```
# Install required libraries (only needed in Colab)
!pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/d
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-package
```

```
# Import libraries
import nltk
import string
import numpy as np
import pandas as pd

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

from nltk.corpus import stopwords, wordnet
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
```

```
# Download required NLTK resources
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt_tab')
```

```
[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]   Unzipping tokenizers/punkt_tab.zip.
True
```

```
documents = [
    # Sports
    "The football team won the championship match",
    "Cricket players trained hard for the tournament",
    "The athlete broke the world record",
    "The coach planned a new strategy for the game",
    "Fans celebrated the victory in the stadium",

    # Politics
    "The government announced new economic policies",
    "Elections were conducted peacefully across the country",
```

```
    "The parliament passed a new law",
    "Political parties debated healthcare reforms",
    "The president addressed the nation",

    # Health
    "Doctors recommend regular exercise for good health",
    "The hospital treated patients with care",
    "Healthy diet reduces the risk of diseases",
    "Vaccination prevents serious illnesses",
    "Mental health awareness is important",

    # Technology
    "Artificial intelligence is transforming technology",
    "Cybersecurity is crucial for data protection",
    "Software developers write efficient code",
    "Machine learning improves prediction accuracy",
    "New smartphones use advanced processors"
]
```

```
# Convert to DataFrame
df = pd.DataFrame(documents, columns=["Text"])
df.head()
```

	Text
0	The football team won the championship match
1	Cricket players trained hard for the tournament
2	The athlete broke the world record
3	The coach planned a new strategy for the game
4	Fans celebrated the victory in the stadium

Next steps: Generate code with df | New interactive sheet

```
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def preprocess_text(text):
    # Lowercase
    text = text.lower()

    # Remove punctuation and numbers
    text = text.translate(str.maketrans('', '', string.punctuation))

    # Tokenization
    tokens = word_tokenize(text)

    # Remove stopwords & lemmatize
    tokens = [lemmatizer.lemmatize(word) for word in tokens if word not in s
```

```
    return tokens
```

```
tfidf = TfidfVectorizer()
tfidf_matrix = tfidf.fit_transform(df["Text"])
```

```
cosine_sim = cosine_similarity(tfidf_matrix)

cosine_df = pd.DataFrame(cosine_sim, index=df["Text"], columns=df["Text"])
cosine_df.head()
```

Next steps: Generate code with cosine_df New interactive sheet

```
# Print sample similarity values
for i in range(5):
    for j in range(i+1, 5):
        print(f"Cosine similarity between sentence {i+1} and {j+1}: {cosine
```

```
Cosine similarity between sentence 1 and 2: 0.075
Cosine similarity between sentence 1 and 3: 0.162
Cosine similarity between sentence 1 and 4: 0.147
Cosine similarity between sentence 1 and 5: 0.148
Cosine similarity between sentence 2 and 3: 0.082
Cosine similarity between sentence 2 and 4: 0.165
Cosine similarity between sentence 2 and 5: 0.075
Cosine similarity between sentence 3 and 4: 0.161
Cosine similarity between sentence 3 and 5: 0.162
Cosine similarity between sentence 4 and 5: 0.147
```

```
def jaccard_similarity(tokens1, tokens2):
    set1 = set(tokens1)
    set2 = set(tokens2)
    return len(set1 & set2) / len(set1 | set2)
```

```
# STEP 7: Jaccard Similarity - Single Cell Working Code

import nltk
import string
from nltk.corpus import stopwords, wordnet
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

# Download required NLTK data (safe to run again)
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt_tab')

# Stopwords and Lemmatizer
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

# Preprocessing function
def preprocess_for_jaccard(text):
    text = text.lower()
    text = text.translate(str.maketrans('', '', string.punctuation))
    tokens = word_tokenize(text)
    # Add lemmatization here
    tokens = [lemmatizer.lemmatize(word) for word in tokens if word not in s
    return tokens

# Jaccard similarity function
def jaccard_similarity(text1, text2):
    set1 = set(preprocess_for_jaccard(text1))
    set2 = set(preprocess_for_jaccard(text2))
    return len(set1.intersection(set2)) / len(set1.union(set2))

# The DataFrame `df` is already created from `documents` in a previous cell.
# No need to load a new CSV.
```

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

```
Jaccard similarity between Doc 1 and Doc 2: 0.000
Jaccard similarity between Doc 1 and Doc 3: 0.000
Jaccard similarity between Doc 1 and Doc 4: 0.000
Jaccard similarity between Doc 1 and Doc 5: 0.000
Jaccard similarity between Doc 2 and Doc 3: 0.000
Jaccard similarity between Doc 2 and Doc 4: 0.000
Jaccard similarity between Doc 2 and Doc 5: 0.000
Jaccard similarity between Doc 3 and Doc 4: 0.000
Jaccard similarity between Doc 3 and Doc 5: 0.000
Jaccard similarity between Doc 4 and Doc 5: 0.000
[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!
```

```
def wordnet_similarity(word1, word2):
    syn1 = wordnet.synsets(word1)
    syn2 = wordnet.synsets(word2)

    if syn1 and syn2:
        return syn1[0].wup_similarity(syn2[0])
    else:
        return None
```

```
# Test semantic similarity
word_pairs = [
    ("doctor", "physician"),
    ("football", "cricket"),
    ("health", "fitness"),
    ("government", "parliament"),
    ("computer", "technology"),
    ("disease", "illness"),
    ("law", "policy"),
    ("athlete", "player"),
    ("software", "program"),
    ("election", "vote")
]

for w1, w2 in word_pairs:
    print(f"{w1} - {w2} similarity:", wordnet_similarity(w1, w2))
```

```
doctor - physician similarity: 1.0
football - cricket similarity: 0.09090909090909091
```

```
health - fitness similarity: 0.375
government - parliament similarity: 0.5333333333333333
computer - technology similarity: 0.1111111111111111
disease - illness similarity: 0.9473684210526315
law - policy similarity: 0.2857142857142857
athlete - player similarity: 0.6666666666666666
software - program similarity: 0.26666666666666666
election - vote similarity: 0.625
```