```
!pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from nltk) (8.3.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages (from nltk) (1.5.3)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nltk) (2025.11.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from nltk) (4.67.1)
```

```
import pandas as pd
import numpy as np
import re
import string

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

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

```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[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!
True
```

```
documents = [
# Sports
"Football is a popular outdoor sport played worldwide",
"Cricket is a bat and ball game enjoyed by many countries",
"Basketball requires agility and teamwork",
"Tennis is played with a racket and a ball",
"Players train daily to improve their fitness",

# Politics
"The government passed a new economic policy",
"Elections are conducted to choose leaders",
"Parliament debated the new law for hours",
"Political parties campaign before elections",
"Citizens vote to select their representatives",

# Health
"Doctors recommend regular exercise for good health",
"A balanced diet includes fruits and vegetables",
"Vaccination prevents serious diseases",
"Drinking water keeps the body hydrated",
"Mental health is as important as physical health",

# Technology
"Artificial intelligence is transforming industries",
"Computers process data at high speed",
"Cyber security protects digital information",
"Software development requires logical thinking",
"Smartphones connect people through the internet"
]

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

	Text
0	Football is a popular outdoor sport played wor...
1	Cricket is a bat and ball game enjoyed by many...
2	Basketball requires agility and teamwork
3	Tennis is played with a racket and a ball
4	Players train daily to improve their fitness

Next steps: Generate code with df | New interactive sheet

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

def preprocess(text):
    text = text.lower()
    text = re.sub(r'[^a-z\s]', '', text)
    tokens = word_tokenize(text)
    tokens = [w for w in tokens if w not in stop_words]
    tokens = [lemmatizer.lemmatize(w) for w in tokens]
    return " ".join(tokens)

df["Clean_Text"] = df["Text"].apply(preprocess)
df.head()
```

	Text	Clean_Text
0	Football is a popular outdoor sport played wor...	football popular outdoor sport played worldwide
1	Cricket is a bat and ball game enjoyed by many...	cricket bat ball game enjoyed many country
2	Basketball requires agility and teamwork	basketball requires agility teamwork
3	Tennis is played with a racket and a ball	tennis played racket ball
4	Players train daily to improve their fitness	player train daily improve fitness

Next steps: Generate code with df | New interactive sheet

```
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(df["Clean_Text"])
```

```
cosine_sim_matrix = cosine_similarity(tfidf_matrix)

cosine_df = pd.DataFrame(
    cosine_sim_matrix,
    index=df["Text"],
    columns=df["Text"]
)

cosine_df.head()
```

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

```
             Football   Cricket               Tennis  Players
```

```
from nltk.tokenize import word_tokenize

def jaccard_similarity(text1, text2):
    tokens1 = set(word_tokenize(text1))
    tokens2 = set(word_tokenize(text2))

    intersection = tokens1.intersection(tokens2)
    union = tokens1.union(tokens2)

    return len(intersection) / len(union)

# Compare first 10 pairs (0 with 1, 2 with 3, ...)
for i in range(0, 10, 2):
    t1 = df["Clean_Text"][i]
    t2 = df["Clean_Text"][i+1]

    score = jaccard_similarity(t1, t2)

    print("Text 1:", df["Text"][i])
    print("Text 2:", df["Text"][i+1])
    print("Jaccard Similarity:", round(score, 3))
    print("-" * 60)
```

```
Text 1: Football is a popular outdoor sport played worldwide
Text 2: Cricket is a bat and ball game enjoyed by many countries
Jaccard Similarity: 0.0
------------------------------------------------------------
Text 1: Basketball requires agility and teamwork
Text 2: Tennis is played with a racket and a ball
Jaccard Similarity: 0.0
------------------------------------------------------------
Text 1: Players train daily to improve their fitness
Text 2: The government passed a new economic policy
Jaccard Similarity: 0.0
------------------------------------------------------------
Text 1: Elections are conducted to choose leaders
```

Tennis is played with a racket and a ball Players train daily to improve their fitness

0.170795 0.157683 0.0 1.000000 0.0 0.0 0.0 0.0 0.0

0.000000 0.000000 0.0 0.000000 1.0 0.0 0.0 0.0 0.0

```
Text 2: Parliament debated the new law for hours
Jaccard Similarity: 0.0
------------------------------------------------------------
Text 1: Political parties campaign before elections
Text 2: Citizens vote to select their representatives
Jaccard Similarity: 0.0
------------------------------------------------------------
```

```
def wordnet_sim(w1, w2):
    syn1 = wordnet.synsets(w1)
    syn2 = wordnet.synsets(w2)
    if syn1 and syn2:
        return syn1[0].wup_similarity(syn2[0]) or 0
    return 0

def sentence_similarity(s1, s2):
    words1 = s1.split()
    words2 = s2.split()
    scores = []
    for w1 in words1:
        for w2 in words2:
            scores.append(wordnet_sim(w1, w2))
    return np.mean(scores)

# check first 10 pairs
for i in range(10):
    print(i, "vs", i+1, ":",
          sentence_similarity(df["Clean_Text"][i], df["Clean_Text"][i+1]))
```

```
0 vs 1 : 0.24580577661073016
1 vs 2 : 0.22580309052832273
2 vs 3 : 0.2910143546898578
3 vs 4 : 0.22685641542839688
4 vs 5 : 0.19640992340992341
5 vs 6 : 0.24637015762015765
6 vs 7 : 0.21854260935143288
7 vs 8 : 0.28003361834244184
```

```
8 vs 9 : 0.2385151307026307
9 vs 10 : 0.2366872914667032
```