```
import pandas as pd
import nltk
from nltk import RegexpParser
from nltk.parse.stanford import StanfordParser
import spacy
```

```
nltk.download('maxent_treebank_pos_tagger')
nltk.download('treebank')
nltk.download('punkt')
nltk.download('words')
nltk.download('maxent_ne_chunker')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package maxent_treebank_pos_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/maxent_treebank_pos_tagger.zip.
[nltk_data] Downloading package treebank to /root/nltk_data...
[nltk_data]   Unzipping corpora/treebank.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Unzipping corpora/words.zip.
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping chunkers/maxent_ne_chunker.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
True
```

```
df = pd.read_csv('/content/lemmatized_data.csv')
df.head(5)
```

	Unnamed: 0.1	Unnamed: 0	id	movie_name	synopsis	genre	filtered_synopsis	lemmatized_synopsis
0	0	0	44978	Super Me	A young scriptwriter starts bringing valuable ...	fantasy	young scriptwriter starts bringing valuable ob...	young scriptwriter start bringing valuable obj...
1	1	1	50185	Entity Project	A director and her friends renting a haunted h...	horror	director friends renting haunted house capture...	director friend renting haunted house capture ...
2	2	2	34131	Behavioral Family Therapy for Serious Psychiat...	This is an educational video for families and ...	family	educational video families family therapists d...	educational video family family therapist desc...
3	3	3	78522	Blood Glacier	Scientists working in the Austrian Alps discov...	scifi	Scientists working Austrian Alps discover glac...	Scientists working Austrian Alps discover glac...
4	4	4	2206	Apat na anino	Buy Day - Four Men Widely - Apart in	action	Buy Day - Four Men Widely	Buy Day - Four Men Widely -

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df['tokenized'] = df['synopsis'].apply(nltk.word_tokenize)
df['tokenized']
```

	tokenized
0	[A, young, scriptwriter, starts, bringing, val...
1	[A, director, and, her, friends, renting, a, h...
2	[This, is, an, educational, video, for, famili...
3	[Scientists, working, in, the, Austrian, Alps,...
4	[Buy, Day, -, Four, Men, Widely, -, Apart, in,...
...	...
42102	[A, ragtag, gang, of, international, talking-d...
42103	[A, seductive, woman, gets, involved, in, rela...
42104	[Duyen, ,, a, wedding, dress, staff, ,, who, d...
42105	[The, people, of, a, crowded, colony, in, Coim...
42106	[Margo, is, a, little, mouse, that, lives, qui...

42107 rows × 1 columns

```
df['entities'] = df['tokenized'].apply(nltk.pos_tag)
df['entities']
```

	entities
0	[(A, DT), (young, JJ), (scriptwriter, NN), (st...
1	[(A, DT), (director, NN), (and, CC), (her, PRP...
2	[(This, DT), (is, VBZ), (an, DT), (educational...
3	[(Scientists, NNS), (working, VBG), (in, IN), ...
4	[(Buy, NNP), (Day, NNP), (-, :), (Four, CD), (...
...	...
42102	[(A, DT), (ragtag, NN), (gang, NN), (of, IN), ...
42103	[(A, DT), (seductive, JJ), (woman, NN), (gets,...
42104	[(Duyen, NNP), (,, ,), (a, DT), (wedding, NN),...
42105	[(The, DT), (people, NNS), (of, IN), (a, DT), ...
42106	[(Margo, NNP), (is, VBZ), (a, DT), (little, JJ...

42107 rows × 1 columns

```
grammar_pattern = """
    GP: {<JJ.*|VBG><NN.*>+}
 """

"""
 GP stands for genre phrase which gives datapoints that help get information
 regarding the genre of the movie- JJ stands for adjective, JJ.* could parse superlative or comparitive
 adjective used in the synopsis- and NN.* stands for the nouns used in the movie synopsis- VBG stands for gerund where any verbs ending
 It will either parse and adjective - noun combination or a gerund noun
 combination to gain inofrmation on the genre of the movie by its synopsis
 """
```

```
'\n GP stands for genre phrase which gives datapoints that help get information\n regarding the genre of the movie- JJ stands for a
djective, JJ.* could parse superlative or comparitive\n adjective used in the synopsis- and NN.* stands for the nouns used in the m
ovie synopsis- VBG stands for gerund where any verbs ending with "ing" would be parsed\n It will either parse and adjective - noun
```

```
chunker = RegexpParser(grammar_pattern)
```

```
df['chunks'] = df['entities'].apply(chunker.parse)
df['chunks']
```

	chunks
0	[(A, DT), [(young, JJ), (scriptwriter, NN)], (...
1	[(A, DT), (director, NN), (and, CC), (her, PRP...
2	[(This, DT), (is, VBZ), (an, DT), [(educationa...
3	[(Scientists, NNS), (working, VBG), (in, IN), ...
4	[(Buy, NNP), (Day, NNP), (-, :), (Four, CD), (...
...	...
42102	[(A, DT), (ragtag, NN), (gang, NN), (of, IN), ...
42103	[(A, DT), [(seductive, JJ), (woman, NN)], (get...
42104	[(Duyen, NNP), (,, ,), (a, DT), (wedding, NN),...
42105	[(The, DT), (people, NNS), (of, IN), (a, DT), ...
42106	[(Margo, NNP), (is, VBZ), (a, DT), [(little, J...

42107 rows × 1 columns

```
nltk.Tree.fromstring(str(df['chunks'][42105])).pretty_print()
```

```
                                                                                                                  S
  ________________________________________________________________________________________________________________|_____
 |        |        |    |     |         |            |      |          |        |    |    |      |          |            |
 |        |        |    |     |         |            |      |          |        |    |    |      |          |            |
The/DT people/NNS of/IN a/DT in/IN Coimbatore/NNP city/NN go/VBP through/IN a/DT as/IN a/DT few/JJ heavily/RB armed/VBN criminals/NN
```

Deep Parsing

```
nlp = spacy.load('en_core_web_sm')
```

```
deep_parse_results = []
for sentence in df['synopsis']:
    doc = nlp(sentence)
```

```
dependencies = []

for token in doc:
    dependencies.append({
        "word":token.text,
        "lemma":token.lemma_,
        "pos":token.pos_,
        "dep":token.dep_,
        "head":token.head.text
    })
    deep_parse_results.append(dependencies)
```

```
deep_parse_results[0]
```

```
{'word': 'who', 'lemma': 'who', 'pos': 'PRON', 'dep': 'nsubj', 'head': 'are'},
{'word': 'are',
 'lemma': 'be',
 'pos': 'AUX',
 'dep': 'relcl',
 'head': 'people'},
{'word': 'not', 'lemma': 'not', 'pos': 'PART', 'dep': 'neg', 'head': 'are'},
{'word': 'willing',
 'lemma': 'willing',
 'pos': 'ADJ',
 'dep': 'acomp',
 'head': 'are'},
{'word': 'to', 'lemma': 'to', 'pos': 'PART', 'dep': 'aux', 'head': 'make'},
{'word': 'make',
 'lemma': 'make',
 'pos': 'VERB',
 'dep': 'xcomp',
 'head': 'willing'},
{'word': 'life',
 'lemma': 'life',
 'pos': 'NOUN',
 'dep': 'nsubj',
 'head': 'easy'},
{'word': 'easy',
 'lemma': 'easy',
 'pos': 'ADJ',
 'dep': 'ccomp',
 'head': 'make'},
{'word': 'for', 'lemma': 'for', 'pos': 'ADP', 'dep': 'prep', 'head': 'make'},
{'word': 'Margo',
 'lemma': 'Margo',
 'pos': 'PROPN',
 'dep': 'pobj',
 'head': 'for'},
{'word': '.', 'lemma': '.', 'pos': 'PUNCT', 'dep': 'punct', 'head': 'are'}]
```

```
from spacy import displacy
text = nlp(df['synopsis'][0])
```

```
displacy.serve(text, style="dep")
```

```
/usr/local/lib/python3.10/dist-packages/spacy/displacy/__init__.py:106: UserWarning: [W011] It looks like you're calling displacy.se
  warnings.warn(Warnings.W011)
```