```
pip install nltk pandas matplotlib seaborn
```

Show hidden output

```
# import the necessary library
import pandas as pd
import pandas as pd
import nltk
import seaborn as sns
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from collections import Counter

nltk.download('punkt_tab')
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
True
```

```
df = pd.read_csv("/content/customer_support_tickets.csv.zip")
```

```
df.head()
```

	Ticket ID	Customer Name	Customer Email	Customer Age	Customer Gender	Product Purchased	Date of Purchase	Ticket Type	Ticket Subject	
0	1	Marisa Obrien	carrollallison@example.com	32	Other	GoPro Hero	2021-03-22	Technical issue	Product setup	I'm {proc
1	2	Jessica Rios	clarkeashley@example.com	42	Female	LG Smart TV	2021-05-22	Technical issue	Peripheral compatibility	I'm {proc
2	3	Christopher Robbins	gonzalestracy@example.com	48	Other	Dell XPS	2020-07-14	Technical issue	Network problem	I'm f {proc
3	4	Christina Dillon	bradleyolson@example.org	27	Female	Microsoft Office	2020-11-13	Billing inquiry	Account access	I'm {proc
4	5	Alexander Carroll	bradleymark@example.com	67	Female	Autodesk AutoCAD	2020-02-04	Billing inquiry	Data loss	I'm {proc

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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8469 entries, 0 to 8468
Data columns (total 17 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Ticket ID                     8469 non-null   int64
 1   Customer Name                 8469 non-null   object
 2   Customer Email                8469 non-null   object
 3   Customer Age                  8469 non-null   int64
 4   Customer Gender               8469 non-null   object
 5   Product Purchased             8469 non-null   object
 6   Date of Purchase              8469 non-null   object
 7   Ticket Type                   8469 non-null   object
 8   Ticket Subject                8469 non-null   object
 9   Ticket Description            8469 non-null   object
 10  Ticket Status                 8469 non-null   object
 11  Resolution                    2769 non-null   object
 12  Ticket Priority               8469 non-null   object
 13  Ticket Channel                8469 non-null   object
 14  First Response Time           5650 non-null   object
 15  Time to Resolution            2769 non-null   object
 16  Customer Satisfaction Rating  2769 non-null   float64
dtypes: float64(1), int64(2), object(14)
memory usage: 1.1+ MB
```

```
# missing values
df.isnull()
```

	Ticket ID	Customer Name	Customer Email	Customer Age	Customer Gender	Product Purchased	Date of Purchase	Ticket Type	Ticket Subject	Ticket Description	Ticket Status	Re
0	False	False	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	
8464	False	False	False	False	False	False	False	False	False	False	False	
8465	False	False	False	False	False	False	False	False	False	False	False	
8466	False	False	False	False	False	False	False	False	False	False	False	
8467	False	False	False	False	False	False	False	False	False	False	False	
8468	False	False	False	False	False	False	False	False	False	False	False	

8469 rows × 17 columns

```
#handling missing values
df = df.assign(
    Resolution=df['Resolution'].fillna("Not Provided"),
    Time_to_Resolution=df['Time to Resolution'].fillna("Not Provided"),
    Customer_Satisfaction_Rating=df['Customer Satisfaction Rating'].fillna(-1),
    First_Response_Time=df['First Response Time'].fillna("Not Provided")
)
# assign() creates a new dataframe and assign the new cleaned column
```

```
# convert date to datetime data type
df['Date of Purchase'] = pd.to_datetime(df['Date of Purchase'], errors='coerce')
df['First Response Time'] = pd.to_datetime(df['First Response Time'], errors='coerce')
df['Time to Resolution'] = pd.to_datetime(df['Time to Resolution'], format="%Y-%m-%d %H:%M:%S", errors='coerce')
```

```
# formatting
df['Customer Gender'] = df['Customer Gender'].str.capitalize()
df['Ticket Description'] = df['Ticket Description'].str.replace(r'{.*?}', 'the product', regex=True)
```

```
# removing duplicates
df.drop_duplicates(inplace=True)
```

```
# save the cleaned dataset
df.to_csv("cleaned_customer_support_tickets.csv", index=False)
```

```
# load the cleaned dataset
df = pd.read_csv("cleaned_customer_support_tickets.csv")  # Load the cleaned dataset
print(df.head())  # Preview the data
print(df.info())  # Check for missing values & data types
```

```
3  I'm having an issue with the the product. Plea...
4  I'm having an issue with the the product. Plea...

                Ticket Status                                   Resolution  \
0  Pending Customer Response                                  Not Provided
1  Pending Customer Response                                  Not Provided
2                     Closed   Case maybe show recently my computer follow.
3                     Closed  Try capital clearly never color toward story.
4                     Closed                    West decision evidence bit.

  Ticket Priority Ticket Channel  First Response Time   Time to Resolution  \
0        Critical   Social media  2023-06-01 12:15:36                  NaN
1        Critical           Chat  2023-06-01 16:45:38                  NaN
2             Low   Social media  2023-06-01 11:14:38  2023-06-01 18:05:38
3             Low   Social media  2023-06-01 07:29:40  2023-06-01 01:57:40
4             Low          Email  2023-06-01 00:12:42  2023-06-01 19:53:42
```

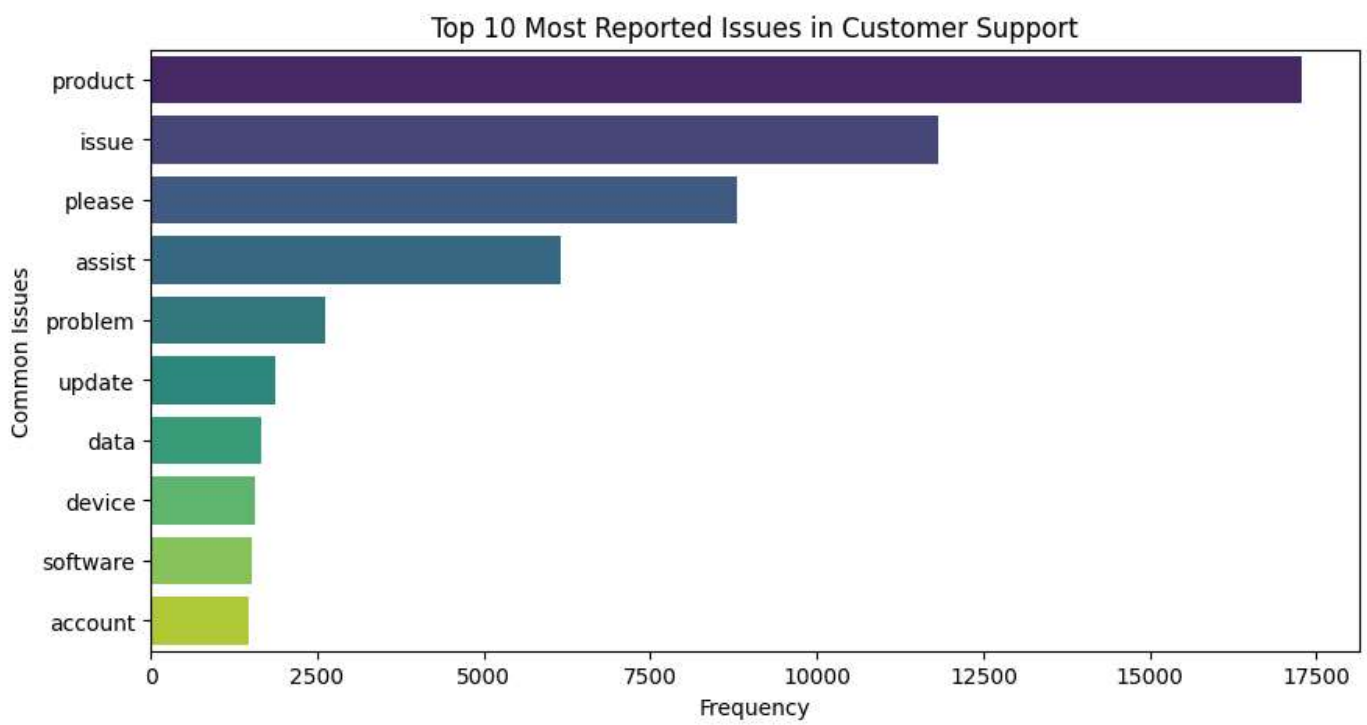
```
 6   Date of Purchase              8469 non-null   object
 7   Ticket Type                   8469 non-null   object
 8   Ticket Subject                8469 non-null   object
 9   Ticket Description            8469 non-null   object
 10  Ticket Status                 8469 non-null   object
 11  Resolution                    8469 non-null   object
 12  Ticket Priority               8469 non-null   object
 13  Ticket Channel                8469 non-null   object
 14  First Response Time           5650 non-null   object
 15  Time to Resolution            2769 non-null   object
 16  Customer Satisfaction Rating  2769 non-null   float64
 17  Time_to_Resolution            8469 non-null   object
 18  Customer_Satisfaction_Rating  8469 non-null   float64
 19  First_Response_Time           8469 non-null   object
dtypes: float64(2), int64(2), object(16)
memory usage: 1.3+ MB
None
```

```
# preprocess the data
def clean_text(text):
    if pd.isna(text):  # Handle missing values
        return ""

    text = text.lower()  # Convert to lowercase
    words = word_tokenize(text)  # Tokenization
    words = [word for word in words if word.isalpha()]  # Remove punctuation & numbers
    words = [word for word in words if word not in stopwords.words('english')]  # Remove stopwords

    lemmatizer = WordNetLemmatizer()
    words = [lemmatizer.lemmatize(word) for word in words]  # Lemmatization

    return " ".join(words)

df["Cleaned_Issue"] = df["Ticket Description"].apply(clean_text)
```

```
# finding most common issue
all_words = " ".join(df["Cleaned_Issue"]).split()  # Combine all text
common_issues = Counter(all_words).most_common(10)  # Find top 10 most common words
print(common_issues)
```

```
[('product', 17284), ('issue', 11829), ('please', 8810), ('assist', 6147), ('problem', 2609), ('update', 1865), ('
```

```
# frequent reported issues
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter

# Count the most common words
all_words = " ".join(df["Cleaned_Issue"]).split()
common_issues = Counter(all_words).most_common(10)  # Top 10

# Convert to DataFrame for visualization
common_df = pd.DataFrame(common_issues, columns=['Issue', 'Count'])

# Plot
plt.figure(figsize=(10,5))
sns.barplot(data=common_df, x="Count", y="Issue",hue="Issue", palette="viridis")
plt.xlabel("Frequency")
plt.ylabel("Common Issues")
plt.title("Top 10 Most Reported Issues in Customer Support")
plt.show()
```



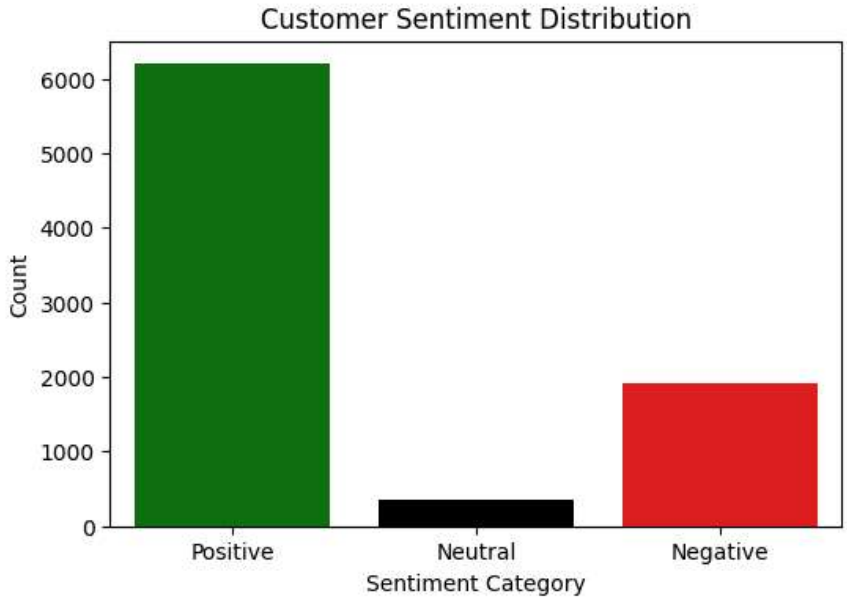

```
# perform a sentiment analysis
from nltk.sentiment import SentimentIntensityAnalyzer

nltk.download("vader_lexicon")
sia = SentimentIntensityAnalyzer()
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
```

```
# sentiment analysis on customer feedback
def get_sentiment(text):
    if pd.isna(text) or text.strip() == "":
        return "Neutral"  # Handle empty text

    score = sia.polarity_scores(text)["compound"]
    if score >= 0.05:
        return "Positive"
    elif score <= -0.05:
        return "Negative"
    else:
        return "Neutral"

df["Sentiment"] = df["Ticket Description"].apply(get_sentiment)
```

```
# sentiment visuals
plt.figure(figsize=(6,4))
sns.countplot(data=df, x="Sentiment", hue="Sentiment", palette=["green", "black", "red"])
plt.xlabel("Sentiment Category")
plt.ylabel("Count")
plt.title("Customer Sentiment Distribution")
plt.show()

# Calculate sentiment counts
sentiment_counts = df["Sentiment"].value_counts()
# Print percentage breakdown
sentiment_percentages = (sentiment_counts / sentiment_counts.sum()) * 100
```

```
print(sentiment_percentages)
```




```
Sentiment
Positive    73.149132
Negative    22.659110
Neutral      4.191758
Name: count, dtype: float64
```

```
# Convert 'Cleaned_Issue' to lowercase for consistency
df["Cleaned_Issue"] = df["Cleaned_Issue"].str.lower()

# Define top issues as lowercase
top_issues = ["product", "issue", "please", "assist", "problem", "update", "data", "device", "software", "account"]

# Filter rows where 'Cleaned_Issue' contains any top issue keyword
filtered_df = df[df["Cleaned_Issue"].apply(lambda x: any(word in x for word in top_issues))]

# Group and count sentiment distribution for filtered issues
issue_sentiment = filtered_df.groupby(["Cleaned_Issue", "Sentiment"]).size().unstack().fillna(0)

# Print result
print(issue_sentiment)
```

```
Sentiment                                          Negative  Neutral  \
Cleaned_Issue
accidentally deleted important data product way...       1.0      0.0
accidentally deleted important data product way...       0.0      0.0
accidentally deleted important data product way...       1.0      0.0
accidentally deleted important data product way...       1.0      0.0
accidentally deleted important data product way...       1.0      0.0
...                                                      ...      ...
unable access product account keep displaying c...       1.0      0.0
unable access product account keep displaying c...       1.0      0.0
unable access product account keep displaying c...       4.0      0.0
unable access product account keep displaying c...       0.0      0.0
unable access product account keep displaying c...       0.0      1.0

Sentiment                                          Positive
Cleaned_Issue
accidentally deleted important data product way...       0.0
accidentally deleted important data product way...       1.0
accidentally deleted important data product way...       0.0
accidentally deleted important data product way...       0.0
accidentally deleted important data product way...       0.0
```

```
...                                                       ...
unable access product account keep displaying c...       0.0
unable access product account keep displaying c...       0.0
unable access product account keep displaying c...       0.0
unable access product account keep displaying c...       1.0
unable access product account keep displaying c...       0.0

[7902 rows x 3 columns]
```