In [1]:
```
from bs4 import BeautifulSoup
from urllib.request import urlopen
from datetime import datetime
from collections import Counter
from collections import defaultdict
from matplotlib.dates import date2num, num2date

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import requests
import seaborn as sns
```

Obtaining All Names (gg)

In [2]:
```
#Web scraping the wikia for Genshin's playable character list.
#Note: Scraping only one page from .gg

urlnames_gg = "https://genshin.gg/"

pagenames_gg = requests.get(urlnames_gg)

soupname_gg = BeautifulSoup(pagenames_gg.text, "html.parser")
```

In [3]:
```
#Scraping through to get the html, identifying which class and identifier to

imgtag_gg = soupname_gg.find_all("h2", class_= "character-name")
#imgtag_gg
```

In [4]:
```
characters_gg = [element.get_text() for element in imgtag_gg]
#characters

def process_names(names_list):
    return [name.replace(" ", "").lower() for name in names_list]

characters_gg = process_names(characters_gg)
characters_gg = sorted(characters_gg)

print(characters_gg)
```

```
['albedo', 'alhaitham', 'aloy', 'amber', 'ayaka', 'ayato', 'baizhu', 'barba
ra', 'beidou', 'bennett', 'candace', 'charlotte', 'chevreuse', 'childe', 'c
hiori', 'chongyun', 'collei', 'cyno', 'dehya', 'diluc', 'diona', 'dori', 'e
ula', 'faruzan', 'fischl', 'freminet', 'furina', 'gaming', 'ganyu', 'gorou'
, 'heizou', 'hutao', 'itto', 'jean', 'kaeya', 'kaveh', 'kazuha', 'keqing',
'kirara', 'klee', 'kokomi', 'kukishinobu', 'layla', 'lisa', 'lynette', 'lyn
ey', 'mika', 'mona', 'nahida', 'navia', 'neuvillette', 'nilou', 'ningguang'
, 'noelle', 'qiqi', 'raiden', 'razor', 'rosaria', 'sara', 'sayu', 'shenhe',
'sucrose', 'thoma', 'tighnari', 'traveler(anemo)', 'traveler(dendro)', 'tra
veler(electro)', 'traveler(geo)', 'traveler(hydro)', 'venti', 'wanderer', '
wriothesley', 'xiangling', 'xianyun', 'xiao', 'xingqiu', 'xinyan', 'yaemiko
', 'yanfei', 'yaoyao', 'yelan', 'yoimiya', 'yunjin', 'zhongli']
```

Tidying Up Names

In [5]:
```
remove_gg = ["itto", "kazuha", "heizou", "sara", "childe", "ayato", "ayaka",


characters_gg = [item for item in characters_gg if item not in remove_gg]
characters_gg = characters_gg + remove_gg
print(characters_gg)
```

```
['albedo', 'alhaitham', 'aloy', 'amber', 'baizhu', 'barbara', 'beidou', 'be
nnett', 'candace', 'charlotte', 'chevreuse', 'chiori', 'chongyun', 'collei'
, 'cyno', 'dehya', 'diluc', 'diona', 'dori', 'eula', 'faruzan', 'fischl', '
freminet', 'furina', 'gaming', 'ganyu', 'gorou', 'hutao', 'jean', 'kaeya',
'kaveh', 'keqing', 'kirara', 'klee', 'kukishinobu', 'layla', 'lisa', 'lynet
te', 'lyney', 'mika', 'mona', 'nahida', 'navia', 'neuvillette', 'nilou', 'n
ingguang', 'noelle', 'qiqi', 'raiden', 'razor', 'rosaria', 'sayu', 'shenhe'
, 'sucrose', 'thoma', 'tighnari', 'traveler(anemo)', 'traveler(dendro)', 't
raveler(electro)', 'traveler(geo)', 'traveler(hydro)', 'venti', 'wanderer',
'wriothesley', 'xiangling', 'xianyun', 'xiao', 'xingqiu', 'xinyan', 'yaemik
o', 'yanfei', 'yaoyao', 'yelan', 'yoimiya', 'yunjin', 'zhongli', 'itto', 'k
azuha', 'heizou', 'sara', 'childe', 'ayato', 'ayaka', 'kokomi']
```

Data Preprocessing, with g8 instead of wk

In [6]:
```
url_g8 = "https://game8.co/games/Genshin-Impact/archives/307054"
```

In [7]:
```
#page_chara_g8 = requests.get(url_g8)
#soup_chara_g8 = BeautifulSoup(page_chara_g8.content, "html.parser")
#soup_chara_g8.find_all("a")
#pandas_table_g8[9]
```

In [8]:
```
#Using the Built-in Pandas html reader

pandas_table_g8 = pd.read_html(url_g8)
```

Scraping Names and Dates from g8

In [9]:
```
url_g8 = "https://game8.co/games/Genshin-Impact/archives/307054"


#Using the Built-in Pandas html reader
pandas_table_g8 = pd.read_html(url_g8)

dates_full_g8 = pandas_table_g8[7]        #Selecting the correct table with

#Removing the irrelevant data
dates_full_g8 = dates_full_g8.drop(columns = ['Element'])
dates_full_g8 = dates_full_g8.dropna()

#Sorting values
dates_full_g8 = dates_full_g8.sort_values(by = ["Character"], axis = 0)
dates_full_g8.reset_index(drop=True, inplace=True)


#Fixing Aloy, Childe, and Kuki Shinobu: specific characters with issues that
#with the naming and release date conventions
dates_full_g8.loc[2, 'Release Date'] = "10/13/2021"
dates_full_g8.loc[44, 'Character'] = "childe"
dates_full_g8 = dates_full_g8.drop(dates_full_g8[dates_full_g8['Character']

#print(dates_full_g8)




#Fixing Rarity Tag
dates_full_g8['Rarity'] = dates_full_g8['Rarity'].str.replace(r'★ (\d+)', r'

#Fixing Release Date
dates_full_g8['Release Date'] = pd.to_datetime(dates_full_g8['Release Date']

#print(dates_full_g8)



#Converting character names into simpler format and obtaining information or
characters_g8 = dates_full_g8['Character'].tolist()
characters_g8 = [x.lower() for x in characters_g8]        #Similar to previc
characters_g8 = process_names(characters_g8)              #Use Previous func

#print(len(characters_g8))
#print(characters_g8)



#Listing out Missing Characters, comparing complete set FROM gg and the incc
characters_gg_g8 = list(set(characters_gg) - set(characters_g8))

#print(characters_gg_g8)



#Isolating Day 1 Characters into g8
characters_gg_g8 = list(set(characters_gg_g8) - set('kukishinobu'))    #kuki

#print(characters_gg_g8)
```

```
#print(characters_gg_g8)



#Creating 5star (legendary) and 4star (epic) dataframes for concatenation
Legendary = ['diluc', 'jean', 'traveler(geo)', 'traveler(electro)',
             'traveler(hydro)','traveler(anemo)', 'traveler(dendro)',
             'qiqi', 'mona', 'keqing']
Epic = list(set(characters_gg_g8) - set(Legendary))        #Manual action is



#Converting Epic list into DataFrame
df_Epic = pd.DataFrame(Epic, columns=['Character'])
df_Epic['Rarity'] = '4star'
df_Epic['Release Date'] = '2020/9/28'
df_Epic['Release Date'] = pd.to_datetime(df_Epic['Release Date'])

kuki = pd.DataFrame([{'Character': 'kukishinobu',         #Adding back kukis
                      'Rarity': '4star',
                      'Release Date': '21/6/2022'}])
kuki['Release Date'] = pd.to_datetime(kuki['Release Date'])


df_Epic = pd.concat([df_Epic, kuki])
df_Epic.reset_index(drop = True, inplace = True)

#print(df_Epic)



#Converting Legendary list into DataFrame
df_Legend = pd.DataFrame(Legendary, columns=['Character'])
df_Legend['Rarity'] = '5star'
df_Legend['Release Date'] = '2020/9/28'
df_Legend['Release Date'] = pd.to_datetime(df_Epic['Release Date'])

#print(df_Legend)



#Combining the 5s and 4s dataframes for the Day One Characters (and kukishir
df_Day1 = pd.concat([df_Epic, df_Legend])
df_Day1.reset_index(drop = True, inplace = True)

#print(df_Day1)



#Combining the Day One Characters with the Initial dates_full_g8 Frame
dates_full_g8 = pd.concat([dates_full_g8, df_Day1])
dates_full_g8.reset_index(inplace = True, drop = True)
```

In [10]:
```
print(len(dates_full_g8))
#dates_full_g8
```

```
85
```

In [11]:
```
characters_g8 = dates_full_g8['Character'].tolist()
characters_g8 = [x.lower() for x in characters_g8]
characters_g8 = process_names(characters_g8)

#print(len(characters_g8))
#characters_g8
```

Scraping Character Data from .gg

In [12]:
```
#Web Scraping Constellation Data from Genshin.gg - Complete

characters_gg_test = characters_g8#[0:60]
cons_full_g8 = []

for names in characters_gg_test:

    url_chara_gg = "https://genshin.gg/characters/"+names+"/"
    page_chara_gg = requests.get(url_chara_gg)
    soup_chara_gg = BeautifulSoup(page_chara_gg.content, "html.parser")


    #Obtaining the constellation html info

    if soup_chara_gg.find("div", id='constellations') != None:
        cons1 = soup_chara_gg.find("div", id="constellations").find_all("div
        cons2 = [con.get_text() for con in cons1]
        cons2 = " ".join(cons2)
        #print(cons2)
        cons_full_g8.append(cons2)

    else:
        #print("No Constellations")
        notext = "This character has no constellations."
        cons_full_g8.append(notext)
```

In [13]:
```
len(cons_full_g8)

#cons_full_g8
#dates_full_g8
```

Out[13]:
```
85
```

In [14]:
```
#Note: Set limit as desired
pd.set_option('display.max_rows', 10)
dates_full_g8['Constellation Description'] = cons_full_g8


#Renaming into new DF
df1 = dates_full_g8
#DF_FULL['Character'] = DF_FULL['Character'].apply(process_names)
df1
```

Out[14]:

	Release Date	Character	Rarity	Constellation Description
0	2020-12-23	Albedo	5star	Transient Blossoms generated by Albedo's Abiog...
1	2023-01-18	Alhaitham	5star	When a Projection Attack hits an opponent, Uni...
2	2021-10-13	Aloy	5star	This character has no constellations.
3	2021-07-01	Ayaka	5star	When Kamisato Ayaka's Normal or Charged Attack...
4	2022-03-30	Ayato	5star	Shunsuiken DMG is increased by 40% against opp...
...	...	...	...	...
80	2020-09-28	traveler(anemo)	5star	Palm Vortex pulls in enemies within a 5m radiu...
81	2020-09-28	traveler(dendro)	5star	After Razorgrass Blade hits an opponent, it wi...
82	2020-09-28	qiqi	5star	When the Herald of Frost hits an enemy marked ...
83	2020-09-28	mona	5star	The effects of Hydro-related Elemental Reactio...
84	2020-09-28	keqing	5star	Recasting Stellar Restoration while a Lightnin...

85 rows × 4 columns

In [15]:
```
#Saving Output from Data Frame 1 into csv (to decrease uneccesary web scrape
#df1.to_csv("/Users/ignatiustobiassoetjianto/Desktop/df1.csv", index = False
```

In [ ]:
```
#df1 = pd.read_csv("/Users/ignatiustobiassoetjianto/Desktop/Project/BEE2041_
#df1
```

After web scraping, save df1 using pandas.

In [17]:
```
#Generating a simple word count from the constellation descriptions.

List_count_g8 = []

for paragraph in cons_full_g8:
    words = paragraph.split()
    word_count = sum(Counter(words).values())
    List_count_g8.append(word_count)

print(List_count_g8)
len(List_count_g8)
```

```
[141, 252, 5, 138, 121, 167, 143, 196, 224, 303, 205, 255, 242, 136, 160, 1
21, 180, 123, 338, 116, 146, 193, 155, 188, 164, 121, 149, 219, 125, 175, 2
30, 100, 149, 172, 234, 240, 236, 183, 171, 96, 111, 193, 135, 143, 110, 14
9, 119, 271, 358, 263, 106, 109, 106, 102, 201, 155, 135, 84, 106, 86, 116,
108, 111, 129, 106, 142, 92, 93, 111, 112, 117, 120, 66, 145, 145, 139, 110
, 124, 123, 180, 80, 126, 102, 165, 137]
```

Out[17]:
```
85
```

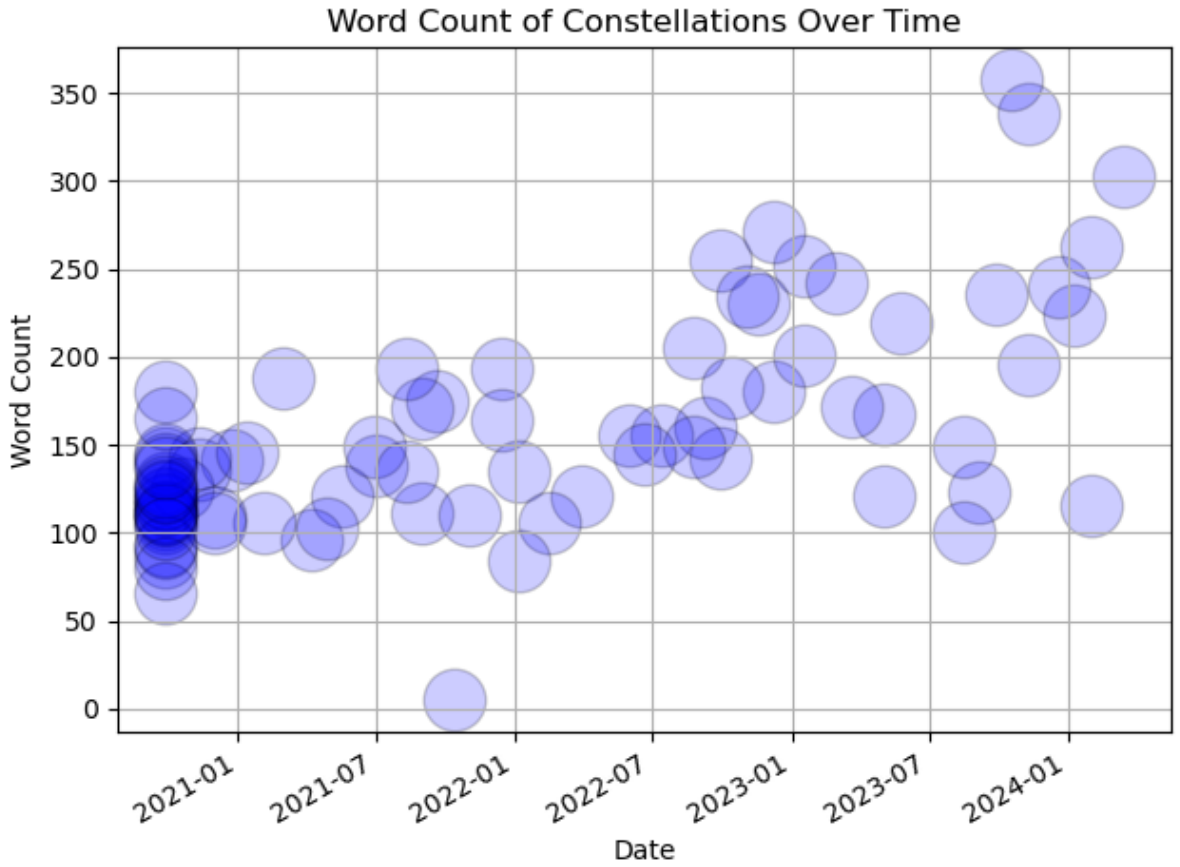
In [18]:
```
df1['Word Count'] = List_count_g8
#df1
```

In [19]:
```
#Saving Output from Data Frame 1 into csv (to decrease uneccesary web scrape

#df1.to_csv("/Users/ignatiustobiassoetjianto/Desktop/df1_wc.csv", index = Fa
```

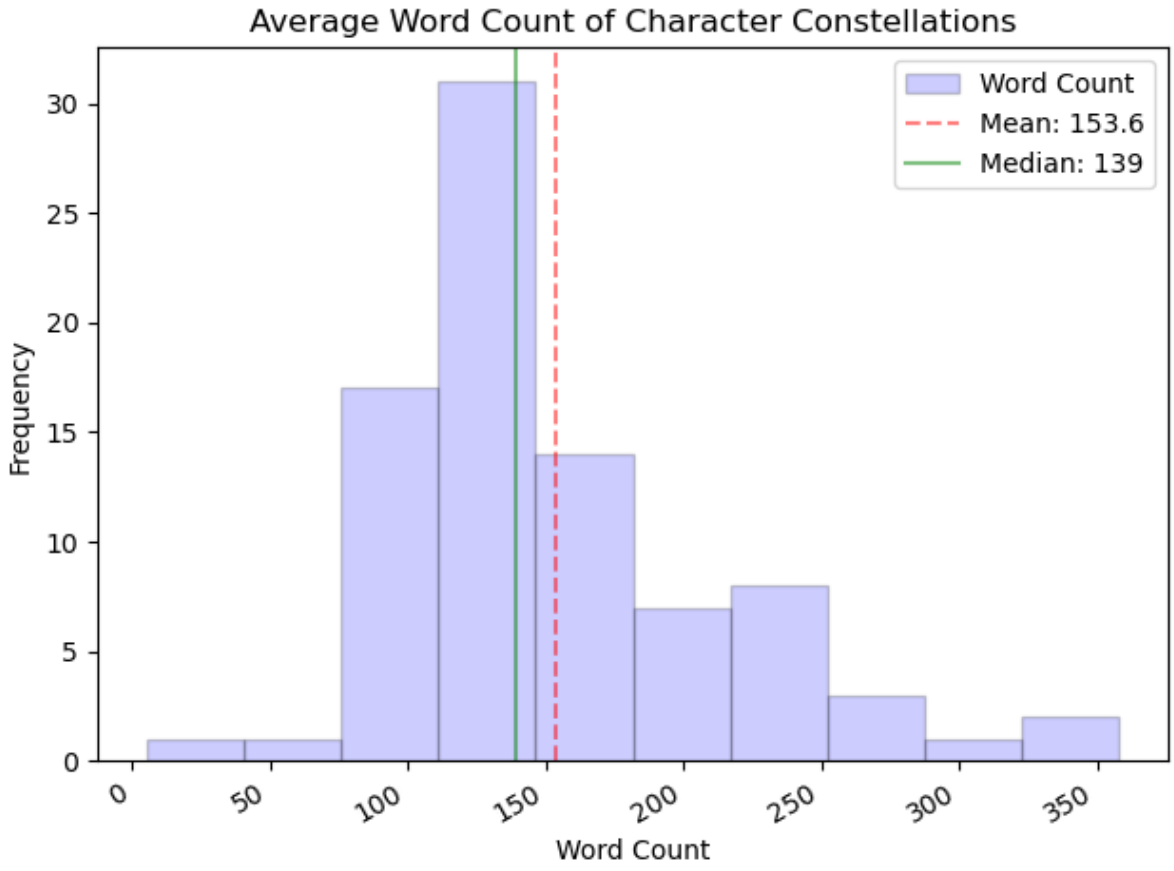
In [20]:
```
#df1 = pd.read_csv('/Users/ignatiustobiassoetjianto/Desktop/Project/BEE2041_
#df1
```

Plots (no discrimination)

In [21]:
```
#Creating plots with the new data frame.

df1.plot.scatter(x='Release Date', y = 'Word Count', color = 'blue',
                 marker = 'o', edgecolor = 'black', alpha = 0.20, s=550)

# Adding labels and title
plt.xlabel('Date')
plt.ylabel('Word Count')
plt.title('Word Count of Constellations Over Time')

# Formatting x-axis ticks as dates
plt.xticks(rotation=30, ha='right')

# Display the plot
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [22]:
```
#Histogram of Frequency of "Word Count:" Average Word Count of Character Cor

df1.plot.hist(y='Word Count', color='blue', edgecolor='black', alpha = 0.20)

#Calculate Mean and Median
mean_stat = df1['Word Count'].mean()
median_stat = df1['Word Count'].median()

plt.axvline(x=mean_stat, color = 'red', linestyle = '--',
            label = f'Mean: {round(mean_stat, 1)}', alpha = 0.5)
plt.axvline(x=median_stat, color = 'green', linestyle = '-',
            label = f'Median: {round(median_stat)}', alpha = 0.5)


#Adding labels and title
plt.xlabel('Word Count')
plt.ylabel('Frequency')
plt.title('Average Word Count of Character Constellations')
plt.legend()

#Formatting x-axis ticks as dates
plt.xticks(rotation=30, ha='right')

#Display the plot
plt.tight_layout()
plt.show()
```



Grouping by Rarity

In [23]:
```
#Summary Statistics according to character rarity

df1_summary = df1.groupby('Rarity')
display(df1_summary.mean())
```

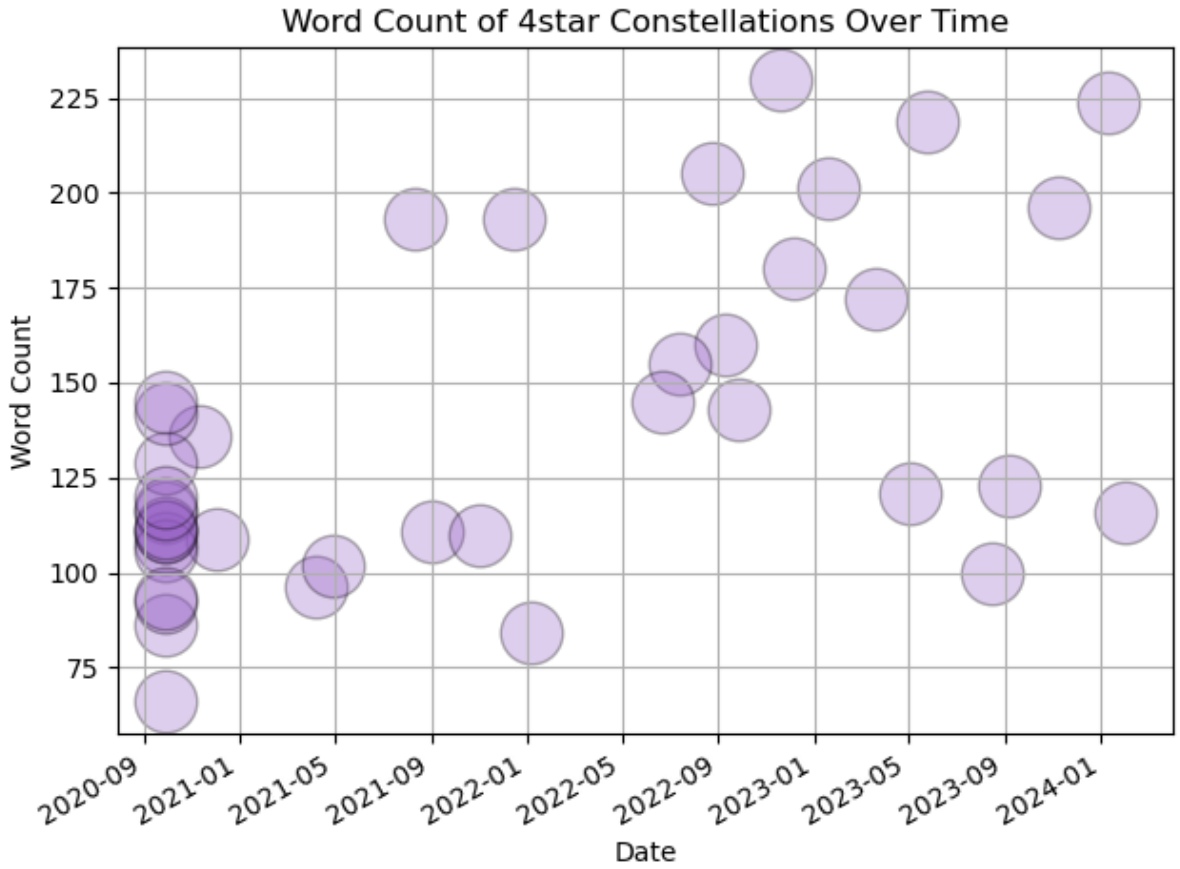
Rarity	Word Count
4star	136.950000
5star	168.333333

In [24]:
```
#Splitting original dataframe into a list, and then converting the list into

Rarity = df1.groupby('Rarity')
df1_Rarity = [group_df for _, group_df in Rarity]
df1_Epic, df1_Legendary = df1_Rarity
```

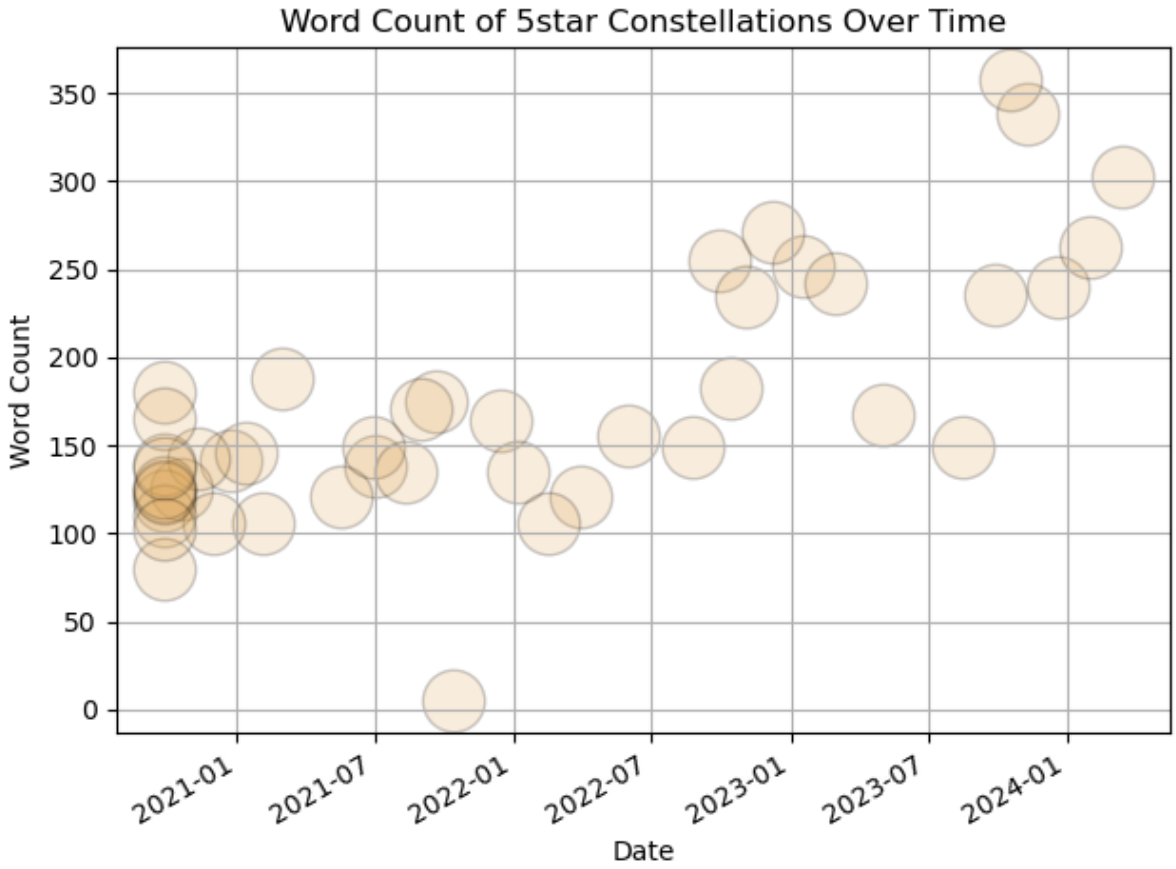
In [25]:
```
df1_Epic.reset_index(inplace = True, drop = True)
df1_Legendary.reset_index(inplace = True, drop = True)

#display(df_Epic_2)
#display(df_Legendary_2)
```

Scatter Plots (with discrimination)

In [26]:
```
#Creating plots for Epic Rarities

df1_Epic.plot.scatter(x = 'Release Date', y = 'Word Count', color='#945dc4',
                      marker = 'o', edgecolor = 'black', alpha = 0.30, s = 550)

# Adding labels and title
plt.xlabel('Date')
plt.ylabel('Word Count')
plt.title('Word Count of 4star Constellations Over Time')

# Formatting x-axis ticks as dates
plt.xticks(rotation=30, ha='right')

# Display the plot
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [27]:
```
#Creating plots for Legendary Rarities

df1_Legendary.plot.scatter(x = 'Release Date', y = 'Word Count', color = '#c
                marker = 'o', edgecolor = 'black', alpha = 0.20, s = 550)

# Adding labels and title
plt.xlabel('Date')
plt.ylabel('Word Count')
plt.title('Word Count of 5star Constellations Over Time')

# Formatting x-axis ticks as dates
plt.xticks(rotation=30, ha='right')

# Display the plot
plt.grid(True)
plt.tight_layout()
plt.show()
```


Word Count of 5star Constellations Over Time
Word Count
350
300
250
200
150
100
50
0
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07
2024-01
Date

Creating Subplot

fig,ax = plt.subplots(nrows = 1, ncols = 2, figsize = (14, 4))

4 Star

X0 = df_Epic_2['Release Date'] Y0 = df_Epic_2['Word Count']

ax[0].set_title('4 Star') ax[0].scatter(X0, Y0, color = '#945dc4', marker = 'o', edgecolor = 'black', alpha = 0.20, s = 550)

Formatting x-axis ticks as dates

ax[0].set_xticks(X0)

ax[0].set_xticklabels(X0, rotation=30, ha='right')

Display the plot

ax[0].grid(True)

5 Star

X1 = df_Legendary_2['Release Date'] Y1 = df_Legendary_2['Word Count']

ax[1].set_title('5 Star') ax[1].scatter(X1, Y1, color = '#dca454', marker = 'o', edgecolor = 'black', alpha = 0.20, s = 550)

Formatting x-axis ticks as dates

ax[1].set_xticks(X1)

ax[1].set_xticklabels(X1, rotation=30

ax[1].set_xticklabels(x1, rotation=30, ha='right')

Display the plot

ax[1].grid(True)

Insight: word frequency distribution

In [28]:
```
pd.set_option('display.max_rows', 10)
```

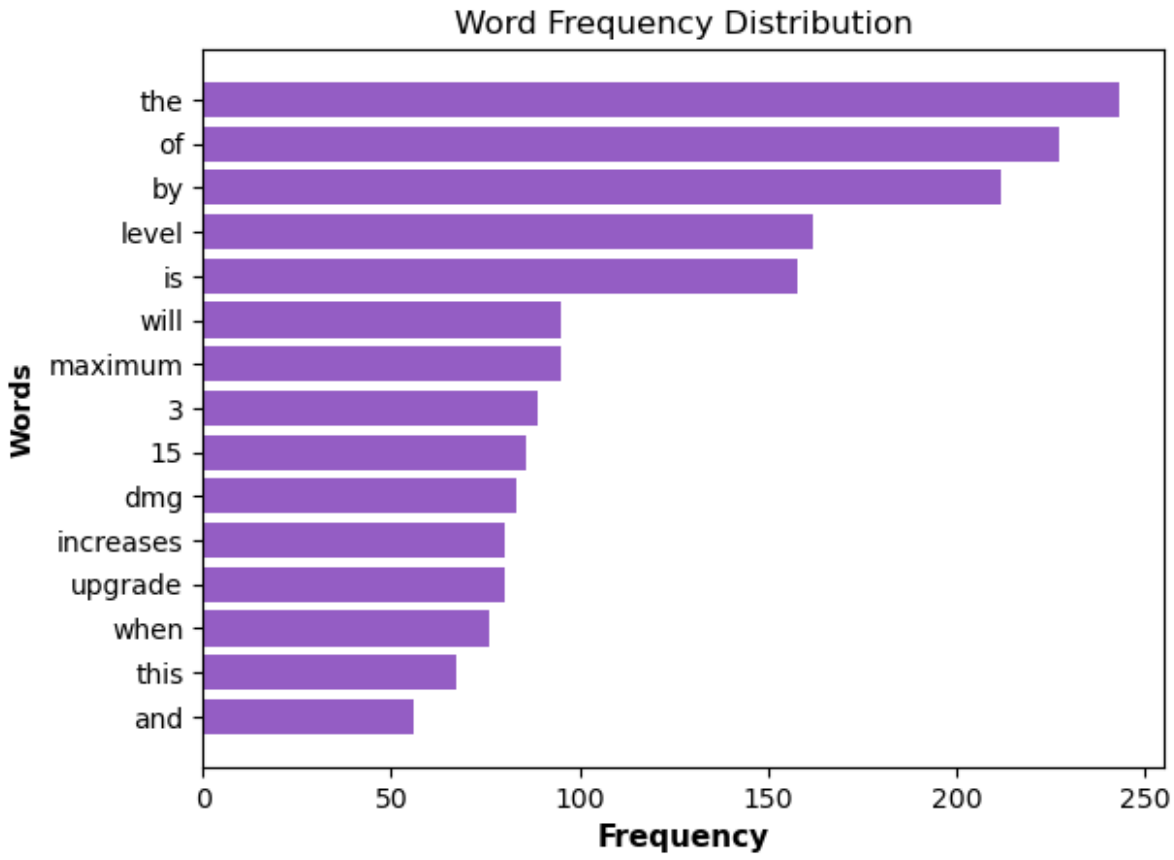
In [29]:
```
df1_Epic
```

Out[29]:

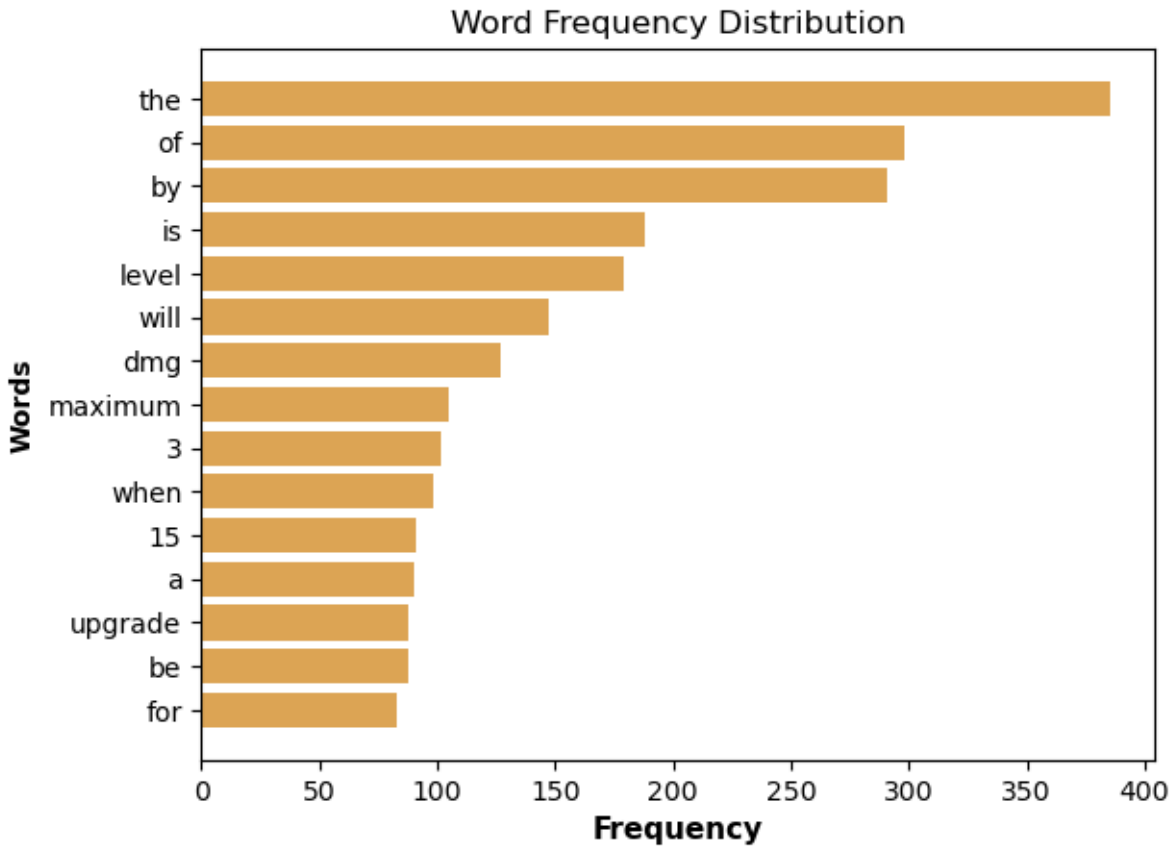
	Release Date	Character	Rarity	Constellation Description	Word Count
0	2022-09-28	Candace	4star	The duration of Prayer of the Crimson Crown ef...	143
1	2023-11-08	Charlotte	4star	After Still Photo: Comprehensive Confirmation ...	196
2	2024-01-09	Chevreuse	4star	When the active character with the "Coordinate...	224
3	2022-08-24	Collei	4star	When in the party and not on the field, Collei...	205
4	2020-11-11	Diona	4star	Regenerates 15 Energy for Diona after the effe...	136
...	...	...	...	...	...
35	2020-09-28	barbara	4star	Barbara regenerates 1 Energy every 10s. Decrea...	117
36	2020-09-28	kaeya	4star	The CRIT Rate of Normal Attack and Charged Att...	120
37	2020-09-28	ningguang	4star	When a Normal Attack hits, it deals AoE DMG. W...	66
38	2020-09-28	kukishinobu	4star	Gyoei Narukami Kariyama Rite's AoE is increase...	145
39	2022-06-21	kukishinobu	4star	Gyoei Narukami Kariyama Rite's AoE is increase...	145

40 rows × 5 columns

In [30]:
```
#Aggregating all 4 stars into one paragraph.
Epic_cons_all = ' '.join(df1_Epic['Constellation Description'])

#Cleaning the paragraph
testing1 = Epic_cons_all.replace(".", "").replace('(', "").replace(')', "")
#testing1

#Splitting paragraphs by word
testing2 = testing1.lower().split()
#testing2

#Making a word count and distribution
testing3 = Counter(testing2).most_common(15)        #Note: Choose the top fr
#testing3

#Converting into usable DataFrame
testing4 = pd.DataFrame(testing3, columns=['Word', 'Frequency'])
testing4 = testing4[::-1]
#testing4[4:10]

#Plotting into bar chart
plt.barh(testing4['Word'], testing4['Frequency'], color = "#945dc4")

plt.xlabel('Frequency', fontsize = 11, fontweight = 'bold')
plt.ylabel('Words', fontsize = 10, fontweight = 'bold')

plt.title('Word Frequency Distribution')
plt.xticks(rotation=0, fontsize = 10)

plt.show()
```



In [31]:
```
#Aggregating all 5 stars into one paragraph.
Legendary_cons_all = ' '.join(df1_Legendary['Constellation Description'])

#Cleaning the paragraph
testing1 = Legendary_cons_all.replace(".", "").replace('(', "").replace(')',
#testing1

#Splitting paragraphs by word
testing2 = testing1.lower().split()
#testing2

#Making a word count and distribution
testing3 = Counter(testing2).most_common(15)        #Note: Choose the top fr
#testing3

#Converting into usable DataFrame
testing4 = pd.DataFrame(testing3, columns=['Word', 'Frequency'])
testing4 = testing4[::-1]
#testing4[4:10]

#Plotting into bar chart
plt.barh(testing4['Word'], testing4['Frequency'], color = "#dca454")

plt.xlabel('Frequency', fontsize = 11, fontweight = 'bold')
plt.ylabel('Words', fontsize = 10, fontweight = 'bold')

plt.title('Word Frequency Distribution')
plt.xticks(rotation=0, fontsize = 10)

plt.show()
```



Insight: specific words used

In [32]:
```
pd.set_option('display.max_rows', 10)
```

In [33]:
```
#Create Function that count specific words/strings

def count_target_words(Input, Target):
    words = Input.split()
    count = 0


    for target in Target:
        for word in words:
#         word = word.strip('.,?!:;-()[]{}\'"')

            if word == target:
                count += 1

    return count
```

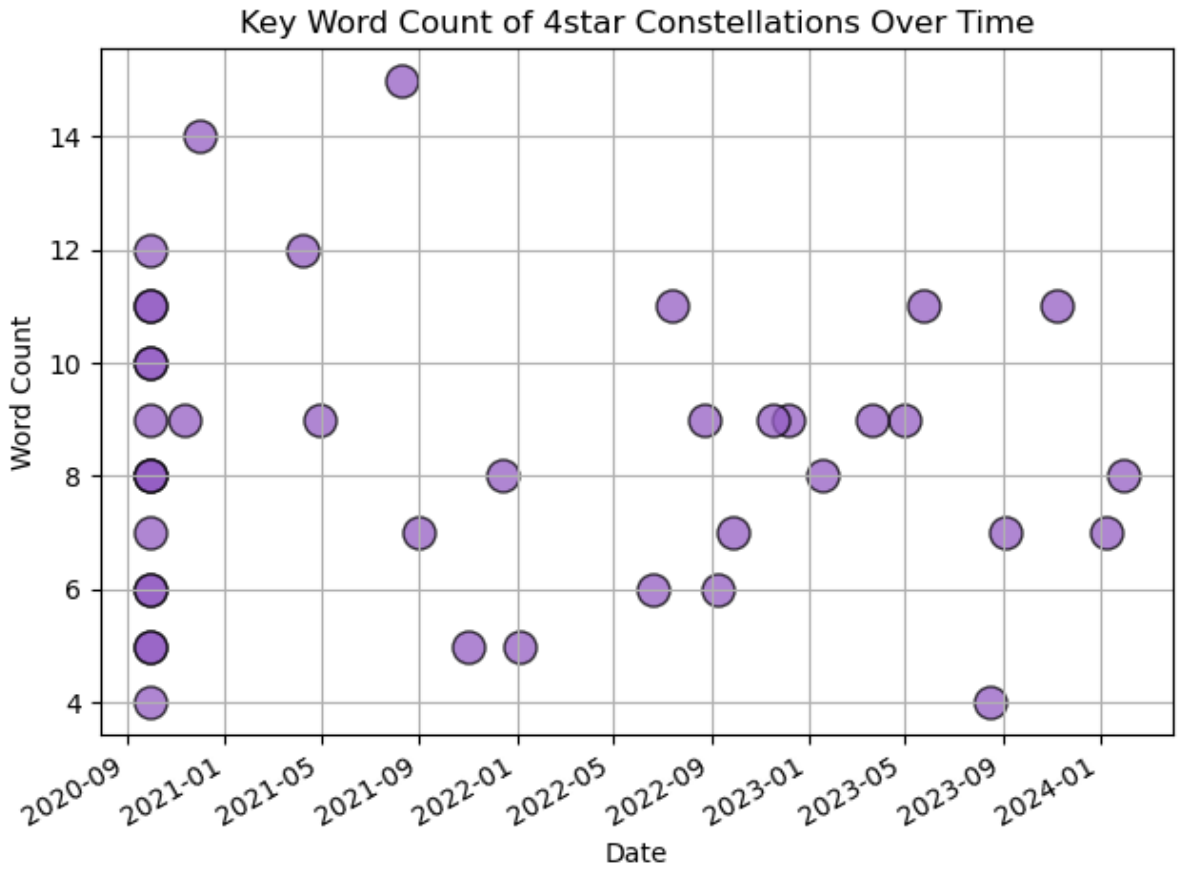
In [34]:
```
#Testing with 4stars

#Epic_cons_all

testing1 = Epic_cons_all.replace(".", "").replace('(', "").replace(')', "").
#testing1

count_target_words(testing1, ["crit", "dmg", "atk", ""])
```

Out[34]:
```
134
```

In [35]:
```
#Creating new dataframe with different counts (df2), on the 4 star character

df2_Epic = df1_Epic#.drop(columns = ['Word Count'])
df2_Epic['Constellation Desc'] = df2_Epic['Constellation Description'].repla

#df['Text'] = df['Text'].apply(lambda x: x.lower())
df2_Epic['Constellation Desc'] = df2_Epic['Constellation Desc'].apply(lambda
df2_Epic = df2_Epic.drop(columns = ['Constellation Description'])


#Critical
#df2_Epic['crit Count'] = df2_Epic['Constellation Desc'].apply(lambda x: cou

#Attack
#df2_Epic['atk Count'] = df2_Epic['Constellation Desc'].apply(lambda x: cour

#Increase
#df2_Epic['increase Count'] = df2_Epic['Constellation Desc'].apply(lambda x:

#Energy
#df2_Epic['energy Count'] = df2_Epic['Constellation Desc'].apply(lambda x: c


#Aggregate of all "Meaningful" key word stats.
df2_Epic['Key Count'] = df2_Epic['Constellation Desc'].apply(lambda x: count
```

In [36]:
```
#df2_Epic
```

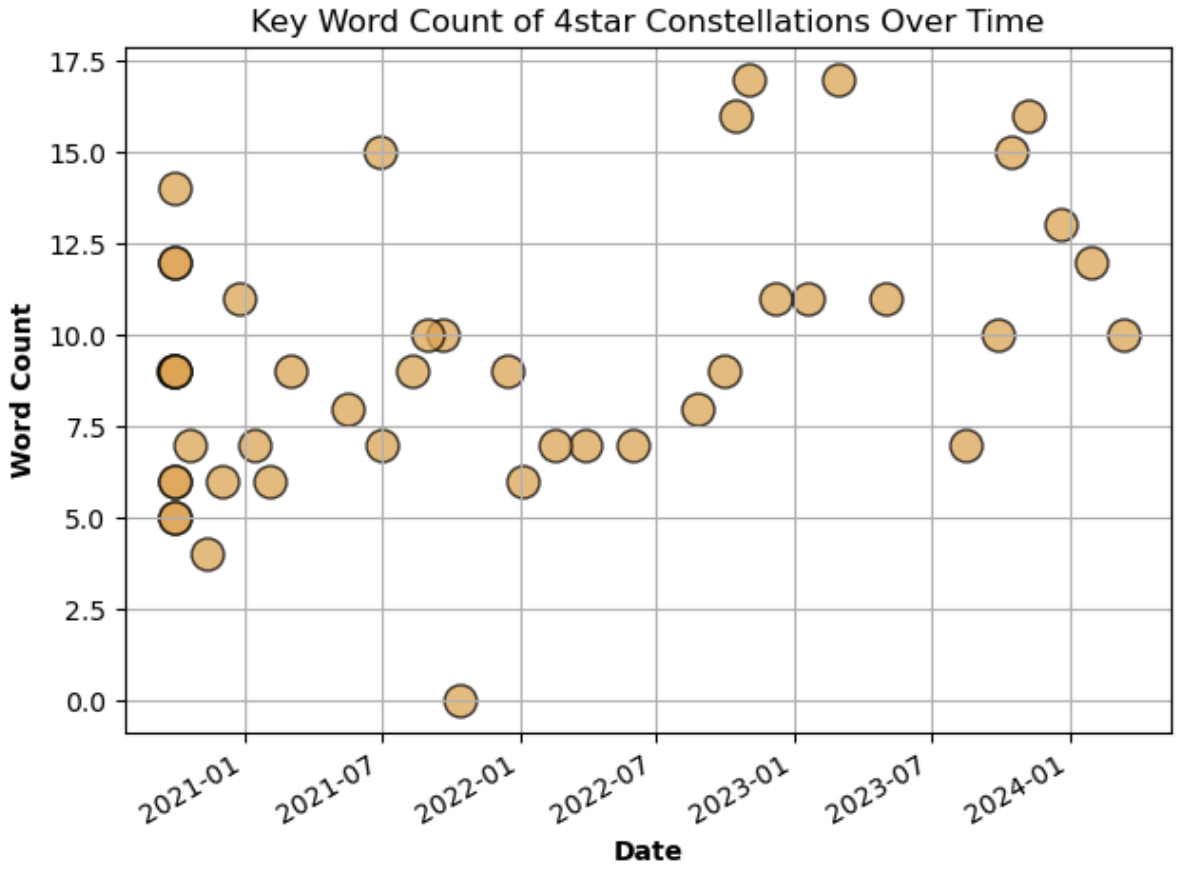
In [37]:
```
#Creating plots for 4 Star Rarities key words

df2_Epic.plot.scatter(x = 'Release Date', y = 'Key Count', color='#945dc4',
                      marker = 'o', edgecolor = 'black', alpha = 0.75, s = 150)

#Adding labels and title
plt.xlabel('Date')
plt.ylabel('Word Count')
plt.title('Key Word Count of 4star Constellations Over Time')

#Formatting x-axis ticks as dates
plt.xticks(rotation=30, ha='right')

#Creating Line of Best Fit
#df2_Epic_set = np.polyfit(df2_Epic['Release Date'], df2_Epic['Key Count'],
#df2_Epic_line = np.poly1d(df2_Epic_set)
#plt.plot(df2_Epic['Release Date'], df2_Epic_line(df2_Epic['Release Date']),

#Display the plot
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [38]:
```
#Creating new dataframe with different counts (df2), on the 5 star character

df2_Legendary = df1_Legendary#.drop(columns = ['Word Count'])
df2_Legendary['Constellation Desc'] = df2_Legendary['Constellation Descripti

#df['Text'] = df['Text'].apply(lambda x: x.lower())
df2_Legendary['Constellation Desc'] = df2_Legendary['Constellation Desc'].ap
df2_Legendary = df2_Legendary.drop(columns = ['Constellation Description'])


#Critical
#df2_Epic['crit Count'] = df2_Epic['Constellation Desc'].apply(lambda x: cou

#Attack
#df2_Epic['atk Count'] = df2_Epic['Constellation Desc'].apply(lambda x: cour

#Increase
#df2_Epic['increase Count'] = df2_Epic['Constellation Desc'].apply(lambda x:

#Energy
#df2_Epic['energy Count'] = df2_Epic['Constellation Desc'].apply(lambda x: c


#Aggregate of all "Meaningful" key word stats.
df2_Legendary['Key Count'] = df2_Legendary['Constellation Desc'].apply(lambc
```

In [39]:
```
#df2_Legendary
```

In [40]:
```
#Creating plots for 4 Star Rarities key words



df2_Legendary.plot.scatter(x = 'Release Date', y = 'Key Count', color='#dca4
                           marker = 'o', edgecolor = 'black', alpha = 0.75,

# Adding labels and title
plt.xlabel('Date', fontweight = 'bold')
plt.ylabel('Word Count', fontweight = 'bold')
plt.title('Key Word Count of 4star Constellations Over Time')

# Formatting x-axis ticks as dates
plt.xticks(rotation=30, ha='right')

# Display the plot
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [ ]:

Testing

In [41]:
```
def count_specific_words_test(Input, Target):
    words = Input.split()
    count = 0


    for target in Target:
        for word in words:
#         word = word.strip('.,?!:;-()[]{}\'"')

            if word == target:
                count += 1

    return count
```

In [42]:
```
count_specific_words_test(testing1, ["crit", "dmg", "atk", ""])
```

Out[42]:
```
134
```

In [45]:
```
#Creating plots for Epic Rarities

#df_Epic_3.plot.scatter(x = 'Release Date', y = 'increase Count', color='#94
#                  marker = 'o', edgecolor = 'black', alpha = 0.30, s = 550)

#Adding labels and title
#plt.xlabel('Date')
#plt.ylabel('Word Count')
#plt.title('Word Count of 4star Constellations Over Time')

#Formatting x-axis ticks as dates
#plt.xticks(rotation=30, ha='right')

#Display the plot
#plt.grid(True)
#plt.tight_layout()
#plt.show()
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [47]:
```
#def count_specific_words(text, target_words):
    # Split the text into words
#    words = text.split()

    # Create a dictionary to store word frequencies
#    word_counts = {word: 0 for word in target_words}

    #iterate over each word and update the counts
#    for word in words:
        # Remove any punctuation from the word
#        word = word.strip('.,?!:;-()[]{}\'"')

        #convert the word to lowercase
#        word = word.lower()

        #update the count for this word if it's in the target words list
#        if word in target_words:
#            word_counts[word] += 1

#    return word_counts

#text = "This is a sample text. It contains some words, some of which are re
#target_words = ['sample', 'text', 'words']

#text = testing1
#target_words = ['crit', 'increase', 'increases', 'atk', 'elemental', 'maste

#word_counts = count_specific_words(text, target_words)
#print(word_counts)
```

In [ ]:
```
#count_words()
```

In [ ]:
```

```

In [ ]:
```
#test = Epic_cons_all.lower()
#test = test.split()


#Cleaning paragraphs
#Epic_Cons_all = Epic_cons_all.replace("\\", "")
#Epic_Cons_all


#Counting Distribution
#test_count = Counter(test)                #Use Counter lib
#test_list = test_count.most_common()
```

In [ ]:
```
#Cleaning the paragraph

#testing1 = Epic_cons_all.replace(".", "").replace('(', "").replace(')', "")
#testing1
```

In [ ]:
```
#testing2 = testing1.replace('(', "")
#testing2 = testing2.replace(')', "")

#testing2 = testing1.lower().split()
#testing2
```

In [ ]:
```
#Making a word count and distribution
#testing3 = Counter(testing2).most_common(15)        #Note: Choose the top 1
#testing3
```

In [ ]:
```
#Creating DataFrame
#testing4 = pd.DataFrame(testing3, columns=['Word', 'Frequency'])
#testing4#[4:10]
```

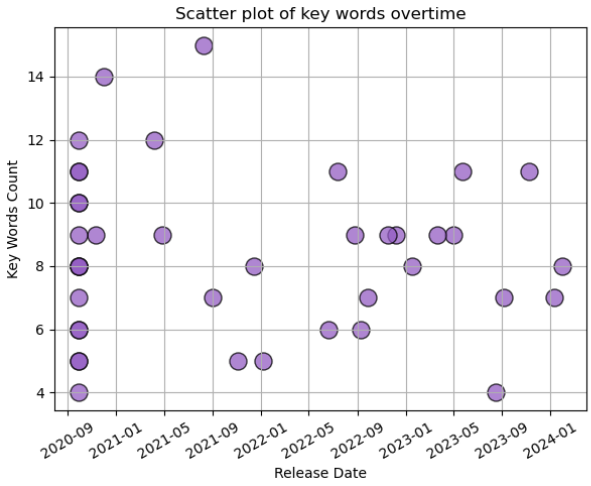
In [ ]:
```
#plt.barh(testing4['Word'], testing4['Frequency'], color = "#945dc4")

#plt.xlabel('Words', fontsize = 13)
#plt.ylabel('Frequency', fontsize = 10, fontweight = 'bold')

#plt.title('Word Frequency Distribution')
#plt.xticks(rotation=0, fontsize = 10)

#plt.show()
```

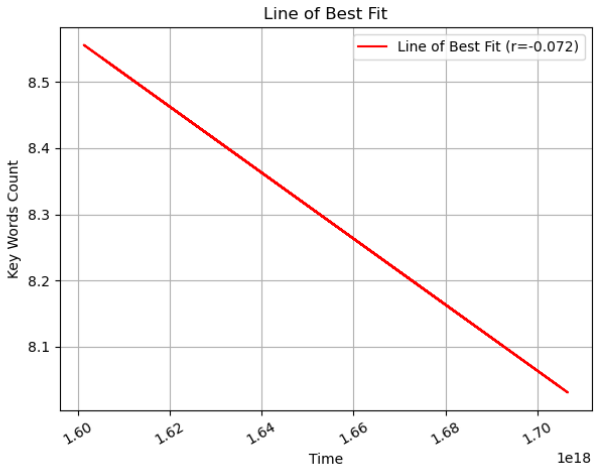
In [ ]:
```
#testing4
```

In [51]:
```
#Setting up for 4 star

#X1_test_int = X1_test.astype(int)
#df2_Epic_set = np.polyfit(X1_test_int, Y1_test, 1)
#df2_Epic_line = np.poly1d(df2_Epic_set)
```

In [48]:
```
#Creating subplots for 4 Star Rarities key words

#Data
X1_test = df2_Epic['Release Date']
X1_test_int = X1_test.astype(int)

Y1_test = df2_Epic['Key Count']

df2_Epic_set = np.polyfit(X1_test_int, Y1_test, 1)
df2_Epic_line = np.poly1d(df2_Epic_set)

r_value = np.corrcoef(X1_test_int, Y1_test)[0, 1]



#Setting the Subplot
df2_TEST_1, ax = plt.subplots(1, 2, figsize=(12, 5))


#Scatter Plot
ax[0].scatter(X1_test, Y1_test, color='#945dc4',
              marker = 'o', edgecolor = 'black', alpha = 0.75, s = 150)


ax[0].set_xlabel('Release Date')
ax[0].set_ylabel('Key Words Count')
ax[0].set_title('Scatter plot of key words overtime')



#Creating Line of Best Fit

#ax[1].plot(X1_test_int, poly(X1_test_int), color='red', label='Line of Best

ax[1].plot(X1_test_int, df2_Epic_line(X1_test_int), color = 'red', label = f

ax[1].set_xlabel('Time')
ax[1].set_ylabel('Key Words Count')
ax[1].set_title('Line of Best Fit')



#df2_Epic_set = np.polyfit(X1_test_int, Y1_test, 1)
#df2_Epic_line = np.poly1d(df2_Epic_set)




#plt.plot(X1_test_int, df2_Epic_line(X1_test_int), color='red', label='Line

#Convert from data back into datetime?
#pd.to_datetime(X1_test)



#Formatting x-axis ticks as dates

for axis in ax:
    axis.tick_params(axis = 'x', rotation=20)
```

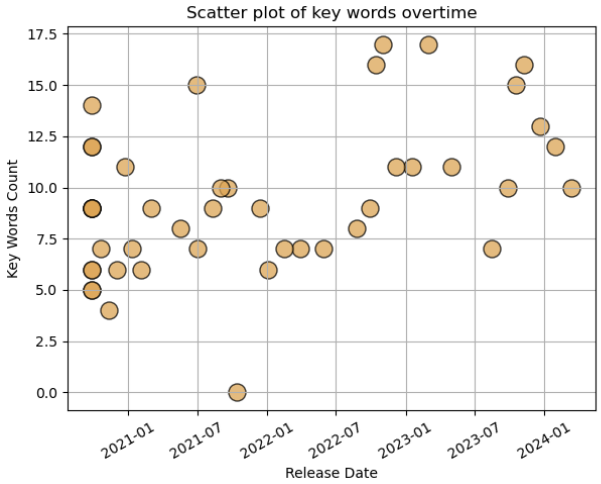
```
    axis.tick_params(axis = 'x', rotation=30)
    axis.grid(True)
#     axis.tight_layout()


#Display the plot
#plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

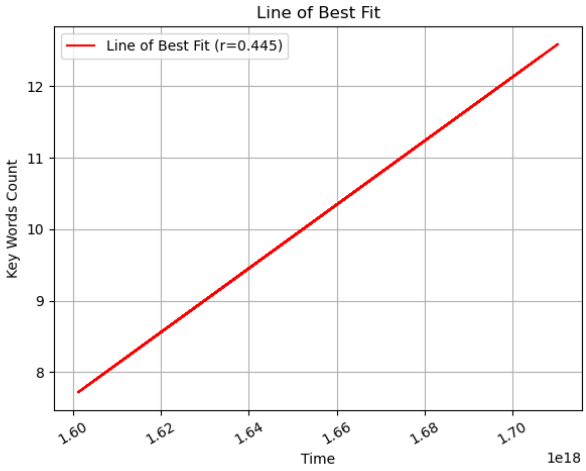




In [50]:
```
#Setting up for 5 star

#X1_test_int = X1_test.astype(int)
#df2_Epic_set = np.polyfit(X1_test_int, Y1_test, 1)
#df2_Epic_line = np.poly1d(df2_Epic_set)
```

In [49]:
```
#Creating subplots for 5 Star Rarities key words

#Data
X1_test = df2_Legendary['Release Date']
X1_test_int = X1_test.astype(int)

Y1_test = df2_Legendary['Key Count']

df2_Legendary_set = np.polyfit(X1_test_int, Y1_test, 1)
df2_Legendary_line = np.poly1d(df2_Legendary_set)

r_value = np.corrcoef(X1_test_int, Y1_test)[0, 1]


#Setting Subplots
df2_TEST_2, ax = plt.subplots(1, 2, figsize=(12, 5))


#Scatter Plot
ax[0].scatter(X1_test, Y1_test, color='#dca454',
              marker = 'o', edgecolor = 'black', alpha = 0.75, s = 150)

ax[0].set_xlabel('Release Date')
ax[0].set_ylabel('Key Words Count')
ax[0].set_title('Scatter plot of key words overtime')



#Creating Line of Best Fit
ax[1].plot(X1_test_int, df2_Legendary_line(X1_test_int), color = 'red', labe

ax[1].set_xlabel('Time')
ax[1].set_ylabel('Key Words Count')
ax[1].set_title('Line of Best Fit')



#Formatting x-axis

for axis in ax:
    axis.tick_params(axis = 'x', rotation=30)
    axis.grid(True)


#Display the plot
plt.tight_layout()
plt.legend()
plt.show()
```



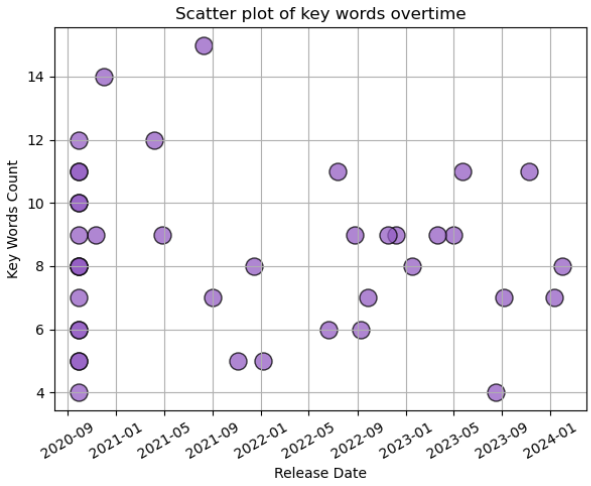


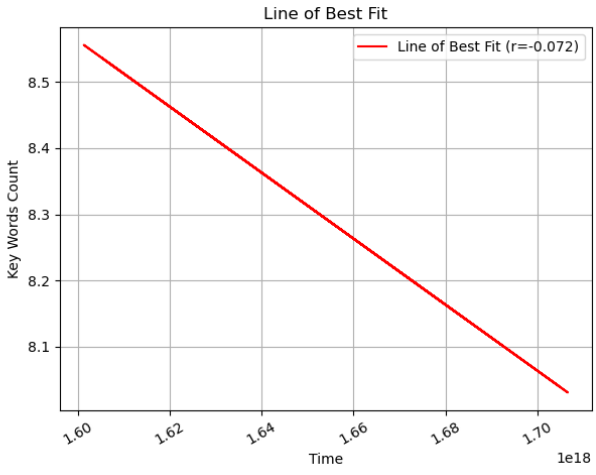
In [52]:
```
#How to add line of best fit

#list(X1_test)
#X1_test.to_julian_date()

#X1_test = X1_test.astype(int)

#Creating Line of Best Fit
#df2_Epic_set = np.polyfit(X1_test, Y1_test, 1)
#df2_Epic_line = np.poly1d(df2_Epic_set)
#plt.plot(X1_test, df2_Epic_line(X1_test), color='red', label='Line of Best



#Convert from data back into datetime?
#pd.to_datetime(X1_test)
```

In [53]:
```
#X1_test = df2_Epic['Release Date']
#X1_test_int = X1_test.astype(int)
```

In [54]:
```
df2_TEST_1
```

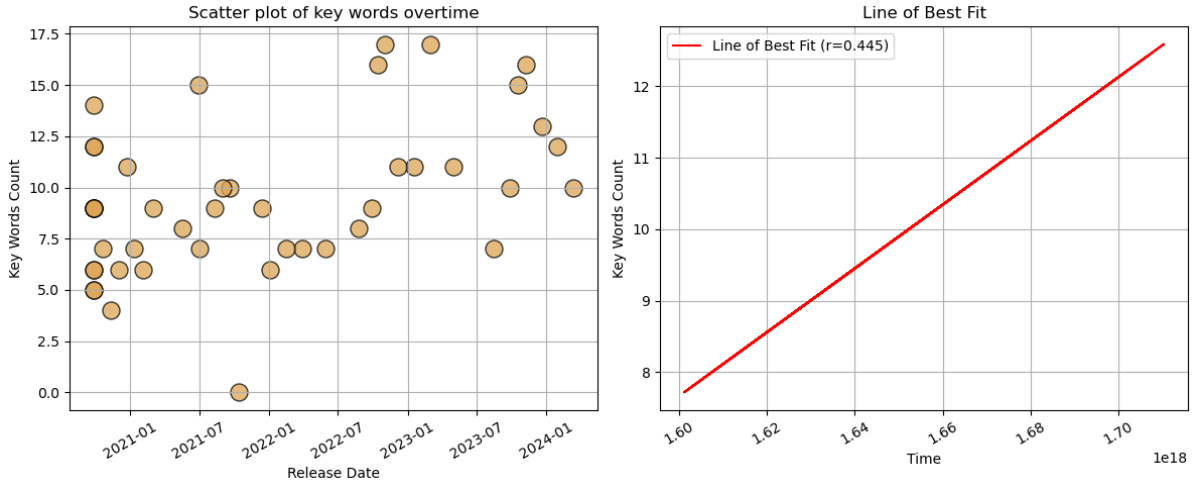
Out[54]:





In [55]:
```
df2_TEST_2
```

Out[55]:


Scatter plot of key words overtime
Key Words Count
Release Date
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07
2024-01
Line of Best Fit
Line of Best Fit (r=0.445)
Key Words Count
Time
1e18
12
11
10
9
8
1.60
1.62
1.64
1.66
1.68
1.70

In [ ]: