```
import pandas as pd # Import pandas for data manipulation
import re # Import re for regular expressions
import nltk # Import nltk for natural language processing
import matplotlib.pyplot as plt # Import matplotlib for plotting

from nltk.corpus import stopwords # Import stopwords from nltk
from nltk.tokenize import word_tokenize # Import word_tokenize from nltk for tokenization
from sklearn.feature_extraction.text import TfidfVectorizer # Import TfidfVectorizer for TF-IDF
from wordcloud import WordCloud # Import WordCloud for generating word clouds
```

```
df = pd.read_csv("/content/Tweets.csv") # Load the dataset from CSV into a pandas DataFrame
print(df.columns) # Print the column names to inspect them
df = df[['text', 'airline_sentiment']] # Select only the 'text' and 'airline_sentiment' columns
df.head() # Display the first 5 rows of the DataFrame
```

```
Index(['tweet_id', 'airline_sentiment', 'airline_sentiment_confidence',
       'negativereason', 'negativereason_confidence', 'airline',
       'airline_sentiment_gold', 'name', 'negativereason_gold',
       'retweet_count', 'text', 'tweet_coord', 'tweet_created',
       'tweet_location', 'user_timezone'],
      dtype='object')
```

| | text | airline_sentiment |
|---|---|---|
| 0 | @VirginAmerica What @dhepburn said. | neutral |
| 1 | @VirginAmerica plus you've added commercials t... | positive |
| 2 | @VirginAmerica I didn't today... Must mean I n... | neutral |
| 3 | @VirginAmerica it's really aggressive to blast... | negative |
| 4 | @VirginAmerica and it's a really big bad thing... | negative |

Next steps: Generate code with df | New interactive sheet

```
stop_words = set(stopwords.words('english')) # Define a set of English stop words

def clean_text(text):
    text = text.lower() # Convert text to lowercase
    text = re.sub(r"http\S+", "", text) # Remove URLs
    text = re.sub(r"@\w+", "", text) # Remove mentions
    text = re.sub(r"#\w+", "", text) # Remove hashtags
    text = re.sub(r"[^a-z\s]", "", text) # Remove non-alphabetic characters

    tokens = word_tokenize(text) # Tokenize the text into individual words
    tokens = [word for word in tokens if word not in stop_words] # Remove stop words

    return " ".join(tokens) # Join the cleaned tokens back into a string
```

```
df['clean_text'] = df['text'].apply(clean_text) # Apply the clean_text function to the 'text' co
df.head() # Display the first 5 rows with the new 'clean_text' column
```

| | text | airline_sentiment | clean_text |
|---|---|---|---|
| 0 | @VirginAmerica What @dhepburn said. | neutral | said |
| 1 | @VirginAmerica plus you've added commercials t... | positive | plus youve added commercials experience tacky |
| 2 | @VirginAmerica I didn't today... Must mean I n... | neutral | didnt today must mean need take another trip |
| 3 | @VirginAmerica it's really aggressive to blast... | negative | really aggressive blast obnoxious entertainmen... |
| 4 | @VirginAmerica and it's a really big bad thing... | negative | really big bad thing |

Next steps: Generate code with df New interactive sheet

```
negative_tweets = df[df['airline_sentiment'] == 'negative']['clean_text'] # Filter for tweets wi
```

```
tfidf = TfidfVectorizer(max_features=20) # Initialize TF-IDF Vectorizer to get top 20 features
tfidf_matrix = tfidf.fit_transform(negative_tweets) # Fit and transform the negative tweets into

tfidf_df = pd.DataFrame( # Create a DataFrame from the TF-IDF matrix
    tfidf_matrix.toarray(), # Convert sparse matrix to dense array
    columns=tfidf.get_feature_names_out() # Use feature names as column headers
)

tfidf_df.head() # Display the first 5 rows of the TF-IDF DataFrame
```

| | amp | call | cancelled | cant | customer | delayed | flight | flightled | get | help | hold | hour | hours |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 3 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 4 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 |

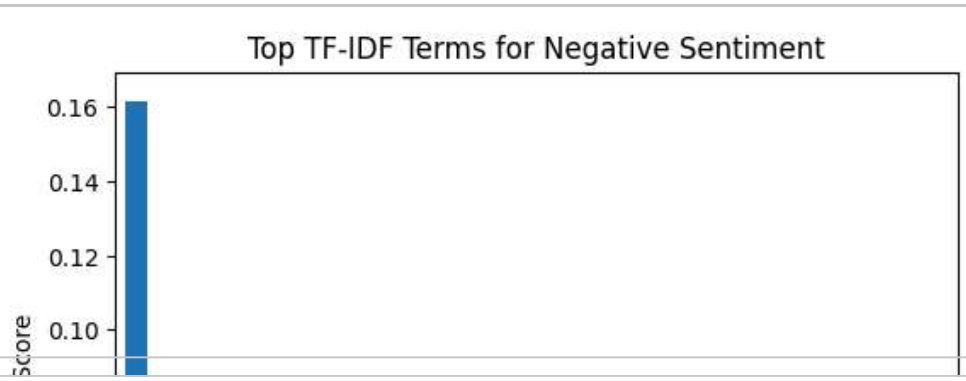
Next steps: Generate code with tfidf_df New interactive sheet

```
top_terms = tfidf_df.mean().sort_values(ascending=False) # Calculate the mean TF-IDF score for e
top_terms # Display the top terms and their scores
```

| | 0 |
|---|---|
| **flight** | 0.161364 |
| **get** | 0.068515 |
| **cancelled** | 0.053937 |
| **service** | 0.052331 |
| **time** | 0.044687 |
| **hours** | 0.044553 |
| **help** | 0.043687 |
| **im** | 0.042881 |
| **customer** | 0.042700 |
| **hold** | 0.040962 |
| **plane** | 0.039804 |
| **us** | 0.038649 |
| **still** | 0.037706 |
| **delayed** | 0.037145 |
| **cant** | 0.036345 |
| **one** | 0.034583 |
| **amp** | 0.034461 |
| **hour** | 0.033592 |
| **call** | 0.032677 |
| **flightled** | 0.027828 |

**dtype:** float64

```
top_terms.plot(kind='bar') # Create a bar plot of the top terms
plt.title("Top TF-IDF Terms for Negative Sentiment") # Set the title of the plot
plt.xlabel("Words") # Set the x-axis label
plt.ylabel("TF-IDF Score") # Set the y-axis label
plt.show() # Display the plot
```




```
wordcloud = WordCloud( # Initialize the WordCloud object
    background_color='white', # Set background color to white
    width=800, # Set width of the word cloud
    height=400 # Set height of the word cloud
).generate(" ".join(negative_tweets)) # Generate word cloud from combined negative tweets

plt.figure(figsize=(10,5)) # Create a figure for the word cloud
plt.imshow(wordcloud, interpolation='bilinear') # Display the generated image
plt.axis('off') # Turn off axis labels and ticks
plt.show() # Show the word cloud plot
```