FDS CODE- 1.1

Code for NER and knowledge graph

```
import pandas as pd
import nltk
import spacy
import collections
import networkx as nx
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

# Download necessary NLTK resources
nltk.download('punkt')
nltk.download('stopwords')

# Load the CSV file
# Replace 'your_file.csv' with the path to your CSV file
csv_file = 'your_file.csv'
df = pd.read_csv(csv_file)

# Combine all comments into a single string
text_data = ' '.join(df['comments'].astype(str).tolist())  # Assuming
the column is named 'comments'

# Step 1: Eliminate stop words and get word counts
stop_words = set(stopwords.words('english'))
words = word_tokenize(text_data.lower())  # Tokenize and lower case
filtered_words = [word for word in words if word.isalpha() and word
not in stop_words]

# Count the top 10 words
word_counts = collections.Counter(filtered_words)
top_10_words = word_counts.most_common(10)

print("Top 10 words:")
for word, count in top_10_words:
    print(f"{word}: {count}")

# Step 2: Named Entity Recognition (NER)
nlp = spacy.load("en_core_web_sm")
doc = nlp(text_data)

print("\nNamed Entities:")
```

```
for ent in doc.ents:
    print(f"{ent.text} - {ent.label_}")

# Step 3: Derive Knowledge Graph
edges = []
for ent in doc.ents:
    edges.append((ent.text, ent.label_))

# Create a graph from the edges
graph = nx.Graph()
graph.add_edges_from(edges)

# Step 4: Draw the knowledge graph
plt.figure(figsize=(10, 6))
nx.draw(graph, with_labels=True, node_color='skyblue', node_size=2000,
font_size=10, font_color='black')
plt.title('Knowledge Graph')
plt.show()
```

Code for Wordcloud

```
from wordcloud import WordCloud

# Generate a word cloud
wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(text_data)

# Display the word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud')
plt.show()
```

Code for N-gram analysis

```
from nltk import ngrams

# Generate bigrams
bigrams = ngrams(filtered_words, 2)
bigram_counts = collections.Counter(bigrams)
top_10_bigrams = bigram_counts.most_common(10)

print("\nTop 10 Bigrams:")
for bigram, count in top_10_bigrams:
    print(f"{' '.join(bigram)}: {count}")
```

Code for TF-IDF vectorization

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(stop_words='english')
tfidf_matrix = vectorizer.fit_transform(df['comments'])

# Convert to DataFrame for better visualization
tfidf_df = pd.DataFrame(tfidf_matrix.toarray(),
columns=vectorizer.get_feature_names_out())
print("\nTF-IDF Matrix:")
print(tfidf_df.head())
```