Problem Statement and Hypothesis

Historically, funding for programs involved in space exploration has been cut or decreased when public interest has decreased or is negative. To keep the momentum currently being experienced in this sector, the companies and agencies involved will need to keep the general public on their side. This study will use messages written by the public on the social media network Twitter, or "tweets", to gauge the public sentiment towards entities in the space industry to answer the question "Is there a significant difference in the sentiment expressed in tweets involving commercial entities compared to government agencies involved space exploration?"

For this research project, the null hypothesis is "There is no difference in sentiment between commercial and government entities", and the alternative hypothesis is "There is a significant difference in sentiment between commercial and government entities."

Data Gathering

To perform the analysis, data was gathered in the form of tweets from the Twitter website (https://twitter.com/ (https://twitter.com/)). For this project the date and time of the tweet, the individual tweet ID, the user name of the author, and the actual text of the tweet were used.

The data was collected using the tool Python tool snscrape. To get a feel for the sentiment over a period of time, the first 2,000 tweets were collected for each month per entity for a period of two years. The parameters for the snscrape tool were set to loop through a list of months (from April 2019 through March 2021) and another list of entities ('SpaceX', 'Virgin Galactic', 'Blue Origin', 'NASA', 'ESA', 'JPL'), collecting the first 2,000 tweets for each. During the loop, the entity name was added to the tweet data. The code below resulted in a dataframe with 285,167 rows and five columns with 0% sparsity.

In [19]:
```
# Import standard libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

import warnings
warnings.filterwarnings('ignore') # Ignore warning messages for readability
```

In [2]:
```
# Create lists of entities and start/end dates
entities = ['SpaceX', 'Virgin Galactic', 'Blue Origin', 'NASA', 'ESA', 'JPL']
month_start = ['2021-03-01', '2021-02-01', '2021-01-01', '2020-12-01', '2020-11-01', '2020-10-01', '2020-09-0
1',
               '2020-08-01', '2020-07-01', '2020-06-01', '2020-05-01', '2020-04-01', '2020-03-01', '2020-02-0
1',
               '2020-01-01', '2019-12-01', '2019-11-01', '2019-10-01', '2019-09-01', '2019-08-01', '2019-07-0
1',
               '2019-06-01', '2019-05-01', '2019-04-01']
month_end   = ['2021-03-31', '2021-02-28', '2021-01-31', '2020-12-31', '2020-11-30', '2020-10-31', '2020-09-3
0',
               '2020-08-31', '2020-07-31', '2020-06-30', '2020-05-31', '2020-04-30', '2020-03-31', '2020-02-2
9',
               '2020-01-31', '2019-12-31', '2019-11-30', '2019-10-31', '2019-09-30', '2019-08-31', '2019-07-3
1',
               '2019-06-30', '2019-05-31', '2019-04-30']
```

In [3]:
```
# Create list to append tweet data to
tweets_list = []
```

In [4]:
```
# Using TwitterSearchScraper to scrape data and append tweets to list

import snscrape.modules.twitter as sntwitter # import scraping tool

for entity in entities:
    for j in range(len(month_start)):
        for i,tweet in enumerate(sntwitter.TwitterSearchScraper(f"'{entity} since:{month_start[j]} until:{mont
h_end[j]}'").get_items()):
            if i>2000:
                break
            tweets_list.append([tweet.date, tweet.id, tweet.content, tweet.user.username, entity])
```

In [5]:
```
# Creating a dataframe from the tweets list above
tweets_df = pd.DataFrame(tweets_list, columns=['Datetime', 'Tweet Id', 'Text', 'Username', 'Company'])
```

In [6]:
```
# View head
tweets_df.head()
```

Out[6]:

	Datetime	Tweet Id	Text	Username	Company
0	2021-03-30 23:59:47+00:00	1377047938938060803	@danahjon @NASAKennedy @ArceneauxHayley @DrSia...	rookisaacman	SpaceX
1	2021-03-30 23:59:38+00:00	1377047902044872706	@SpaceTfrs So grateful for your contribution t...	BlueJ11274903	SpaceX
2	2021-03-30 23:59:34+00:00	1377047884101656576	Guess What Just Happened to The Latest SpaceX ...	Pauloaep	SpaceX
3	2021-03-30 23:59:31+00:00	1377047870424104964	@WhatAFool6 @PushTheFrontier @TooChill_Javi No...	RubenJHenriquez	SpaceX
4	2021-03-30 23:59:17+00:00	1377047811057979394	Guess What Just Happened to The Latest SpaceX ...	ScienceAlert	SpaceX

In [7]:
```
# Save dataframe to CSV
tweets_df.to_csv('tweets.csv', index = False, encoding = 'utf-8')
```

In [11]:
```
# View length of dataset
original = tweets_df.shape[0]
print("There are", original, "rows of data.")
```

```
There are 285132 rows of data.
```

Data Extraction and Preparation

- The first step in the data preparation is to add a column for the sector of each entity, commercial or government. This was accomplished by applying a simple function using an if statement to test the name of the entity.

In [12]:
```
# Define a function to return the sector based on the entity name
def sector(name):
    return_type = ''
    if name == 'SpaceX' or name == 'Virgin Galactic' or name == 'Blue Origin':
        return_type = 'Commercial'
    else: return_type = 'Government'
    return return_type
```

In [13]:
```
# Apply sector function to each row of the dataframe and view head
tweets_df['Type'] = tweets_df['Company'].apply(sector)
tweets_df.head()
```

Out[13]:

	Datetime	Tweet Id	Text	Username	Company	Type
0	2021-03-30 23:59:47+00:00	1377047938938060803	@danahjon @NASAKennedy @ArceneauxHayley @DrSia...	rookisaacman	SpaceX	Commercial
1	2021-03-30 23:59:38+00:00	1377047902044872706	@SpaceTfrs So grateful for your contribution t...	BlueJ11274903	SpaceX	Commercial
2	2021-03-30 23:59:34+00:00	1377047884101656576	Guess What Just Happened to The Latest SpaceX ...	Pauloaep	SpaceX	Commercial
3	2021-03-30 23:59:31+00:00	1377047870424104964	@WhatAFool6 @PushTheFrontier @TooChill_Javi No...	RubenJHenriquez	SpaceX	Commercial
4	2021-03-30 23:59:17+00:00	1377047811057979394	Guess What Just Happened to The Latest SpaceX ...	ScienceAlert	SpaceX	Commercial

- Next, any rows with duplicate text were dropped from the dataframe.

In [14]:
```
# Investigate duplicates
print(tweets_df[tweets_df.duplicated(subset='Text')])
```

```
                       Datetime             Tweet Id  \
325    2021-03-30 23:19:26+00:00  1377037785962516484
564    2021-03-30 22:56:11+00:00  1377031932182167554
837    2021-03-30 22:36:46+00:00  1377027045268598786
851    2021-03-30 22:36:06+00:00  1377026880864403456
923    2021-03-30 22:33:03+00:00  1377026111104806914
...                          ...                  ...
284903 2019-04-24 01:50:16+00:00  1120867521614073856
284999 2019-04-23 19:30:15+00:00  1120771888517541888
285025 2019-04-23 18:25:17+00:00  1120755537845477376
285099 2019-04-23 13:07:25+00:00  1120675542875787264
285109 2019-04-23 12:07:16+00:00  1120660406786072576

                                                     Text        Username  \
325     @jdeshetler @NASASpaceflight @SpaceX @BocaChic...    BeirMauricio
564     RT elonmusk: @jdeshetler @NASASpaceflight @Spa...   someguyjimmy1
837     @elonmusk @jdeshetler @NASASpaceflight @SpaceX...      creamcheez
851     @elonmusk @jdeshetler @NASASpaceflight @SpaceX...     Somoschile1
923     elonmusk: @jdeshetler @NASASpaceflight @SpaceX...  ElonMuskLegacy
...                                                   ...             ...
284903  アメリカ航空宇宙局のジェット推進研究所の動画などをお楽しみ下さい。\nNASA Jet Pr...  someresearcher
284999  星空・・・\n良い物ですよ！\nお天気が良かったら窓を開けて観てみませんか？\n↓\nhtt...  someresearcher
285025                                @realDonaldTrump 🤔          dr_jpl
285099  Enjoy the video of the NASA JPL.\nNASA Jet Pro...  someresearcher
285109  天体観測情報 (NASAHubbleSite,NASA Jet Propulsion Labo...  someresearcher

       Company        Type
325     SpaceX  Commercial
564     SpaceX  Commercial
837     SpaceX  Commercial
851     SpaceX  Commercial
923     SpaceX  Commercial
...        ...         ...
284903     JPL  Government
284999     JPL  Government
285025     JPL  Government
285099     JPL  Government
285109     JPL  Government

[8998 rows x 6 columns]
```

In [15]:
```
# Remove duplicate text and save in a new df
df_clean = tweets_df.drop_duplicates(subset='Text')
```

In [16]:
```
# Look at info
df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 276134 entries, 0 to 285131
Data columns (total 6 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   Datetime  276134 non-null  datetime64[ns, UTC]
 1   Tweet Id  276134 non-null  int64
 2   Text      276134 non-null  object
 3   Username  276134 non-null  object
 4   Company   276134 non-null  object
 5   Type      276134 non-null  object
dtypes: datetime64[ns, UTC](1), int64(1), object(4)
memory usage: 14.7+ MB
```

- The next step was to convert the "Datetime" column to a datetime data type.

In [20]:
```
# Convert Datetime column to Datetime type
df_clean['Datetime']= pd.to_datetime(df_clean['Datetime'])
df_clean.dtypes
```

Out[20]:
```
Datetime    datetime64[ns, UTC]
Tweet Id                  int64
Text                     object
Username                 object
Company                  object
Type                     object
dtype: object
```

- To eliminate bias, rows with tweets from the official accounts of the entities were dropped from the dataframe. Upon inspection, this involved less than 200 tweets.

In [21]:
```
# Check for tweets from official entity Twitter accounts
print("The number of tweets from the official SpaceX account: ", len(df_clean[df_clean['Username'] == 'SpaceX'
]))
print("The number of tweets from the official Virgin Galactic account: ", len(df_clean[df_clean['Username'] ==
                                                                                'virgingalactic']))
print("The number of tweets from the official Blue Origin account: ", len(df_clean[df_clean['Username'] == 'bl
ueorigin']))
print("The number of tweets from the official NASA account: ", len(df_clean[df_clean['Username'] == 'NASA']))
print("The number of tweets from the official ESA account: ", len(df_clean[df_clean['Username'] == 'esaoperati
ons']))
print("The number of tweets from the official JPL account: ", len(df_clean[df_clean['Username'] == 'NASAJPL'
]))
```

```
The number of tweets from the official SpaceX account:  0
The number of tweets from the official Virgin Galactic account:  67
The number of tweets from the official Blue Origin account:  3
The number of tweets from the official NASA account:  27
The number of tweets from the official ESA account:  1
The number of tweets from the official JPL account:  17
```

In [22]:
```
# Get names of indexes for which column Username is one of the studied entities
indexNames = df_clean[(df_clean['Username'] == 'SpaceX') | (df_clean['Username'] == 'virgingalactic')
                      | (df_clean['Username'] == 'blueorigin') | (df_clean['Username'] == 'NASA')
                      | (df_clean['Username'] == 'esaoperations') | (df_clean['Username'] == 'NASAJPL')].index
# Delete these row indexes from dataFrame
df_clean.drop(indexNames , inplace=True)
```

- The text for the tweets was then cleaned. Mentions (ex. @NASA), hashtags (ex. #NASA), URL links, non-alphabetic characters, and retweets were removed. This was accomplished with the use of regular expression syntax in a user-defined function. This function was then applied to the “Text” column of each row in the data frame.

In [26]:
```
# Cleaning the tweets
import re # import regex

def cleanUpTweet(txt):
    # Remove mentions
    txt = re.sub(r'@[A-Za-z0-9_]+', '', txt)
    # Remove hashtags
    txt = re.sub(r'#', '', txt)
    # Remove retweets:
    txt = re.sub(r'RT : ', '', txt)
    # Remove other undesirable text
    txt = re.sub(r'b\'', '', txt)
    # Remove urls
    txt = re.sub(r'https?:\/\/[A-Za-z0-9\.\/]+', '', txt)
    return txt
```

In [27]:
```
# Apply cleaning method to each tweet
df_clean['Text'] = df_clean['Text'].apply(cleanUpTweet)
```

In [28]:
```
# View head
df_clean.head()
```

Out[28]:

	Datetime	Tweet Id	Text	Username	Company	Type
0	2021-03-30 23:59:47+00:00	1377047938938060803	Frak.	rookisaacman	SpaceX	Commercial
1	2021-03-30 23:59:38+00:00	1377047902044872706	So grateful for your contribution to helping ...	BlueJ11274903	SpaceX	Commercial
2	2021-03-30 23:59:34+00:00	1377047884101656576	Guess What Just Happened to The Latest SpaceX ...	Pauloaep	SpaceX	Commercial
3	2021-03-30 23:59:31+00:00	1377047870424104964	Not sci-fi at all. I get there is a weird f...	RubenJHenriquez	SpaceX	Commercial
4	2021-03-30 23:59:17+00:00	1377047811057979394	Guess What Just Happened to The Latest SpaceX ...	ScienceAlert	SpaceX	Commercial

- After applying the data cleaning function, some of the “Text” fields were left empty. Therefore, those rows were dropped from the dataframe.

In [42]:
```
# Ensure there are no zero value columns after cleaning
print("There are", len(df_clean[df_clean['Text'] == 0]), "records with a value of zero in the 'Text' field.")
```

```
There are 0 records with a value of zero in the 'Text' field.
```

In [43]:
```
# Ensure there are no empty value columns after cleaning
len(df_clean[df_clean['Text'] == ''])
print("There are", len(df_clean[df_clean['Text'] == '']), "records with an empty 'Text' field.")
```

```
There are 51 records with an empty 'Text' field.
```

In [44]:
```
# Drop rows with empty text column
df_clean = df_clean.drop(df_clean[df_clean['Text'] == ''].index)
```

- Next I looked at how many rows are left after cleaning

In [45]:
```
# Calculate how many rows were affected by the cleaning
clean = df_clean.shape[0]
diff = original - clean
print("There were", original, "rows of data in the original dataset.")
print("There were", diff, "rows removed during cleaning, resulting in", clean, "rows of data to be analyzed.")
```

```
There were 285132 rows of data in the original dataset.
There were 9164 rows removed during cleaning, resulting in 275968 rows of data to be analyzed.
```

- Finally, a wordcloud is generated to see the words that occur most in the "Text" column of the dataset.

In [84]:
```
# Build list of words for wordcloud
comment_words = ''
for i in df_clean.Text:
    i = str(i)
    separate = i.split()
    for j in range(len(separate)):
        separate[j] = separate[j].lower()

    comment_words += " ".join(separate)+" "
```

In [85]:
```
# View WordCloud
from wordcloud import WordCloud, STOPWORDS
from PIL import Image


mask = np.array(Image.open('rocket2.jpg'))
wc = WordCloud(stopwords=STOPWORDS, mask=mask, background_color="black", max_words=500, max_font_size=256,
               random_state=42, width=mask.shape[1], height=mask.shape[0])
wc.generate(comment_words)

plt.figure(figsize=(10,15))
plt.imshow(wc, interpolation="bilinear")
plt.tight_layout(pad = 0)
plt.axis('off');
```

Applying Sentiment Labels

The final step to prepare the data for analysis was to add the sentiment labels. This was accomplished using TextBlob, which is a library that can be used for NLP tasks including calculating the polarity of text. First, TextBlob was called in a user-defined function to assign a polarity score to the "Polarity" column of each tweet. This score has a range of negative one to one (-1 to 1). Then, another user-defined function was created to assign each score to a text label. Negative polarity values were assigned a "negative" label, positive values were given the label "positive", and a score of zero was assigned the label "neutral".

In [46]:
```
# Calculate polarity of each tweet
from textblob import TextBlob

def getTextPolarity(txt):
    return TextBlob(txt).sentiment.polarity
```

In [47]:
```
# Add the polarity values to each tweet
df_clean['Polarity'] = df_clean['Text'].apply(getTextPolarity)
```

In [48]:
```
# Analyze text for negative, neutral, positive sentiment
def getTextAnalysis(a):
    if a < 0:
        return "Negative"
    elif a == 0:
        return "Neutral"
    else:
        return "Positive"
```

In [49]:
```
# Apply sentiment label to each text
df_clean['Score'] = df_clean['Polarity'].apply(getTextAnalysis)
df_clean_clean.head()
```

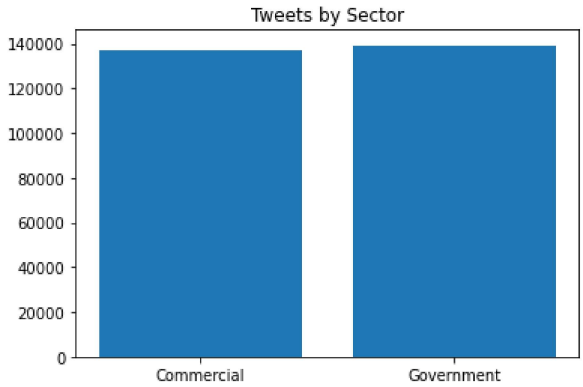
Out[49]:

	Datetime	Tweet Id	Text	Username	Company	Type	Polarity	Score
0	2021-03-30 23:59:47+00:00	1377047938938060803	Frak.	rookisaacman	SpaceX	Commercial	0.0000	Neutral
1	2021-03-30 23:59:38+00:00	1377047902044872706	So grateful for your contribution to helping ...	BlueJ11274903	SpaceX	Commercial	0.0000	Neutral
2	2021-03-30 23:59:34+00:00	1377047884101656576	Guess What Just Happened to The Latest SpaceX ...	Pauloaep	SpaceX	Commercial	0.5000	Positive
3	2021-03-30 23:59:31+00:00	1377047870424104964	Not sci-fi at all. I get there is a weird f...	RubenJHenriquez	SpaceX	Commercial	-0.3725	Negative
4	2021-03-30 23:59:17+00:00	1377047811057979394	Guess What Just Happened to The Latest SpaceX ...	ScienceAlert	SpaceX	Commercial	0.5000	Positive

Visualizing Sentiment Distribution

- After cleaning and preparing the data, the number of tweets per sector is fairly even. There are approximately 2,500 more tweets in the data labeled as “Government”, but since the total dataset contains nearly 300,000 tweets, this is less than a 1% difference, therefore it should not affect the results of the analysis.

In [50]:
```
# Plot overall count of tweets by sector
labels = df_clean.groupby('Type').count().index.values
values = df_clean.groupby('Type').size().values
plt.bar(labels, values)
plt.title('Tweets by Sector');
```

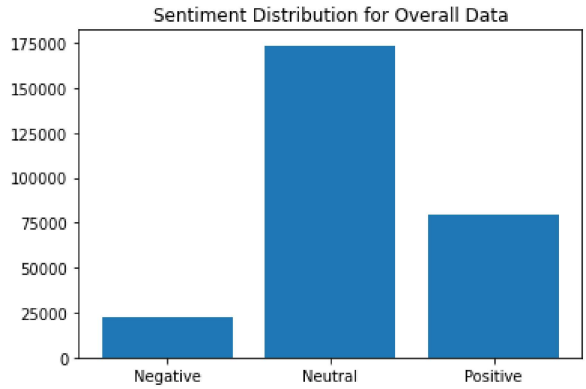


In [51]:
```
# Print value counts for each sentiment label
for i in range(len(labels)):
    print(labels[i], values[i])
```

```
Commercial 136768
Government 139200
```

- In viewing the overall sentiment on a bar chart, it seems that the majority of the text in the tweets is classified as neutral sentiment. In terms of polarized sentiment, positive sentiment occurs much more often than negative sentiment.

In [61]:
```
# Plot overall sentiment values
labels = df_clean.groupby('Score').count().index.values
values = df_clean.groupby('Score').size().values
plt.bar(labels, values)
plt.title('Sentiment Distribution for Overall Data');
```



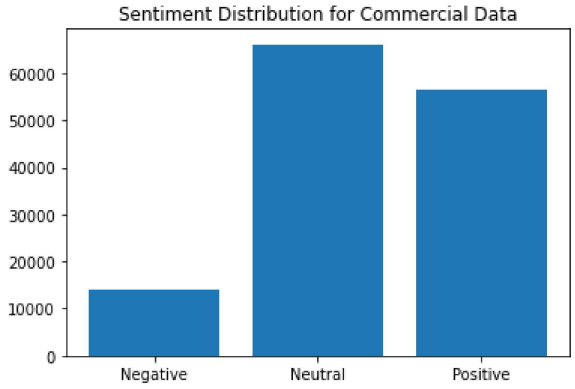
In [62]:
```
# Print value counts for each sentiment label
for i in range(len(labels)):
    print(labels[i], values[i])
```

```
Negative 22256
Neutral 173806
Positive 79906
```

- When looking at each sector individually, there are more instances of neutral sentiment than other categories. In the commercial sector, however, positive sentiment is nearly equal to neutral sentiment and much greater than negative sentiment. In the government sector, the neutral category is far greater than the other two labels.

In [63]:
```
# Create dataframes for government and commercial
df_gov = df_clean[df_clean['Type'] == 'Government']
df_com = df_clean[df_clean['Type'] == 'Commercial']
```

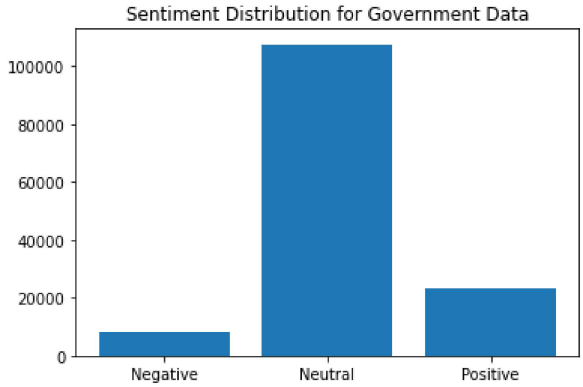
In [64]:
```
# Plot overall sentiment values for commercial sector
labels = df_com.groupby('Score').count().index.values
values = df_com.groupby('Score').size().values
plt.bar(labels, values)
plt.title('Sentiment Distribution for Commercial Data');
```



In [65]:
```
# Print value counts for each sentiment label
for i in range(len(labels)):
    print(labels[i], values[i])
```

```
Negative 14132
Neutral 66154
Positive 56482
```

In [66]:
```
# Plot overall sentiment values for government sector
labels = df_gov.groupby('Score').count().index.values
values = df_gov.groupby('Score').size().values
plt.bar(labels, values)
plt.title('Sentiment Distribution for Government Data');
```

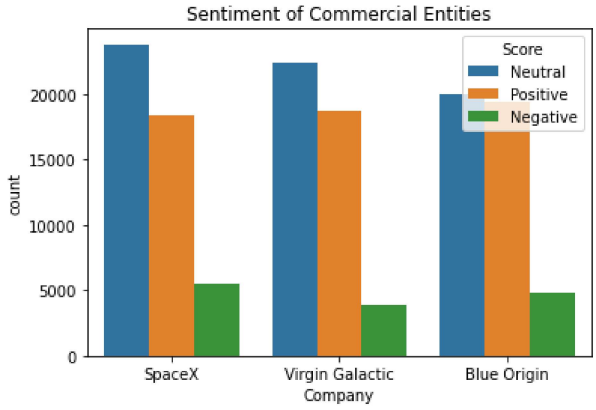


In [67]:
```
# Print value counts for each sentiment label
for i in range(len(labels)):
    print(labels[i], values[i])
```

```
Negative 8124
Neutral 107652
Positive 23424
```

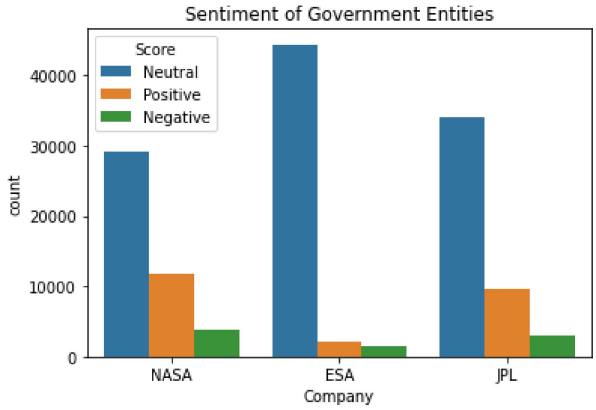
- Each entity in the commercial sector seems pretty evenly matched, with positive sentiment far exceeding negative sentiment. In each case, the positive sentiment is very close in number to the tweets labeled neutral.

In [68]:
```
# Plot Sentiment Values By Commercial Entity
sns.countplot(x="Company", hue="Score", data=df_com).set_title("Sentiment of Commercial Entities");
```



- The government sector appears to be much different from the commercial sector in its sentiment distribution. Overall, there is much less polarized sentiment than neutral sentiment for each entity.

In [69]:
```
# Plot Sentiment Values By Government Entity
sns.countplot(x="Company", hue="Score", data=df_gov).set_title("Sentiment of Government Entities");
```

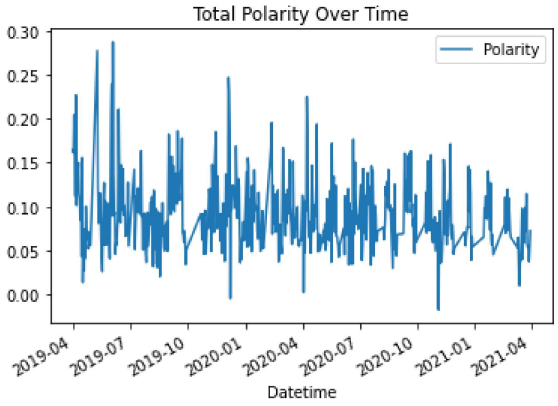


Visualizing Sentiment over Time

- The polarity score can also be viewed over time to get an idea if there are any trends in changing sentiment. To accomplish this, the data is grouped by day and then the mean is calculated. Since the 2,000 tweet cutoff may limit the range of the data, some days may have no data associated with it and therefore any null values for the mean are dropped. For the overall tweet dataframe, there do not appear to be any trends, and the distribution appears random. The overall daily scores seem to stay above zero.

In [70]:
```
# Calculate daily average polarity for all tweets
df_time = df_clean.groupby(pd.Grouper(key='Datetime', freq="D")).mean().dropna()
```

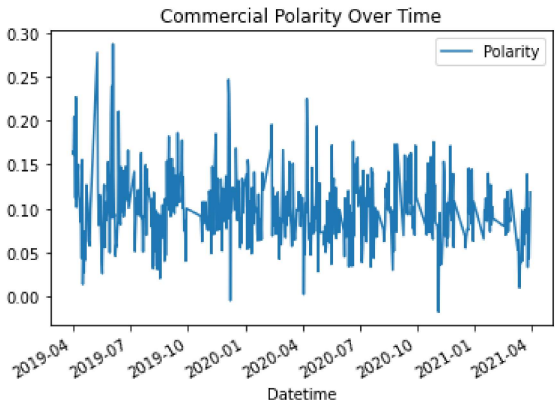
In [71]:
```
# Plot total polarity over time
df_time.plot(y='Polarity')
plt.title('Total Polarity Over Time');
```



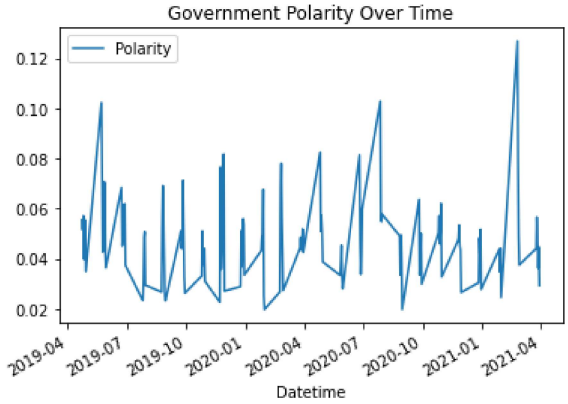
- The average daily sentiment is then calculated for each sector. There appear to be no trends when looking at the commercial sector polarity scores over time as the distribution appears random and above zero. The government distribution appears to be less noisy, but this could be due to having a smaller range for polarity score than the commercial sector.

In [72]:
```
# Calculate daily average polarity for each sector
df_com_time = df_com.groupby(pd.Grouper(key='Datetime', freq="D")).mean().dropna()
df_gov_time = df_gov.groupby(pd.Grouper(key='Datetime', freq="D")).mean().dropna()
```

In [73]:
```
# Plot commercial polarity over time
df_com_time.plot(y='Polarity')
plt.title('Commercial Polarity Over Time');
```



In [74]:
```
# Plot government polarity over time
df_gov_time.plot(y='Polarity')
plt.title('Government Polarity Over Time');
```

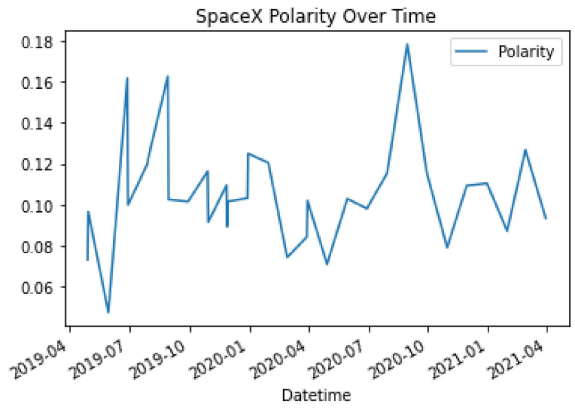


- Next, the sentiment over time for each entity was calculated and plotted. This was done by first breaking the data into separate data frames by each entity. Then the average polarity by day was calculated and any null values dropped. The SpaceX, NASA, and ESA plots are less noisy than Virgin Galactic, Blue Origin, and JPL. This means that those entities with less noisy plots may experience trends in polarity. ESA stands out as having more negative average daily sentiment than any of the other entities, all of which have mostly positive daily sentiment.

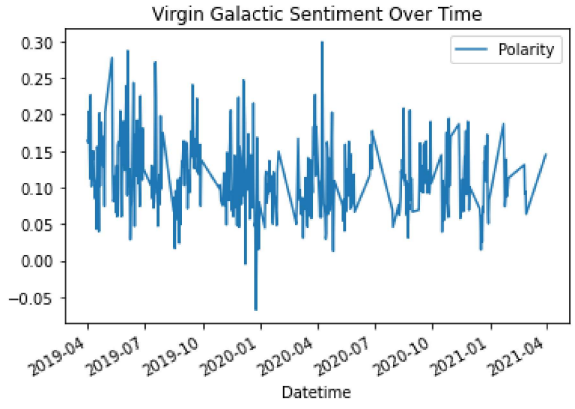
In [75]:
```
# Separate data by entity
df_com_spa = df_com[df_com['Company'] == 'SpaceX']
df_com_vir = df_com[df_com['Company'] == 'Virgin Galactic']
df_com_blu = df_com[df_com['Company'] == 'Blue Origin']
df_gov_nas = df_gov[df_gov['Company'] == 'NASA']
df_gov_esa = df_gov[df_gov['Company'] == 'ESA']
df_gov_jpl = df_gov[df_gov['Company'] == 'JPL']
```

In [76]:
```
# Calculate daily average polarity for each entity
df_com_spa_time = df_com_spa.groupby(pd.Grouper(key='Datetime', freq="D")).mean().dropna()
df_com_vir_time = df_com_vir.groupby(pd.Grouper(key='Datetime', freq="D")).mean().dropna()
df_com_blu_time = df_com_blu.groupby(pd.Grouper(key='Datetime', freq="D")).mean().dropna()
df_gov_nas_time = df_gov_nas.groupby(pd.Grouper(key='Datetime', freq="D")).mean().dropna()
df_gov_esa_time = df_gov_esa.groupby(pd.Grouper(key='Datetime', freq="D")).mean().dropna()
df_gov_jpl_time = df_gov_jpl.groupby(pd.Grouper(key='Datetime', freq="D")).mean().dropna()
```

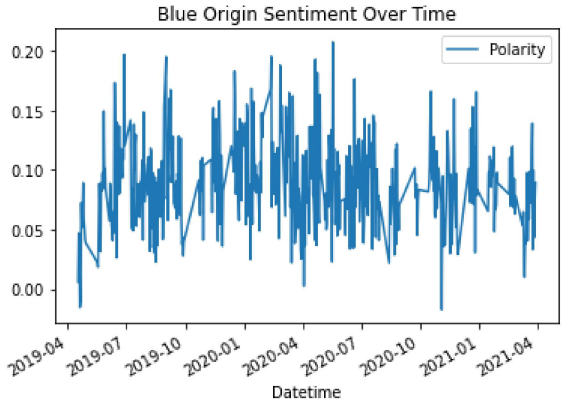
In [77]:
```
# Plot SpaceX polarity over time
df_com_spa_time.plot(y='Polarity')
plt.title('SpaceX Polarity Over Time');
```



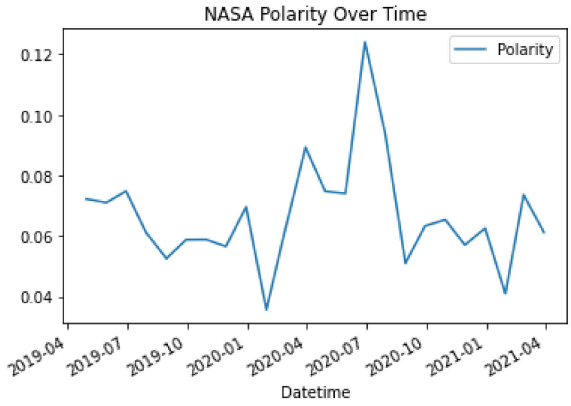
In [78]:
```
# Plot Virgin Galactic polarity over time
df_com_vir_time.plot(y='Polarity')
plt.title('Virgin Galactic Sentiment Over Time');
```



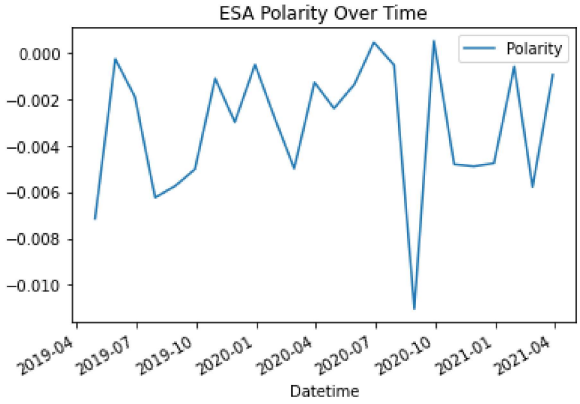
In [79]:
```
# Plot Blue Origin polarity over time
df_com_blu_time.plot(y='Polarity')
plt.title('Blue Origin Sentiment Over Time');
```



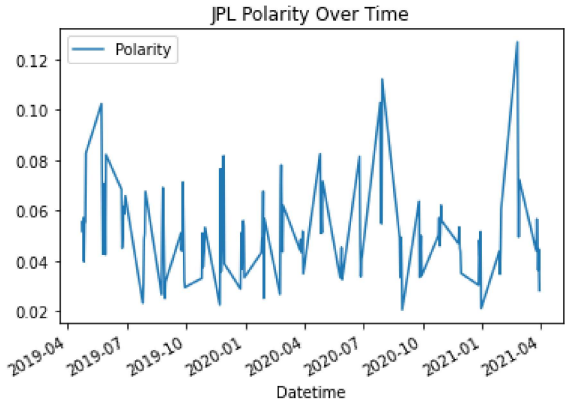
In [80]:
```
# Plot NASA polarity over time
df_gov_nas_time.plot(y='Polarity')
plt.title('NASA Polarity Over Time');
```



In [81]:
```
# Plot ESA polarity over time
df_gov_esa_time.plot(y='Polarity')
plt.title('ESA Polarity Over Time');
```



In [82]:
```
# Plot JPL polarity over time
df_gov_jpl_time.plot(y='Polarity')
plt.title('JPL Polarity Over Time');
```



Chi-square Analysis

Hypothesis testing will be done using the Chi-square test for independence. This test compares two categorical variables to determine if the distributions vary from each other. The test statistic is calculated using the observed and expected values from a frequency table. This statistic is a single value that describes the amount of difference between the observed frequency and the frequency that is expected if there were no relationship. The test statistic can be used to calculate a p-value to determine the significance of the relationship between the variables.

- The first step in the Chi-square analysis is to create a contingency table, which is the frequency of each sentiment label broken out by sector. This is done by using the pandas.crosstab() function.

In [86]:
```
# Perform Chi-square analysis
from scipy.stats import chi2_contingency

# Create contingency table
contigency= pd.crosstab(df_clean['Type'], df_clean['Score'])
contigency
```

Out[86]:

Score	Negative	Neutral	Positive
Type			
Commercial	14132	66154	56482
Government	8124	107652	23424

- This can also be viewed in terms of a percentage of sentiment for each sector. This is done by normalizing the data in the table created using the pandas.crosstab() function.

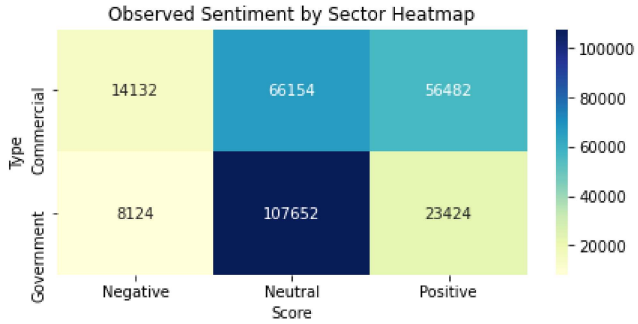
In [87]:
```
# Calculate sentiment percentage by sector
contigency_pct = pd.crosstab(df_clean['Type'], df_clean['Score'], normalize='index')
round(contigency_pct, 2)
```

Out[87]:

Score	Negative	Neutral	Positive
Type			
Commercial	0.10	0.48	0.41
Government	0.06	0.77	0.17

- The table can also be viewed as a heatmap

In [88]:
```
# View contingency table on a heatmap
plt.figure(figsize=(7,3))
sns.heatmap(contigency, annot=True, cmap='YlGnBu', fmt='g').set_title("Observed Sentiment by Sector Heatmap");
```



- Next, the chi-square test for independence was performed. This step created variables for the test statistic, p-value, degrees of freedom, and the expected values array. The p-value was then printed, as this value is used to determine the significance of the difference between the data for both sectors. This value is the determining factor on whether the null hypothesis is accepted or rejected. The p-value of 0.0 is statistically significant, therefore the null hypothesis can be rejected.

In [89]:
```
# Chi-square test of independence
c, p, dof, expected = chi2_contingency(contigency)
# Print the p-value
print("The p-value of the Chi-square analysis is:", p)
```

```
The p-value of the Chi-square analysis is: 0.0
```

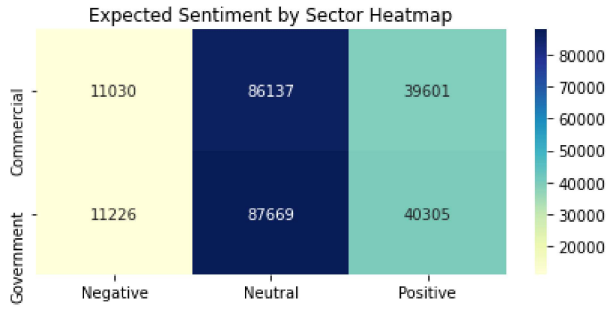
- Finally, the expected values table and heatmap are displayed as a comparison to the contingency table values.

In [91]:
```
# Create expected values table
expected_tbl = pd.DataFrame(data=np.round(expected, decimals=0), index=['Commercial', 'Government'],
                            columns=['Negative', 'Neutral', 'Positive'])
expected_tbl.astype(int)
```

Out[91]:

	Negative	Neutral	Positive
Commercial	11030	86137	39601
Government	11226	87669	40305

In [92]:
```
# View expected table on a heatmap
plt.figure(figsize=(7,3))
sns.heatmap(expected_tbl, annot=True, cmap='YlGnBu', fmt='g').set_title("Expected Sentiment by Sector Heatmap"
);
```



Data Summary and Implications

A chi-square p-value of less than five percent indicates that the sentiment values for the government and commercial space sectors are significantly different. The chi-square test returned a value of 0.0, which is grounds to reject the null hypothesis “There is no difference in sentiment between commercial and government entities”. As seen in the "Sentiment Distribution for Government Data" barplot above, government sentiment seems to be overwhelmingly neutral. The commercial sentiment seems to be a bit more polarized, but as the "Sentiment Distribution for Commercial Data" barplot shows, that sentiment tends to be more positive.

Limitations of Study

One limitation of this study is in the way the data was gathered. The method used to extract the data from Twitter did not allow the ability to choose the monthly sampling at random. Some entities may be mentioned more often, therefore the 2,000 tweet limit may only capture a period of a few days. The odd number of records in the collected data means that some entities may not have had 2,000 tweets in particular months. Without having an equal amount of tweets for each entity dispersed randomly throughout each month, the analysis could be impacted by bias.

This study is also limited by the small list of entities studied. There are many more nations, government agencies, and private sector companies involved in the exploration of space.

Proposed Actions

One recommendation for future studies would be to include more than just the six entities used in this study. This would provide a more well-rounded world view of public interest in the exploration of space. Another proposal for future studies would be to gather data not just from Twitter but to include other social media networks. Different types of people tend to have different preferences on the preferred social media platform, therefore adding more information would add to the perspective.

Expected Benefits of Study

I would encourage groups involved in the space domain to use the methods discussed in the study to conduct a similar analysis periodically to get a feel for public opinion both at an industry and organizational level. Since both government space programs and private corporations rely on the use of money from taxpayers to continue operation, it could be advantageous to monitor public attitudes. They can then use the results to develop methods and programs to foster public enthusiasm. This may result in less cuts to funding for future space exploration.