```
import pandas as pd

df = pd.read_csv("complaints.csv")

df.head()
```

```
C:\Users\GANESH\AppData\Local\Temp\ipykernel_14500\3056042945.py:3:
DtypeWarning: Columns (16) have mixed types. Specify dtype option on
import or set low_memory=False.
  df = pd.read_csv("complaints.csv")
```

```
  Date received                                         Product  \
0    2024-01-23  Credit reporting or other personal consumer re...
1    2024-01-24  Credit reporting or other personal consumer re...
2    2024-01-24  Credit reporting or other personal consumer re...
3    2024-01-23  Credit reporting or other personal consumer re...
4    2024-01-23  Credit reporting or other personal consumer re...

        Sub-product                                  Issue  \
0  Credit reporting  Incorrect information on your report
1  Credit reporting  Incorrect information on your report
2  Credit reporting           Improper use of your report
3  Credit reporting           Improper use of your report
4  Credit reporting           Improper use of your report

                                          Sub-issue  \
0               Information belongs to someone else
1               Information belongs to someone else
2  Credit inquiries on your report that you don't...
3      Reporting company used your report improperly
4      Reporting company used your report improperly

                       Consumer complaint narrative  \
0                                               NaN
1                                               NaN
2                                               NaN
3  In accordance with the Fair Credit Reporting a...
4  I have observed several deviations from mandat...

                            Company public response  \
0  Company has responded to the consumer and the ...
1  Company has responded to the consumer and the ...
2  Company has responded to the consumer and the ...
3  Company has responded to the consumer and the ...
4  Company has responded to the consumer and the ...

                                  Company State ZIP code Tags  \
0  TRANSUNION INTERMEDIATE HOLDINGS, INC.    ME    04005  NaN
1  TRANSUNION INTERMEDIATE HOLDINGS, INC.    FL    33311  NaN
2  TRANSUNION INTERMEDIATE HOLDINGS, INC.    PA    175XX  NaN
```

```
3  TRANSUNION INTERMEDIATE HOLDINGS, INC.    TX    79907  NaN
4  TRANSUNION INTERMEDIATE HOLDINGS, INC.    NY    10075  NaN

  Consumer consent provided? Submitted via Date sent to company  \
0       Consent not provided           Web           2024-01-23
1                      Other           Web           2024-01-24
2                      Other           Web           2024-01-24
3           Consent provided           Web           2024-01-23
4           Consent provided           Web           2024-01-23

       Company response to consumer Timely response? Consumer disputed?
\
0  Closed with non-monetary relief              Yes                 NaN

1  Closed with non-monetary relief              Yes                 NaN

2  Closed with non-monetary relief              Yes                 NaN

3  Closed with non-monetary relief              Yes                 NaN

4  Closed with non-monetary relief              Yes                 NaN


   Complaint ID
0       8206605
1       8211390
2       8211362
3       8210433
4       8209430

df.shape

(5134967, 16)

(df.isna().mean()*100).round(2)

Date received                   0.00
Product                         0.00
Sub-product                     4.58
Issue                           0.00
Sub-issue                      14.31
Consumer complaint narrative   64.84
Company public response        52.14
Company                         0.00
State                           0.89
ZIP code                        0.59
Tags                           90.62
Consumer consent provided?     19.95
Submitted via                   0.00
Date sent to company            0.00
Company response to consumer    0.00
```

```
Timely response?                0.00
Consumer disputed?             85.04
Complaint ID                    0.00
dtype: float64
```

Understanding of each column in the dataset

- **Date received**: The date when the complaint was received by the agency.
- **Product**: The category of product or service related to the complaint.
- **Sub-product**: A more specific category within the product category.
- **Issue**: The type of issue or problem the consumer is experiencing.
- **Sub-issue**: A more specific description of the issue.
- **Consumer complaint narrative**: A detailed description of the complaint provided by the consumer.
- **Company public response**: The company's official response to the complaint, which may be publicly visible.
- **Company**: The name of the company involved in the complaint.
- **State**: The state where the consumer is located.
- **ZIP code**: The consumer's ZIP code.
- **Tags**: Keywords or categories assigned to the complaint for easier filtering or searching.
- **Consumer consent provided?**: Whether the consumer has given consent for their complaint to be shared publicly.
- **Submitted via**: The method by which the complaint was submitted (e.g., web, phone, mail).
- **Date sent to company**: The date when the complaint was forwarded to the company.
- **Company response to consumer**: The company's response to the consumer, which may not be publicly visible.
- **Timely response?**: Whether the company responded to the complaint in a timely manner.
- **Consumer disputed?**: Whether the consumer disputed the company's response.
- **Complaint ID**: A unique identifier assigned to the complaint.

```
df.columns
```

```
Index(['Date received', 'Product', 'Sub-product', 'Issue', 'Sub-
issue',
       'Consumer complaint narrative', 'Company public response',
'Company',
       'State', 'ZIP code', 'Tags', 'Consumer consent provided?',
       'Submitted via', 'Date sent to company', 'Company response to
consumer',
       'Timely response?', 'Consumer disputed?', 'Complaint ID'],
      dtype='object')
```

Removing columns having more than 70% null values

```
df.drop(columns=["Tags","Consumer disputed?"],inplace=True)
```

```
(df.isna().mean()*100).round(2)
```

```
Date received                   0.00
Product                         0.00
Sub-product                     4.58
Issue                           0.00
Sub-issue                      14.31
Consumer complaint narrative   64.84
Company public response        52.14
Company                         0.00
State                           0.89
ZIP code                        0.59
Consumer consent provided?     19.95
Submitted via                   0.00
Date sent to company            0.00
Company response to consumer    0.00
Timely response?                0.00
Complaint ID                    0.00
dtype: float64
```

```
df["Sub-product"].value_counts()
```

```
Credit reporting                                  3069348
Checking account                                   227762
General-purpose credit card or charge card         197036
I do not know                                      133125
Other debt                                         109001
                                                    ...
Transit card                                           37
Earned wage access                                     36
Student loan debt relief                               21
Electronic Benefit Transfer / EBT card                 12
Tax refund anticipation loan or check                   8
Name: Sub-product, Length: 86, dtype: int64
```

```
df["Sub-product"] = df["Sub-product"].fillna(df["Sub-product"].mode()
[0])
```

```
(df.isna().mean()*100).round(2)
```

```
Date received                   0.00
Product                         0.00
Sub-product                     0.00
Issue                           0.00
Sub-issue                      14.31
Consumer complaint narrative   64.84
Company public response        52.14
Company                         0.00
State                           0.89
ZIP code                        0.59
Consumer consent provided?     19.95
```

```
Submitted via                    0.00
Date sent to company             0.00
Company response to consumer     0.00
Timely response?                 0.00
Complaint ID                     0.00
dtype: float64
```

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

```
Date received                         0
Product                               0
Sub-product                           0
Issue                                 2
Sub-issue                        734684
Consumer complaint narrative    3329405
Company public response         2677245
Company                               0
State                             45517
ZIP code                          30225
Consumer consent provided?      1024493
Submitted via                         0
Date sent to company                  0
Company response to consumer         14
Timely response?                      0
Complaint ID                          0
dtype: int64
```

```
df["Issue"] = df["Issue"].fillna(df["Issue"].mode()[0])
```

```
df["Sub-issue"].value_counts()
```

```
Information belongs to someone else
988837
Reporting company used your report improperly
517881
Their investigation did not fix an error on your report
452281
Credit inquiries on your report that you don't recognize
268236
Account information incorrect
180125

...
Problem with a credit reporting company's investigation into an
existing problem         5
Issues with financial aid services
4
Problem with fraud alerts or security freezes
1
Credit monitoring or identity theft protection services
```

```
1
Improper use of your report
1
Name: Sub-issue, Length: 272, dtype: int64
```

```
df["Sub-issue"].fillna("Unknown",inplace=True)
```

```
(df.isna().mean()*100).round(2)
```

```
Date received                   0.00
Product                         0.00
Sub-product                     0.00
Issue                           0.00
Sub-issue                       0.00
Consumer complaint narrative   64.84
Company public response        52.14
Company                         0.00
State                           0.89
ZIP code                        0.59
Consumer consent provided?     19.95
Submitted via                   0.00
Date sent to company            0.00
Company response to consumer    0.00
Timely response?                0.00
Complaint ID                    0.00
dtype: float64
```

```
df["Consumer complaint narrative"].fillna("Not found",inplace=True)
```

```
df["Company public response"].fillna("Not found",inplace=True)
```

```
df["Consumer consent provided?"].fillna("Not found",inplace=True)
```

```
(df.isna().mean()*100).round(2)
```

```
Date received                   0.00
Product                         0.00
Sub-product                     0.00
Issue                           0.00
Sub-issue                       0.00
Consumer complaint narrative    0.00
Company public response         0.00
Company                         0.00
State                           0.89
ZIP code                        0.59
Consumer consent provided?      0.00
Submitted via                   0.00
Date sent to company            0.00
Company response to consumer    0.00
Timely response?                0.00
```

```
Complaint ID                    0.00
dtype: float64
```

```
df.isna().sum()
```

```
Date received                       0
Product                             0
Sub-product                         0
Issue                               0
Sub-issue                           0
Consumer complaint narrative        0
Company public response             0
Company                             0
State                           45517
ZIP code                        30225
Consumer consent provided?          0
Submitted via                       0
Date sent to company                0
Company response to consumer       14
Timely response?                    0
Complaint ID                        0
dtype: int64
```

```
df["State"].fillna("Not mentioned",inplace=True)
```

```
df["ZIP code"].fillna("Not mentioned",inplace=True)
```

```
df.isna().sum()
```

```
Date received                     0
Product                           0
Sub-product                       0
Issue                             0
Sub-issue                         0
Consumer complaint narrative      0
Company public response           0
Company                           0
State                             0
ZIP code                          0
Consumer consent provided?        0
Submitted via                     0
Date sent to company              0
Company response to consumer     14
Timely response?                  0
Complaint ID                      0
dtype: int64
```

```
df["Company response to consumer"].fillna("not showned",inplace=True)
```

```
df.isna().sum()
```

```
Date received                   0
Product                         0
Sub-product                     0
Issue                           0
Sub-issue                       0
Consumer complaint narrative    0
Company public response         0
Company                         0
State                           0
ZIP code                        0
Consumer consent provided?      0
Submitted via                   0
Date sent to company            0
Company response to consumer    0
Timely response?                0
Complaint ID                    0
dtype: int64
```

```
df.columns
```

```
Index(['Date received', 'Product', 'Sub-product', 'Issue', 'Sub-
issue',
       'Consumer complaint narrative', 'Company public response',
'Company',
       'State', 'ZIP code', 'Consumer consent provided?', 'Submitted
via',
       'Date sent to company', 'Company response to consumer',
       'Timely response?', 'Complaint ID'],
      dtype='object')
```

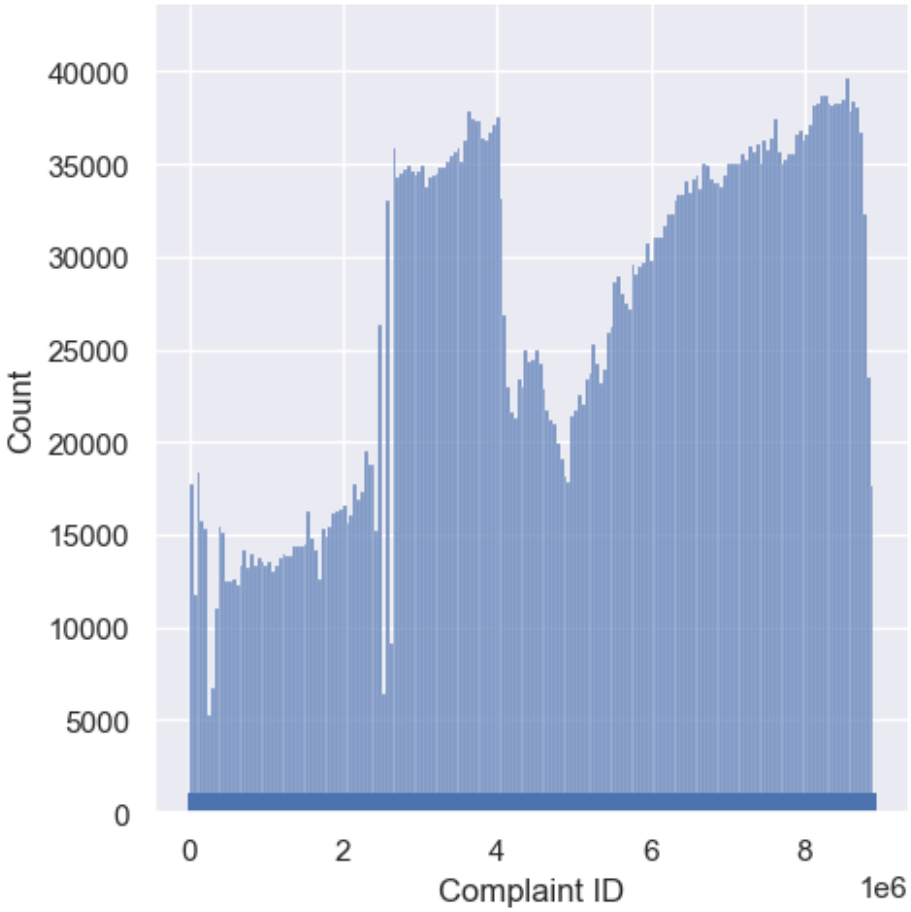
Exploratory Data Analysis [EDA]

```
import seaborn as sns
sns.set(color_codes=True) # to apply standard color codes of seaborn
to the plots
```

Distribution of Complaints using distplot

```
sns.displot(df["Complaint ID"],kde=False,rug=True)
```

```
<seaborn.axisgrid.FacetGrid at 0x1ba50228d90>
```

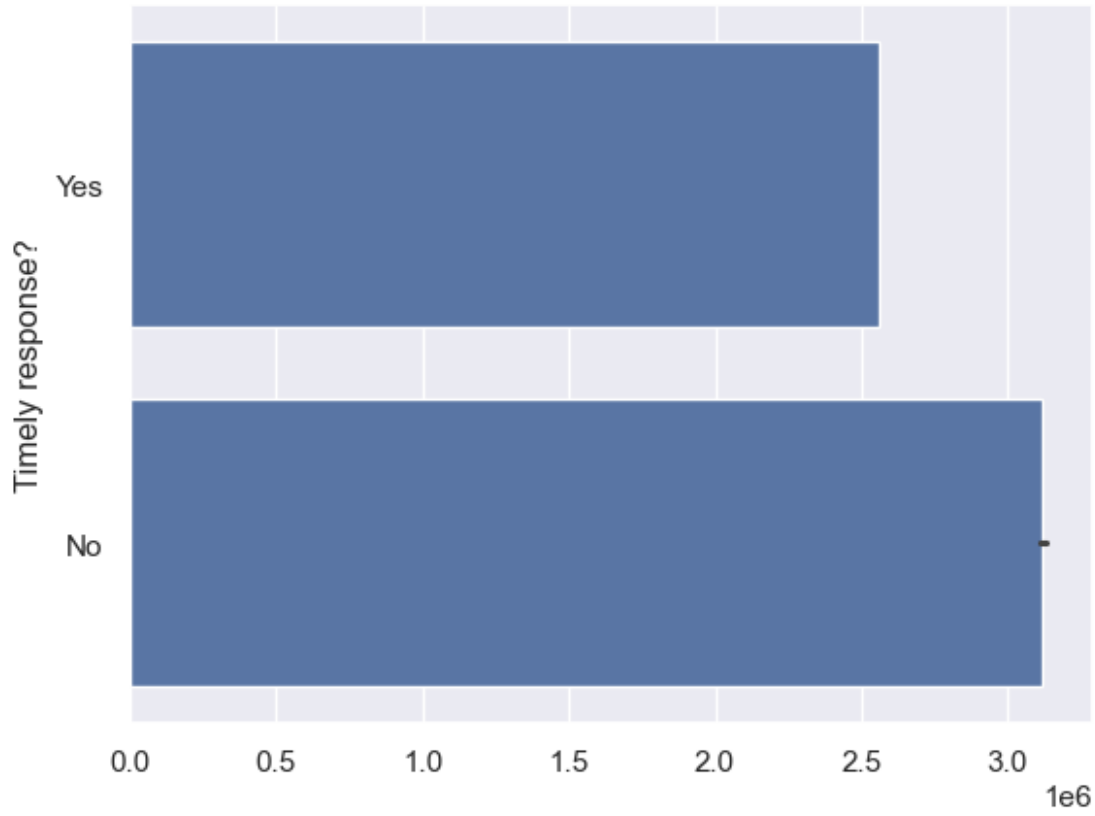


Timely response?: Whether the company responded to the complaint in a timely manner.

The distribution of timely responses

```
sns.barplot(df["Timely response?"])
```

```
<Axes: ylabel='Timely response?'>
```



trends or patterns in complaint volume over time

Month-wise complaints line graph

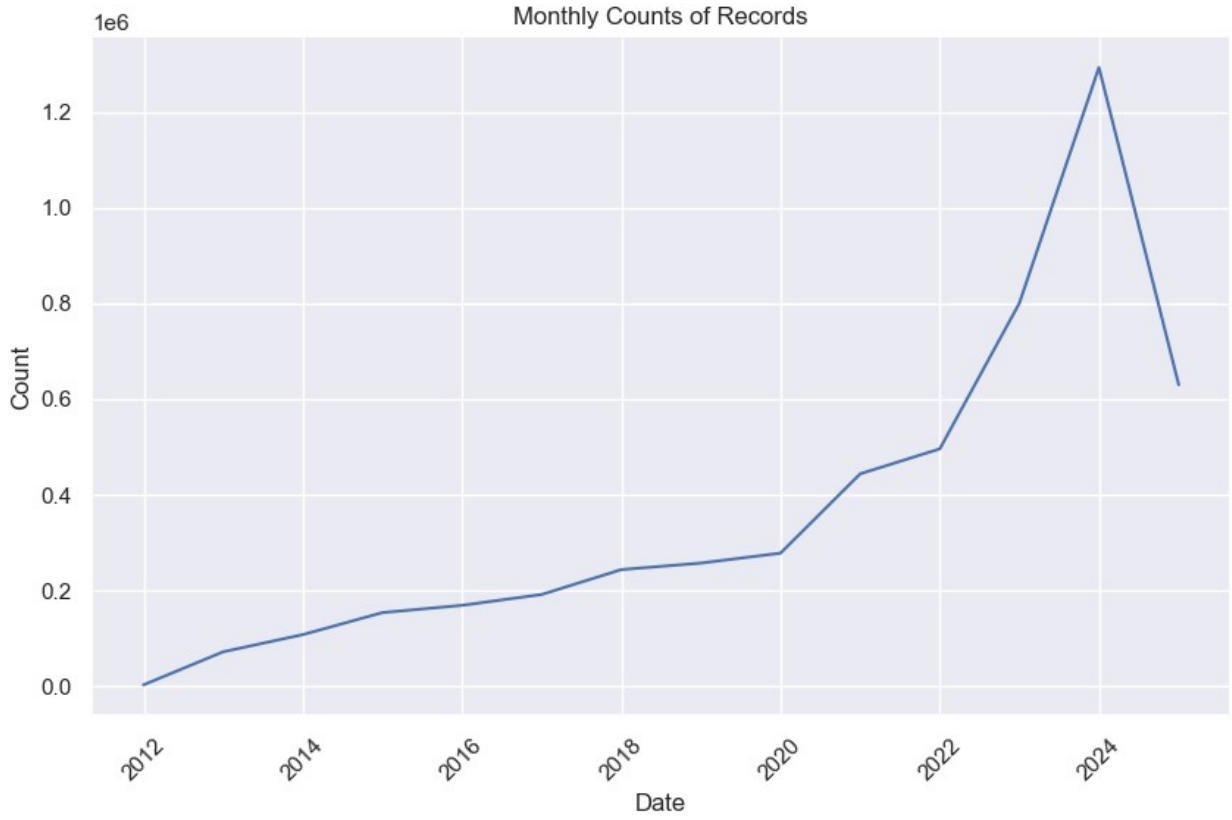
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming df is your DataFrame with a column "Date sent to company"
# Convert the "Date sent to company" column to datetime if it's not
already
df["Date sent to company"] = pd.to_datetime(df["Date sent to
company"])

# Aggregate the data based on some criteria, for example, monthly
averages
df_agg = df.resample('M', on='Date sent to
company').size().reset_index(name='Count')

# Plot the aggregated data
plt.figure(figsize=(10, 6))
sns.lineplot(data=df_agg, x='Date sent to company', y='Count')
plt.title('Monthly Counts of Records')
plt.xlabel('Date')
plt.ylabel('Count')
```

```
plt.xticks(rotation=45)
plt.show()
```

Year-wise complaints line-graph

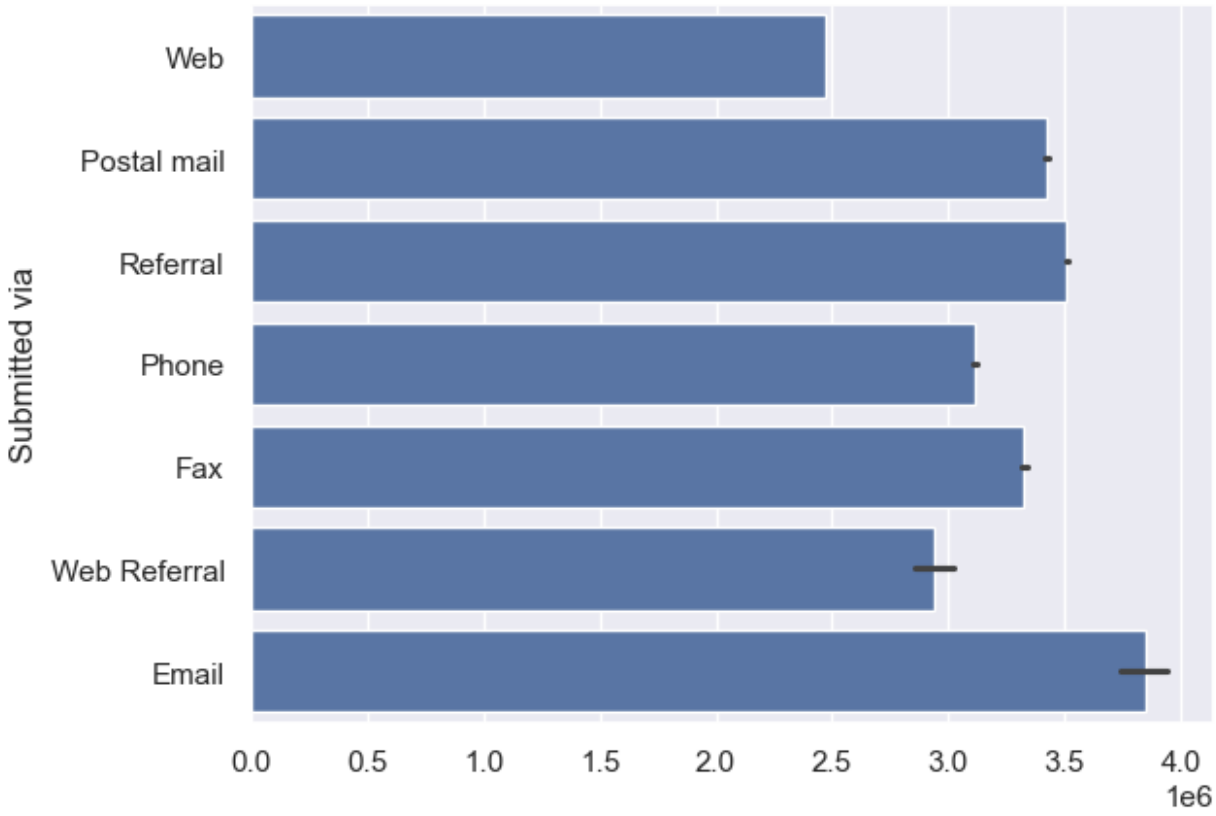
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming df is your DataFrame with a column "Date sent to company"
# Convert the "Date sent to company" column to datetime if it's not
already
df["Date sent to company"] = pd.to_datetime(df["Date sent to
company"])

# Aggregate the data based on some criteria, for example, monthly
averages
df_agg = df.resample('Y', on='Date sent to
company').size().reset_index(name='Count')

# Plot the aggregated data
plt.figure(figsize=(10, 6))
sns.lineplot(data=df_agg, x='Date sent to company', y='Count')
plt.title('Monthly Counts of Records')
plt.xlabel('Date')
```

```
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```




```
sns.barplot(df["Submitted via"])
```

```
<Axes: ylabel='Submitted via'>
```



The most common words and phrases used in company public responses

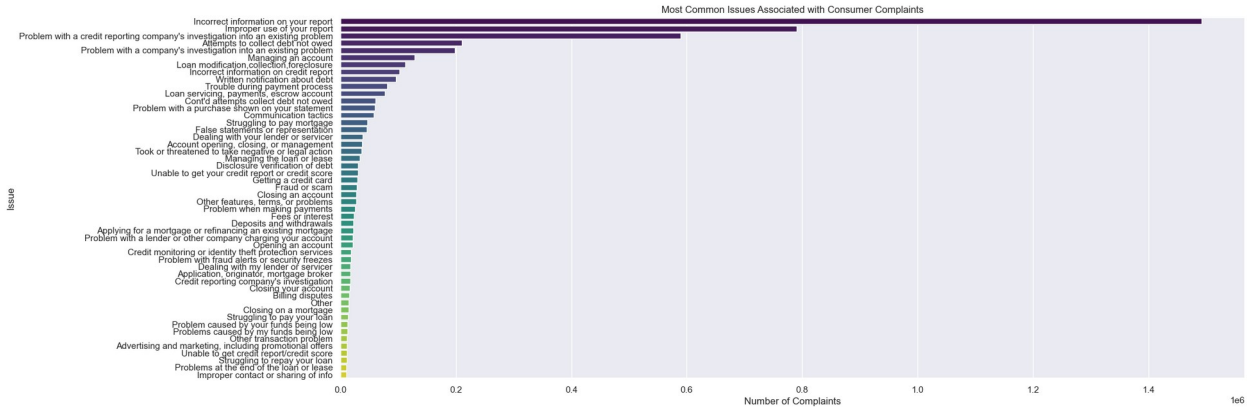
```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Assuming df is your DataFrame with a column "Company public
response"
# Concatenate all responses into a single string
text = ' '.join(df['Company public response'].dropna())

# Generate the word cloud
wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(text)

# Plot the word cloud
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.title('Word Cloud for Company Public Response')
plt.axis('off')
plt.show()
```

Top 50 Issues faced by the product consumers

```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming df is your DataFrame with columns "Issue" and "Sub-issue"
# Count the occurrences of each issue and sub-issue
issue_counts = df['Issue'].value_counts()[:50] # Top 50 issues faced
by the consumers
sub_issue_counts = df['Sub-issue'].value_counts()[:50] # Top 50 sub-
issues faced by the consumers

# Plot the bar plot for issues
plt.figure(figsize=(20, 8))
sns.barplot(x=issue_counts.values, y=issue_counts.index,
palette='viridis')
plt.title('Most Common Issues Associated with Consumer Complaints')
plt.xlabel('Number of Complaints')
plt.ylabel('Issue')
plt.show()
```

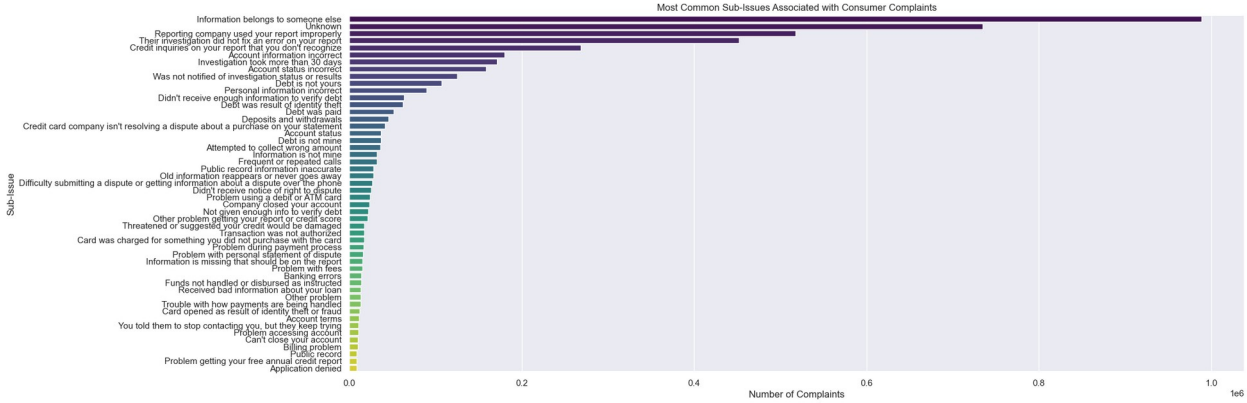
```
C:\Users\GANESH\AppData\Local\Temp\ipykernel_14500\1138860561.py:11:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.
```

```
  sns.barplot(x=issue_counts.values, y=issue_counts.index,
palette='viridis')
```



Top 50 sub-Issues faced by the product consumers

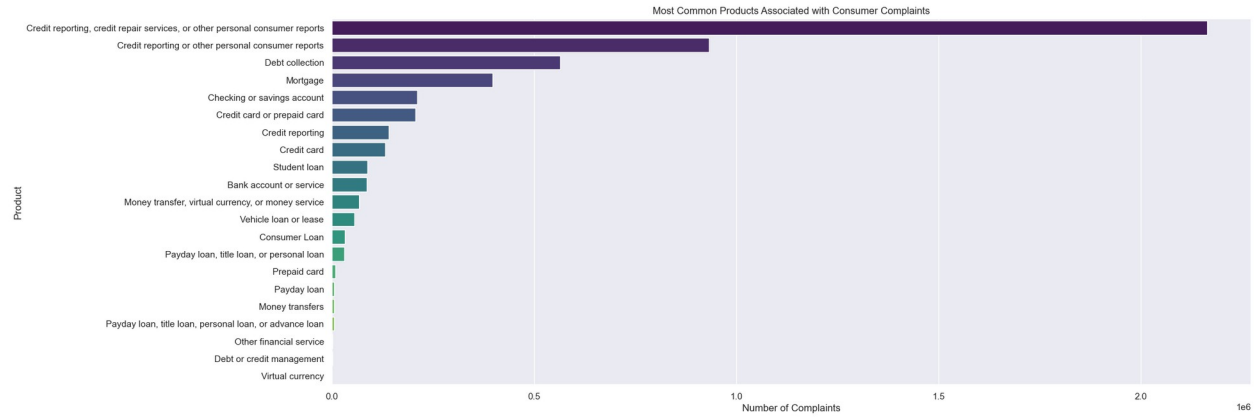
```
# Plot the bar plot for sub-issues
plt.figure(figsize=(20,8))
sns.barplot(x=sub_issue_counts.values, y=sub_issue_counts.index,
palette='viridis')
plt.title('Most Common Sub-Issues Associated with Consumer
Complaints')
plt.xlabel('Number of Complaints')
plt.ylabel('Sub-Issue')
plt.show()

C:\Users\GANESH\AppData\Local\Temp\ipykernel_14500\2963492590.py:3:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=sub_issue_counts.values, y=sub_issue_counts.index,
palette='viridis')
```



Top 50 Products having complaints by the consumers

```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming df is your DataFrame with columns "Product" and "Sub-
product"
# Count the occurrences of each product and sub-product
product_counts = df['Product'].value_counts()
sub_product_counts = df['Sub-product'].value_counts()

# Plot the bar plot for products
plt.figure(figsize=(20,8))
sns.barplot(x=product_counts.values, y=product_counts.index,
palette='viridis')
plt.title('Most Common Products Associated with Consumer Complaints')
plt.xlabel('Number of Complaints')
plt.ylabel('Product')
plt.show()
```

```
C:\Users\GANESH\AppData\Local\Temp\ipykernel_14500\3118708769.py:11:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=product_counts.values, y=product_counts.index,
palette='viridis')
```

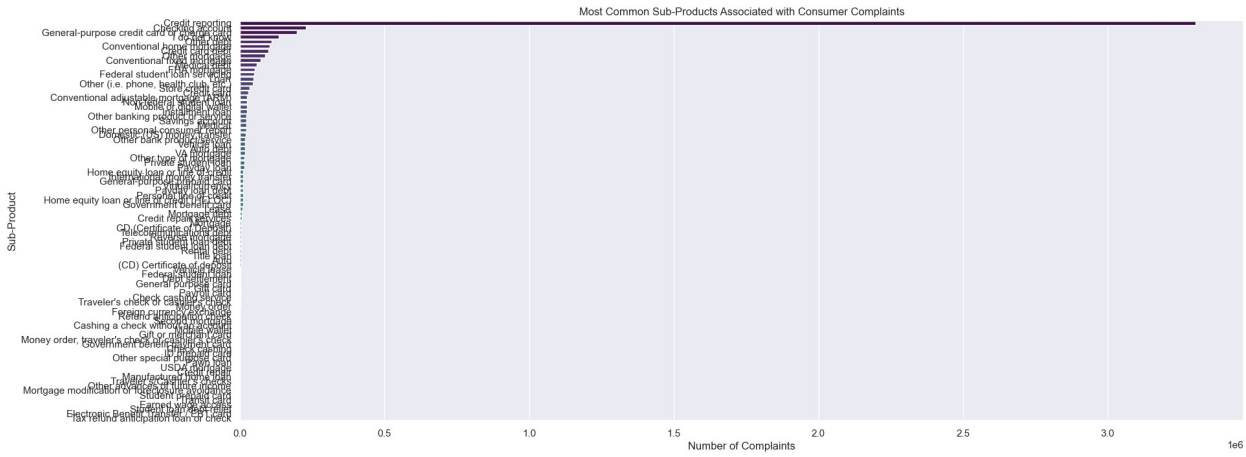


Top 50 Sub-products having complaints by the consumers

```
# Plot the bar plot for sub-products
plt.figure(figsize=(20,8))
sns.barplot(x=sub_product_counts.values, y=sub_product_counts.index,
palette='viridis')
plt.title('Most Common Sub-Products Associated with Consumer
```

```
Complaints')
plt.xlabel('Number of Complaints')
plt.ylabel('Sub-Product')
plt.show()
```

```
C:\Users\GANESH\AppData\Local\Temp\ipykernel_14500\1494137716.py:3:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=sub_product_counts.values, y=sub_product_counts.index,
palette='viridis')
```

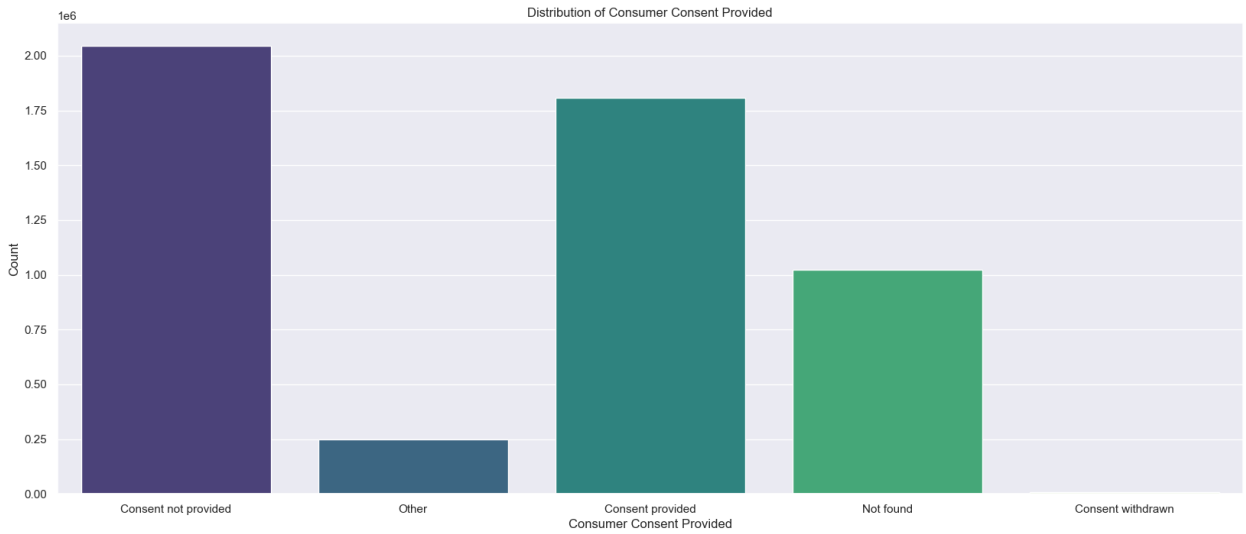


The distribution of consumer consent provided

```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming df is your DataFrame with the column "Consumer consent
provided?"
# Plot the count plot for consumer consent provided
plt.figure(figsize=(20, 8))
sns.countplot(x='Consumer consent provided?', data=df,
palette='viridis')
plt.title('Distribution of Consumer Consent Provided')
plt.xlabel('Consumer Consent Provided')
plt.ylabel('Count')
plt.show()
```

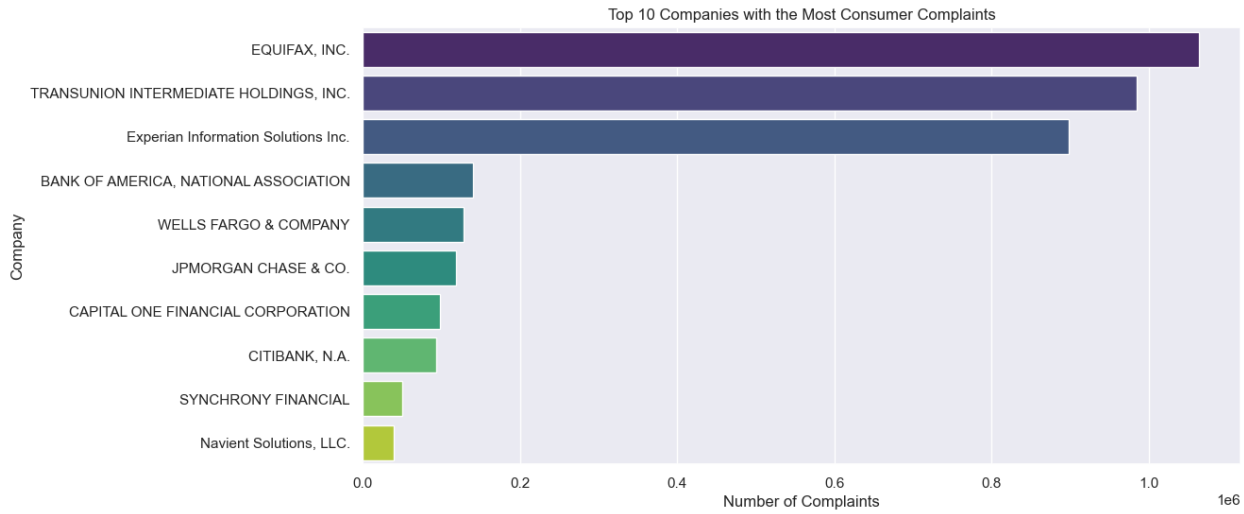
```
C:\Users\GANESH\AppData\Local\Temp\ipykernel_14500\3615125836.py:7:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
```

```
`legend=False` for the same effect.

  sns.countplot(x='Consumer consent provided?', data=df,
palette='viridis')
```



Top 10 companies with the most consumer complaints

```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming df is your DataFrame with the column "Company"
# Count the occurrences of each company
company_counts = df['Company'].value_counts()[:10]

# Plot the bar plot for companies
plt.figure(figsize=(12, 6))
sns.barplot(x=company_counts.values[:10], y=company_counts.index[:10],
palette='viridis')
plt.title('Top 10 Companies with the Most Consumer Complaints')
plt.xlabel('Number of Complaints')
plt.ylabel('Company')
plt.show()

C:\Users\GANESH\AppData\Local\Temp\ipykernel_14500\2103550010.py:10:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=company_counts.values[:10],
y=company_counts.index[:10], palette='viridis')
```



2. Given an unsorted array of integers, find the length of the longest continuous increasing subsequence (subarray)

```
def findLengthOfLCIS(nums):
    if not nums:  # If the input list is empty
        return 0

    max_length = 1  # Initialize the maximum length to 1
    current_length = 1  # Initialize the length of the current
increasing subsequence to 1

    for i in range(1, len(nums)):
        if nums[i] > nums[i - 1]:  # If the current number is greater
than the previous one
            current_length += 1  # Increment the length of the current
increasing subsequence
            max_length = max(max_length, current_length)  # Update the
maximum length if needed
        else:
            current_length = 1  # Reset the length of the current
increasing subsequence

    return max_length

# Test cases
print(findLengthOfLCIS([1, 3, 5, 4, 7]))  # Output: 3
print(findLengthOfLCIS([2, 2, 2, 2, 2]))  # Output: 1

li = list(map(int,input().split())) # 20 10 30 ----> 2
print(findLengthOfLCIS(li))
```

```
3
1
2
```

3. Given a list of non negative integers, arrange them such that they form the largest number.

```
def largestNumber(nums):

    # Convert integers to strings for comparison
    num = list(map(str, nums))

    # Sort the numbers using the custom comparison function
    num.sort(key=lambda x: (x[0], x[1 % len(x)]), reverse=True)
    # Sort the numbers based on the comparison function. We use a
lambda function here
    # to extract the first digit of each number and then alternate
between the digits
    # if the lengths of the numbers are different. This effectively
achieves the same
    # sorting behavior as the previous custom comparison function.
    # 'reverse=True' is used to sort in descending order.

    # Join the sorted numbers into a single string
    largest_num = ''.join(num)

    # Remove leading zeros if any
    largest_num = largest_num.lstrip('0')

    # If all numbers were zeros, return '0'
    return largest_num if largest_num else '0'

# Test cases
print(largestNumber([10, 2]))  # Output: "210"
print(largestNumber([3, 30, 34, 5, 9]))  # Output: "9534330"

li = list(map(int,input().split())) # 10 20 30 ----->302010
print(largestNumber(li))
```

```
210
9534330
302010
```

4. Store all the "servlet-name", and "servlet-class" to a csv file from the attached sample_json.json file using Python.

First Method

```
import pandas as pd # importing pandas library to read json file
df = pd.read_json("sample_json.json").T # transposing result of json
to access servlet key and its elements
df
```

```
                                                      servlet  \
web-app  [{'servlet-name': 'cofaxCDS', 'servlet-class':...

                                                servlet-mapping  \
web-app  {'cofaxCDS': '/', 'cofaxEmail': '/cofaxutil/ae...

                                                       taglib
web-app  {'taglib-uri': 'cofax.tld', 'taglib-location':...
```

```
dic = {"servlet-name":[],"servlet-class":[]} # creating dictionary to
append the servlet name , servlet class makes easier without passing
column names to the dataframe
for i in df["servlet"]: # looping over servlet
  for j in i: # in-order to access nested documents
    dic["servlet-name"].append(j["servlet-name"]) #
    dic["servlet-class"].append(j["servlet-class"])

res = pd.DataFrame(dic)
res
```

```
  servlet-name                    servlet-class
0     cofaxCDS         org.cofax.cds.CDSServlet
1   cofaxEmail       org.cofax.cds.EmailServlet
2   cofaxAdmin       org.cofax.cds.AdminServlet
3  fileServlet        org.cofax.cds.FileServlet
4   cofaxTools  org.cofax.cms.CofaxToolsServlet
```

```
res.to_csv("servlet_data.csv",index=False)
```

Second method

```
import json
with open("sample_json.json","r") as file: # to read JSON file without
raising any exception
  R = json.load(file)

dic1 = {"servlet-name":[],"servlet-class":[]}
i = 0
while i < len(R["web-app"]["servlet"]): # here we are iterating loop
```

```
upto length of servlet document
  dic1["servlet-name"].append(R["web-app"]["servlet"][i]["servlet-
name"]) # here we are accessing embedded document of servlet-name
using key traversing
  dic1["servlet-class"].append(R["web-app"]["servlet"][i]["servlet-
class"]) # here we are accessing embedded document of servlet-class
using key traversing
  i += 1
else:
  print("list out of bound")
```

```
list out of bound
```

```
df1 = pd.DataFrame(dic1)
df1
```

	servlet-name	servlet-class
0	cofaxCDS	org.cofax.cds.CDSServlet
1	cofaxEmail	org.cofax.cds.EmailServlet
2	cofaxAdmin	org.cofax.cds.AdminServlet
3	fileServlet	org.cofax.cds.FileServlet
4	cofaxTools	org.cofax.cms.CofaxToolsServlet