```
# Text preprocessing
import re
import string

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

# Vectorization and similarity
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

# Data handling
import pandas as pd
```

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

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
True
```

```
data = {
    "Sentence 1": [
        "The doctor treated the patient",
        "Cats are playing in the garden",
        "I love machine learning",
        "The car is very fast",
        "He is eating food",
        "The teacher explains the lesson",
        "Dogs are loyal animals",
        "She bought a new phone",
        "The child is happy",
        "He reads a book"
    ],
    "Sentence 2": [
        "The physician helped the sick person",
        "Dogs are playing outside",
        "I enjoy artificial intelligence",
        "The automobile moves quickly",
        "He consumes a meal",
        "The professor teaches the class",
        "Animals are faithful",
        "She purchased a smartphone",
        "The kid feels joyful",
        "He is reading a novel"
    ]
}

df = pd.DataFrame(data)
df
```

	Sentence 1	Sentence 2
0	The doctor treated the patient	The physician helped the sick person
1	Cats are playing in the garden	Dogs are playing outside
2	I love machine learning	I enjoy artificial intelligence
3	The car is very fast	The automobile moves quickly
4	He is eating food	He consumes a meal
5	The teacher explains the lesson	The professor teaches the class
6	Dogs are loyal animals	Animals are faithful
7	She bought a new phone	She purchased a smartphone
8	The child is happy	The kid feels joyful
9	He reads a book	He is reading a novel

Next steps: Generate code with df | New interactive sheet

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

def preprocess(text):
    # Convert to lowercase
    text = text.lower()

    # Remove punctuation and numbers
    text = re.sub(r'[^a-z\s]', '', text)

    # Tokenize
    tokens = word_tokenize(text)

    # Remove stopwords and lemmatize
    tokens = [lemmatizer.lemmatize(word) for word in tokens if word not in stop_words]

    return tokens
```

```
nltk.download('punkt_tab')

# Join tokens back into strings for TF-IDF
df["Clean S1"] = df["Sentence 1"].apply(lambda x: " ".join(preprocess(x)))
df["Clean S2"] = df["Sentence 2"].apply(lambda x: " ".join(preprocess(x)))

vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(
    df["Clean S1"].tolist() + df["Clean S2"].tolist()
)
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
```

```
cosine_scores = []

for i in range(len(df)):
    vec1 = tfidf_matrix[i]
    vec2 = tfidf_matrix[i + len(df)]
    score = cosine_similarity(vec1, vec2)[0][0]
    cosine_scores.append(score)

df["Cosine Similarity"] = cosine_scores
df[["Sentence 1", "Sentence 2", "Cosine Similarity"]]
```

	Sentence 1	Sentence 2	Cosine Similarity
0	The doctor treated the patient	The physician helped the sick person	0.000000
1	Cats are playing in the garden	Dogs are playing outside	0.290852
2	I love machine learning	I enjoy artificial intelligence	0.000000
3	The car is very fast	The automobile moves quickly	0.000000
4	He is eating food	He consumes a meal	0.000000
5	The teacher explains the lesson	The professor teaches the class	0.000000
6	Dogs are loyal animals	Animals are faithful	0.363753
7	She bought a new phone	She purchased a smartphone	0.000000
8	The child is happy	The kid feels joyful	0.000000
9	He reads a book	He is reading a novel	0.000000

```
def jaccard_similarity(s1, s2):
    set1 = set(preprocess(s1))
    set2 = set(preprocess(s2))
    return len(set1 & set2) / len(set1 | set2)

df["Jaccard Similarity"] = df.apply(
    lambda row: jaccard_similarity(row["Sentence 1"], row["Sentence 2"]), axis=1
)

df[["Sentence 1", "Sentence 2", "Jaccard Similarity"]]
```

	Sentence 1	Sentence 2	Jaccard Similarity
0	The doctor treated the patient	The physician helped the sick person	0.00
1	Cats are playing in the garden	Dogs are playing outside	0.20
2	I love machine learning	I enjoy artificial intelligence	0.00
3	The car is very fast	The automobile moves quickly	0.00
4	He is eating food	He consumes a meal	0.00
5	The teacher explains the lesson	The professor teaches the class	0.00
6	Dogs are loyal animals	Animals are faithful	0.25
7	She bought a new phone	She purchased a smartphone	0.00
8	The child is happy	The kid feels joyful	0.00
9	He reads a book	He is reading a novel	0.00

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

    if not syn1 or not syn2:
        return 0

    return syn1[0].wup_similarity(syn2[0]) or 0
```

```
semantic_scores = []

for i in range(len(df)):
    words1 = preprocess(df.iloc[i]["Sentence 1"])
    words2 = preprocess(df.iloc[i]["Sentence 2"])

    scores = []
    for w1 in words1:
        for w2 in words2:
            sim = wordnet_similarity(w1, w2)
            if sim:
                scores.append(sim)

    semantic_scores.append(sum(scores)/len(scores) if scores else 0)

df["WordNet Similarity"] = semantic_scores
df[["Sentence 1", "Sentence 2", "WordNet Similarity"]]
```

	Sentence 1	Sentence 2	WordNet Similarity
0	The doctor treated the patient	The physician helped the sick person	0.368450
1	Cats are playing in the garden	Dogs are playing outside	0.380162
2	I love machine learning	I enjoy artificial intelligence	0.247730
3	The car is very fast	The automobile moves quickly	0.336376