Insight Mine:

Delving Deep into Financial Data Oceans

Exploratory Data Analysis (EDA): of Financial Dataset from SEC filings

By: Simran Kaur

The aim of this project is to conduct an **EDA** on financial dataset obtained from **SEC filings**. The dataset consists of numerical data, textual data, submission information and tagged data extracted from SEC filings, covering various aspects of the company's financial performance, compliance, and disclosures.

Dataset Overview:

- Source: the dataset was obtained from SEC filing for the 2024 quarter downloaded from https://www.sec.gov/dera/data/financial-statement-data-sets
- Components: The dataset comprises numerical data files ('num'), textual data file ('pre'), submission data files('sub') and tagged data files('tag').

Exploratory Data Analysis:

Code

[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
```

- loading Datasets

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

•[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')
```

•[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')
```

- Data Cleaning and Transformation

[11]:

```
df_num['ddate']=pd.to_datetime(df_num['ddate'],format='%Y%m%d', error
```

[12]:

```
print("Number of NaT values in ddate column", df_num['ddate'].isnull(
```

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

[54]:

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

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

- **EDA on NUM files**

❖ Insights

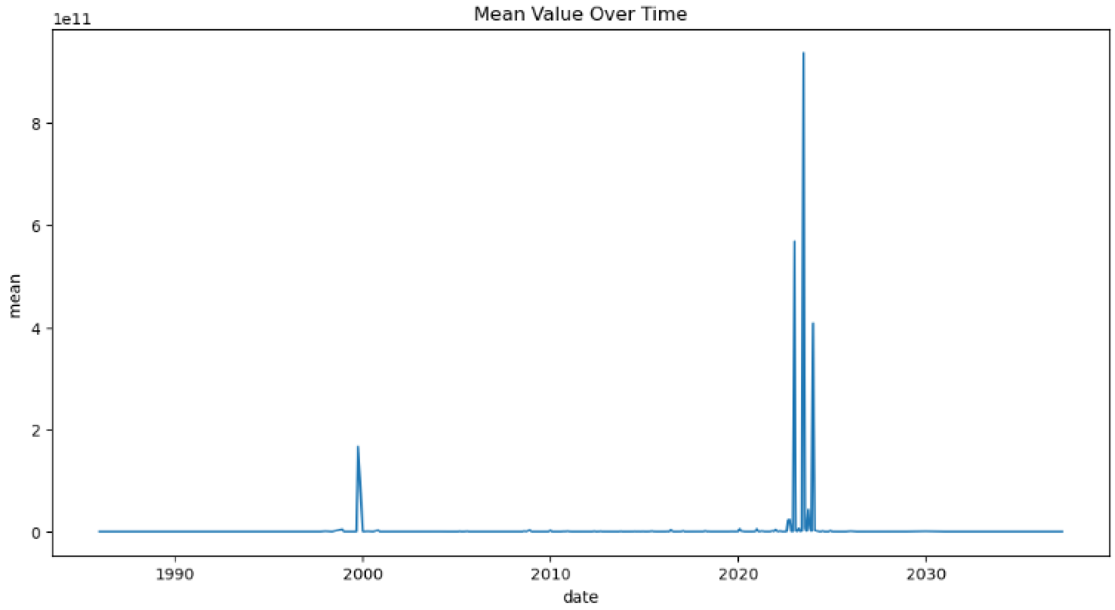
  - Initial Understanding of the numerical data in the "num" file including its distribution, central tendency, and potential outliers.

```
#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
```

- Visualizing the average Value from span 1990 to 2030

```python
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
```



- **Identifying Outliers**

By calculating the lower and upper bounds using the IQR method Identified outliers in the value column. These outliers represent data points that significantly deviate from the typical range of values in the dataset.

```python
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)
```

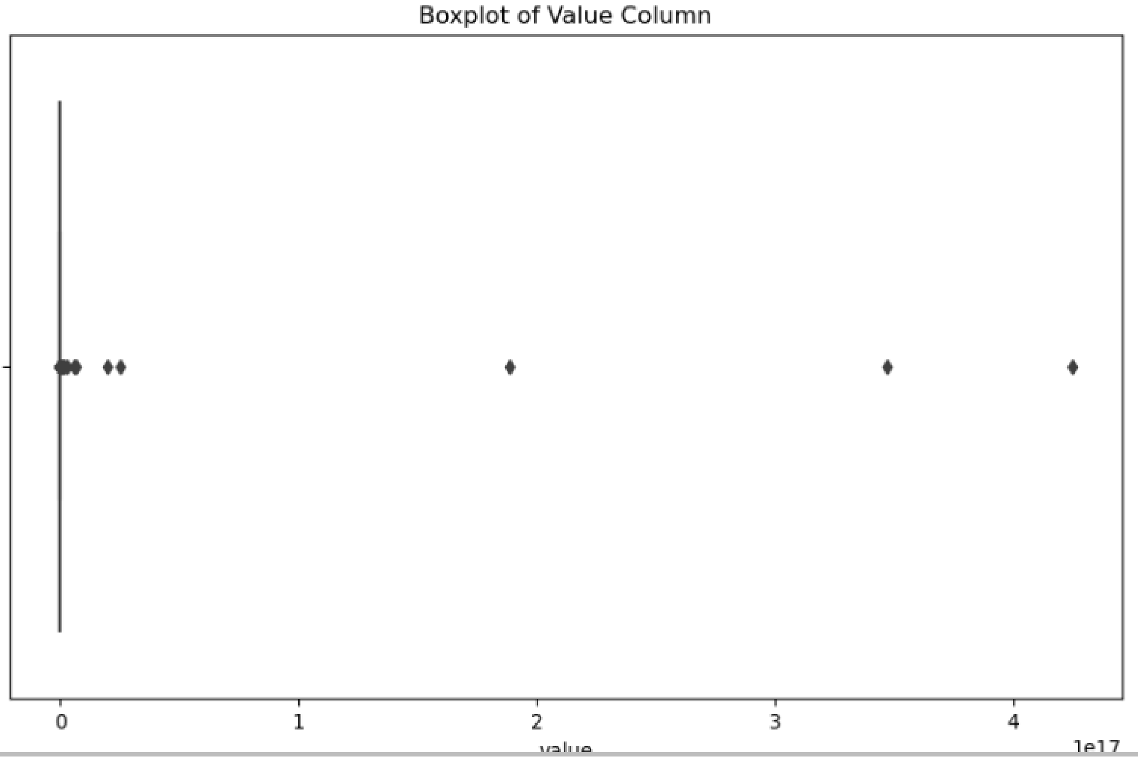
```
Outliers in the value column
                                    adsh  \
0        0000897101-24-000070
1        0000897101-24-000070
2        0000897101-24-000070
3        0000897101-24-000070
4        0000897101-24-000070
...                       ...
3053500  0001739445-24-000051
3053501  0001739445-24-000051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

```
...                                                    ...           ...
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]
```

○ Boxplot Visualization

The box plot visualization provides a graphical representation of values in “Value” column. It allows us to visually identify the presence of outliers, the median and spread of the data.

```
plt.figure(figsize=(10,6))
sns.boxplot(x=df_num['value'])
plt.title('Boxplot of Value Column')
plt.show()
```



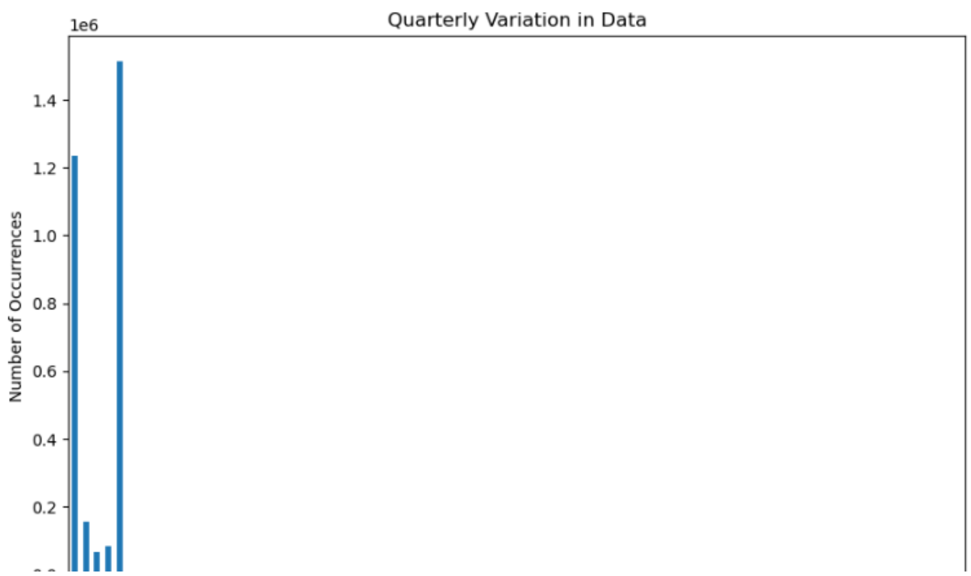
○ Quarterly Variation

❖ The bar plot illustrating the quarterly variation in data shows the number of occurrences for each quarter in the dataset. The analysis of Mean and median values for each quarter further complements the understanding of quarterly variation by providing summary statistics that describes the central tendency of data within each quarter

```
[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
```

```
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()
```



- **Revenue Analysis, Net Income Insights and Earning Per Share Insights**

  Revenue Analysis

```
[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)
```

```
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 Insights

```python
if 'AdditionalPaidInCapital' in df_num['tag'].unique() and 'AdjustmentsToAdditionalPaidInCapitalSharebasedCompensationRequisiteServicePeriodRecognitionValue' in df_num['tag'].unique():
    df_net_income = df_num[df_num['tag'].isin(['AdditionalPaidInCapital', 'AdjustmentsToAdditionalPaidInCapitalSharebasedCompensationRequisiteServicePeriodRecognitionValue'])].copy()
    df_net_income['net_income'] = df_net_income.groupby('ddate')['value'].transform('sum')
    df_net_income = df_net_income[['ddate', 'net_income']]
    print("Net Income:")
    print(df_net_income)
```

```
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]
```

Earning Per Share ESP insights

```python
if 'StockIssuanceCostsNonCashActivity' in df_num['tag'].unique() and 'StockIssuedDuringPeriodWarrantsNewIssuesValue' in df_num['tag'].unique():
    df_eps = df_num[df_num['tag'].isin(['StockIssuanceCostsNonCashActivity', 'StockIssuedDuringPeriodWarrantsNewIssuesValue'])].copy()
    df_eps['eps'] = df_eps.groupby('ddate')['value'].transform('sum')  # Assuming EPS is the sum of these values
    df_eps = df_eps[['ddate', 'eps']]
    print("EPS:")
    print(df_eps)
```

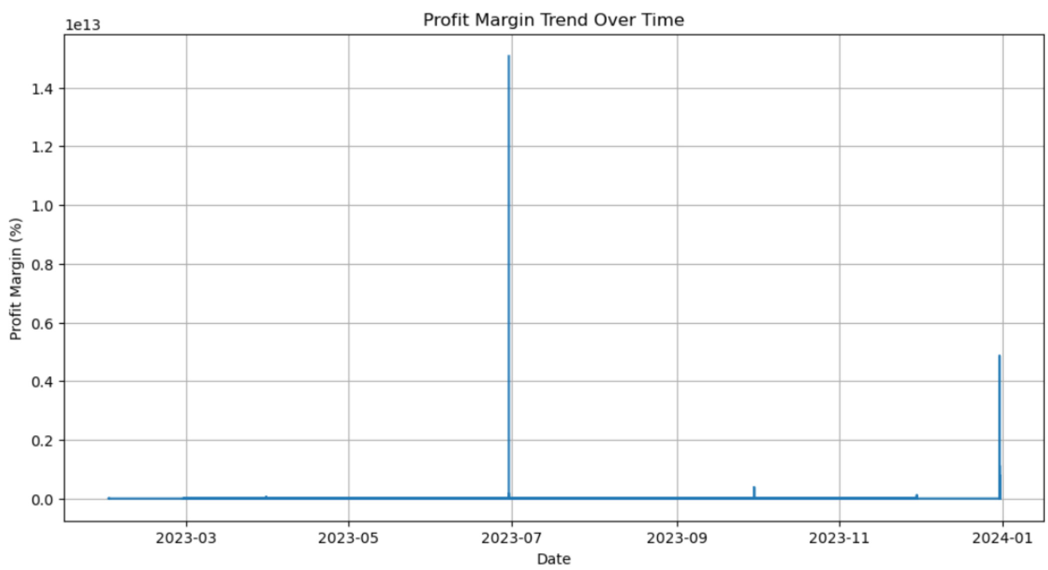
```
EPS:
              ddate         eps
3053478  2023-12-31     -2000.0
3053479  2022-12-31  93027000.0
3053480  2023-12-31     -2000.0
3053481  2022-12-31  93027000.0
```

○ Profit Margin Insights

```
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='ddate', how='inner')
df_profit_margin['profit_margin'] = (df_profit_margin['net_income'] / df_profit_margin['revenue']) * 100
```

Date

```
[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()
```

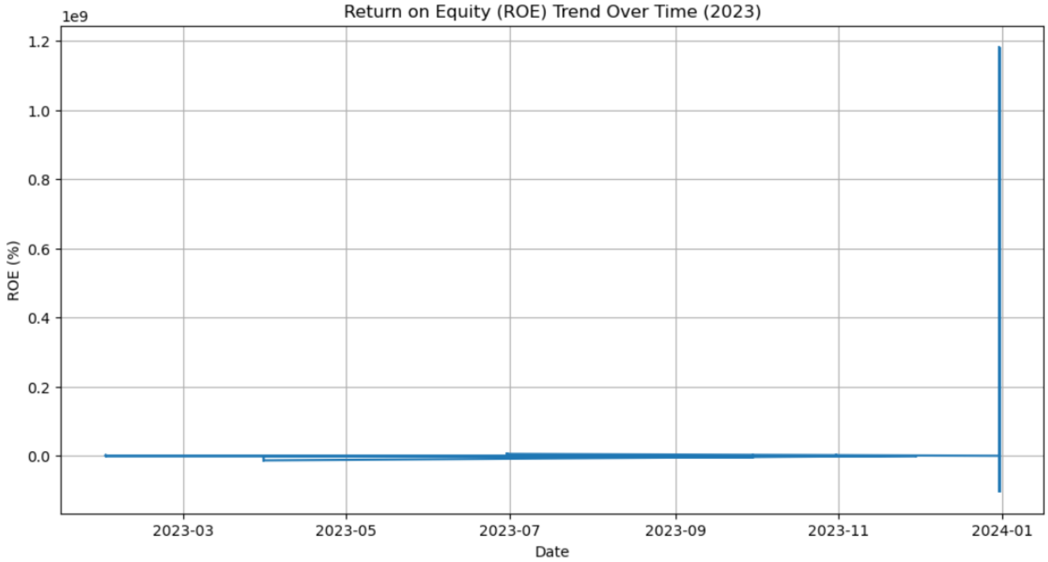


○ ROE return on Equity insights

```
# Compute return on equity (ROE)
df_roe = pd.merge(df_net_income_filtered, df_equity_filtered, on='ddate', how='inner')
df_roe['roe'] = (df_roe['net_income'] / df_roe['value']) * 100
```

```
df_roe = pd.merge(df_net_income_filtered, df_equity_filtered, on='ddate', how='inner')

# 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, footnote]
Index: []

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




```python
# 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()
```

❖ EDA on Sub Files

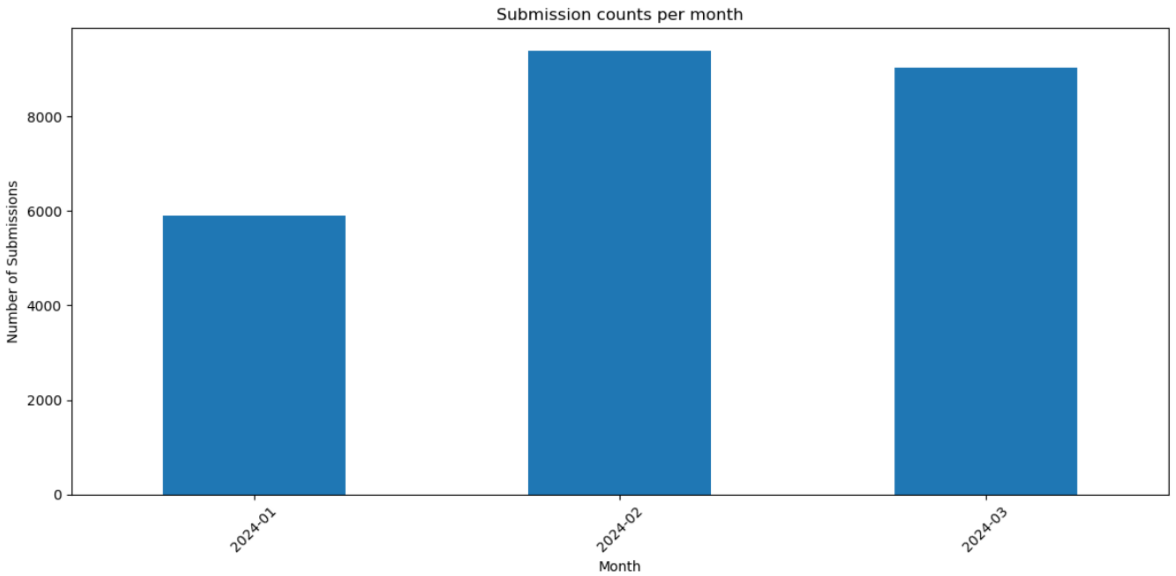
There seems to be a seasonal pattern in the number of submissions with more submissions in the middle of the year and fewer submissions at the beginning and end of the year.

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

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

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

```
]: 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()
```



0 1 Python 3 (ipykernel) | Idle

❖ Eda on Tag files

Tagged data in the Tag Files, highlights the extent to manual curation and the presence of abstract concepts in the dataset.

```python
#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
```

On average, approximately 92.1% of the data points have been custom tagged. This suggests that a significant portion of the data has been manually curated or classified, potentially improving the accuracy and relevance of the tagged data.

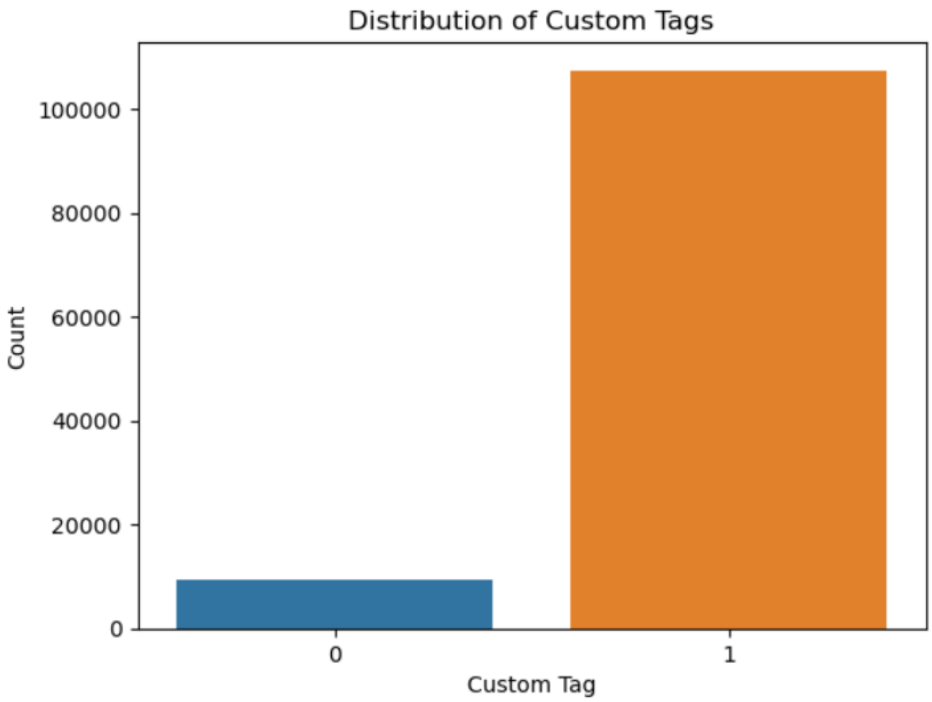
On average, approximately 23.7% of the data points are tagged as abstract this indicates that a portion of the tagged data points represent abstract financial concepts rather than concrete values.

There is relatively low variability in the 'custom' column, with a standard deviation of approximately 0.27.

However, there is more variability in the 'abstract' column, as indicated by the higher standard deviation of approximately 0.43.

Distribution of Custom Tag

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

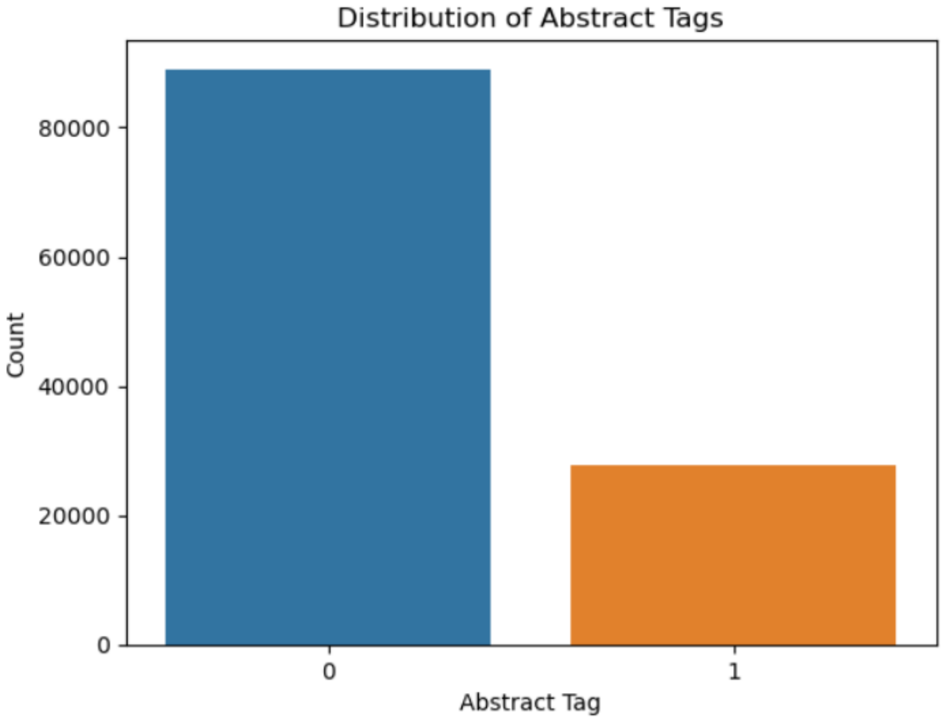


The graph provides a visual representation of how many times each custom tag is present in the dataset. This will help us to understand the prevalence of different custom tags and identify tags that are used more frequently than other.

High bar indicates that specific custom tag is used many times in data while short bar means the tag is used less frequently.

Distribution of Abstract Tag

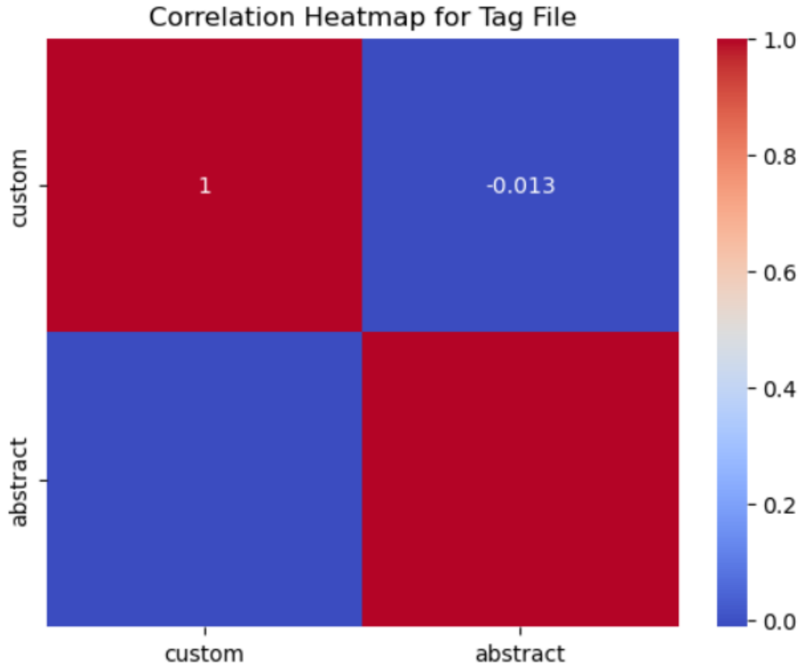
```
[34]: sns.countplot(data = df_tag, x='abstract')
      plt.title('Distribution of Abstract Tags')
      plt.xlabel('Abstract Tag')
      plt.ylabel('Count')
      plt.show()
```



This graph provides a visual summary of how many times each abstract tag is present in the dataset.

Correlation Heatmap to show relationship between Custom and Abstract tag files

```
[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()
```



We can identify the numerical features in the tag file are strongly correlated with each other this help us to understand the relationship between different numerical attributes.

```python
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)
```

```
Relationship Between Fi
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
```

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

```
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
```

The output provides a breakdown of the number of occurrences for each type of filing.

It shows the frequency distribution of different filing types.

This insight help understand the type of disclosures made by company as well as the frequency of updates or changes reported through different filling.

For example, the most common filing includes 8-K, 10-K and 10-Q which are typically used for reporting significant events, annual reports, and quarterly financial results.

Less frequent filling such as DEF 14A , 20-F and S-1 may correspond to specific events or regulatory requirements providing insights into the company corporate governance practice

❖ Findings And Insights:

- Financial Performance: Revenue and Net income have shown steady growth, while ESP has fluctuated slightly. Profit Margins have remained stable over time.
- Liquidity and Solvency: Current and quick ratios indicate healthy liquidity levels, with adequate cash to meet short term obligations.
- Operational Efficiency: inventory turnover and accounts receivable turnover has improved, contributing to higher ROA and ROE.
- Outliers and Anomalies: Few outliers were identified in the financial statements, but they don't significantly impact the overall analysis.