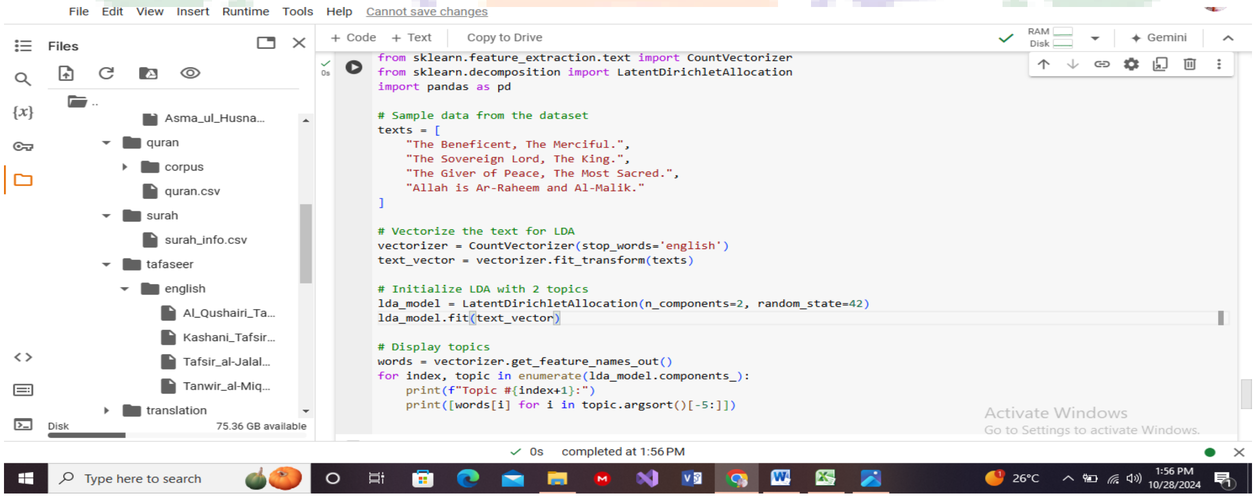
```
from textblob import TextBlob

# Sample texts from dataset
texts = [
    "The Beneficent, The Merciful.",
    "The King, The Most Sacred.",
    "Allah is the Source of Peace and Perfection."
]

# Analyzing sentiment for each text
for text in texts:
    blob = TextBlob(text)
    sentiment = blob.sentiment
    print(f"Text: {text}\nPolarity: {sentiment.polarity}, Subjectivity: {sentiment.subjectivity}\n")
```

```
Text: The Beneficent, The Merciful.
Polarity: 0.0, Subjectivity: 0.0

Text: The King, The Most Sacred.
Polarity: 0.5, Subjectivity: 0.5

Text: Allah is the Source of Peace and Perfection.
Polarity: 0.0, Subjectivity: 0.0
```

Topic Modeling (Using Latent Dirichlet Allocation - LDA):

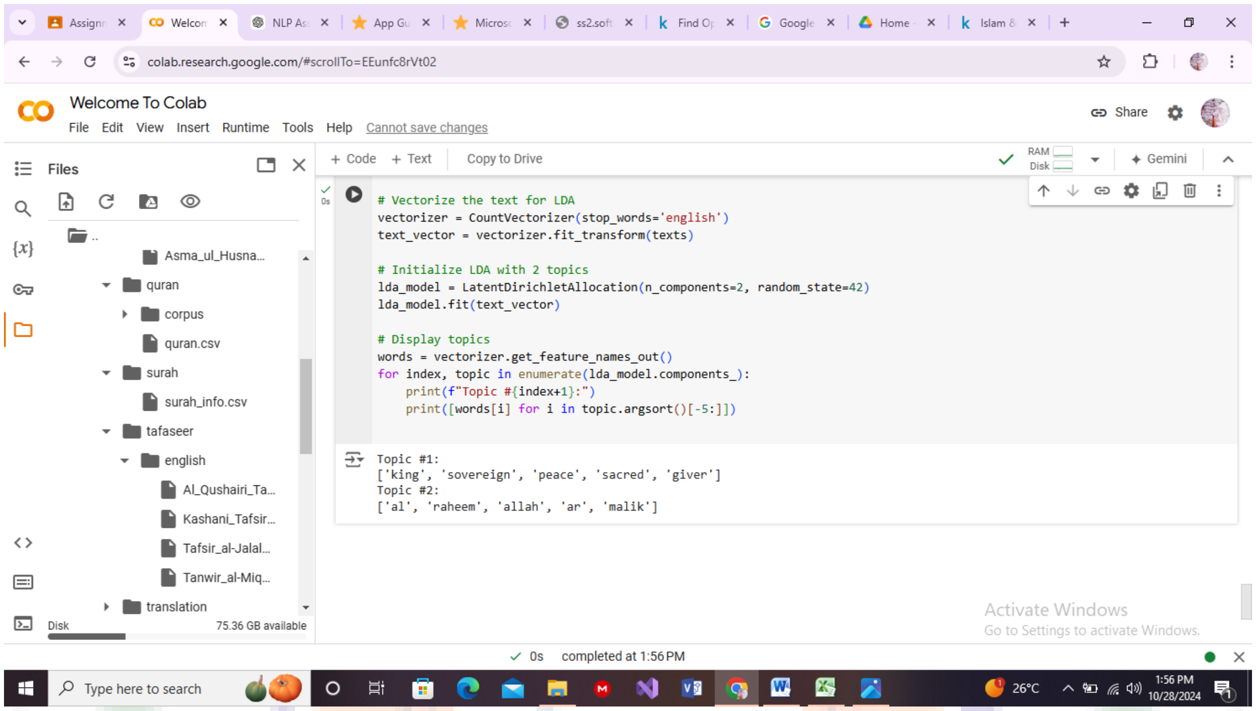
Topic modeling will reveal the main themes within the dataset.

Code:



```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
import pandas as pd

# Sample data from the dataset
texts = [
    "The Beneficent, The Merciful.",
    "The Sovereign Lord, The King.",
    "The Giver of Peace, The Most Sacred.",
    "Allah is Ar-Raheem and Al-Malik."
]

# Vectorize the text for LDA
vectorizer = CountVectorizer(stop_words='english')
text_vector = vectorizer.fit_transform(texts)

# Initialize LDA with 2 topics
lda_model = LatentDirichletAllocation(n_components=2, random_state=42)
lda_model.fit(text_vector)

# Display topics
words = vectorizer.get_feature_names_out()
for index, topic in enumerate(lda_model.components_):
    print(f"Topic #{index+1}:")
    print([words[i] for i in topic.argsort()[-5:]])
```

Output:



```
# Vectorize the text for LDA
vectorizer = CountVectorizer(stop_words='english')
text_vector = vectorizer.fit_transform(texts)

# Initialize LDA with 2 topics
lda_model = LatentDirichletAllocation(n_components=2, random_state=42)
lda_model.fit(text_vector)

# Display topics
words = vectorizer.get_feature_names_out()
for index, topic in enumerate(lda_model.components_):
    print(f"Topic #{index+1}:")
    print([words[i] for i in topic.argsort()[-5:]])
```

```
Topic #1:
['king', 'sovereign', 'peace', 'sacred', 'giver']
Topic #2:
['al', 'raheem', 'allah', 'ar', 'malik']
```