SENTIMENT ANALYSIS ON FASHION NOVA

FASHION NOVA IS A PROMINENT FAST FASHION RETAILER THAT HAS MADE WAVES WITH ITS TRENDY, AFFORDABLE, AND HIGH-QUALITY CLOTHING. KNOWN FOR ITS BOLD AND STYLISH DESIGNS, THE BRAND CATERS PRIMARILY TO YOUNG ADULTS WHO ARE FASHION-CONSCIOUS AND BUDGET-SAVVY.

DATA OVERVIEW

- REVIEWER NAME: NAME OF THE PERSON WHO WROTE THE REVIEW.
- PROFILE LINK: URL TO THE REVIEWER'S PROFILE.
- COUNTRY: COUNTRY WHERE THE REVIEWER IS LOCATED.
- REVIEW COUNT: TOTAL NUMBER OF REVIEWS WRITTEN BY THE REVIEWER.
- REVIEW DATE: DATE WHEN THE REVIEW WAS POSTED.
- RATING: SCORE GIVEN BY THE REVIEWER, OFTEN ON A SCALE OF 1 TO 5.
- REVIEW TITLE: TITLE OR HEADLINE OF THE REVIEW.
- REVIEW TEXT: DETAILED CONTENT OF THE REVIEW.
- DATE OF EXPERIENCE: DATE WHEN THE REVIEWER USED THE PRODUCT OR SERVICE.

IMPORT ALL THE NECESSARY LIBRARIES

[61]

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from nltk.stem.porter import PorterStemmer
nltk.download('stopwords')
from nltk.corpus import stopwords
STOPWORDS = set(stopwords.words('english'))

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.feature_extraction.text import CountVectorizer
from wordcloud import WordCloud
import pickle
import re
import warnings
warnings.filterwarnings('ignore')
```

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

The dataset includes reviewer information, including their name, profile link, and country of residence, along with metrics such as the number of reviews written, review dates, ratings, review titles, detailed review texts, and the dates of their experiences.

```
[4] df= pd.read_csv('fashionnova_reviews.csv')
```

```
df.head()
```

	Reviewer Name	Profile Link	Country	Review Count	Review Date	Rating	Review Title	Review Text	Date of Experience
0	Champagne	/users/66c78240087b6269ffbcb5fb	US	1 review	2024-08-22T20:24:02.000Z	Rated 5 out of 5 stars	I love ordering from fashion nova	I love ordering from fashion nova. The clothes...	August 22, 2024
1	Vg customer	/users/6618fdb53d4198001210cbe7	VG	3 reviews	2024-08-21T05:43:11.000Z	Rated 5 out of 5 stars	Top tier content for fashion nova	Always amazing clothes and the fast shipping i...	August 18, 2024
2	Colleen Burgher	/users/64e9595206be1a001244ff73	US	3 reviews	2024-08-21T17:09:14.000Z	Rated 5 out of 5 stars	Prices and quality of products are...	Prices and quality of products are GREAT Would...	August 21, 2024
3	R.G.M	/users/66c58ad1c6ab36352a08f57a	US	1 review	2024-08-21T08:36:03.000Z	Rated 5 out of 5 stars	Great customer service	Great customer service. I was helped until the...	August 20, 2024
4	Rosalyn Cousar	/users/60ad4b6ef3788e001adbb8e3	US	5 reviews	2024-08-22T00:46:16.000Z	Rated 3 out of 5 stars	False advertising	Disappointing experience. You don't live up to...	August 21, 2024

CHECKING THE NULL VALUES FROM THE DATA

Reviewer name , country and Review Title contain Null values

```
[6] df.columns
```

```
Index(['Reviewer Name', 'Profile Link', 'Country', 'Review Count',
       'Review Date', 'Rating', 'Review Title', 'Review Text',
       'Date of Experience'],
      dtype='object')
```

```
df.isnull().sum()
```

	0
Reviewer Name	63
Profile Link	0
Country	4
Review Count	0
Review Date	0
Rating	0
Review Title	48
Review Text	0
Date of Experience	0

dtype: int64

- This dataset consist of 131880 rows
- Removing and Checking the null values from the data

```
[8] df.shape
```

```
(131980, 9)
```

```
[9] df[df['Review Title'].isnull()]
    df=df.dropna()
```

```
df.isnull().sum()
```

	0
Reviewer Name	0
Profile Link	0
Country	0
Review Count	0
Review Date	0
Rating	0
Review Title	0
Review Text	0
Date of Experience	0

dtype: int64

- DROPPING THE IRRELEVANT COLUMNS ACCORDING TO OUR PRBLEM STATEMENT

```
[11] df.drop(columns=['Reviewer Name','Profile Link'] , axis=0,inplace=True)
     df.drop(columns=['Review Date'] , axis=0,inplace=True)
```

```
[12] df.columns
```

```
Index(['Country', 'Review Count', 'Rating', 'Review Title', 'Review Text',
       'Date of Experience'],
      dtype='object')
```

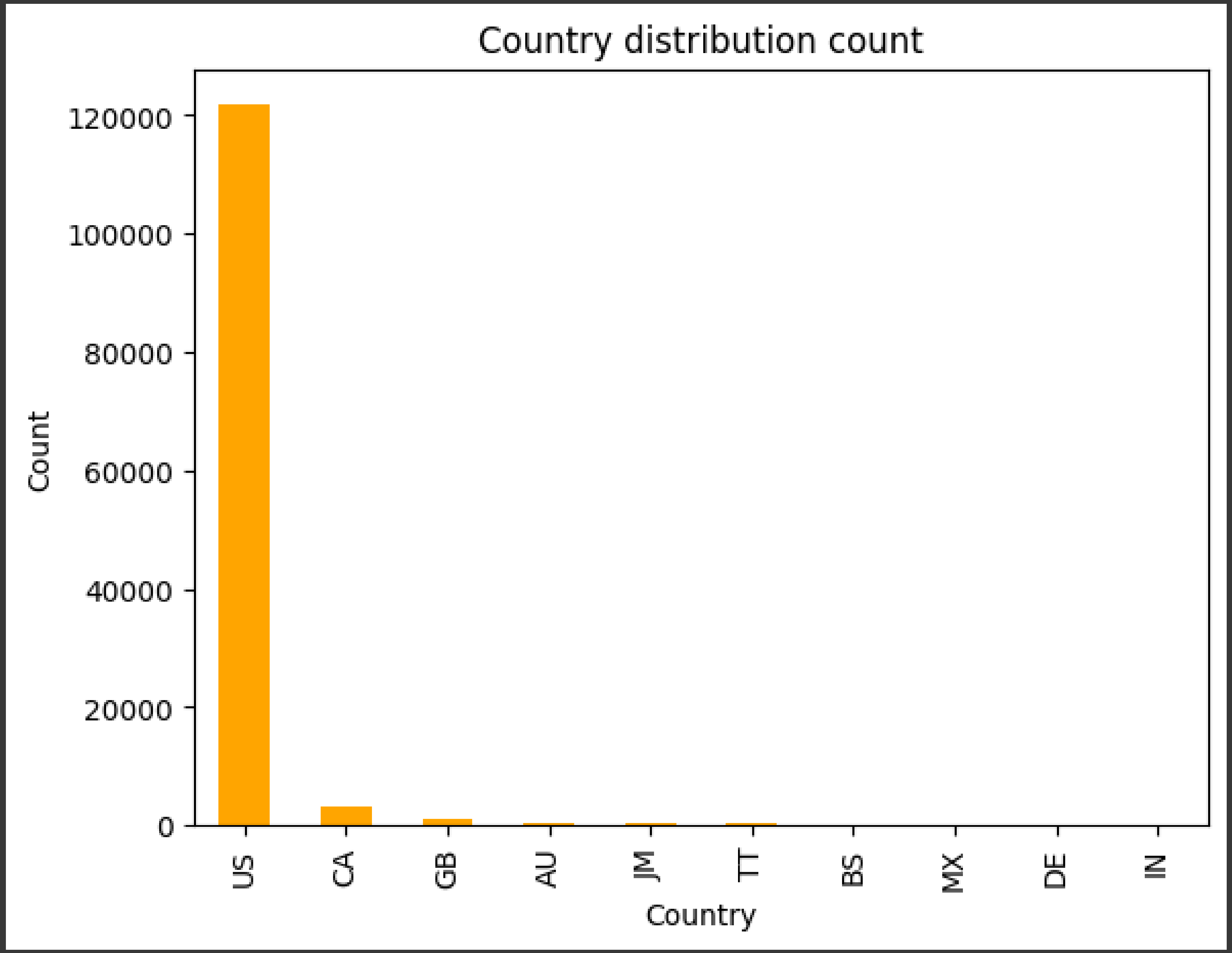
START PERFORMING EDA

```
data = df['Country'].value_counts()
```

```
[14] data = data.head(10)
```

```
0s
#Bar plot to visualize the total counts of each rating
data.plot.bar(color = '#FFA500')
plt.title('Country distribution count')
plt.xlabel('Country')
plt.ylabel('Count')
plt.show()
```

The majority of shoppers are U.S. citizens.

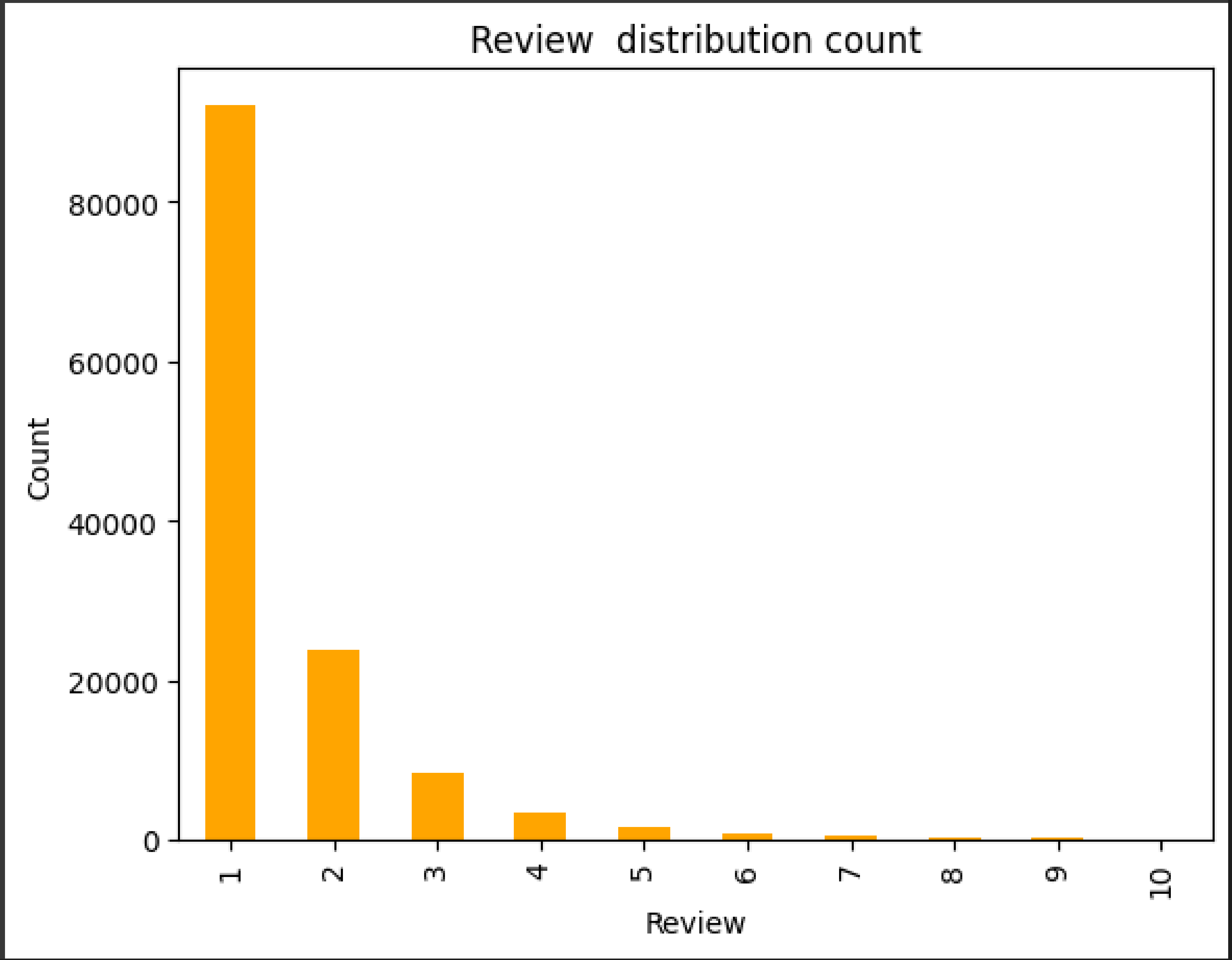


[16]
```python
# df['Review Count'].str.split(' ')
df['Review Count']=df['Review Count'].str.extract('(\d+)').astype(int)
```

[17]
```python
data2=df['Review Count'].value_counts()
data2=data2.head(10)
```

0s
```python
#Bar plot to visualize the total counts of each rating
data2.plot.bar(color = '#FFA500')
plt.title('Review  distribution count')
plt.xlabel('Review')
plt.ylabel('Count')
plt.show()
```

Majority of people gives at least 1 review



Rating
Rated 5 out of 5 stars
Rated 5 out of 5 stars
Rated 5 out of 5 stars
Rated 5 out of 5 stars
Rated 3 out of 5 stars

The Rating column contains text entries with ratings embedded within. We will extract the first numeric value from each entry to determine the actual rating.

```
[ ]  df['Rating'] = df['Rating'].str.extract('(\d+)').astype(int)
```

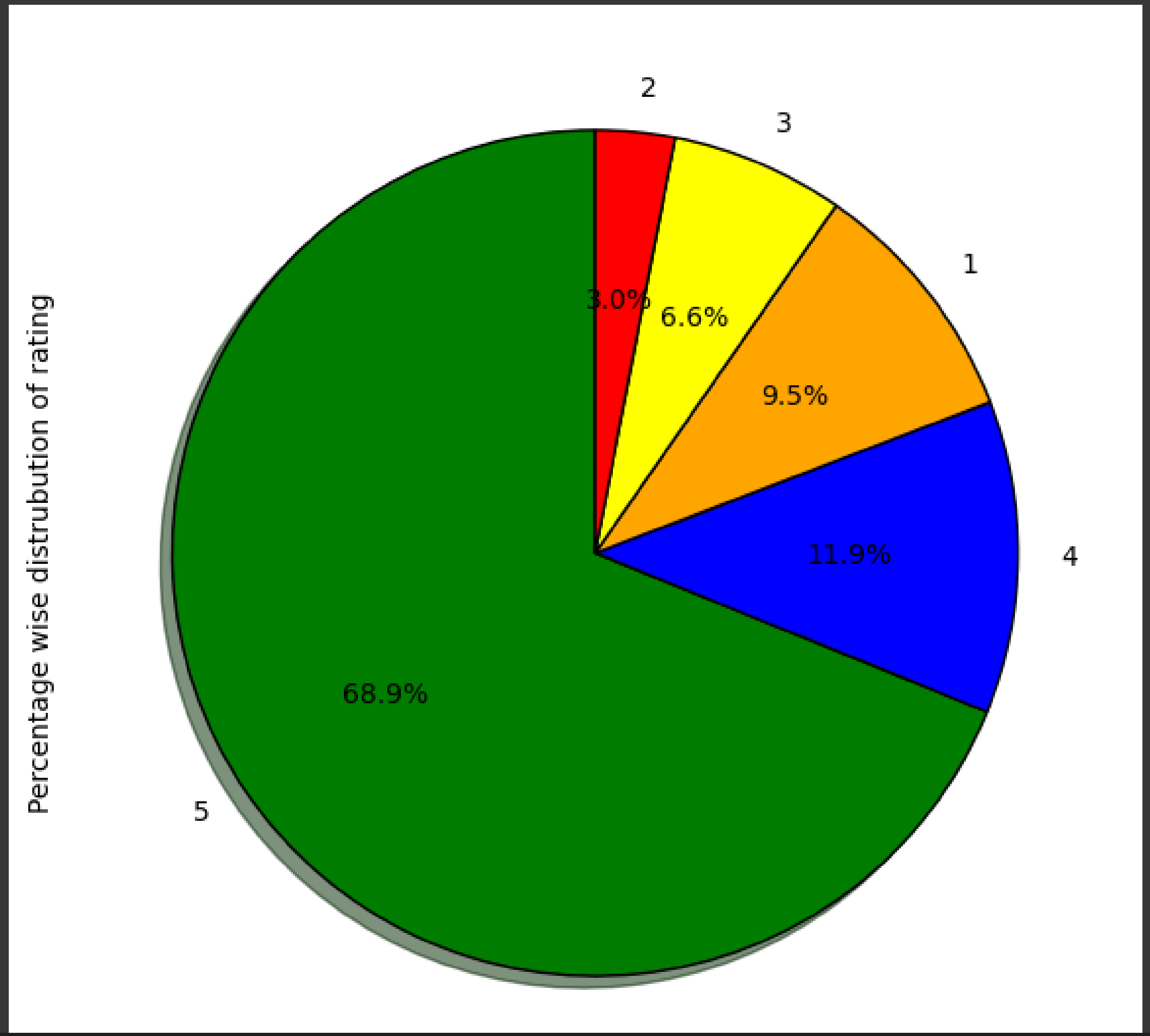
```python
df['Rating'].value_counts()
```

Rating	count
5	90830
4	15731
1	12575
3	8734
2	3998

dtype: int64

The majority of individuals provide ratings of 4 and 5, while there are also instances of ratings as low as 1.

```python
fig = plt.figure(figsize=(7,7))

colors = ( 'green', 'blue','orange','yellow','red')

wp = {'linewidth':1, "edgecolor":'black'}

tags = df['Rating'].value_counts()/data.shape[0]

tags.plot(kind='pie', autopct="%1.1f%%", shadow=True, colors=colors, startangle=90, wedgeprops=wp,  label='Percentage wise distrubution of rating')
```

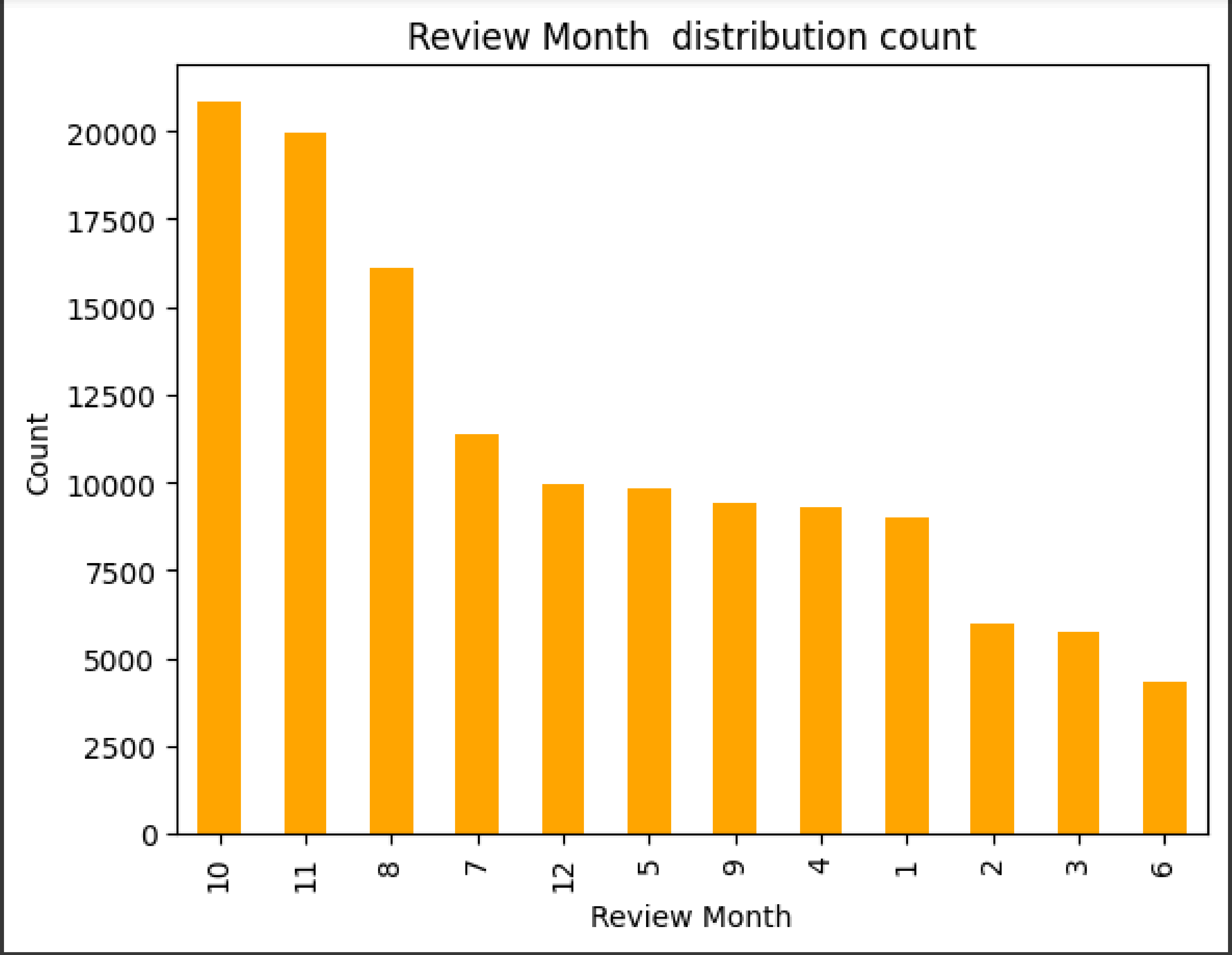


```
[ ] df['Date of Experience']=pd.to_datetime(df['Date of Experience'])
    # df['Date of Experience'].dt.month()
```

```
[ ] df['Review month']=df['Date of Experience'].dt.month
```

```
[ ] data3=df['Review month'].value_counts()
```

```
#Bar plot to visualize the total counts of each rating
data3.plot.bar(color = '#FFA500')
plt.title('Review Month  distribution count')
plt.xlabel('Review Month')
plt.ylabel('Count')
plt.show()
```

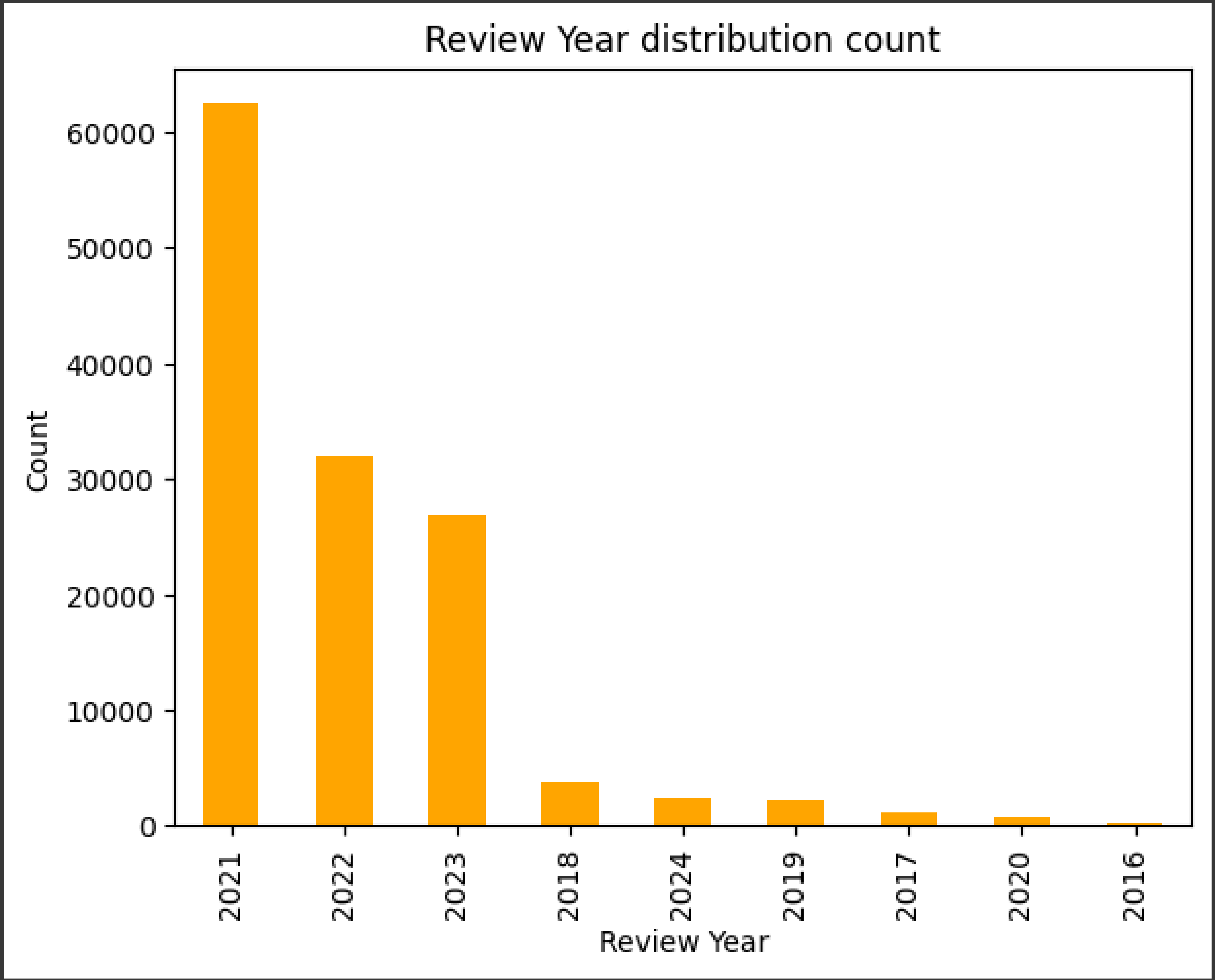


Winter months, particularly October,November and December, exhibit a higher volume of reviews. Additionally, in the UK, many festivals occur during these months.

```python
df['Review year']=df['Date of Experience'].dt.year
```

```python
data4=df['Review year'].value_counts()
```

```python
#Bar plot to visualize the total counts of each rating
data4.plot.bar(color = '#FFA500')
plt.title('Review Year distribution count')
plt.xlabel('Review Year')
plt.ylabel('Count')
plt.show()
```



Review Year distribution count
Count
Review Year
60000
50000
40000
30000
20000
10000
0
2021
2022
2023
2018
2024
2019
2017
2020
2016

The year 2021 saw a significant increase in review volume, likely due to the COVID-19 pandemic, which led to a surge in online shopping and higher sales during that period.

```python
df['Review Title']
```

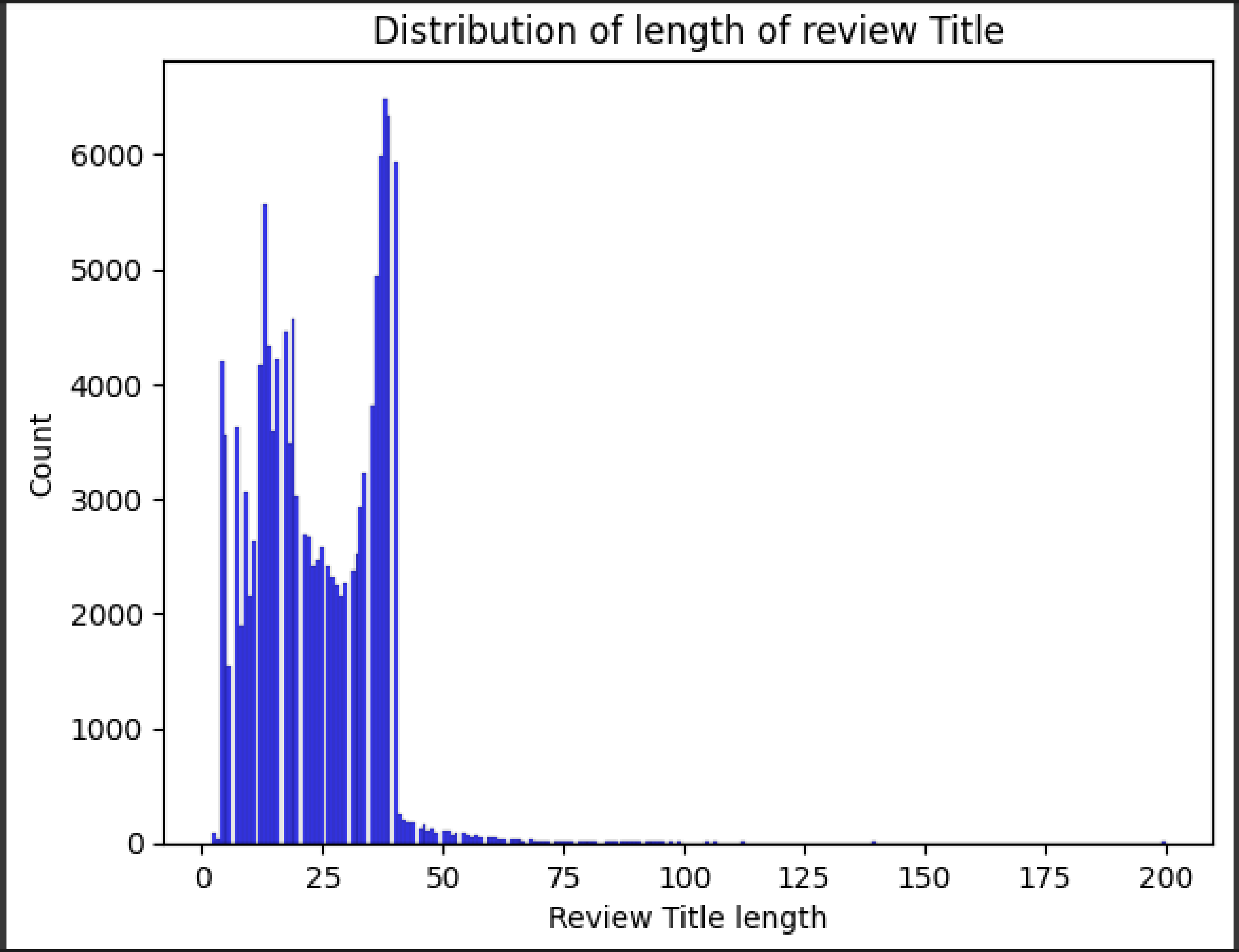
	Review Title
0	I love ordering from fashion nova
1	Top tier content for fashion nova
2	Prices and quality of products are...
3	Great customer service
4	False advertising
...	...
131975	My experience was horrible
131976	amazing
131977	Very helpful
131978	Courteous treatment will make a customer a wal...
131979	Wonderful staff

131868 rows × 1 columns

This is a review title column we will perform analysis on it

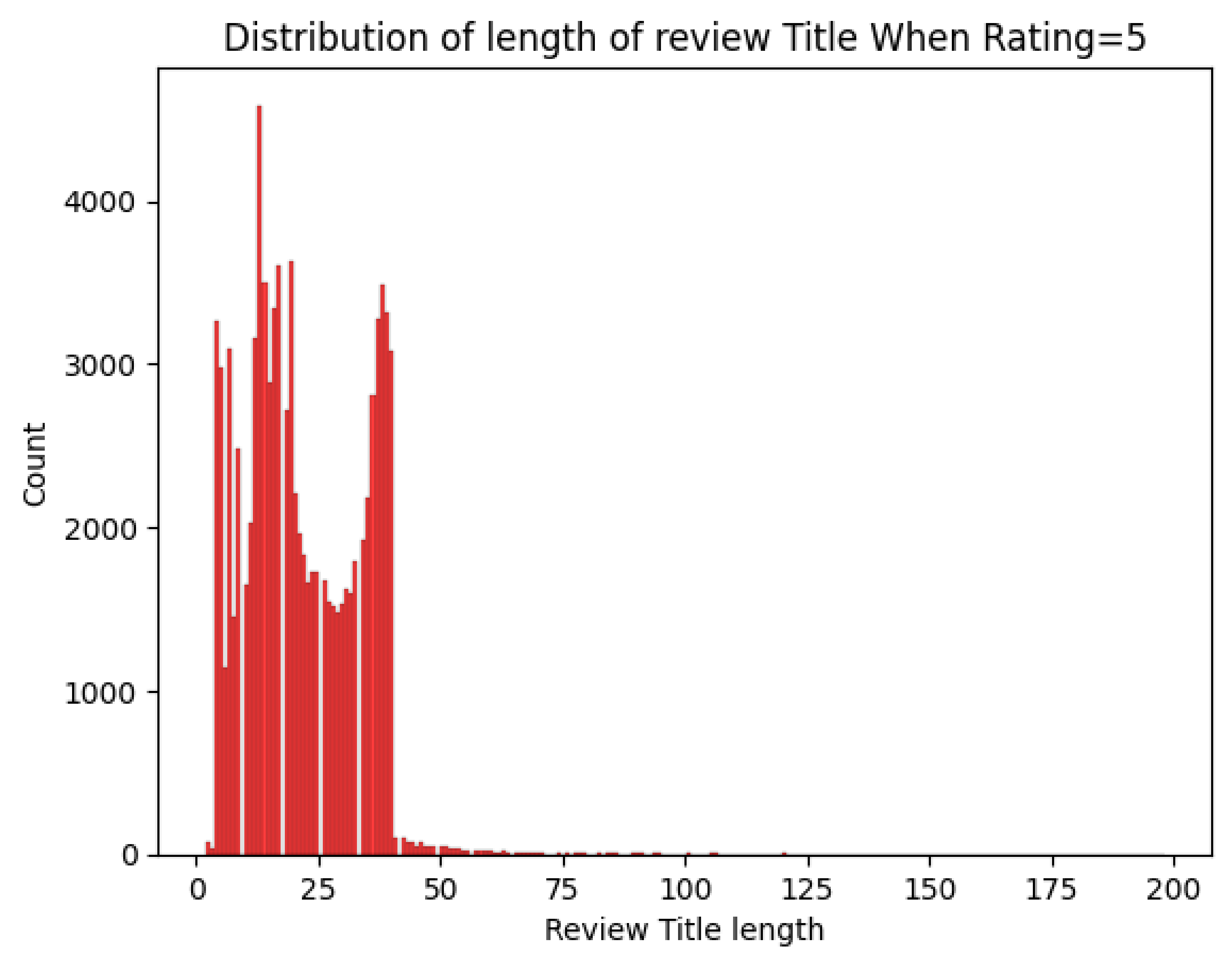
```
[ ] #Creating a new column 'length' that will contain the length of the string in 'verified_reviews' column
    df['Review Title length'] = df['Review Title'].apply(len)
```

```
sns.histplot(df['Review Title length'],color='blue').set(title='Distribution of length of review Title')
```



```
sns.histplot(df[df['Rating']==5]['Review Title length'],color='red').set(title='Distribution of length of review Title When Rating=5')
```

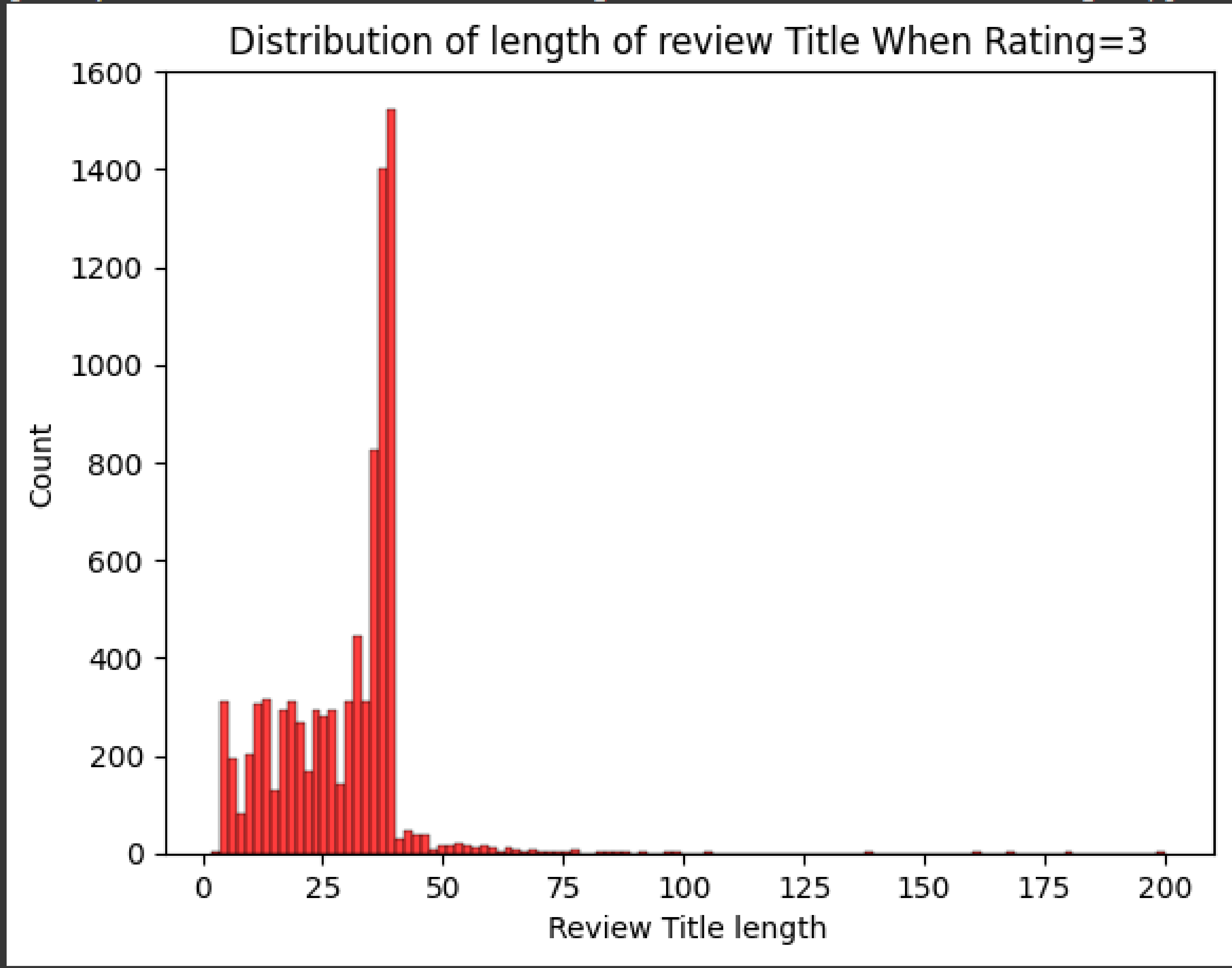
```
[Text(0.5, 1.0, 'Distribution of length of review Title When Rating=5')]
```



The graph indicates that individuals who give a rating of 5 tend to write longer reviews.

```
sns.histplot(df[df['Rating']==3]['Review Title length'],color='red').set(title='Distribution of length of review Title When Rating=3')
```

```
[Text(0.5, 1.0, 'Distribution of length of review Title When Rating=3')]
```



The graph indicates that individuals who give a rating of 3 tend to write shorter reviews which can leads to that people may not like the service.

```
[98] from sklearn.feature_extraction.text import CountVectorizer

     cv = CountVectorizer(stop_words='english')
     words = cv.fit_transform(df['Review Title'])
```

```
# Combine all reviews
reviews = " ".join([review for review in df['Review Title']
])

# Initialize wordcloud object
wc = WordCloud(background_color='white', max_words=100)

# Generate and plot wordcloud
plt.figure(figsize=(10,10))
plt.imshow(wc.generate(reviews))
plt.title('Wordcloud for all reviews Title', fontsize=10)
plt.axis('off')
plt.show()
```

Positive Review title from the people include words like Fashion,Love,Good,Awesome,Thank,bst,Quick etc

1s [100]

```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(stop_words='english')
words = cv.fit_transform(df['Review Text'])
```

5s

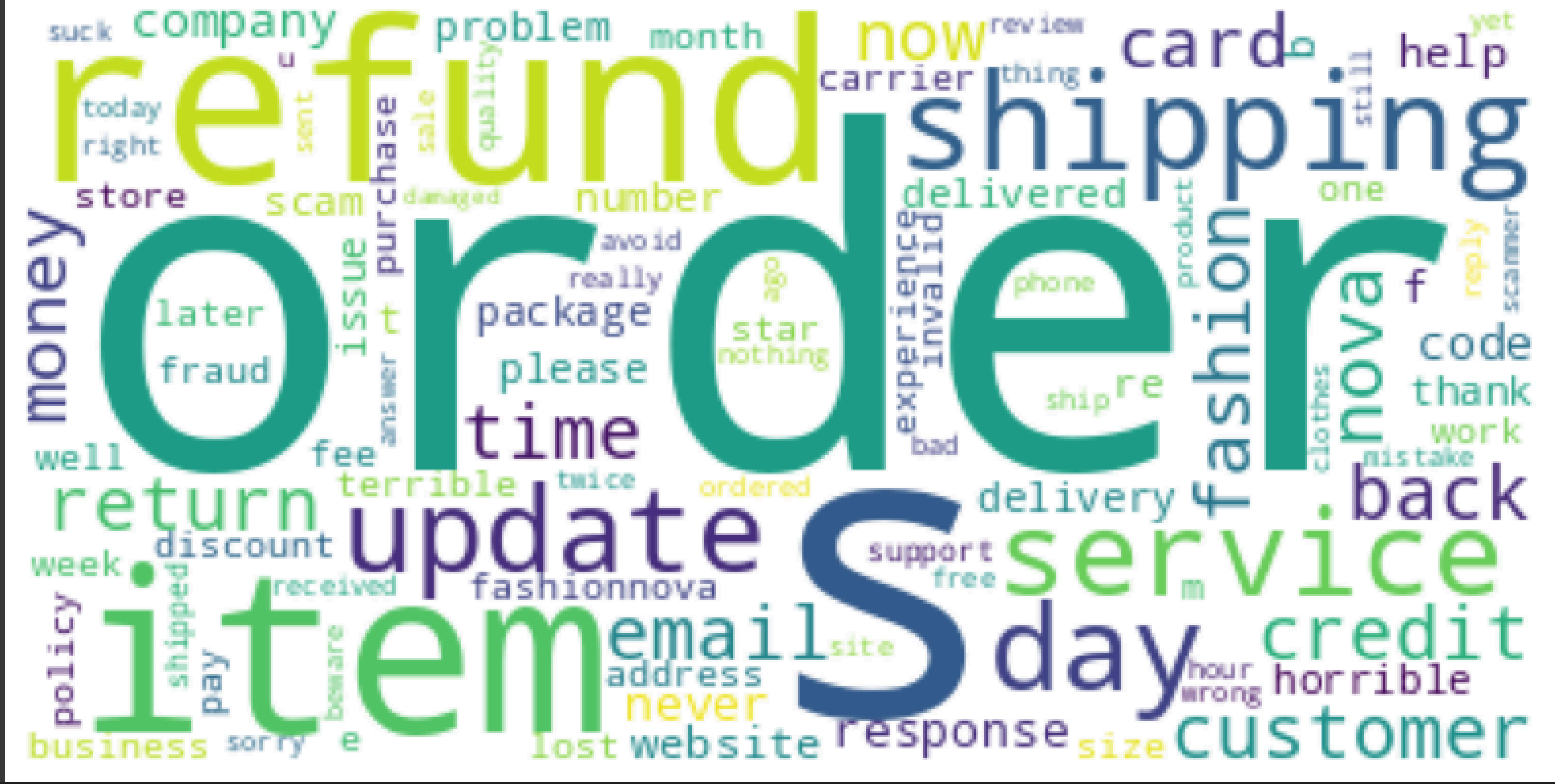
```
# Combine all reviews
reviews = " ".join([review for review in df['Review Text']
])

# Initialize wordcloud object
wc = WordCloud(background_color='white', max_words=100)

# Generate and plot wordcloud
plt.figure(figsize=(10,10))
plt.imshow(wc.generate(reviews))
plt.title('Wordcloud for all reviews', fontsize=10)
plt.axis('off')
plt.show()
```

Wordcloud for all reviews

```python
from collections import Counter

# Combine all reviews for each feedback category and count word frequencies
negative_reviews = Counter(" ".join(df[df['Rating'] <= 2]['Review Text']).lower().split())
positive_reviews = Counter(" ".join(df[df['Rating'] >= 3]['Review Text']).lower().split())

# Convert to sets of words
negative_words = set(negative_reviews)
positive_words = set(positive_reviews)

# Finding words unique to each feedback category
unique_negative_words = negative_words - positive_words
unique_positive_words = positive_words - negative_words

# Join them into strings if necessary
unique_negative = " ".join(unique_negative_words)
unique_positive = " ".join(unique_positive_words)
```

```python
# Generate and plot wordcloud
plt.figure(figsize=(10,10))
plt.imshow(wc.generate(unique_negative))
plt.title('Wordcloud for negative reviews', fontsize=10)
plt.axis('off')
plt.show()
```

Wordcloud for negative reviews



Horrible,never,terrible,bad,hour wrong , late,scam are such words which are showig negative reviews about the service

```
wc = WordCloud(background_color='white', max_words=150)

# Generate and plot wordcloud
plt.figure(figsize=(10,10))
plt.imshow(wc.generate(unique_positive))
plt.title('Wordcloud for positive reviews', fontsize=10)
plt.axis('off')
plt.show()
```

Wordcloud for positive reviews

0s

```python
from textblob import TextBlob
def classify_sentiment(review):
    analysis = TextBlob(review)
    if analysis.sentiment.polarity > 0:
        return 1
    elif analysis.sentiment.polarity < 0:
        return -1
    else:
        return 0
```

26s

```python
[108] df['review_class'] = df['Review Text'].apply(classify_sentiment)
```

```
[109] corpus = []
      stemmer = PorterStemmer()
      for i in range(0, df.shape[0]):
        review = re.sub('[^a-zA-Z]', ' ', df.iloc[i]['Review Text'])
        review = review.lower().split()
        review = [stemmer.stem(word) for word in review if not word in STOPWORDS]
        review = ' '.join(review)
        corpus.append(review)
```

```
[110] cv = CountVectorizer(max_features = 2500)

      #Storing independent and dependent variables in X and y
      X = cv.fit_transform(corpus).toarray()
      y = df['review_class'].values
```

```
[111] #Saving the Count Vectorizer
      pickle.dump(cv, open('countVectorizer.pkl', 'wb'))
```

```
print(f"X shape: {X.shape}")
print(f"y shape: {y.shape}")
```

```
X shape: (131868, 2500)
y shape: (131868,)
```

✓ 0s [113]
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 15)

print(f"X train: {X_train.shape}")
print(f"y train: {y_train.shape}")
print(f"X test: {X_test.shape}")
print(f"y test: {y_test.shape}")
```

```
X train: (92307, 2500)
y train: (92307,)
X test: (39561, 2500)
y test: (39561,)
```

✓ 3s
```python
scaler = MinMaxScaler()

X_train_scl = scaler.fit_transform(X_train)
X_test_scl = scaler.transform(X_test)
```

✓ 0s [115]
```python
#Saving the scaler model
pickle.dump(scaler, open('scaler.pkl', 'wb'))
```

```python
from sklearn.model_selection import cross_val_score, KFold
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score, classification_report, confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, BaggingClassifier, ExtraTreesClassifier, GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB

# List of classifiers
classifiers = {
    'Logistic Regression': LogisticRegression(multi_class='ovr'),
    'Random Forest': RandomForestClassifier(),
    'Decision Tree': DecisionTreeClassifier(),
    'Naive Bayes': GaussianNB()
}

# Dictionaries to store results
results_train = {}
results_test = {}

# K-Fold cross-validation
kfold = KFold(n_splits=2, shuffle=True, random_state=42)
```

```python
# K-Fold cross-validation
kfold = KFold(n_splits=2, shuffle=True, random_state=42)

for name, clf in classifiers.items():
    # Cross-validation
    cv_results = cross_val_score(clf, X_train_scl, y_train, cv=kfold, scoring='accuracy')
    results_train[name] = {
        'CrossVal_Score_Mean': cv_results.mean(),
        'CrossVal_Error': cv_results.std()
    }

    # Train the model
    clf.fit(X_train, y_train)

    # Make predictions on the test set
    y_pred = clf.predict(X_test)
    y_pred_proba = clf.predict_proba(X_test_scl) if hasattr(clf, "predict_proba") else None

    # Evaluate the predictions
    accuracy = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred, average='weighted')  # Use 'weighted' for multiclass
    roc_auc = roc_auc_score(y_test, y_pred_proba, multi_class='ovr', average='weighted') if y_pred_proba is not None else 'N/A'
    clf_report = classification_report(y_test, y_pred)
    cm = confusion_matrix(y_test, y_pred)
```

```python
    results_test[name] = {
        'Accuracy': accuracy,
        'F1_Score': f1,
        'ROC_AUC_Score': roc_auc,
        'Classification_Report': clf_report,
        'Confusion_Matrix': cm
    }

# Print the cross-validation results
for name, result in results_train.items():
    print(f"{name} (Training):")
    print(f"  CrossVal_Score_Mean: {result['CrossVal_Score_Mean']:.4f}")
    print(f"  CrossVal_Error: {result['CrossVal_Error']:.4f}")
    print()

# Print the test results
for name, result in results_test.items():
    print(f"{name} (Test):")
    print(f"  Accuracy: {result['Accuracy']:.4f}")
    print(f"  F1_Score: {result['F1_Score']:.4f}")
    print(f"  ROC_AUC_Score: {result['ROC_AUC_Score']}")
    print(f"  Classification_Report:\n{result['Classification_Report']}")
    print(f"  Confusion_Matrix:\n{result['Confusion_Matrix']}\n")
```

```
Logistic Regression (Training):
  CrossVal_Score_Mean: 0.9080
  CrossVal_Error: 0.0007

Random Forest (Training):
  CrossVal_Score_Mean: 0.9210
  CrossVal_Error: 0.0001

Decision Tree (Training):
  CrossVal_Score_Mean: 0.9223
  CrossVal_Error: 0.0028

Naive Bayes (Training):
  CrossVal_Score_Mean: 0.4645
  CrossVal_Error: 0.0041
```

```
Logistic Regression (Test):
  Accuracy: 0.9497
  F1_Score: 0.9487
  ROC_AUC_Score: 0.9744706548820037
  Classification_Report:
              precision    recall  f1-score   support

          -1       0.83      0.71      0.77      3019
           0       0.96      0.97      0.97     14482
           1       0.96      0.97      0.96     22060

    accuracy                           0.95     39561
   macro avg       0.91      0.88      0.90     39561
weighted avg       0.95      0.95      0.95     39561

  Confusion_Matrix:
[[ 2154   237   628]
 [   83 14095   304]
 [  364   374 21322]]
```

```
Random Forest (Test):
  Accuracy: 0.9234
  F1_Score: 0.9188
  ROC_AUC_Score: 0.5478357330968104
  Classification_Report:
              precision    recall  f1-score   support

          -1       0.79      0.50      0.61      3019
           0       0.94      0.95      0.95     14482
           1       0.92      0.96      0.94     22060

    accuracy                           0.92     39561
   macro avg       0.89      0.80      0.83     39561
weighted avg       0.92      0.92      0.92     39561

  Confusion_Matrix:
[[ 1512   374  1133]
 [   82 13742   658]
 [  308   474 21278]]
```

```
Decision Tree (Test):
  Accuracy: 0.9236
  F1_Score: 0.9232
  ROC_AUC_Score: 0.5111788947584278
  Classification_Report:
              precision    recall  f1-score   support

          -1       0.67      0.64      0.65      3019
           0       0.95      0.96      0.95     14482
           1       0.94      0.94      0.94     22060

    accuracy                           0.92     39561
   macro avg       0.85      0.85      0.85     39561
weighted avg       0.92      0.92      0.92     39561

  Confusion_Matrix:
[[ 1937   247   835]
 [  169 13853   460]
 [  798   514 20748]]
```

```
Naive Bayes (Test):
  Accuracy: 0.5858
  F1_Score: 0.5594
  ROC_AUC_Score: 0.6207236883701452
  Classification_Report:
              precision    recall  f1-score   support

          -1       0.31      0.72      0.44      3019
           0       0.55      0.95      0.70     14482
           1       0.94      0.33      0.48     22060

    accuracy                           0.59     39561
   macro avg       0.60      0.67      0.54     39561
weighted avg       0.75      0.59      0.56     39561

  Confusion_Matrix:
[[ 2182   520   317]
 [  519 13802   161]
 [ 4227 10642  7191]]
```

Out of all the models logistic regression is performing well with the accuracy of 94.9


Thank you
rawpixel