In [3]:
```
pip install textblob
```

```
Requirement already satisfied: textblob in c:\users\kaurs\anaconda3\lib\site-pack
ages (0.18.0.post0)Note: you may need to restart the kernel to use updated packag
es.

Requirement already satisfied: nltk>=3.8 in c:\users\kaurs\anaconda3\lib\site-pac
kages (from textblob) (3.8.1)
Requirement already satisfied: click in c:\users\kaurs\anaconda3\lib\site-package
s (from nltk>=3.8->textblob) (8.1.7)
Requirement already satisfied: joblib in c:\users\kaurs\anaconda3\lib\site-packag
es (from nltk>=3.8->textblob) (1.2.0)
Requirement already satisfied: regex>=2021.8.3 in c:\users\kaurs\anaconda3\lib\si
te-packages (from nltk>=3.8->textblob) (2023.10.3)
Requirement already satisfied: tqdm in c:\users\kaurs\anaconda3\lib\site-packages
(from nltk>=3.8->textblob) (4.65.0)
Requirement already satisfied: colorama in c:\users\kaurs\anaconda3\lib\site-pack
ages (from click->nltk>=3.8->textblob) (0.4.6)
```

In [4]:
```
pip install wordcloud
```

```
Requirement already satisfied: wordcloud in c:\users\kaurs\anaconda3\lib\site-pac
kages (1.9.3)
Requirement already satisfied: numpy>=1.6.1 in c:\users\kaurs\anaconda3\lib\site-
packages (from wordcloud) (1.26.4)
Requirement already satisfied: pillow in c:\users\kaurs\anaconda3\lib\site-packag
es (from wordcloud) (10.2.0)
Requirement already satisfied: matplotlib in c:\users\kaurs\anaconda3\lib\site-pa
ckages (from wordcloud) (3.8.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\kaurs\anaconda3\lib\s
ite-packages (from matplotlib->wordcloud) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\kaurs\anaconda3\lib\site-
packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\kaurs\anaconda3\lib
\site-packages (from matplotlib->wordcloud) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\kaurs\anaconda3\lib
\site-packages (from matplotlib->wordcloud) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\kaurs\anaconda3\lib\si
te-packages (from matplotlib->wordcloud) (23.1)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\kaurs\anaconda3\lib\s
ite-packages (from matplotlib->wordcloud) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\kaurs\anaconda3\l
ib\site-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\kaurs\anaconda3\lib\site-pack
ages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [5]:
```python
#importing necessary modules
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from textblob import TextBlob
from wordcloud import WordCloud
import os
```

In [6]:
```python
#defining the directory where our SEC dataset files are stored
sec_data ="C:/Users/kaurs/Downloads/2024q1"
```

In [7]:
```
#Load files
num_file = os.path.join(sec_data,'num.txt')
sub_file = os.path.join(sec_data,'sub.txt')
tag_file = os.path.join(sec_data, 'tag.txt')
```

In [8]:
```
#Load Data into DataFrames
df_num = pd.read_csv(num_file,sep='\t')
df_sub = pd.read_csv(sub_file,sep='\t')
df_tag = pd.read_csv(tag_file,sep='\t')
```

In [59]:
```
print("Unique tags in df_num",df_num['tag'].unique())
```

```
Unique tags in df_num ['AccountsPayableCurrent' 'AdditionalPaidInCapital'
 'AdjustmentsToAdditionalPaidInCapitalSharebasedCompensationRequisiteServicePerio
dRecognitionValue'
 ... 'StockIssuanceCostsNonCashActivity'
 'StockIssuedDuringPeriodWarrantsNewIssuesValue'
 'UnrealizedForeignCurrencyTransactionLossBeforeTax']
```

In [10]:
```
#EDA on NUM files
#calculating Summary statistics
print("Summary Statistics for num file")
print(df_num.describe())
```

```
Summary Statistics for num file
              ddate          qtrs         value
count  3.053505e+06  3.053505e+06  3.000013e+06
mean   2.022423e+07  2.165653e+00  3.511402e+11
std    1.613759e+04  1.993730e+00  3.352469e+14
min    1.985123e+07  0.000000e+00 -4.261655e+13
25%    2.022123e+07  0.000000e+00  1.015000e+01
50%    2.022123e+07  3.000000e+00  3.273000e+06
75%    2.023123e+07  4.000000e+00  6.084100e+07
max    2.923123e+07  1.200000e+02  4.244000e+17
```

In [11]:
```
df_num['ddate']=pd.to_datetime(df_num['ddate'],format='%Y%m%d', errors ='coerce'
```

In [12]:
```
print("Number of NaT values in ddate column", df_num['ddate'].isnull().sum())
```

```
Number of NaT values in ddate column 7
```

In [54]:
```
#df_num= df_num.dropna(subset=['ddate'])
print(df_num.columns)
```

```
Index(['adsh', 'tag', 'version', 'coreg', 'ddate', 'qtrs', 'uom', 'value',
       'footnote'],
      dtype='object')
```

In [63]:
```
if 'AccountsPayableCurrent' in df_num['tag'].unique():
    df_revenue = df_num[df_num['tag'] == 'AccountsPayableCurrent'].copy()
    df_revenue['revenue'] = df_revenue['value']
    df_revenue = df_revenue[['ddate', 'revenue']]
    print("Revenue:")
    print(df_revenue)

if 'AdditionalPaidInCapital' in df_num['tag'].unique() and 'AdjustmentsToAdditio
    df_net_income = df_num[df_num['tag'].isin(['AdditionalPaidInCapital', 'Adjus
    df_net_income['net_income'] = df_net_income.groupby('ddate')['value'].transf
    df_net_income = df_net_income[['ddate', 'net_income']]
    print("Net Income:")
    print(df_net_income)

if 'StockIssuanceCostsNonCashActivity' in df_num['tag'].unique() and 'StockIssue
    df_eps = df_num[df_num['tag'].isin(['StockIssuanceCostsNonCashActivity', 'St
    df_eps['eps'] = df_eps.groupby('ddate')['value'].transform('sum')  # Assumin
    df_eps = df_eps[['ddate', 'eps']]
    print("EPS:")
    print(df_eps)
```

```
Revenue:
             ddate    revenue
0       2023-12-31  1041000.0
1       2023-06-30  1372000.0
246     2023-12-31   339897.0
247     2023-06-30  1005059.0
1479    2023-11-30   317000.0
...            ...        ...
3051357 2022-12-31   554247.0
3051491 2023-12-31   271244.0
3051492 2022-12-31   280384.0
3051630 2023-12-31   492000.0
3051631 2022-12-31   513000.0

[7057 rows x 2 columns]
Net Income:
             ddate    net_income
2       2023-12-31  4.808975e+12
3       2023-06-30  1.507344e+11
4       2022-09-30  5.371518e+09
5       2022-12-31  4.317727e+12
6       2023-09-30  1.000722e+11
...            ...           ...
3050476 2023-12-31  4.808975e+12
3050609 2023-12-31  4.808975e+12
3050610 2022-12-31  4.317727e+12
3052867 2022-12-31  4.317727e+12
3052868 2023-12-31  4.808975e+12

[14889 rows x 2 columns]
EPS:
             ddate         eps
3053478 2023-12-31     -2000.0
3053479 2022-12-31  93027000.0
3053480 2023-12-31     -2000.0
3053481 2022-12-31  93027000.0
```

In [97]:

```
df_revenue_filtered = df_revenue[df_revenue['ddate'].dt.year == 2023]
df_net_income_filtered = df_net_income[df_net_income['ddate'].dt.year == 2023]
# Merge filtered data
df_profit_margin = pd.merge(df_revenue_filtered, df_net_income_filtered, on='dda
df_profit_margin['profit_margin'] = (df_profit_margin['net_income'] / df_profit_
```

In [85]:

```
df_equity_filtered = df_num[(df_num['tag'] == 'Equity') & (df_num['ddate'].dt.ye
df_liabilities_filtered = df_num[(df_num['tag'] == 'liabilities') & (df_num['dda

# Compute return on equity (ROE)
```

```
df_roe = pd.merge(df_net_income_filtered, df_equity_filtered, on='ddate', how='i
df_roe['roe'] = (df_roe['net_income'] / df_roe['value']) * 100
```

```
Debt- to - Equity Ratio DataFrame:
Empty DataFrame
Columns: [adsh_x, tag_x, version_x, coreg_x, qtrs_x, uom_x, value_x, footnote_x,
adsh_y, tag_y, version_y, coreg_y, ddate, qtrs_y, uom_y, value_y, footnote_y, deb
t_equity_ratio]
Index: []
```

In [86]:

```
# Check unique dates in df_net_income_filtered
print("Unique Dates in df_net_income_filtered:")
print(df_net_income_filtered['ddate'].unique())

# Check unique dates in df_equity_filtered
print("\nUnique Dates in df_equity_filtered:")
print(df_equity_filtered['ddate'].unique())

# Print filtering criteria for df_equity_filtered
print("Filtering Criteria for df_equity_filtered:")
print(df_equity_filtered.head())

# Check if df_num contains data for equity
print("\nUnique Tags in df_num:")
print(df_num['tag'].unique())
```

```
Unique Dates in df_net_income_filtered:
<DatetimeArray>
['2023-12-31 00:00:00', '2023-06-30 00:00:00', '2023-09-30 00:00:00',
 '2023-03-31 00:00:00', '2023-11-30 00:00:00', '2023-02-28 00:00:00',
 '2023-05-31 00:00:00', '2023-08-31 00:00:00', '2023-07-31 00:00:00',
 '2023-01-31 00:00:00', '2023-04-30 00:00:00', '2023-10-31 00:00:00']
Length: 12, dtype: datetime64[ns]

Unique Dates in df_equity_filtered:
<DatetimeArray>
['2023-09-30 00:00:00', '2023-12-31 00:00:00', '2023-03-31 00:00:00',
 '2023-06-30 00:00:00', '2023-11-30 00:00:00', '2023-08-31 00:00:00',
 '2023-01-31 00:00:00', '2023-10-31 00:00:00', '2023-04-30 00:00:00',
 '2023-07-31 00:00:00']
Length: 10, dtype: datetime64[ns]
Filtering Criteria for df_equity_filtered:
                       adsh     tag    version coreg      ddate  qtrs  uom  \
9771   0001213900-24-013460  Equity  ifrs/2023   NaN 2023-09-30     0  CAD   
31693  0001213900-24-022367  Equity  ifrs/2023   NaN 2023-12-31     0  USD   
61275  0001193125-24-067358  Equity  ifrs/2023   NaN 2023-12-31     0  GBP   
63738  0001213900-24-012567  Equity  ifrs/2023   NaN 2023-03-31     0  USD   
70801  0001178913-24-000941  Equity  ifrs/2023   NaN 2023-12-31     0  USD   

              value footnote  
9771  -2.094224e+06      NaN  
31693  1.318100e+07      NaN  
61275  3.988000e+09      NaN  
63738  8.596131e+06      NaN  
70801  6.037000e+09      NaN  

Unique Tags in df_num:
['AccountsPayableCurrent' 'AdditionalPaidInCapital'
 'AdjustmentsToAdditionalPaidInCapitalSharebasedCompensationRequisiteServicePerio
dRecognitionValue'
 ... 'StockIssuanceCostsNonCashActivity'
 'StockIssuedDuringPeriodWarrantsNewIssuesValue'
 'UnrealizedForeignCurrencyTransactionLossBeforeTax']
```

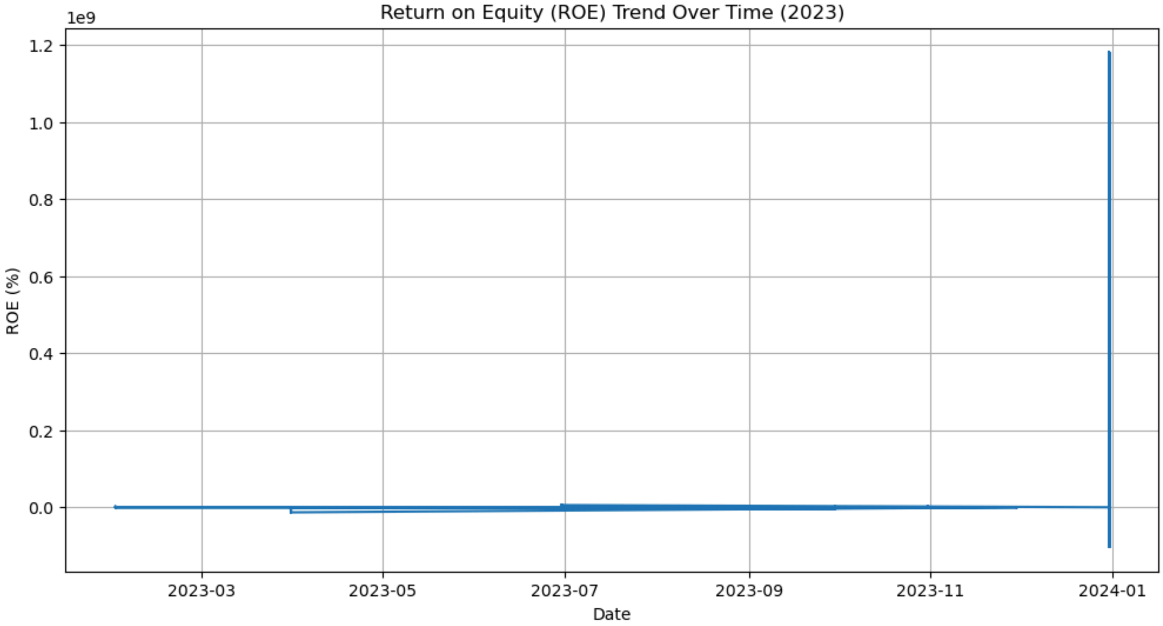
In [80]:

```python
df_roe = pd.merge(df_net_income_filtered, df_equity_filtered, on='ddate', how='i

# Print the first few rows of the merged dataframe
print("Merged DataFrame:")
print(df_roe.head())

# Check the column names of the merged dataframe
print("\nColumn Names:")
print(df_roe.columns)
```

```
Merged DataFrame:
Empty DataFrame
Columns: [ddate, net_income, adsh, tag, version, coreg, qtrs, uom, value, footnot
e]
Index: []

Column Names:
Index(['ddate', 'net_income', 'adsh', 'tag', 'version', 'coreg', 'qtrs', 'uom',
       'value', 'footnote'],
      dtype='object')
```

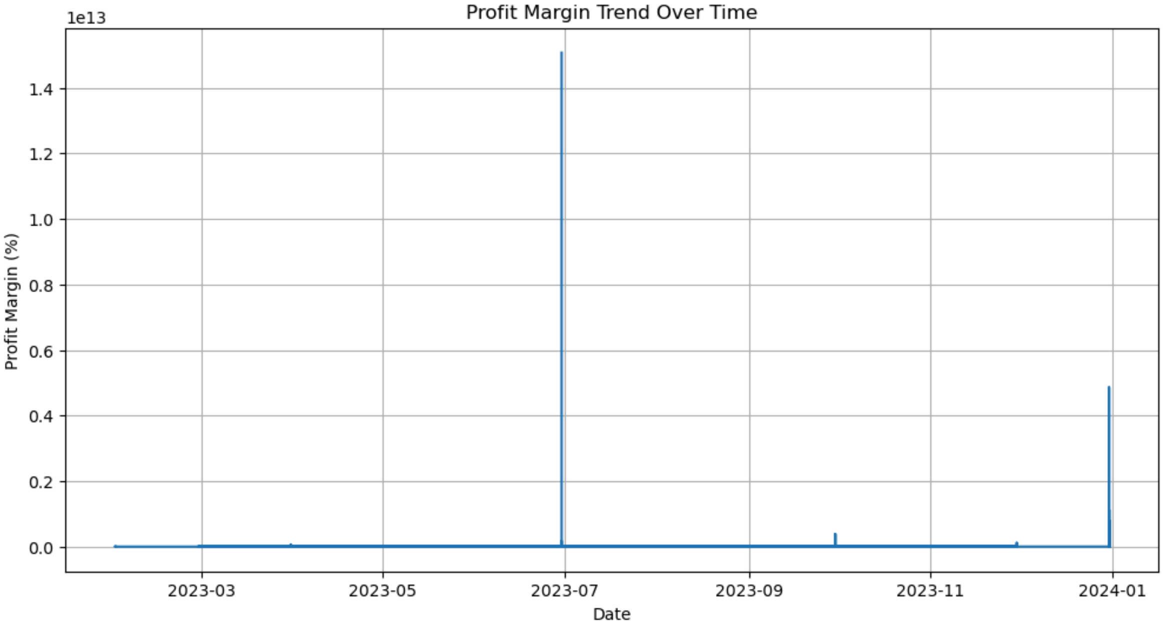
In [87]:

```
# Visualize ROE trend
plt.figure(figsize=(12, 6))
plt.plot(df_roe['ddate'], df_roe['roe'])
plt.title('Return on Equity (ROE) Trend Over Time (2023)')
plt.xlabel('Date')
plt.ylabel('ROE (%)')
plt.grid(True)
plt.show()
```



In [96]:

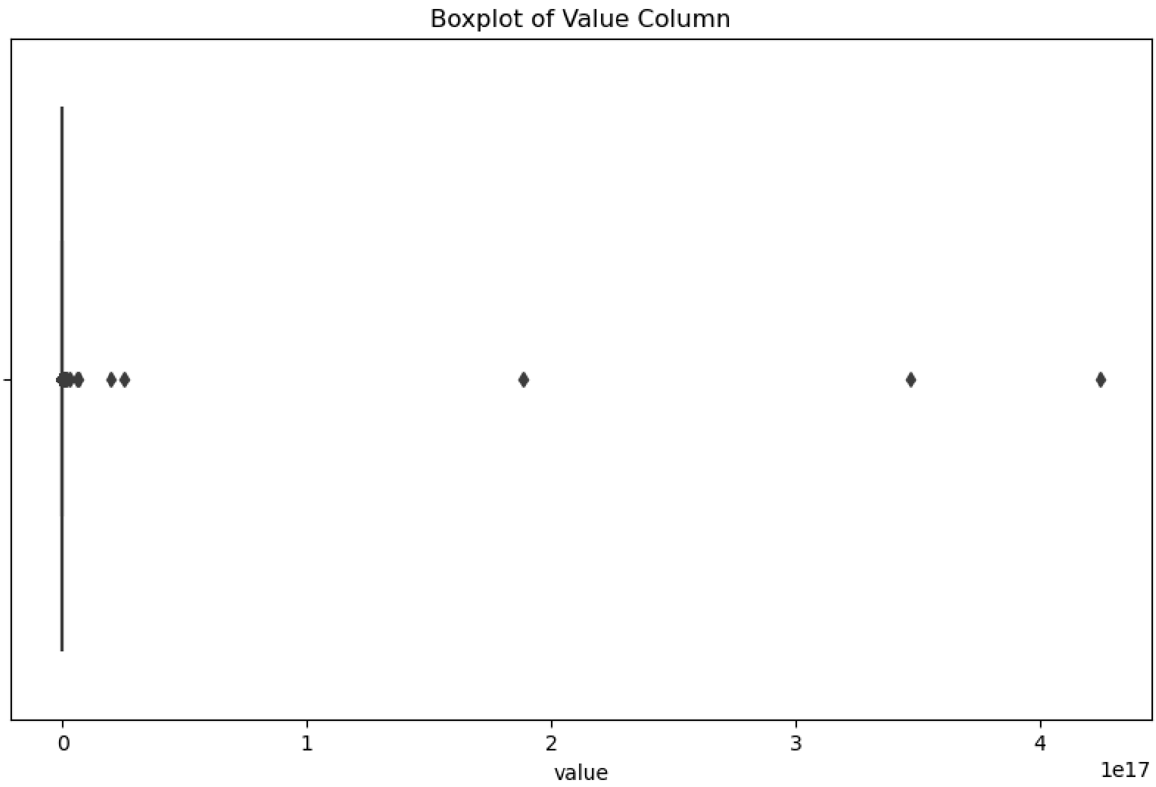
```
plt.figure(figsize=(12, 6))
plt.plot(df_profit_margin['ddate'], df_profit_margin['profit_margin'])
plt.title('Profit Margin Trend Over Time')
plt.xlabel('Date')
plt.ylabel('Profit Margin (%)')
plt.grid(True)
plt.show()
```



In [14]:

```
plt.figure(figsize=(10,6))
sns.boxplot(x=df_num['value'])
```

```
plt.title('Boxplot of Value Column')
plt.show()
```



In [ ]:

In [ ]:

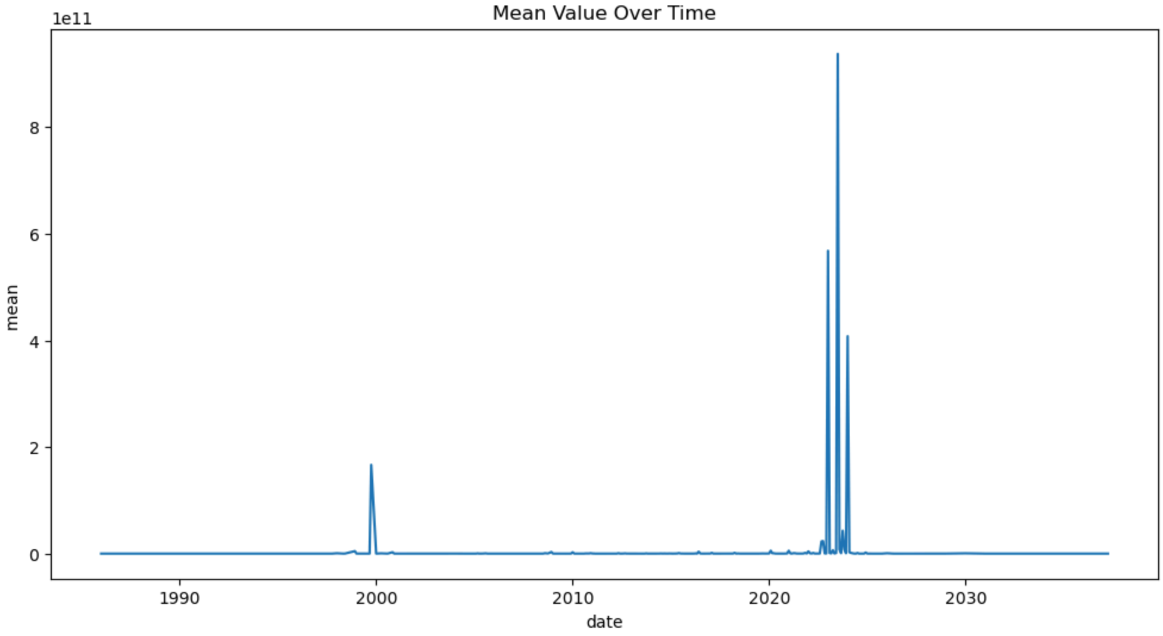
In [15]:

```
Q1 = df_num['value'].quantile(0.25)
Q3 = df_num['value'].quantile(0.75)
IQR = Q3- Q1
lower_bound = Q1 -1.5*IQR
upper_bound = Q3-1.5*IQR
outliers = df_num[(df_num['value']<lower_bound)| (df_num['value']>upper_bound)]
print("Outliers in the value column")
print(outliers)
```

```
Outliers in the value column
                         adsh  \
0        0000897101-24-000070   
1        0000897101-24-000070   
2        0000897101-24-000070   
3        0000897101-24-000070   
4        0000897101-24-000070   
...                       ...   
3053500  0001739445-24-000051   
3053501  0001739445-24-000051   
3053502  0001739445-24-000051   
3053503  0001739445-24-000051   
3053504  0001739445-24-000051   

                                                       tag       version  \
0                                   AccountsPayableCurrent  us-gaap/2023   
1                                   AccountsPayableCurrent  us-gaap/2023   
2                                  AdditionalPaidInCapital  us-gaap/2023   
3                                  AdditionalPaidInCapital  us-gaap/2023   
4        AdjustmentsToAdditionalPaidInCapitalSharebased...  us-gaap/2023   
...                                                    ...           ...   
3053500                                    PeoTotalCompAmt      ecd/2023   
3053501                             TotalShareholderRtnAmt      ecd/2023   
3053502                             TotalShareholderRtnAmt      ecd/2023   
3053503                             TotalShareholderRtnAmt      ecd/2023   
3053504                             TotalShareholderRtnAmt      ecd/2023   

        coreg       ddate  qtrs  uom       value footnote  
0          NaN 2023-12-31     0  USD   1041000.0      NaN  
1          NaN 2023-06-30     0  USD   1372000.0      NaN  
2          NaN 2023-12-31     0  USD  19634000.0      NaN  
3          NaN 2023-06-30     0  USD  18788000.0      NaN  
4          NaN 2022-09-30     1  USD     95000.0      NaN  
...        ...        ...   ...  ...         ...      ...  
3053500    NaN 2023-12-31     4  USD   6474120.0      NaN  
3053501    NaN 2023-12-31     4  USD       188.0      NaN  
3053502    NaN 2022-12-31     4  USD       123.0      NaN  
3053503    NaN 2021-12-31     4  USD       119.0      NaN  
3053504    NaN 2020-12-31     4  USD       124.0      NaN  

[2960469 rows x 9 columns]
```

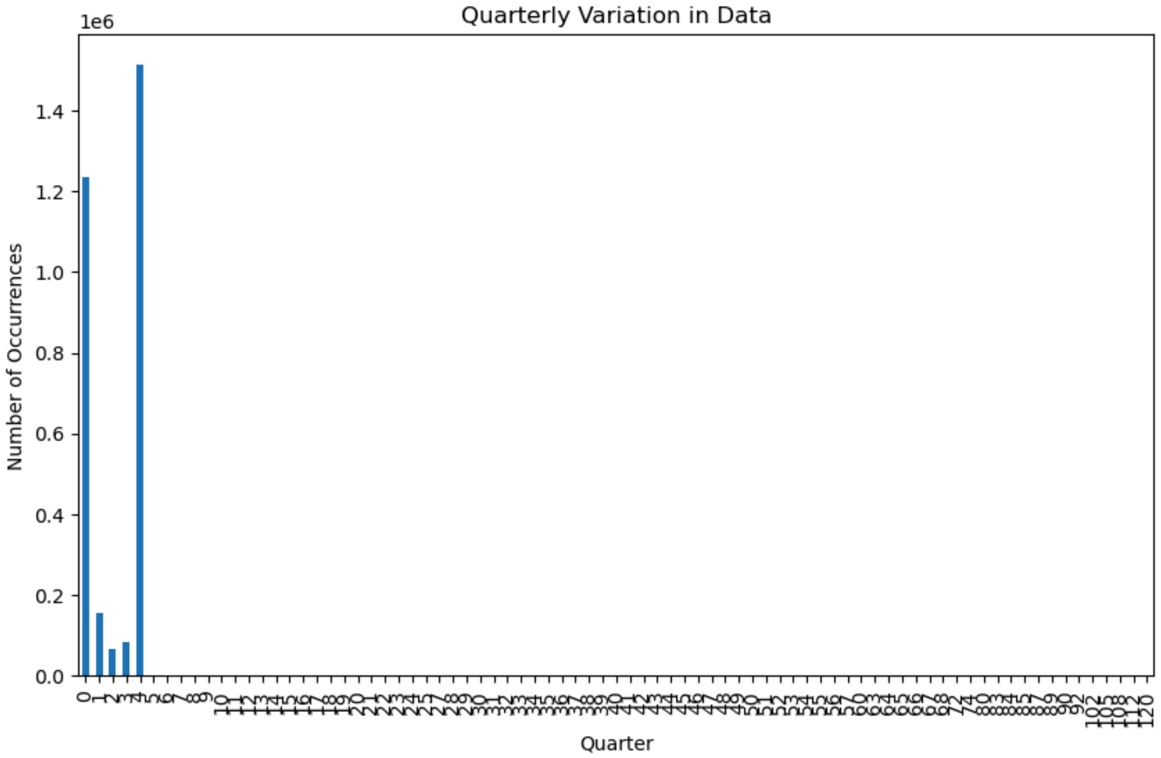
In [16]:
```
date_grouped = df_num.groupby(df_num['ddate']).agg({'value':'mean'})
```

In [17]:
```
plt.figure(figsize=(12,6))
plt.plot(date_grouped.index, date_grouped['value'])
plt.title('Mean Value Over Time')
plt.xlabel('date')
plt.ylabel('mean ')
plt.show() #Analyzing the Mean value over time Time Series Plot
```



In [18]:

```
quarter_counts = df_num['qtrs'].value_counts().sort_index()
plt.figure(figsize=(10,6))
quarter_counts.plot(kind= 'bar')
plt.title('Quarterly Variation in Data')
plt.xlabel('Quarter')
plt.ylabel('Number of Occurrences')
plt.show()
```



In [19]:

```
quarterly_stats= df_num.groupby('qtrs')['value'].agg(['mean','median'])
print("Mean and Median Value for Each Quarter")
print(quarterly_stats)
```

```
Mean and Median Value for Each Quarter
                mean        median
qtrs
0     4.690904e+10  1.282200e+07
1     4.472078e+08  2.254300e+05
2     1.427774e+08  1.982820e+05
3     2.210613e+10  1.800000e+05
4     6.658532e+11  1.350000e+06
...            ...           ...
102   1.195650e+09  1.195650e+09
105   1.188490e+10  1.547000e+08
108   0.000000e+00  0.000000e+00
112   1.250000e-02  1.250000e-02
120   1.496440e+08  1.496440e+08

[79 rows x 2 columns]
```

In [105…

```
df_sub['submission_date']= pd.to_datetime(df_sub['filed'],format='%Y%m%d')
```

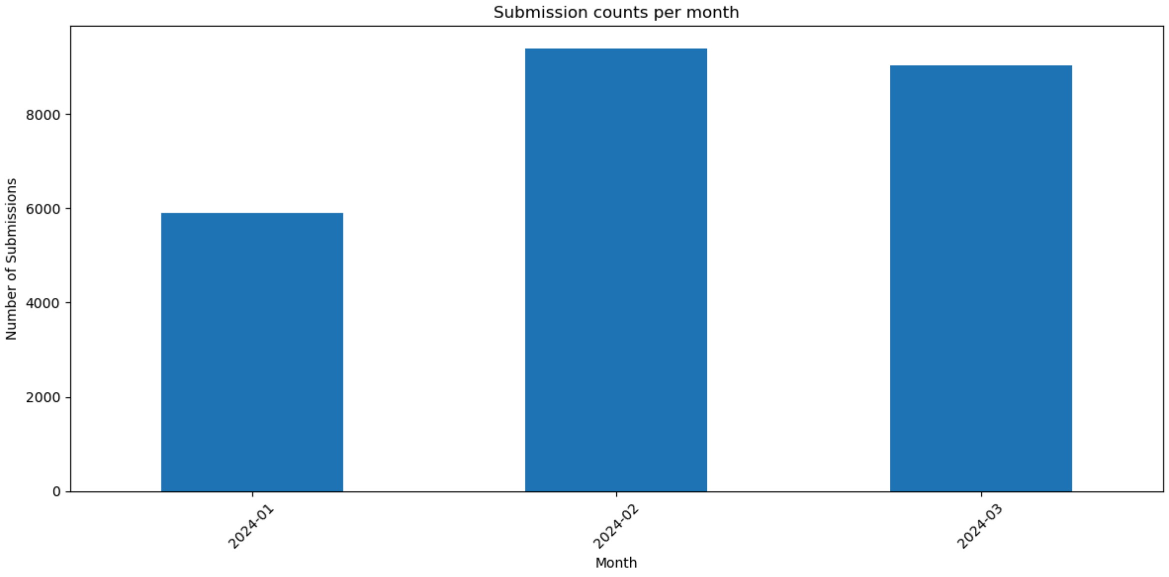
In [103…

```
# eda on sub files
submission_counts = df_sub['submission_date'].dt.to_period('M').value_counts().s
print(submission_counts)
```

```
submission_date
2024-01    5904
2024-02    9398
2024-03    9031
Freq: M, Name: count, dtype: int64
```

In [104…

```
submission_counts.plot(kind ='bar', figsize =(12,6))
plt.title('Submission counts per month')
plt.xlabel('Month')
plt.ylabel('Number of Submissions')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



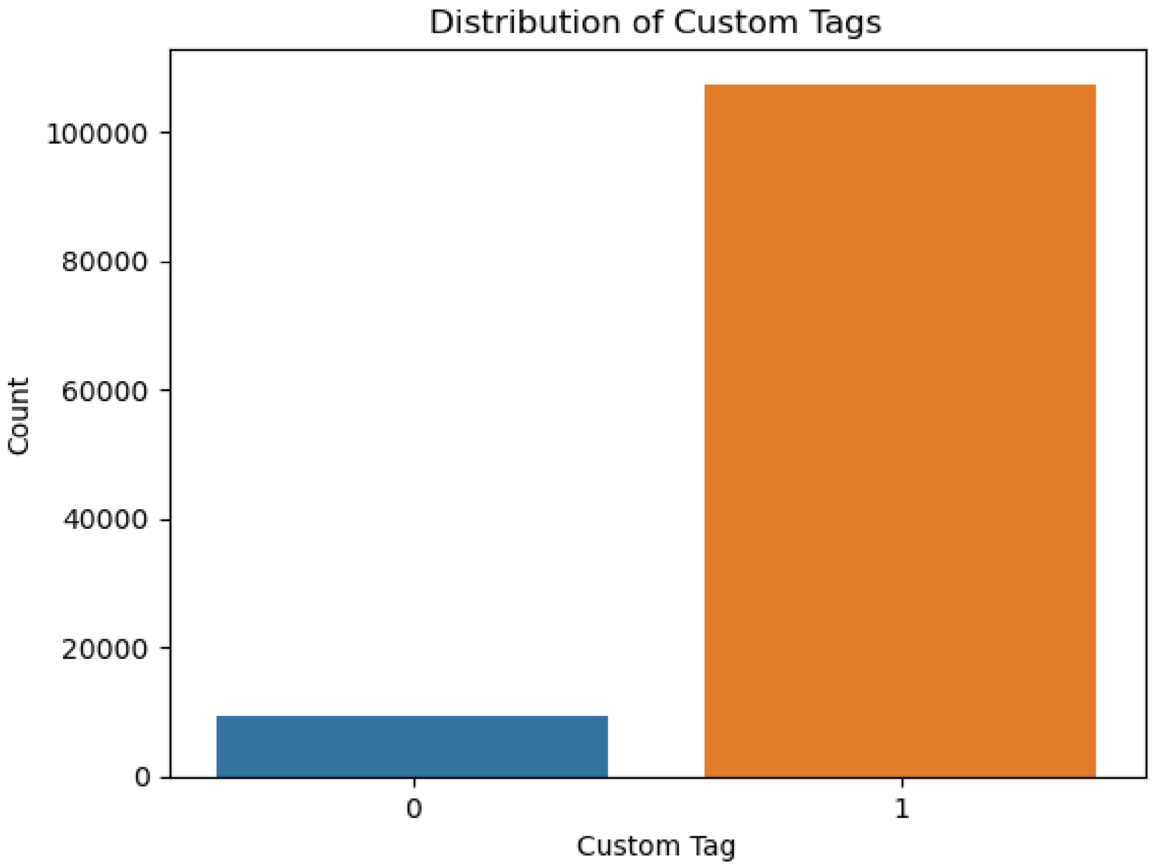
In [32]:

```
#EDA on Tag Files
print("Summary of Tagged Data:")
print(df_tag.describe())
```

```
Summary of Tagged Data:
              custom       abstract
count  116771.000000  116771.000000
mean        0.920554       0.237448
std         0.270435       0.425521
min         0.000000       0.000000
25%         1.000000       0.000000
50%         1.000000       0.000000
75%         1.000000       0.000000
max         1.000000       1.000000
```

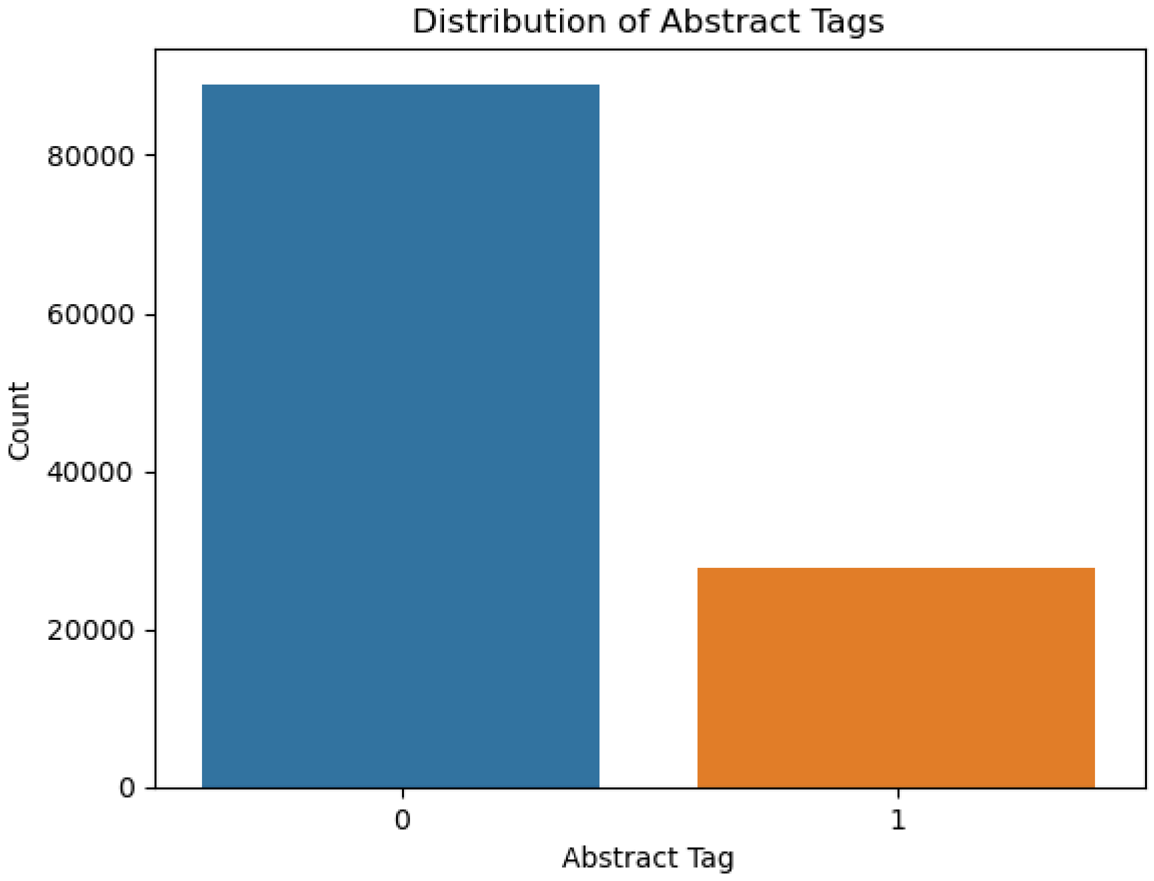
In [33]:

```
sns.countplot(data = df_tag, x='custom')
plt.title('Distribution of Custom Tags')
plt.xlabel('Custom Tag')
plt.ylabel('Count')
plt.show()
```

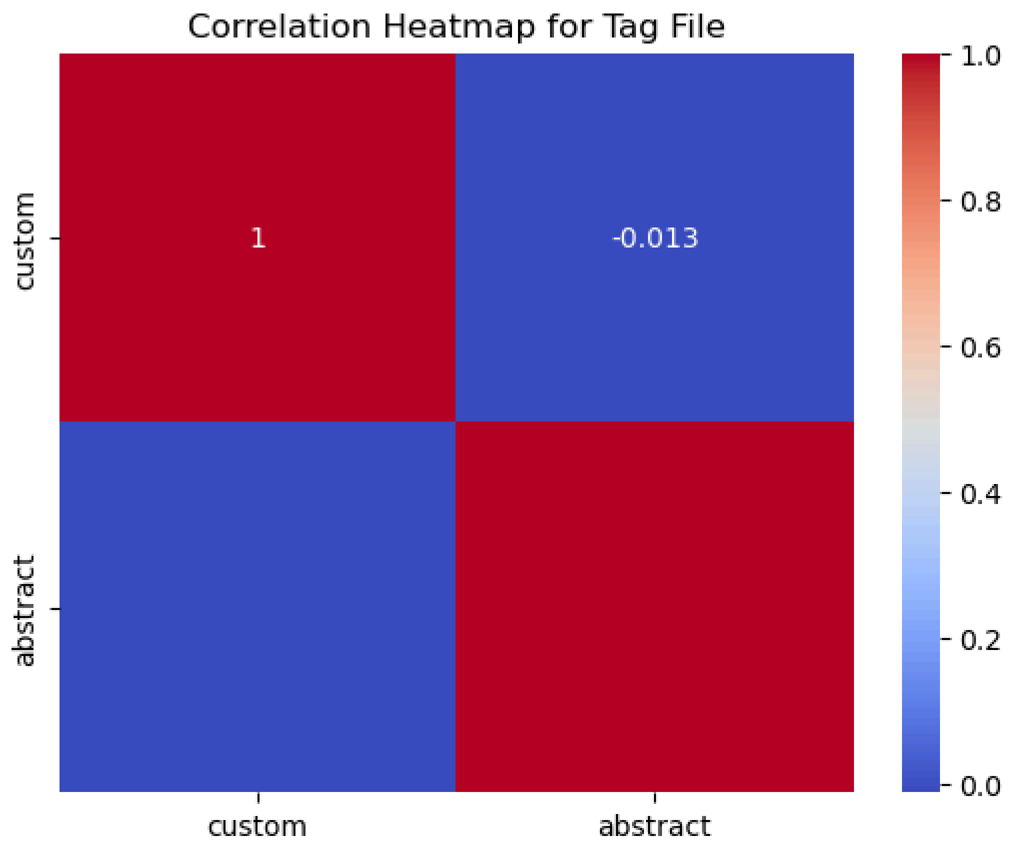


In [34]:

```
sns.countplot(data = df_tag, x='abstract')
plt.title('Distribution of Abstract Tags')
plt.xlabel('Abstract Tag')
plt.ylabel('Count')
plt.show()
```



In [43]:
```
numeric_df_tag = df_tag.select_dtypes(include =['float64','int64'])
correlation_matrix = numeric_df_tag.corr()

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap for Tag File')
plt.show()
```



In [47]:

```
def analyze_relationships(df):
    print("Available Columns in the DataFrame")
    print(df.columns)

    if 'form' in df.columns:
        filing_counts = df['form'].value_counts()
        print("\nRelationship Between Filings:")
        print(filing_counts)
    else:
        print("\n 'Form' column not found in The dataframe")
analyze_relationships(df_sub)
```

```
Available Columns in the DataFrame
Index(['adsh', 'cik', 'name', 'sic', 'countryba', 'stprba', 'cityba', 'zipba',
       'bas1', 'bas2', 'baph', 'countryma', 'stprma', 'cityma', 'zipma',
       'mas1', 'mas2', 'countryinc', 'stprinc', 'ein', 'former', 'changed',
       'afs', 'wksi', 'fye', 'form', 'period', 'fy', 'fp', 'filed', 'accepted',
       'prevrpt', 'detail', 'instance', 'nciks', 'aciks', 'submission_date'],
      dtype='object')

Relationship Between Filings:
form
8-K         16271
10-K         3986
10-Q         1044
DEF 14A       827
8-K/A         429
20-F          319
PRE 14A       238
10-K/A        164
S-1/A         132
N-CSR         112
10-Q/A        108
40-F          107
S-1            84
S-4/A          62
6-K            56
424B3          45
N-2/A          37
424B2          36
POS AM         36
N-CSRS         32
20-F/A         31
POS EX         21
S-4            21
F-1            18
F-1/A          17
N-2            11
PREC14A         8
DEFR14A         7
DEFA14A         7
POS 8C          6
PRER14A         5
10-KT           5
DEFC14A         5
8-K12B          4
POS AMI         4
S-3             4
N-CSR/A         3
DEF 14C         3
F-4             3
6-K/A           3
10-QT           3
N-CSRS/A        2
F-3             2
424B5           2
424B1           2
SP 15D2         2
N-2ASR          2
10-12G          1
8-K12B/A        1
S-11/A          1
```

```
8-K12G3         1
S-3/A           1
F-4/A           1
40-F/A          1
Name: count, dtype: int64
```

```
In [ ]:
```

```
In [ ]:
```