***Data preprocessing and Tagging***

*Data preprocessing in the first part of code*

This code is a Python script that performs text preprocessing, feature extraction, and tag prediction on a dataset containing descriptions of various items. The script can be divided into several sections:

1. Data Loading and Cleaning.

- The script loads a CSV file containing descriptions into a pandas DataFrame.

- It then cleans the data by removing rows with missing descriptions and duplicates.

2. Text Preprocessing.

- The script tokenizes, lemmatizes, and removes stopwords from the descriptions.

- It also creates separate columns for nouns, verbs, and adjectives by performing Part-of-Speech (POS) tagging.

3. Saving Preprocessed Data.

- The preprocessed data is saved to a new CSV file for further analysis or modeling.

4. Tag Extraction.

- The script extracts tags from descriptions by performing POS tagging and extracting the first noun encountered in each description.

- A dictionary is created to map names to their corresponding tags.

5. Tag Prediction.

- The script creates a pipeline using CountVectorizer and Multinomial Naive Bayes to predict tags based on descriptions.

- The model is trained and tested using a portion of the preprocessed data.

- The accuracy of the model is printed.

6. Example Usage.

- The script provides a function to predict tags for a given phrase using the trained model.

The code demonstrates how to perform text preprocessing, feature extraction, and tag prediction using Python libraries such as pandas, NLTK, and scikit-learn. The script can be used as a starting point for more advanced text analysis and machine learning tasks.

*Tagging Solution*

***5th Solution***

The script can be divided into several sections:

1. Data Loading and Preprocessing.

- The script loads a CSV file containing descriptions into a pandas DataFrame.

- It initializes NLTK tools for stopword removal and lemmatization.

- It loads a spaCy model for tokenization.

- The descriptions are preprocessed using spaCy to tokenize, filter out stopwords, punctuation, and lemmatize the tokens.

2. Term Scoring and Ranking.

- The script scores terms based on their frequency in each description.

- The terms are sorted by frequency and the top 7 terms are selected for each description.

3. Mapping Names to Top 7 Tags.

- The script creates a dictionary mapping names to their corresponding top 7 tags.

4. Counting and Naming Tags.

- The script initializes a defaultdict to store counts and names for each tag.

- It updates the counts and names for each tag based on the top 7 tags for each description.

5. Adding Tag Counts and Names to DataFrame.

- The script adds the tag counts and names to the DataFrame for each description.

6. Printing Tag Information.

- The script prints the total count and names associated with each tag.

7. Saving DataFrame to a New CSV File.

- The script saves the DataFrame with added tag information to a new CSV file.