```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
from nltk import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk import pos_tag
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import pandas as pd

sent = "Sachin is considered to be one of the greatest cricket players. Virat is the captain of the Indian cricket team."
print("----- SENTENCES -----")
print(sent)

print("----- WORD TOKENIZATION -----")
print(word_tokenize(sent))
print("----- SENT TOKENIZATION -----")
print(sent_tokenize(sent))

stop_words = set(stopwords.words('english'))

token = word_tokenize(sent)
cleaned_token = [word for word in token if word.lower() not in stop_words]

words = [cleaned_word.lower() for cleaned_word in cleaned_token if cleaned_word.isalpha()]
print("----- WORDS -----")
print(words)

# STEMMING
stemmer = PorterStemmer()
```

```
port_stemmer_output = [stemmer.stem(word) for word in words]
print("----- STEMMING -----")
print(port_stemmer_output)

# LEMMATIZATION
lemmatizer = WordNetLemmatizer()
lemmatizer_output = [lemmatizer.lemmatize(word) for word in words]
print("----- LEMMATIZATION -----")
print(lemmatizer_output)

# POS TAGGING
tagged = pos_tag(cleaned_token)
print("----- POS TAGGING -----")
print(tagged)

docs = [
   "Sachin is considered to be one of the greatest cricket players.",
   "Federer is considered one of the greatest tennis players.",
   "Nadal is considered one of the greatest tennis players.",
   "Virat is the captain of the Indian cricket team."
]

vectorizer = TfidfVectorizer(analyzer="word", norm=None, use_idf=True, smooth_idf=True)
tfidfMat = vectorizer.fit_transform(docs)

features_names = vectorizer.get_feature_names_out()
print("----- FEATURE NAMES -----")
print(features_names)

dense = tfidfMat.todense()
denselist = dense.tolist()
df = pd.DataFrame(denselist, columns=features_names)
```

```
docsList = ['Docs_1', 'Docs_2', 'Docs_3', 'Docs_4']
skDocsIfIdfdf = pd.DataFrame(tfidfMat.todense(), index=docsList, columns=features_names)
print("----- SK DOCS -----")
print(skDocsIfIdfdf)

csim = cosine_similarity(tfidfMat, tfidfMat)

csimDf = pd.DataFrame(csim, index=docsList, columns=docsList)
print("----- COSINE SIMILARITY -----")
print(csimDf)
```

OUTPUT-