Import libraries and modules for text analysis

```
from textblob import TextBlob
import pandas as pd
import matplotlib.pyplot as plt
import string
import nltk
from nltk.util import ngrams
from collections import Counter
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.probability import FreqDist
from wordcloud import WordCloud, STOPWORDS
```

Connect with postgresql

```
import pandas as pd
from sqlalchemy import create_engine
```

```
%load_ext sql
```

```
username = 'postgres'
password = '########'
hostname = 'localhost'
dbname = 'Target_AirFryer_Project'
```

```
connection_string = f'postgresql://{username}:{password}@{hostname}/{dbname}'
```

```
%sql $connection_string
```

Access to the review table

```
review_rating = pd.read_csv(r"C:\Users\Admin\Downloads\classified_review.csv")
```

```
review_rating
```

Analyze the sentiment of the reviews (postive, negative, neutral)

score > 0: positive

score < 0: negative

score = 0: neutral

```
score = []
for comment in review_rating['review']:
    blob = TextBlob(comment)
    polarity = blob.sentiment.polarity
    score.append(polarity)
```

```
review_rating['sentiment_score'] = score
```

```
review_rating
```

Create a wordcloud to see which are the most common words

We will focus on the non-promotion reviews first

```
no_promotion = review_rating[review_rating['promotion_review'] == 0]
```

```
no_promotion
```

```
combined_review_text = no_promotion['review'].str.cat(sep=' ')

# Set stopwords
my_stopwords = set(STOPWORDS)

# Generate the WordCloud
my_cloud = WordCloud(background_color='white', stopwords=my_stopwords).generate(combined_review_text)
```

```
# Create a figure of the generated wordcloud
plt.imshow(my_cloud,interpolation='bilinear')
plt.axis('off')

#Display the wordcloud
plt.show()
```

We can see that 'air', 'fryer', 'review', 'toaster','oven','part','promotion',"'s'","'ve'",'ca',"n't",'thing','got' are the common words that will not be usefull for analyze. We wil get rid of them and other stopwords next

Clean the text

```
def preprocess_text(text):

    # Tokenize the text

    tokens = word_tokenize(text.lower())


    # Remove stop words

    filtered_tokens = [token for token in tokens if token not in stopwords.words('english')]

    #Customized list of stop words:
    customized_stopwords = ['air', 'fryer', 'review', 'toaster','oven','part','promotion','s',"'ve'",'ca',"n't",'thing','got',"'"]
    removed_words = [word for word in filtered_tokens if word not in customized_stopwords]

    # Remove punctuation
    punctuation_list = set(string.punctuation)
    no_pun = [element for element in removed_words if element not in punctuation_list]

    # Lemmatize the tokens

    lemmatizer = WordNetLemmatizer()

    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in no_pun]


    # Join the tokens back into a string

    processed_text = ' '.join(lemmatized_tokens)

    return processed_text

# apply the function df

no_promotion['review']=no_promotion['review'].apply(preprocess_text)
no_promotion
```

Import it into the sql database to combine reviews for each product

```
 no_promotion.to_sql("true_review", connection_string)
```

```
%%sql
SELECT *
FROM true_review
```

```
%%sql
SELECT
    item_id,
    STRING_AGG(review, ' ' ORDER BY review) AS combined_review
FROM true_review_only
GROUP BY item_id;
```

```
output = %sql SELECT item_id, STRING_AGG(review, ' ' ORDER BY review) AS combined_review FROM true_review_only GROUP BY item_id
```

```
combined_review = pd.DataFrame(output)
```

```
combined_review
```

Get the most common pairs of words of each product

```
def get_most_common_bigrams(text):
    tokens = word_tokenize(text)
    bigrams = list(ngrams(tokens, 2))
    bigram_frequency = Counter(bigrams)
    most_common_bigrams = bigram_frequency.most_common(10)
    return most_common_bigrams

# Create a new DataFrame to store the most common bigrams for each product
most_common_bigrams_df = pd.DataFrame(columns=['item_id', 'bigram', 'count'])

# Iterate through each row (product) in the DataFrame
for index, row in combined_review.iterrows():
    item_id = row['item_id']
    combined_review_text = row['combined_review']
    most_common_bigrams = get_most_common_bigrams(combined_review_text)

    # Extract bigrams and counts into separate lists
    bigrams_list = [bigram for bigram, count in most_common_bigrams]
    counts_list = [count for bigram, count in most_common_bigrams]

    # Create a DataFrame for the current product
    product_df = pd.DataFrame({'item_id': item_id, 'bigram': bigrams_list, 'count': counts_list})

    # Append the DataFrame to the main DataFrame
    most_common_bigrams_df = pd.concat([most_common_bigrams_df, product_df], ignore_index=True)

# Display the result for bigrams
print(most_common_bigrams_df)
```

Import it the our database

```
most_common_bigrams_df.to_sql('common_pairs', connection_string)
```

```
%%sql
SELECT *
FROM common_pair
```

Drop the index column since we do not need it

```
%%sql
ALTER TABLE common_pair
DROP COLUMN index ;
```

```
%%sql
SELECT *
FROM common_pair
```