```
# Import necessary libraries
import numpy as np # For numerical operations
import pandas as pd # For data manipulation and analysis
import string # For string operations

from sklearn.feature_extraction.text import TfidfVectorizer # For converting text into TF-IDF features
from sklearn.metrics.pairwise import cosine_similarity # For calculating cosine similarity

import nltk # Natural Language Toolkit for text processing
from nltk.corpus import stopwords, wordnet # For stop words and WordNet lexical database
from nltk.tokenize import word_tokenize # For tokenizing text into words
from nltk.stem import WordNetLemmatizer # For lemmatizing words
```

```
# Download necessary NLTK data resources
nltk.download('punkt') # For sentence tokenization
nltk.download('stopwords') # For a list of common stop words
nltk.download('wordnet') # For the WordNet lexical database (used in WordNet similarity)
nltk.download('punkt_tab') # Another punkt tokenizer resource
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
True
```

```
# Load the dataset from a CSV file into a pandas DataFrame
df = pd.read_csv("Lab7_Text_Similarity_Dataset.csv")
# Display the first 5 rows of the DataFrame to inspect the data
df.head()
```

	Document_ID	Text
0	1	The cricket team won the match by scoring many...
1	2	Football players trained hard for the tournament
2	3	The athlete broke the world record in running
3	4	Tennis matches require speed and accuracy
4	5	The coach planned a new strategy for the game

Next steps: Generate code with df | New interactive sheet

```
# Define a set of English stop words for text cleaning
stop_words = set(stopwords.words('english'))
# Initialize the WordNet Lemmatizer
lemmatizer = WordNetLemmatizer()

# Define a preprocessing function for text
def preprocess(text):
    text = text.lower() # Convert text to lowercase
    text = text.translate(str.maketrans('', '', string.punctuation)) # Remove punctuation
    tokens = word_tokenize(text) # Tokenize the text into words
    tokens = [w for w in tokens if w not in stop_words] # Remove stop words
    tokens = [lemmatizer.lemmatize(w) for w in tokens] # Lemmatize words to their base form
    return tokens
```

```
# Initialize TF-IDF Vectorizer with a custom preprocessor
# The preprocessor function converts text into a string of preprocessed words for TF-IDF calculation
vectorizer = TfidfVectorizer(preprocessor=lambda x: " ".join(preprocess(x)))
# Fit the vectorizer to the 'Text' column of the DataFrame and transform the text into TF-IDF features
tfidf_matrix = vectorizer.fit_transform(df["Text"])
```

```
# Calculate cosine similarity between all pairs of documents based on their TF-IDF vectors
cos_sim = cosine_similarity(tfidf_matrix)
# Convert the cosine similarity matrix into a pandas DataFrame for better readability
cosine_df = pd.DataFrame(cos_sim)
```

```
# Display the first few rows of the cosine similarity DataFrame
cosine_df.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	1.000000	0.0	0.0	0.147205	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.000000	1.0	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.000000	0.0	1.0	0.000000	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.147205	0.0	0.0	1.000000	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.000000	0.0	0.0	0.000000	1.0	0.135906	0.0	0.135906	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Next steps: Generate code with `cosine_df` New interactive sheet

```
# Define a function to calculate Jaccard similarity between two documents
def jaccard_similarity(doc1, doc2):
    set1 = set(preprocess(doc1)) # Preprocess and convert document 1 to a set of words
    set2 = set(preprocess(doc2)) # Preprocess and convert document 2 to a set of words
    # Calculate Jaccard similarity using the formula: |intersection| / |union|
    return len(set1 & set2) / len(set1 | set2)

# Calculate and print Jaccard similarity for the first 5 pairs of consecutive documents
for i in range(5):
    score = jaccard_similarity(df["Text"][i], df["Text"][i+1])
    print(f"Jaccard similarity between Doc {i+1} and Doc {i+2}: {score}")
```

```
Jaccard similarity between Doc 1 and Doc 2: 0.0
Jaccard similarity between Doc 2 and Doc 3: 0.0
Jaccard similarity between Doc 3 and Doc 4: 0.0
Jaccard similarity between Doc 4 and Doc 5: 0.0
Jaccard similarity between Doc 5 and Doc 6: 0.1111111111111111
```

```
# Define a function to calculate WordNet-based semantic similarity between two sentences
def wordnet_similarity(sent1, sent2):
    tokens1 = preprocess(sent1) # Preprocess sentence 1 to get a list of tokens
    tokens2 = preprocess(sent2) # Preprocess sentence 2 to get a list of tokens

    scores = [] # List to store similarity scores between word pairs
    # Iterate through all word pairs from both sentences
    for w1 in tokens1:
        for w2 in tokens2:
            syn1 = wordnet.synsets(w1) # Get WordNet synsets for word 1
            syn2 = wordnet.synsets(w2) # Get WordNet synsets for word 2
            # If both words have synsets, calculate Wu-Palmer similarity
            if syn1 and syn2:
                sim = syn1[0].wup_similarity(syn2[0]) # Use the first synset for each word
                if sim:
                    scores.append(sim) # Add similarity score if it's not None
    # Return the mean of all calculated similarity scores, or 0 if no scores were calculated
    return np.mean(scores) if scores else 0
```

```
# Define pairs of document indices for which to calculate WordNet similarity
pairs = [
    (0, 1), (2, 3), (5, 6), (10, 11), (12, 13),
    (15, 16), (17, 18), (8, 9), (3, 4), (14, 19)
]

# Iterate through the defined pairs and print their WordNet similarity scores
for i, j in pairs:
    print(f"WordNet similarity Doc {i+1} & Doc {j+1}:",
          wordnet_similarity(df["Text"][i], df["Text"][j]))
```

```
WordNet similarity Doc 1 & Doc 2: 0.23563927472596205
WordNet similarity Doc 3 & Doc 4: 0.2261880994822171
WordNet similarity Doc 6 & Doc 7: 0.25648102877514645
WordNet similarity Doc 11 & Doc 12: 0.19409732682178502
WordNet similarity Doc 13 & Doc 14: 0.20449811518890468
WordNet similarity Doc 16 & Doc 17: 0.30928839678839676
WordNet similarity Doc 18 & Doc 19: 0.2502995305348247
WordNet similarity Doc 9 & Doc 10: 0.22425266082386824
WordNet similarity Doc 4 & Doc 5: 0.25552778044109314
WordNet similarity Doc 15 & Doc 20: 0.22688327668203828
```

```
Start coding or generate with AI.
```