1. The dataset for Amazon Review Electronics Product is available at Amazon Reviews Dataset . Download the 5-core dataset for Electronics Category, under the heading of Small subset for experimentation. Read the file to a dataframe. Remember to keep the product metadata in a distinct dataframe as well.

```
# import pandas as pd
import gzip
import json

# def parse(path):
#     g = gzip.open(path, 'rb')
#     for l in g:
#         yield json.loads(l)

# def getDF(path):
#     df = {}
#     for i, d in enumerate(parse(path)):
#         df[i] = d
#         if (i + 1) % 1000000 == 0:  # Print progress every 1,000,000
records
#             print(f"Processed {i+1} records...")
#     print(f"Finished processing {i+1} records.")
#     return pd.DataFrame.from_dict(df, orient='index')

# df_reviews = getDF('D:\IR ASS 3\Electronics_5.json.gz')

# # Print the first few rows of the dataframe
# print(df_reviews.head())


import pandas as pd

# Adjust dtype argument according to your data types
def fast_getDF(path, chunksize=None, dtype=None):
    try:
        # Use read_json to directly read gzipped JSON file
        chunks = pd.read_json(path, lines=True, compression='gzip',
chunksize=chunksize, dtype=dtype)

        # Concatenate all chunks into a single DataFrame
        df_reviews = pd.concat(chunks, ignore_index=True)
        return df_reviews
    except Exception as e:
        print("Error reading review data:", e)

# Adjust file path accordingly
df_reviews = fast_getDF('D:\IR ASS 3\Electronics_5.json.gz',
chunksize=1000000)

# Print the first few rows of the dataframe
if df_reviews is not None:
```

```
    print(df_reviews.head())
else:
    print("DataFrame creation failed.")
```

```
   overall vote  verified   reviewTime      reviewerID        asin  \
0        5   67      True  09 18, 1999   AAP7PPBU72QFM  0151004714
1        3    5      True  10 23, 2013  A2E168DTVGE6SV  0151004714
2        5    4     False   09 2, 2008  A1ER5AYS3FQ9O3  0151004714
3        5   13     False   09 4, 2000  A1T17LMQABMBN5  0151004714
4        3    8      True   02 4, 2000  A3QHJ0FXK33OBE  0151004714

                             style      reviewerName  \
0       {'Format:': ' Hardcover'}      D. C. Carrad
1  {'Format:': ' Kindle Edition'}               Evy
2       {'Format:': ' Paperback'}             Kcorn
3       {'Format:': ' Hardcover'}   Caf Girl Writes
4       {'Format:': ' Hardcover'}  W. Shane Schmidt

                                          reviewText  \
0  This is the best novel I have read in 2 or 3 y...
1  Pages and pages of introspection, in the style...
2  This is the kind of novel to read when you hav...
3  What gorgeous language! What an incredible wri...
4  I was taken in by reviews that compared this b...

                                             summary  unixReviewTime
image
0                                    A star is born       937612800
NaN
1                     A stream of consciousness novel      1382486400
NaN
2  I'm a huge fan of the author and this one did ...      1220313600
NaN
3          The most beautiful book I have ever read!       968025600
NaN
4                         A dissenting view--In part.       949622400
NaN
```

```
# import pandas as pd
# import gzip
# import json

# def parse(path):
#     with gzip.open(path, 'rb') as g:
#         for l in g:
#             try:
#                 yield json.loads(l)
#             except json.JSONDecodeError:
#                 continue
```

```
# def getDF(path):
#     df = {}
#     for i, d in enumerate(parse(path)):
#         df[i] = d
#         if (i + 1) % 1000000 == 0:  # Print progress every 1,000,000
records
#             print(f"Processed {i+1} records...")
#     print(f"Finished processing {i+1} records.")
#     return pd.DataFrame.from_dict(df, orient='index')

# meta_df = getDF(r'D:\IR ASS 3\meta_Electronics.json.gz')  # Using
raw string

# # Print the first few rows of the dataframe
# print(meta_df.head())


# import pandas as pd

# # Adjust dtype argument according to your data types
# def fast_getDF(path, chunksize=None, dtype=None):
#     try:
#         # Use read_json to directly read gzipped JSON file
#         chunks = pd.read_json(path, lines=True, compression='gzip',
chunksize=chunksize, dtype=dtype)

#         # Concatenate all chunks into a single DataFrame
#         df_reviews = pd.concat(chunks, ignore_index=True)
#         return df_reviews
#     except Exception as e:
#         print("Error reading review data:", e)

# # Adjust file path accordingly
# df_reviews = fast_getDF('D:\IR ASS 3\Electronics_5.json.gz',
chunksize=1000000)

# # Print the first few rows of the dataframe
# if df_reviews is not None:
#     print(df_reviews.head())
# else:
#     print("DataFrame creation failed.")



import gzip
import json
import pandas as pd

def read_metadata_chunks(path, chunksize=1000000):
```

```
    try:
        with gzip.open(path, 'rt') as file:
            chunk = []
            for line in file:
                chunk.append(json.loads(line))
                if len(chunk) == chunksize:
                    yield pd.DataFrame(chunk)
                    chunk = []
            if chunk:
                yield pd.DataFrame(chunk)
    except Exception as e:
        print("Error reading metadata:", e)

# Adjust file path for metadata accordingly
metadata_path = 'D:\IR ASS 3\meta_Electronics.json.gz'

print("Attempting to read metadata in chunks...")

# Read metadata in chunks
metadata_chunks = read_metadata_chunks(metadata_path)

# Process each chunk iteratively
for i, chunk_df in enumerate(metadata_chunks):
    # Process the chunk here (e.g., perform necessary operations,
filtering, etc.)
    # Example: print the first few rows of each chunk
    print(f"Chunk {i+1}:")
    print(chunk_df.head())

print("Metadata reading complete.")
```

```
Attempting to read metadata in chunks...
Chunk 1:
                                              category tech1  \
0  [Electronics, Camera &amp; Photo, Video Survei...
1                  [Electronics, Camera &amp; Photo]
2  [Electronics, eBook Readers &amp; Accessories,...
3  [Electronics, eBook Readers & Accessories, eBo...
4  [Electronics, eBook Readers & Accessories, eBo...

                                         description fit  \
0  [The following camera brands and models have b...
1  [This second edition of the Handbook of Astron...
2  [A zesty tale. (Publishers Weekly)<br /><br />...
3                                                 []
4  [&#8220;sex.lies.murder.fame. is brillllli&#82...

                                               title  \
0  Genuine Geovision 1 Channel 3rd Party NVR IP S...
1  Books "Handbook of Astronomical Image Processi...
```

```
2                                        One Hot Summer
3  Hurray for Hattie Rabbit: Story and pictures (...
4                      sex.lies.murder.fame.: A Novel

                                 also_buy tech2  \
0                                       []
1                             [0999470906]
2                 [0425167798, 039914157X]
3  [0060219521, 0060219580, 0060219394]
4                                       []

                                           brand  \
0                                       GeoVision
1                                    33 Books Co.
2  Visit Amazon's Carolina Garcia Aguilera Page
3           Visit Amazon's Dick Gackenbach Page
4              Visit Amazon's Lolita Files Page

                                                  feature  \
0  [Genuine Geovision 1 Channel NVR IP Software, ...
1  [Detailed chapters cover these fundamental top...
2                                                 []
3                                                 []
4                                                 []

                                                    rank  \
0  [>#3,092 in Tools &amp;amp; Home Improvement &gt; ...
1  [>#55,933 in Camera &amp;amp; Photo (See Top 100 i...
2                                3,105,177 in Books (
3                                2,024,298 in Books (
4                                3,778,828 in Books (

                                also_view            main_cat
similar_item  \
0                                       []  Camera &amp;amp; Photo

1  [0943396670, 1138055360, 0999470906]  Camera &amp;amp; Photo

2                                       []               Books

3  [0060219521, 0060219475, 0060219394]               Books

4                                       []               Books


               date                                                price
\
0  January 28, 2014                                               $65.00

1     June 17, 2003
```

```
2                                                                  $11.49

3                    .a-section.a-spacing-mini{margin-bottom:6px!im...

4                                                                  $13.95


         asin                                                imageURL  \
0  0011300000  [https://images-na.ssl-images-amazon.com/image...
1  0043396828  [https://images-na.ssl-images-amazon.com/image...
2  0060009810                                                    []
3  0060219602                                                    []
4  0060786817                                                    []

                                      imageURLHighRes details
0  [https://images-na.ssl-images-amazon.com/image...     NaN
1  [https://images-na.ssl-images-amazon.com/image...     NaN
2                                                 []     NaN
3                                                 []     NaN
4                                                 []     NaN
Metadata reading complete.
```

```
merged_df = pd.merge(df_reviews, chunk_df, on='asin', how='inner')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call
last)
Cell In[1], line 1
----> 1 merged_df = pd.merge(df_reviews, chunk_df, on='asin',
how='inner')

NameError: name 'pd' is not defined
```

---

1. Choose a product of your choice. Let's say 'Headphones'.
2. Report the total number of rows for the product. Perform appropriate pre-processing as handling missing values, duplicates and other.

```
# Filter metadata for the chosen product ('Headphones')
headphones_metadata =
chunk_df[chunk_df['title'].str.contains('Headphones', case=False)]

# Print the first few rows of the filtered metadata dataframe
print(headphones_metadata.head())
```

```
                                            category tech1  \
8    [Electronics, Headphones, Earbud Headphones]
47                      [Electronics, Headphones]
132  [Electronics, Headphones, Earbud Headphones]
223  [Electronics, Headphones, Earbud Headphones]
229  [Electronics, Headphones, Earbud Headphones]

                                             description fit  \
8    [, <b>True High Definition Sound:</b><br>With ...
47   [Use these high quality headphones for interne...
132  [, <b>True High Definition Sound:</b><br>With ...
223  [, <b>True High Definition Sound:</b><br />Wit...
229  [, <b>True High Definition Sound:</b><br />Wit...

                                                  title also_buy tech2
\
8    Wireless Bluetooth Headphones Earbuds with Mic...       []

47   Polaroid Pbm2200 PC / Gaming Stereo Headphones...       []

132  Bluetooth Workout Headphones for Running and G...       []

223  Bluetooth Workout Headphones for Running and G...       []

229  Bluetooth Workout Headphones for Running and G...       []


                brand
feature  \
8    Enter The Arena  [Superb Sound Quality: Plays crystal clear
aud...
47          Polaroid  [Ideal for PC Internet chatting, PC /
Console ...
132  Enter The Arena  [Superb Sound Quality: Plays crystal clear
aud...
223  Enter The Arena  [Superb Sound Quality: Plays crystal clear
aud...
229  Enter The Arena  [Superb Sound Quality: Plays crystal clear
aud...

                                                    rank also_view  \
8    [>#950 in Cell Phones & Accessories (See Top 1...        []
47   [>#3,548,269 in Cell Phones &amp;amp; Accessories ...        []
132  [>#4,626,934 in Cell Phones & Accessories (See...        []
223  [>#2,654,020 in Cell Phones &amp;amp; Accessories ...        []
229  [>#5,289,289 in Cell Phones &amp;amp; Accessories ...        []

                     main_cat similar_item                date
price  \
8        Home Audio & Theater                October 23, 2017  $7.99
```

```
47            All Electronics                December 13, 2012

132      Home Audio & Theater                December 28, 2015

223  Home Audio &amp; Theater                 October 18, 2015

229  Home Audio &amp; Theater                   April 26, 2013


           asin                                              imageURL  \
8    0132492776  [https://images-na.ssl-images-amazon.com/image...
47   0558835155  [https://images-na.ssl-images-amazon.com/image...
132  0692206280  [https://images-na.ssl-images-amazon.com/image...
223  0983629269  [https://images-na.ssl-images-amazon.com/image...
229  0985262788  [https://images-na.ssl-images-amazon.com/image...

                                        imageURLHighRes details
8    [https://images-na.ssl-images-amazon.com/image...     NaN
47   [https://images-na.ssl-images-amazon.com/image...     NaN
132  [https://images-na.ssl-images-amazon.com/image...     NaN
223  [https://images-na.ssl-images-amazon.com/image...     NaN
229  [https://images-na.ssl-images-amazon.com/image...     NaN
```

```
# Report the total number of rows for 'Headphones'
total_rows_headphones = len(headphones_metadata)
print("Total number of rows for 'Headphones':", total_rows_headphones)


Total number of rows for 'Headphones': 18210

# Handle missing values
headphones_metadata = headphones_metadata.dropna()

# Calculate the total number of rows after handling missing values
total_rows_after_preprocessing = len(headphones_metadata)
print("Total number of rows after handling missing values:",
total_rows_after_preprocessing)

Total number of rows after handling missing values: 18196
```

you should use the review data rather than the metadata. The review data contains information about individual reviews, including the rating score, which is essential for calculating statistics such as the average rating score, number of good ratings, number of bad ratings, and number of reviews corresponding to each rating.

1. Obtain the Descriptive Statistics of the product as : -

a. Number of Reviews.

b. Average Rating Score.

c. Number of Unique Products.

d. Number of Good Rating.

e. Number of Bad Ratings (Set a threshold of >=3 as 'Good' and rest as 'Bad'), and

f. Number of Reviews corresponding to each Rating.

```
# a. Number of Reviews
num_reviews = len(df_reviews)
# b. Average Rating Score
avg_rating = df_reviews['overall'].mean()

# c. Number of Unique Products
num_unique_products = df_reviews['asin'].nunique()


# d. Number of Good Ratings
num_good_ratings = df_reviews[df_reviews['overall'] >= 3]
['overall'].count()

# e. Number of Bad Ratings
num_bad_ratings = df_reviews[df_reviews['overall'] < 3]
['overall'].count()


# f. Number of Reviews corresponding to each Rating
rating_counts = df_reviews['overall'].value_counts().sort_index()



# Displaying the descriptive statistics
print("Descriptive Statistics for Headphones:")
print("a. Number of Reviews:", num_reviews)
print("b. Average Rating Score:", avg_rating)
print("c. Number of Unique Products:", num_unique_products)
print("d. Number of Good Ratings:", num_good_ratings)
print("e. Number of Bad Ratings:", num_bad_ratings)
print("f. Number of Reviews corresponding to each Rating:")
print(rating_counts)
```

```
Descriptive Statistics for Headphones:
a. Number of Reviews: 6739590
b. Average Rating Score: 4.26766835964799
c. Number of Unique Products: 160052
d. Number of Good Ratings: 5965756
e. Number of Bad Ratings: 773834
f. Number of Reviews corresponding to each Rating:
overall
1.0     467158
```

```
2.0     306676
3.0     504781
4.0    1137393
5.0    4323582
Name: count, dtype: int64
```

1. Preprocess the Text

a. Removing the HTML Tags.

b. Removing accented characters.

c. Expanding Acronyms.

d. Removing Special Characters

e. Lemmatization

f. Text Normalizer

```
import re
import unicodedata
import nltk
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer

nltk.download('punkt')
nltk.download('wordnet')


print("Before preprocessing:")
print(df_reviews['reviewText'].head())

def remove_html_tags(text):
    if pd.isnull(text):
        return ""  # Return empty string for missing values
    else:
        clean = re.compile('<.*?>')
        return re.sub(clean, '', str(text))

# Function to remove accented characters
def remove_accented_chars(text):
    text = unicodedata.normalize('NFKD', text).encode('ascii',
'ignore').decode('utf-8', 'ignore')
    return text

# Dictionary of common acronyms to expand
acronyms_dict = {
    "lol": "laugh out loud",
    "brb": "be right back",
    # Add more acronyms as needed
```

```
}

# Function to expand acronyms
def expand_acronyms(text, acronyms_dict):
    for acronym, expanded in acronyms_dict.items():
        text = re.sub(r'\b' + re.escape(acronym) + r'\b', expanded,
text)
    return text

# Function to remove special characters
def remove_special_characters(text):
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    return text

# Lemmatization
lemmatizer = WordNetLemmatizer()

def lemmatize_text(text):
    tokens = nltk.word_tokenize(text)
    lemmatized_text = ' '.join([lemmatizer.lemmatize(word) for word in
tokens])
    return lemmatized_text

def normalize_text(text):
    if isinstance(text, str):
        text = remove_html_tags(text)
        text = remove_accented_chars(text)
        text = expand_acronyms(text, acronyms_dict)
        text = remove_special_characters(text)
        text = lemmatize_text(text)
        return text
    else:
        return ""  # Return empty string for missing values

# Apply text normalization to a column of the dataframe
df_reviews['processed_reviewText'] =
df_reviews['reviewText'].apply(normalize_text)



# Print 'processed_reviewText' column after preprocessing
print("\nAfter preprocessing:")
print(df_reviews['processed_reviewText'].head())


df_reviews.to_pickle('D:\IR ASS 3\preprocessed_reviews.pkl')

[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Dell\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Dell\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!

Before preprocessing:
0    This is the best novel I have read in 2 or 3 y...
1    Pages and pages of introspection, in the style...
2    This is the kind of novel to read when you hav...
3    What gorgeous language! What an incredible wri...
4    I was taken in by reviews that compared this b...
Name: reviewText, dtype: object

After preprocessing:
0    This is the best novel I have read in or year ...
1    Pages and page of introspection in the style o...
2    This is the kind of novel to read when you hav...
3    What gorgeous language What an incredible writ...
4    I wa taken in by review that compared this boo...
Name: processed_reviewText, dtype: object

The Kernel crashed while executing code in the current cell or a
previous cell.

Please review the code in the cell(s) to identify a possible cause of
the failure.

Click <a href='https://aka.ms/vscodeJupyterKernelCrash'>here</a> for
more info.

View Jupyter <a href='command:jupyter.viewOutput'>log</a> for further
details.
```

1. To extract relevant statistics, perform the following EDA -

a. Top 20 most reviewed brands in the category that you have chosen.

b. Top 20 least reviewed brands in the category you have chosen.

c. Which is the most positively reviewed 'Headphone' (Or for any other electronic product you have selected)

d. Show the count of ratings for the product over 5 consecutive years.

e. Form a Word Cloud for 'Good' and 'Bad' ratings. Report the most commonly used words for positive and negative reviews by observing the good and bad word clouds.

f. Plot a pie chart for Distribution of Ratings vs. the No. of Reviews.

g. Report in which year the product got maximum reviews.

h. Which year has the highest number of Customers?

```
pip install wordcloud
```

```
Requirement already satisfied: wordcloud in c:\users\dell\appdata\
local\programs\python\python310\lib\site-packages (1.9.3)
Requirement already satisfied: numpy>=1.6.1 in c:\users\dell\appdata\
local\programs\python\python310\lib\site-packages (from wordcloud)
(1.26.4)
Requirement already satisfied: pillow in c:\users\dell\appdata\
roaming\python\python310\site-packages (from wordcloud) (9.2.0)
Requirement already satisfied: matplotlib in c:\users\dell\appdata\
roaming\python\python310\site-packages (from wordcloud) (3.5.2)
Requirement already satisfied: cycler>=0.10 in c:\users\dell\appdata\
roaming\python\python310\site-packages (from matplotlib->wordcloud)
(0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\dell\
appdata\roaming\python\python310\site-packages (from matplotlib-
>wordcloud) (4.34.4)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\dell\
appdata\roaming\python\python310\site-packages (from matplotlib-
>wordcloud) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\dell\
appdata\roaming\python\python310\site-packages (from matplotlib-
>wordcloud) (21.3)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\dell\
appdata\roaming\python\python310\site-packages (from matplotlib-
>wordcloud) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\dell\
appdata\roaming\python\python310\site-packages (from matplotlib-
>wordcloud) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\dell\appdata\
roaming\python\python310\site-packages (from python-dateutil>=2.7-
>matplotlib->wordcloud) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

```
# # Split 'asin' values to extract brand information
# df_reviews['brand'] = df_reviews['asin'].str.split('-').str[0]

# # Calculate the top 20 most reviewed brands
# top_20_most_reviewed_brands =
df_reviews['brand'].value_counts().head(20)
# print("Top 20 most reviewed brands:")
# print(top_20_most_reviewed_brands)

# Filter out rows with empty brand names
meta_df_filtered = meta_df[meta_df['brand'] != '']

# Calculate the top 20 most reviewed brands
top_20_most_reviewed_brands =
meta_df_filtered['brand'].value_counts().head(20)
```

```
print("Top 20 most reviewed brands:")
print(top_20_most_reviewed_brands)
```

```
Top 20 most reviewed brands:
brand
Sony          12310
Generic       11524
Dell           7586
HP             7559
Samsung        6728
Canon          4727
Panasonic      4384
Asus           3786
Neewer         3783
uxcell         3716
Toshiba        3688
Belkin         3680
Nikon          3586
Fintie         3396
Live2Pedal     3070
Lenovo         3056
Acer           3039
Philips        2860
UPBRIGHT       2742
SanDisk        2600
Name: count, dtype: int64
```

```
# b. Top 20 least reviewed brands
top_20_least_reviewed_brands =
meta_df_filtered['brand'].value_counts().tail(20)
print("\nTop 20 least reviewed brands:")
print(top_20_least_reviewed_brands)
```

```
Top 20 least reviewed brands:
brand
Natural LIfe              1
Fisher Scientific         1
Duke                      1
imito QX                  1
PMID1000 (PMID1000D)      1
Main Street 24/7          1
kidsafe                   1
Kidsafe                   1
Kidsafe Case              1
efoglobal                 1
HCO                       1
Spring Rose               1
HANYA CASE                1
Reborn                    1
```

```
ComfortVu                1
Novastone                1
RALLY                    1
INNOVATIVE SOFTWARE      1
GODIRECT                 1
COOLBOY                  1
Name: count, dtype: int64
```

```
# c. Most positively reviewed headphone (or any other electronic
product)
# most_positively_reviewed_product = meta_df[meta_df['overall'] == 5]
['title'].value_counts().idxmax()
# print("\nMost positively reviewed product:",
most_positively_reviewed_product)

# c. Most positively reviewed headphone (or any other electronic
product)
most_positively_reviewed_product = df_reviews[df_reviews['overall'] ==
5]['asin'].value_counts().idxmax()
print("\nMost positively reviewed product:",
most_positively_reviewed_product)


# # # c. Most positively reviewed 'Headphone'
# most_positively_reviewed = df_reviews[df_reviews['overall'] == 5.0]
['asin'].value_counts().idxmax()
# print("\nMost positively reviewed 'Headphone':",
most_positively_reviewed)
```

```
Most positively reviewed product: B003L1ZYYW
```

```
# # Convert 'reviewTime' to datetime and extract year
df_reviews['reviewYear'] =
pd.to_datetime(df_reviews['reviewTime']).dt.year

# Group by reviewYear and count the number of ratings
ratings_count_per_year = df_reviews.groupby('reviewYear')
['overall'].count()

# # Display the count of ratings for each year
# print("Count of ratings for the product over 5 consecutive years:")
# print(ratings_count_per_year)


years_of_interest = range(2014, 2019)
ratings_count_5_years = ratings_count_per_year.loc[years_of_interest]

# Display the count of ratings for the product over the 5 consecutive
years
print("Count of ratings for the product over 5 consecutive years (2014
```

```
to 2018):")
print(ratings_count_5_years)
```

```
Count of ratings for the product over 5 consecutive years (2014 to
2018):
reviewYear
2014    1034835
2015    1441804
2016    1426496
2017     912915
2018     377430
Name: overall, dtype: int64
```

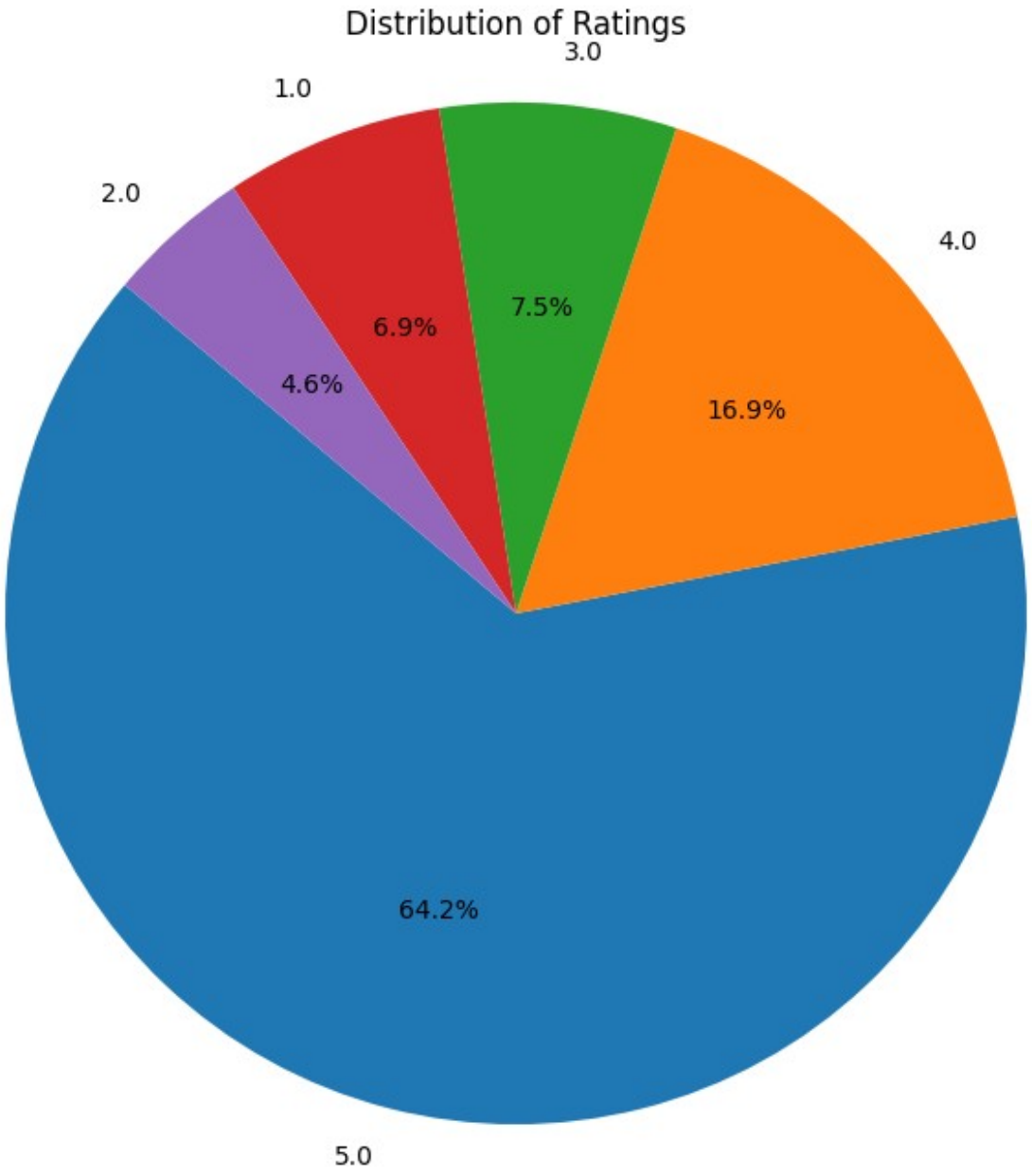
#----------

```
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt

# Function to generate word cloud from text
def generate_wordcloud(text, title):
    # Remove common words (stopwords) from the text
    stopwords = set(STOPWORDS)
    wordcloud = WordCloud(width=800, height=400,
background_color='white', stopwords=stopwords).generate(text)

    # Plot word cloud
    plt.figure(figsize=(10, 6))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.title(title)
    plt.axis('off')
    plt.show()

# Sample a portion of the good and bad reviews text for the entire
category
# Sample a portion of the good and bad reviews text for the entire
category
sample_size = 100000  # Adjust the sample size as needed
good_reviews_text_sample = ' '.join(df_reviews[df_reviews['overall']
>= 3]['reviewText'].astype(str).sample(sample_size))
bad_reviews_text_sample = ' '.join(df_reviews[df_reviews['overall'] <
3]['reviewText'].astype(str).sample(sample_size))

# Generate and plot word clouds
generate_wordcloud(good_reviews_text_sample, "Word Cloud for Good
Reviews (Entire Category)")
generate_wordcloud(bad_reviews_text_sample, "Word Cloud for Bad
Reviews (Entire Category)")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call
last)
Cell In[3], line 18
     15     plt.show()
     17 # Check the size of the DataFrame
---> 18 df_size_good =
len(Headphone_merged_df[Headphone_merged_df['overall'] >= 3])
     19 df_size_bad =
len(Headphone_merged_df[Headphone_merged_df['overall'] < 3])
     21 # Set the sample size to the minimum of df_size and 100000

NameError: name 'Headphone_merged_df' is not defined
```

```python
# from wordcloud import STOPWORDS

# # Function to generate word cloud from text
# def generate_wordcloud(text, title):
#     # Remove common words (stopwords) from the text
#     stopwords = set(STOPWORDS)
#     wordcloud = WordCloud(width=800, height=400,
background_color='white', stopwords=stopwords).generate(text)

#     # Plot word cloud
#     plt.figure(figsize=(10, 6))
#     plt.imshow(wordcloud, interpolation='bilinear')
#     plt.title(title)
#     plt.axis('off')
#     plt.show()

# # Sample a portion of the good and bad reviews text
# sample_size = 100000  # Adjust the sample size as needed
# good_reviews_text_sample = ' '.join(df_reviews[df_reviews['overall']
>= 3]['reviewText'].astype(str).sample(sample_size))
# bad_reviews_text_sample = ' '.join(df_reviews[df_reviews['overall']
< 3]['reviewText'].astype(str).sample(sample_size))

# # Generate and plot word clouds
# generate_wordcloud(good_reviews_text_sample, "Word Cloud for Good
Reviews")
# generate_wordcloud(bad_reviews_text_sample, "Word Cloud for Bad
Reviews")

# Calculate the distribution of ratings
ratings_distribution = df_reviews['overall'].value_counts()
# Plot a pie chart for the distribution of ratings
plt.figure(figsize=(8, 8))
plt.pie(ratings_distribution, labels=ratings_distribution.index,
autopct='%1.1f%%', startangle=140)
```

```
plt.title('Distribution of Ratings')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle
plt.show()
```




```
# # Calculate the count of reviews for each rating
# rating_counts = df_reviews['overall'].value_counts()

# # Plot a pie chart
# plt.figure(figsize=(8, 8))
```

```
# plt.pie(rating_counts, labels=rating_counts.index, autopct='%1.1f%
%', startangle=140)
# plt.title('Distribution of Ratings')
# plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as
a circle.
# plt.show()

# Report in which year the product got maximum reviews
max_reviews_year = df_reviews['reviewYear'].value_counts().idxmax()
print("Year with the maximum reviews:", max_reviews_year)

# Which year has the highest number of customers?
year_with_highest_customers = df_reviews.groupby('reviewYear')
['reviewerID'].nunique().idxmax()
print("Year with the highest number of customers:",
year_with_highest_customers)
```

```
---------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In[2], line 2
      1 # Report in which year the product got maximum reviews
----> 2 max_reviews_year =
df_reviews['reviewYear'].value_counts().idxmax()
      3 print("Year with the maximum reviews:", max_reviews_year)
      5 # Which year has the highest number of customers?

NameError: name 'df_reviews' is not defined
```

1. Use a relevant feature engineering technique to model review text as Bag of Words model, TF-IDF, Hashing Vectorizer or Word2Vec

```
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\dell\appdata\
local\programs\python\python310\lib\site-packages (1.4.1.post1)
Requirement already satisfied: numpy<2.0,>=1.19.5 in c:\users\dell\
appdata\local\programs\python\python310\lib\site-packages (from
scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in c:\users\dell\appdata\
local\programs\python\python310\lib\site-packages (from scikit-learn)
(1.12.0)
Requirement already satisfied: joblib>=1.2.0 in c:\users\dell\appdata\
local\programs\python\python310\lib\site-packages (from scikit-learn)
(1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\dell\
appdata\local\programs\python\python310\lib\site-packages (from
scikit-learn) (3.4.0)
Note: you may need to restart the kernel to use updated packages.
```

```
from sklearn.feature_extraction.text import CountVectorizer

# Define a generator function to yield documents one by one
def document_generator(df):
    for document in df:
        yield document

# Initialize CountVectorizer with max_features parameter
count_vectorizer = CountVectorizer(max_features=500)  # Adjust the
value as needed

# Fit and transform the processed text data using the generator
bow_matrix =
count_vectorizer.fit_transform(document_generator(df_reviews['processe
d_reviewText']))

# Print the shape of the Bag of Words matrix
print("Shape of Bag of Words matrix:", bow_matrix.shape)
```

```
---------------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In[1], line 12
      9 count_vectorizer = CountVectorizer(max_features=500)  # Adjust
the value as needed
     11 # Fit and transform the processed text data using the
generator
---> 12 bow_matrix =
count_vectorizer.fit_transform(document_generator(df_reviews['processe
d_reviewText']))
     14 # Print the shape of the Bag of Words matrix
     15 print("Shape of Bag of Words matrix:", bow_matrix.shape)

NameError: name 'df_reviews' is not defined
```

```
# from sklearn.feature_extraction.text import TfidfVectorizer
# # Fill missing values in 'reviewText' column with an empty string
# # Sample a portion of the data for feature extraction (adjust the
sample size as needed)
# sample_size = 5000  # Adjust the sample size as needed
# df_sample = df_reviews.sample(sample_size, random_state=42)

# # Fill missing values in 'reviewText' column with an empty string
# df_sample['reviewText'].fillna('', inplace=True)

# # Initialize TF-IDF vectorizer with reduced max_features
# tfidf_vectorizer = TfidfVectorizer(max_features=500,  # Adjust the
number of features as needed
#                                    stop_words='english',  # Remove
```

```
common English stopwords
#                                     ngram_range=(1, 2),   # Include
unigrams and bigrams
#                                     max_df=0.9,           # Ignore
terms with a document frequency higher than 90%
#                                     min_df=5)            # Ignore
terms with a document frequency lower than 5

# # Fit and transform the review text data
# tfidf_features =
tfidf_vectorizer.fit_transform(df_sample['reviewText'])

# # Check the shape of the resulting TF-IDF matrix
# print("TF-IDF Matrix Shape:", tfidf_features.shape)
```

```
TF-IDF Matrix Shape: (5000, 500)
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Preprocess the review text
# Assuming you have a dataframe 'df_reviews' containing the review
text column named 'reviewText'

# Fill missing values with an empty string

# Initialize TfidfVectorizer with reduced max_features
max_features = 1000  # Adjust the number of features as needed
tfidf_vectorizer = TfidfVectorizer(max_features=max_features)

# Fit and transform the review text to TF-IDF vectors
tfidf_vectors =
tfidf_vectorizer.fit_transform(df_reviews['reviewText'])

# Convert TF-IDF vectors to a dense array
tfidf_vectors_dense = tfidf_vectors.toarray()

# Now tfidf_vectors_dense contains the TF-IDF representation of the
review text
# You can use this representation for further modeling, such as
classification or clustering
```