```
import pandas as pd
import re
import nltk

nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.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/Twitter_Data.csv")
df
```

```
{"type":"dataframe","variable_name":"df"}
```

```
tweets = df['clean_text'].dropna()
tweets.head()
```

```
0    when modi promised “minimum government maximum...
1    talk all the nonsense and continue all the dra...
2    what did just say vote for modi  welcome bjp t...
3    asking his supporters prefix chowkidar their n...
4    answer who among these the most powerful world...
Name: clean_text, dtype: object
```

```
def clean_tweet(text):
    text = re.sub(r'http\S+', '', text)     # remove URLs
    text = re.sub(r'@\w+', '', text)        # remove mentions
    text = re.sub(r'[^a-zA-Z\s]', '', text) # remove special
characters
    text = text.lower()                     # convert to lowercase
    return text

cleaned_tweets = tweets.apply(clean_tweet)
cleaned_tweets.head()
```

```
0    when modi promised minimum government maximum ...
1    talk all the nonsense and continue all the dra...
2    what did just say vote for modi  welcome bjp t...
3    asking his supporters prefix chowkidar their n...
4    answer who among these the most powerful world...
Name: clean_text, dtype: object
```

```
from nltk.tokenize import word_tokenize

tokenized_tweets = cleaned_tweets.apply(word_tokenize)
tokenized_tweets.head()
```

```
0    [when, modi, promised, minimum, government, ma...
1    [talk, all, the, nonsense, and, continue, all,...
2    [what, did, just, say, vote, for, modi, welcom...
3    [asking, his, supporters, prefix, chowkidar, t...
4    [answer, who, among, these, the, most, powerfu...
Name: clean_text, dtype: object
```

```
import nltk
nltk.download('averaged_perceptron_tagger_eng')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger_eng is already up-to-
[nltk_data]       date!
```

```
True
```

```
pos_tagged_tweets = tokenized_tweets.apply(nltk.pos_tag)
pos_tagged_tweets.head()
```

```
0    [(when, WRB), (modi, NN), (promised, VBD), (mi...
1    [(talk, NN), (all, PDT), (the, DT), (nonsense,...
2    [(what, WP), (did, VBD), (just, RB), (say, VB)...
3    [(asking, VBG), (his, PRP$), (supporters, NNS)...
4    [(answer, NN), (who, WP), (among, IN), (these,...
Name: clean_text, dtype: object
```

```
# See POS tags of one tweet clearly
pos_tagged_tweets.iloc[0]
```

```
[('when', 'WRB'),
 ('modi', 'NN'),
 ('promised', 'VBD'),
 ('minimum', 'JJ'),
 ('government', 'NN'),
 ('maximum', 'JJ'),
 ('governance', 'NN'),
 ('expected', 'VBD'),
 ('him', 'PRP'),
 ('begin', 'VB'),
 ('the', 'DT'),
 ('difficult', 'JJ'),
 ('job', 'NN'),
 ('reforming', 'VBG'),
 ('the', 'DT'),
 ('state', 'NN'),
 ('why', 'WRB'),
 ('does', 'VBZ'),
 ('take', 'VB'),
 ('years', 'NNS'),
 ('get', 'VB'),
```

```
 ('justice', 'NN'),
 ('state', 'NN'),
 ('should', 'MD'),
 ('and', 'CC'),
 ('not', 'RB'),
 ('business', 'NN'),
 ('and', 'CC'),
 ('should', 'MD'),
 ('exit', 'VB'),
 ('psus', 'NN'),
 ('and', 'CC'),
 ('temples', 'NNS')]
```

```
# Extract only POS tags from each tweet
pos_sequences = pos_tagged_tweets.apply(lambda x: [tag for word, tag
in x])
pos_sequences.head()
```

```
0    [WRB, NN, VBD, JJ, NN, JJ, NN, VBD, PRP, VB, D...
1    [NN, PDT, DT, NN, CC, VB, PDT, DT, NN, MD, VB,...
2    [WP, VBD, RB, VB, NN, IN, JJ, JJ, NN, VBD, PRP...
3    [VBG, PRP$, NNS, VBP, VBP, PRP$, NNS, RB, VBD,...
4    [NN, WP, IN, DT, DT, RBS, JJ, NN, NN, NN, VBP,...
Name: clean_text, dtype: object
```

```
from collections import Counter

transitions = Counter()

for tags in pos_sequences:
    for i in range(len(tags) - 1):
        transitions[(tags[i], tags[i+1])] += 1

# Show some transitions
transitions.most_common(10)
```

```
[(('NN', 'NN'), 323492),
 (('JJ', 'NN'), 211961),
 (('DT', 'NN'), 90948),
 (('NN', 'IN'), 84326),
 (('IN', 'NN'), 67206),
 (('NNS', 'VBP'), 63046),
 (('JJ', 'NNS'), 57513),
 (('NN', 'VBD'), 54182),
 (('NN', 'NNS'), 53731),
 (('NN', 'JJ'), 49954)]
```

```
from collections import Counter

words = []
```

```
for tweet in tokenized_tweets:
    words.extend(tweet)

word_freq = Counter(words)

# Rare words (appearing only once)
rare_words = [w for w, c in word_freq.items() if c == 1]

rare_words[:10]
```

```
['crustal',
 'maarkefir',
 'taxthe',
 'tuthukudi',
 'likly',
 'thuthukudi',
 'leadershipwho',
 'thiugh',
 'modiganga',
 'isits']
```

```
pos_tagged_tweets.iloc[1]
```

```
[('talk', 'NN'),
 ('all', 'PDT'),
 ('the', 'DT'),
 ('nonsense', 'NN'),
 ('and', 'CC'),
 ('continue', 'VB'),
 ('all', 'PDT'),
 ('the', 'DT'),
 ('drama', 'NN'),
 ('will', 'MD'),
 ('vote', 'VB'),
 ('for', 'IN'),
 ('modi', 'NN')]
```