Correlation between stocks and sentiment analysis:

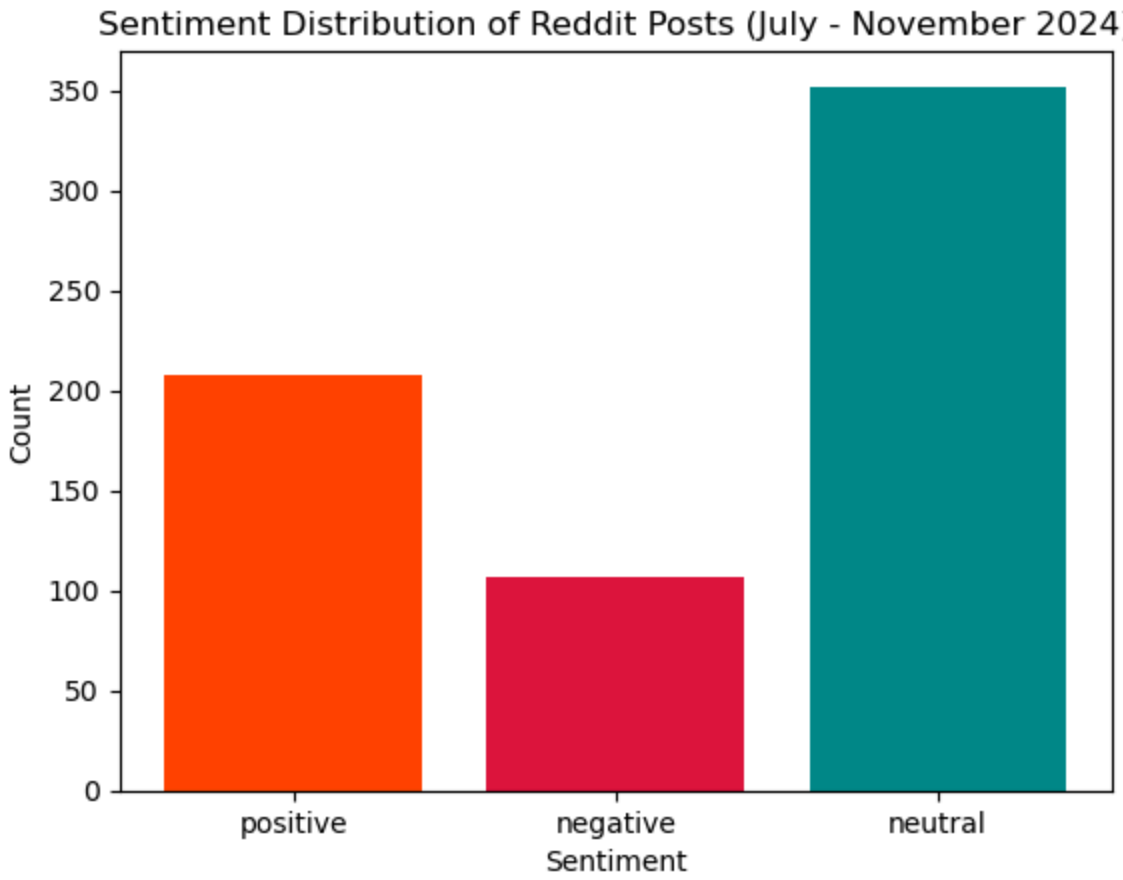
Analysis by month from TRUMP's Nomination for the elections till the election results.

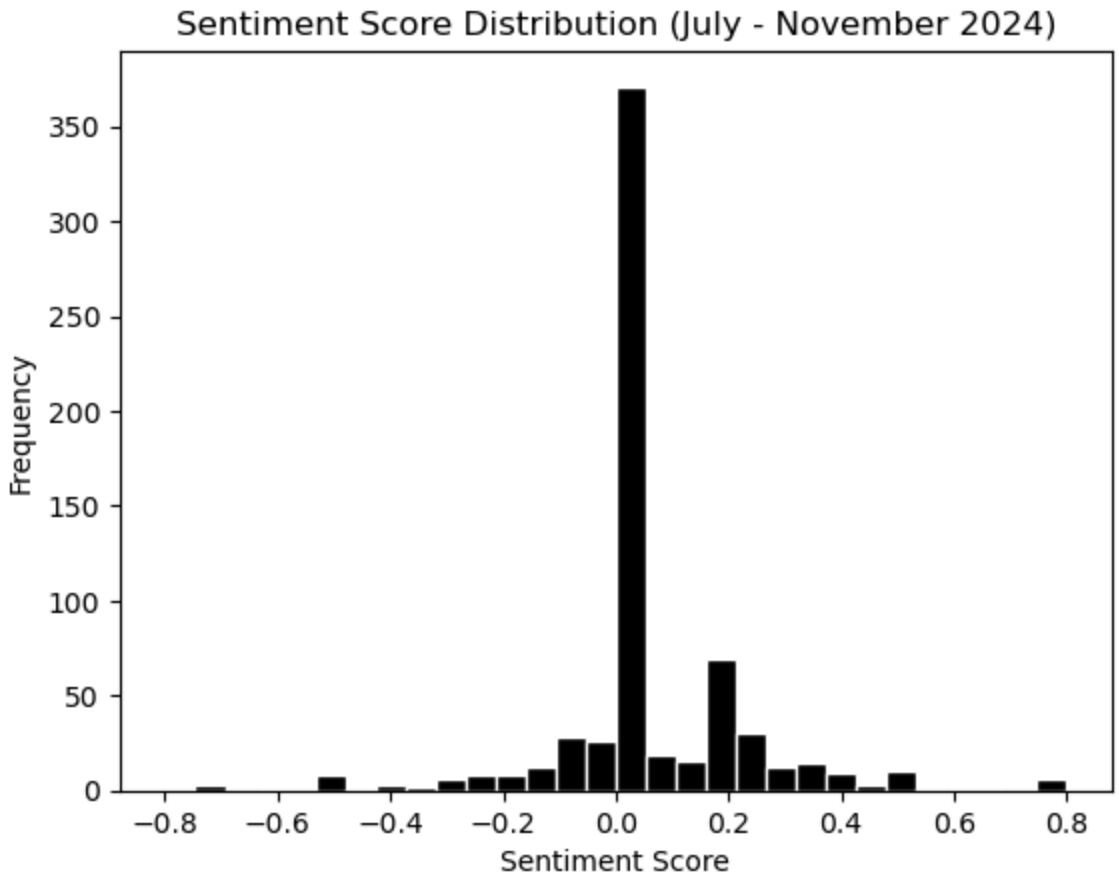
In [29]:

```
import pandas as pd
import json
from datetime import datetime
from textblob import TextBlob
import matplotlib.pyplot as plt

# Load Reddit data
with open("2cleaned_defence_API_Reddit_with_dates.json") as file:
    reddit_data = json.load(file)

# Define the date range for filtering
start_date = datetime(2024, 7, 1).date()  # Month of Trump's Nomination for Presidential Run
end_date = datetime(2024, 11, 30).date()

# Filter Reddit posts for the specified date range
filtered_reddit_data = [
    post for post in reddit_data
    if start_date <= datetime.strptime(post['date'], '%Y-%m-%d').date() <= end_date
]

# Sentiment analysis function using TextBlob
def analyze_sentiment(text):
    blob = TextBlob(text)
    sentiment_score = blob.sentiment.polarity  # Returns a sentiment score between -1 and 1
    if sentiment_score > 0:
        return 'positive', sentiment_score
    elif sentiment_score < 0:
        return 'negative', sentiment_score
    else:
        return 'neutral', sentiment_score

# Apply sentiment analysis to Reddit posts using the 'title' key
for post in filtered_reddit_data:
    sentiment, score = analyze_sentiment(post['title'])
    post['sentiment'] = sentiment
    post['sentiment_score'] = score

# Count sentiment types
sentiment_counts = {"positive": 0, "negative": 0, "neutral": 0}
sentiment_scores = []

for post in filtered_reddit_data:
    sentiment_counts[post['sentiment']] += 1
    sentiment_scores.append(post['sentiment_score'])

# Calculate average sentiment score
average_sentiment_score = sum(sentiment_scores) / len(sentiment_scores) if sentiment_scores else 0
```

```python
# Display sentiment analysis results
print(f"Sentiment Analysis on Reddit Posts from July 1 to November 30, 2024: {sentiment_counts}")
print(f"Average Sentiment Score: {average_sentiment_score:.2f}")

# Visualizing the sentiment distribution
sentiments = list(sentiment_counts.keys())
counts = list(sentiment_counts.values())

# Bar plot of sentiment distribution
plt.bar(sentiments, counts, color=['orangered', 'crimson', 'darkcyan'])
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.title('Sentiment Distribution of Reddit Posts (July - November 2024)')
plt.show()

# Sentiment score distribution (Optional - This will help you understand the intensity)
plt.hist(sentiment_scores, bins=30, color='black', edgecolor='white')
plt.title('Sentiment Score Distribution (July - November 2024)')
plt.xlabel('Sentiment Score')
plt.ylabel('Frequency')
plt.show()
```

```
Sentiment Analysis on Reddit Posts from July 1 to November 30, 2024: {'positive': 208, 'negative': 107, 'neutral': 35
2}
Average Sentiment Score: 0.04
```


Sentiment Distribution of Reddit Posts (July - November 2024)
Count
350
300
250
200
150
100
50
0
positive
negative
neutral
Sentiment

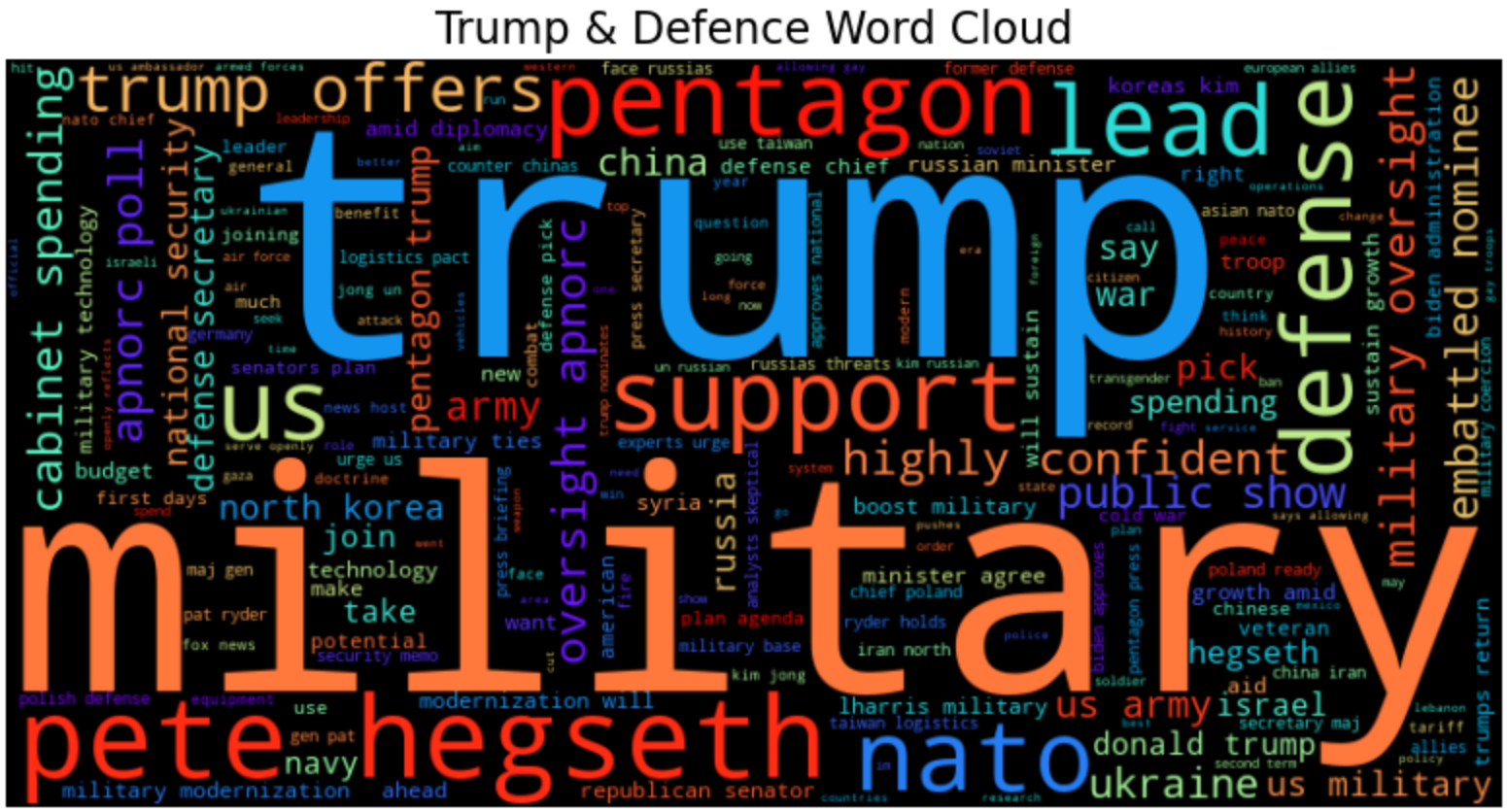


Sentiment Word Cloud

In [30]:

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

# Load JSON file (Make sure the path is correct)
file_path = 'cleaned_defence_API_Reddit_with_dates.json'
with open(file_path, 'r') as f:
    data = json.load(f)

# Define USA, Defense, and Trump-related keywords
defense_keywords = [
    'military', 'defense', 'national defense', 'us military', 'pentagon', 'army',
    'navy', 'air force', 'us army', 'defense spending', 'national security', 'defense policy',
    'military budget', 'defense contractors', 'armed forces', 'veterans', 'defense strategy',
    'trump military', 'trump defense policy', 'trump national security', 'trump military budget',
    'trump pentagon', 'trump military spending', 'us defense', 'us army', 'us navy',
    'military power', 'military strategy', 'military readiness', 'global military presence',
    'nuclear weapons', 'missile defense', 'defense industry', 'homeland security',
    'border security', 'trump homeland security', 'military action', 'military conflict',
    'cyber defense', 'trump military action', 'military alliances', 'nato', 'global security',
    'terrorism', 'trump terrorism policy', 'military innovation', 'military technology',
    'war on terror', 'military aid', 'defense cooperation', 'defense treaties',
    'military personnel', 'military spending', 'veterans affairs', 'trump veterans'
]


# Extract and filter text based on keywords
def filter_relevant_text(entries, keywords):
    """
    Extracts and filters text data based on given keywords.
    """
    filtered_texts = []
    for entry in entries:
        text = entry.get('title', '').lower()  # Assuming 'title' holds the text
        if any(keyword in text for keyword in keywords):  # Check if any keyword exists in text
            filtered_texts.append(text)
    return filtered_texts

# Filter data
filtered_texts = filter_relevant_text(data, defense_keywords)

# Combine all filtered text into a single string
filtered_text = " ".join(filtered_texts)

# Optional: Clean the text (remove URLs, special characters, etc.)
def clean_text(text):
```

```
    text = re.sub(r'http\S+', '', text)  # Remove URLs
    text = re.sub(r'[^a-zA-Z\s]', '', text)  # Remove special characters
    return text.lower()  # Convert to lowercase

cleaned_text = clean_text(filtered_text)

# Generate the word cloud
wordcloud = WordCloud(width=800, height=400, background_color='black', colormap='rainbow').generate(cleaned_text)

# Display the word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title("Trump & Defence Word Cloud", fontsize=16)
plt.show()
```



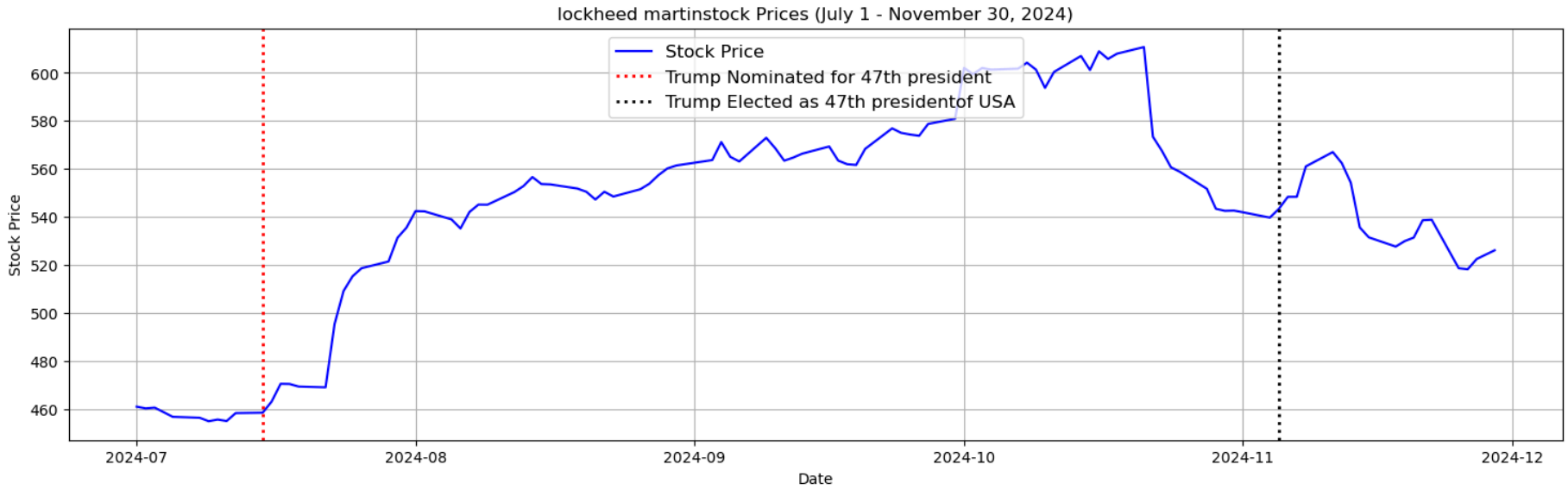
Stock Analysis:

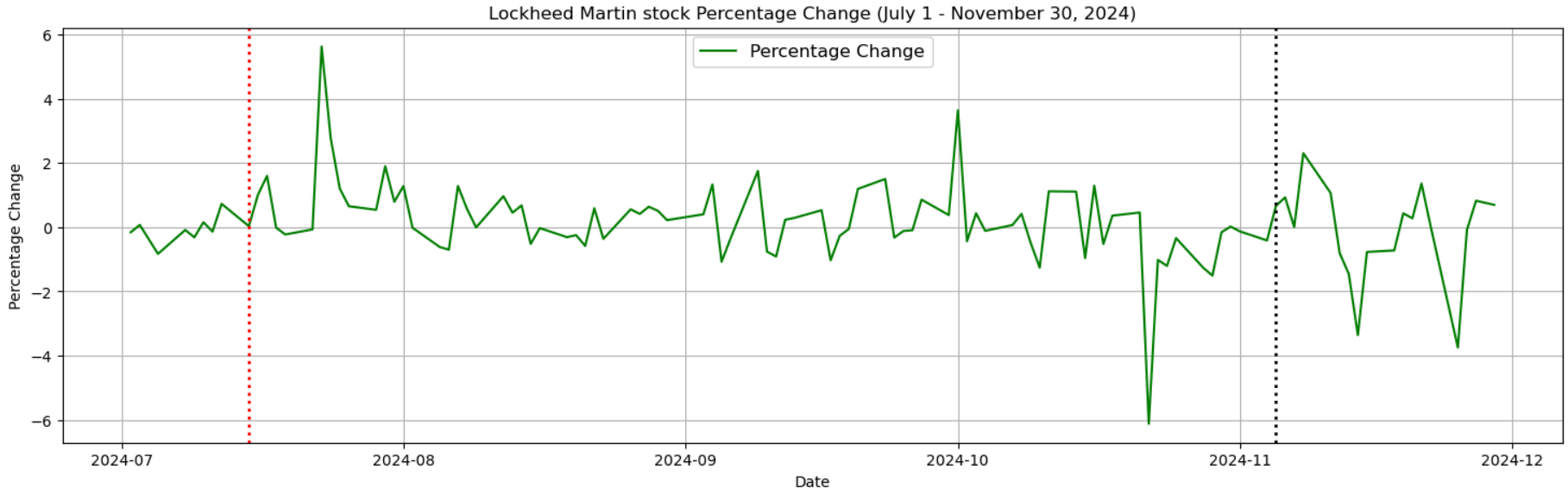
In [10]:

```
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime

# Load stock data for JPMorgan (Assuming you have already loaded this)
stock_data = pd.read_csv("lockheed_martin_stock_data.csv")  # Replace with your actual file path

# Ensure Date column is properly converted to datetime
stock_data['Date'] = pd.to_datetime(stock_data['Date'], errors='coerce')

# Check for invalid dates and drop them
if stock_data['Date'].isnull().any():
    print("Some rows have invalid dates and will be dropped.")
stock_data = stock_data.dropna(subset=['Date'])

# Remove timezone information to make the Date column timezone-naive
stock_data['Date'] = stock_data['Date'].apply(lambda x: x.replace(tzinfo=None) if x.tzinfo else x)

# Define the date range for filtering
start_date = datetime(2024, 7, 1)
end_date = datetime(2024, 11, 30)

# Filter stock data within the date range
stock_data_filtered = stock_data[
    (stock_data['Date'] >= start_date) & (stock_data['Date'] <= end_date)
]

# Calculate percentage change in stock prices
stock_data_filtered['Pct_Change'] = stock_data_filtered['Close'].pct_change() * 100

# Plot stock prices
plt.figure(figsize=(18, 5))
plt.plot(stock_data_filtered['Date'], stock_data_filtered['Close'], label='Stock Price', color='blue')
plt.title('lockheed martinstock Prices (July 1 - November 30, 2024)')
plt.xlabel('Date')
plt.ylabel('Stock Price')

# Add dotted vertical lines on July 15 and November 5
plt.axvline(datetime(2024, 7, 15), color='red', linestyle=':', linewidth=2, label='Trump Nominated for 47th president')
plt.axvline(datetime(2024, 11, 5), color='black', linestyle=':', linewidth=2, label='Trump Elected as 47th presidentof U



# Add legends for the main plot and the vertical lines
plt.legend(loc='upper center', fontsize=12, frameon=True)

# Show the grid and the plot
```

```
plt.grid(True)
plt.show()

# Plot percentage change in stock price
plt.figure(figsize=(18, 5))
plt.plot(stock_data_filtered['Date'], stock_data_filtered['Pct_Change'], label='Percentage Change', color='green')
plt.title('Lockheed Martin stock Percentage Change (July 1 - November 30, 2024)')
plt.xlabel('Date')
plt.ylabel('Percentage Change')

# Add dotted vertical lines on July 15 and November 5
plt.axvline(datetime(2024, 7, 15), color='red', linestyle=':', linewidth=2,)
plt.axvline(datetime(2024, 11, 5), color='black', linestyle=':', linewidth=2)

plt.legend(loc='upper center', fontsize=12)
plt.grid(True)
plt.show()

# Display stock price statistics
print(f"Lockheed Martin Stock Price Statistics (July 1 - November 30, 2024):")
print(stock_data_filtered[['Date', 'Close']].describe())
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11640\349370918.py:29: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  stock_data_filtered['Pct_Change'] = stock_data_filtered['Close'].pct_change() * 100
```





Lockheed Martin Stock Price Statistics (July 1 - November 30, 2024):

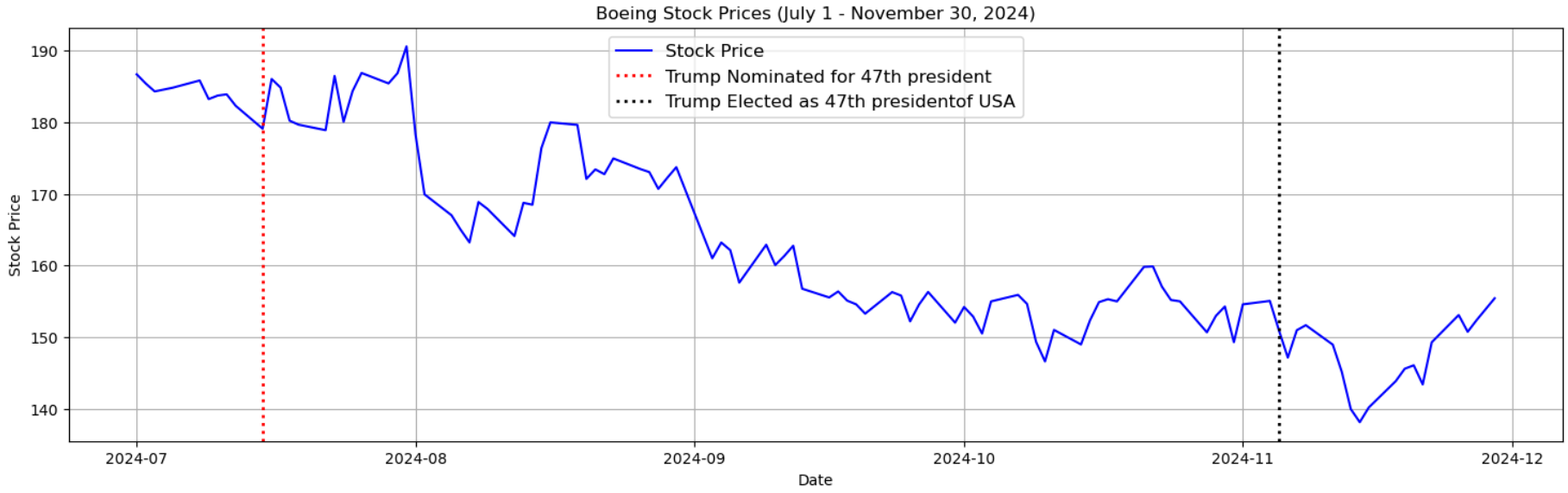
	Date	Close
count	107	107.000000
mean	2024-09-14 14:21:18.504673024	544.818357
min	2024-07-01 00:00:00	454.894409
25%	2024-08-07 12:00:00	531.371399
50%	2024-09-16 00:00:00	551.545898
75%	2024-10-22 12:00:00	567.947571
max	2024-11-29 00:00:00	610.778931
std	NaN	41.549367

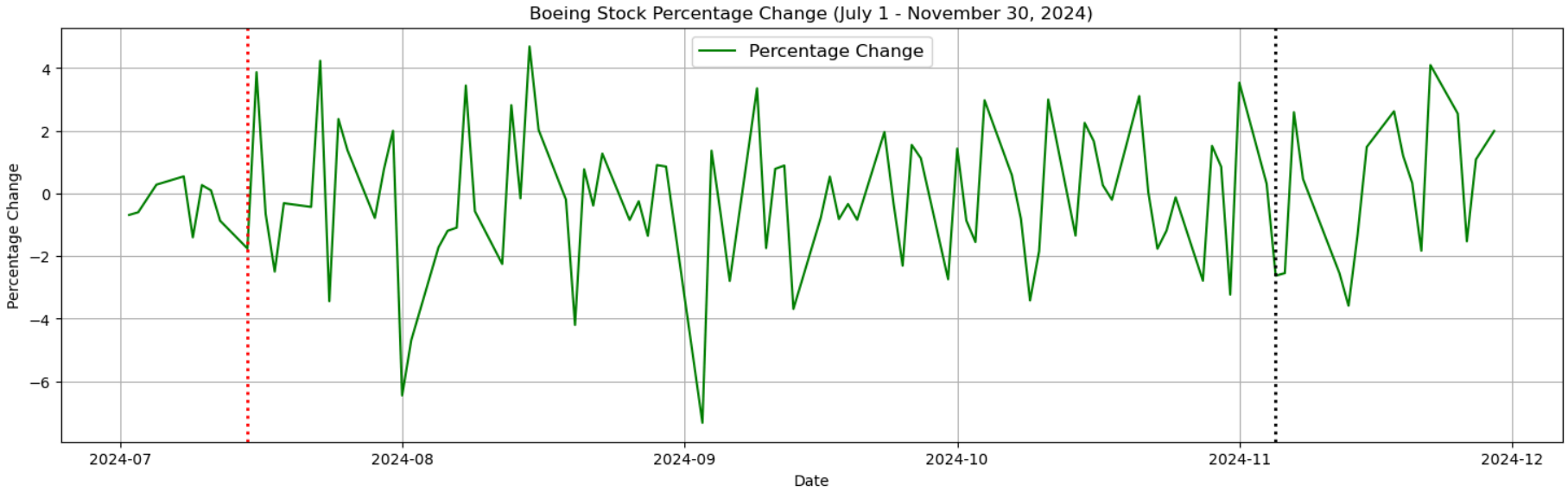
In [8]:

```
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime

# Load stock data for JPMorgan (Assuming you have already loaded this)
stock_data = pd.read_csv("boeing_stock_data.csv")  # Replace with your actual file path

# Ensure Date column is properly converted to datetime
stock_data['Date'] = pd.to_datetime(stock_data['Date'], errors='coerce')

# Check for invalid dates and drop them
if stock_data['Date'].isnull().any():
    print("Some rows have invalid dates and will be dropped.")
stock_data = stock_data.dropna(subset=['Date'])

# Remove timezone information to make the Date column timezone-naive
stock_data['Date'] = stock_data['Date'].apply(lambda x: x.replace(tzinfo=None) if x.tzinfo else x)

# Define the date range for filtering
start_date = datetime(2024, 7, 1)
end_date = datetime(2024, 11, 30)

# Filter stock data within the date range
stock_data_filtered = stock_data[
    (stock_data['Date'] >= start_date) & (stock_data['Date'] <= end_date)
]

# Calculate percentage change in stock prices
stock_data_filtered['Pct_Change'] = stock_data_filtered['Close'].pct_change() * 100

# Plot stock prices
plt.figure(figsize=(18, 5))
plt.plot(stock_data_filtered['Date'], stock_data_filtered['Close'], label='Stock Price', color='blue')
plt.title('Boeing Stock Prices (July 1 - November 30, 2024)')
plt.xlabel('Date')
plt.ylabel('Stock Price')

# Add dotted vertical lines on July 15 and November 5
plt.axvline(datetime(2024, 7, 15), color='red', linestyle=':', linewidth=2, label='Trump Nominated for 47th president')
plt.axvline(datetime(2024, 11, 5), color='black', linestyle=':', linewidth=2, label='Trump Elected as 47th presidentof U



# Add legends for the main plot and the vertical lines
plt.legend(loc='upper center', fontsize=12, frameon=True)

# Show the grid and the plot
```

```
plt.grid(True)
plt.show()

# Plot percentage change in stock price
plt.figure(figsize=(18, 5))
plt.plot(stock_data_filtered['Date'], stock_data_filtered['Pct_Change'], label='Percentage Change', color='green')
plt.title('Boeing Stock Percentage Change (July 1 - November 30, 2024)')
plt.xlabel('Date')
plt.ylabel('Percentage Change')

# Add dotted vertical lines on July 15 and November 5
plt.axvline(datetime(2024, 7, 15), color='red', linestyle=':', linewidth=2,)
plt.axvline(datetime(2024, 11, 5), color='black', linestyle=':', linewidth=2)

plt.legend(loc='upper center', fontsize=12)
plt.grid(True)
plt.show()

# Display stock price statistics
print(f"Boeing Stock Price Statistics (July 1 - November 30, 2024):")
print(stock_data_filtered[['Date', 'Close']].describe())
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_8952\3574451865.py:29: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  stock_data_filtered['Pct_Change'] = stock_data_filtered['Close'].pct_change() * 100
```





Boeing Stock Price Statistics (July 1 - November 30, 2024):

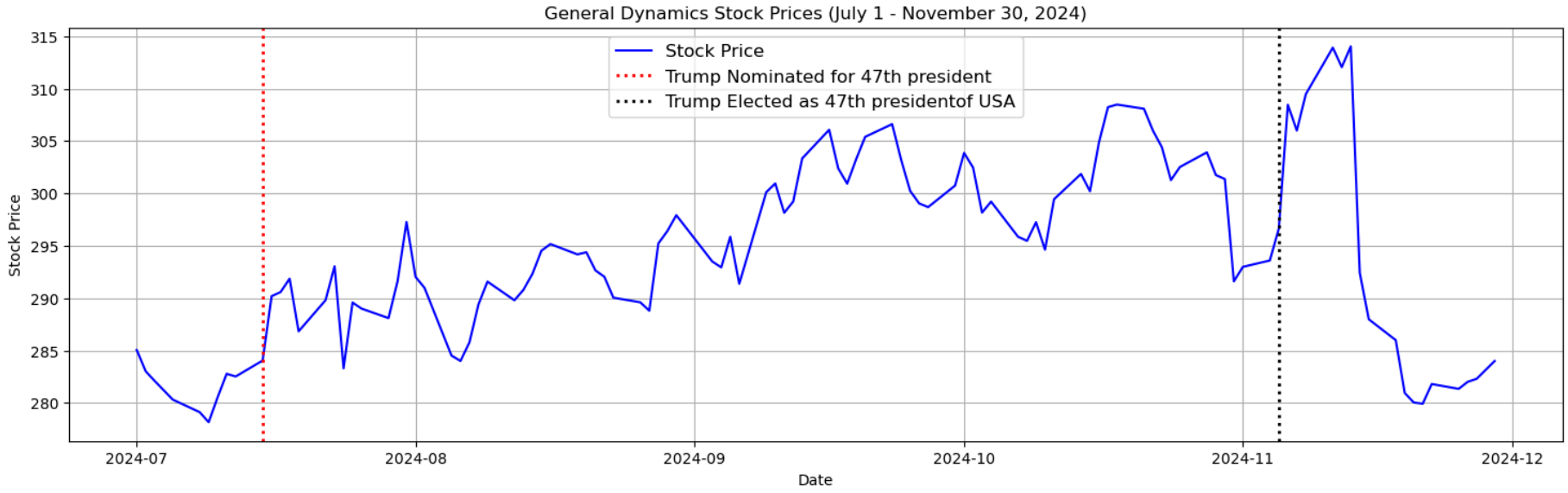
	Date	Close
count	107	107.000000
mean	2024-09-14 14:21:18.504673024	163.251869
min	2024-07-01 00:00:00	138.139999
25%	2024-08-07 12:00:00	152.934998
50%	2024-09-16 00:00:00	157.619995
75%	2024-10-22 12:00:00	174.350006
max	2024-11-29 00:00:00	190.600006
std	NaN	13.780320

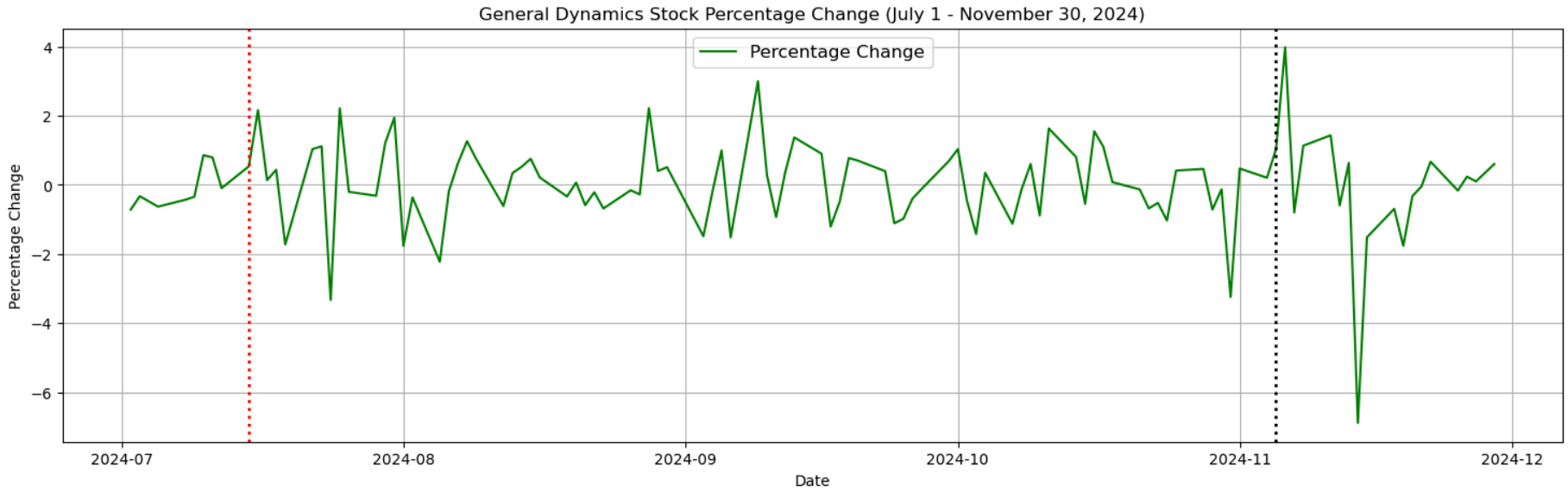
In [9]:

```
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime

# Load stock data for JPMorgan (Assuming you have already loaded this)
stock_data = pd.read_csv("general_dynamics_stock_data.csv")  # Replace with your actual file path

# Ensure Date column is properly converted to datetime
stock_data['Date'] = pd.to_datetime(stock_data['Date'], errors='coerce')

# Check for invalid dates and drop them
if stock_data['Date'].isnull().any():
    print("Some rows have invalid dates and will be dropped.")
stock_data = stock_data.dropna(subset=['Date'])

# Remove timezone information to make the Date column timezone-naive
stock_data['Date'] = stock_data['Date'].apply(lambda x: x.replace(tzinfo=None) if x.tzinfo else x)

# Define the date range for filtering
start_date = datetime(2024, 7, 1)
end_date = datetime(2024, 11, 30)

# Filter stock data within the date range
stock_data_filtered = stock_data[
    (stock_data['Date'] >= start_date) & (stock_data['Date'] <= end_date)
]

# Calculate percentage change in stock prices
stock_data_filtered['Pct_Change'] = stock_data_filtered['Close'].pct_change() * 100

# Plot stock prices
plt.figure(figsize=(18, 5))
plt.plot(stock_data_filtered['Date'], stock_data_filtered['Close'], label='Stock Price', color='blue')
plt.title('General Dynamics Stock Prices (July 1 - November 30, 2024)')
plt.xlabel('Date')
plt.ylabel('Stock Price')

# Add dotted vertical lines on July 15 and November 5
plt.axvline(datetime(2024, 7, 15), color='red', linestyle=':', linewidth=2, label='Trump Nominated for 47th president')
plt.axvline(datetime(2024, 11, 5), color='black', linestyle=':', linewidth=2, label='Trump Elected as 47th presidentof U



# Add legends for the main plot and the vertical lines
plt.legend(loc='upper center', fontsize=12, frameon=True)

# Show the grid and the plot
```

```
plt.grid(True)
plt.show()

# Plot percentage change in stock price
plt.figure(figsize=(18, 5))
plt.plot(stock_data_filtered['Date'], stock_data_filtered['Pct_Change'], label='Percentage Change', color='green')
plt.title('General Dynamics Stock Percentage Change (July 1 - November 30, 2024)')
plt.xlabel('Date')
plt.ylabel('Percentage Change')

# Add dotted vertical lines on July 15 and November 5
plt.axvline(datetime(2024, 7, 15), color='red', linestyle=':', linewidth=2,)
plt.axvline(datetime(2024, 11, 5), color='black', linestyle=':', linewidth=2)

plt.legend(loc='upper center', fontsize=12)
plt.grid(True)
plt.show()

# Display stock price statistics
print(f"General Dynamics stocks Statistics (July 1 - November 30, 2024):")
print(stock_data_filtered[['Date', 'Close']].describe())
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_8952\1006696607.py:29: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  stock_data_filtered['Pct_Change'] = stock_data_filtered['Close'].pct_change() * 100
```





General Dynamics stocks Statistics (July 1 - November 30, 2024):

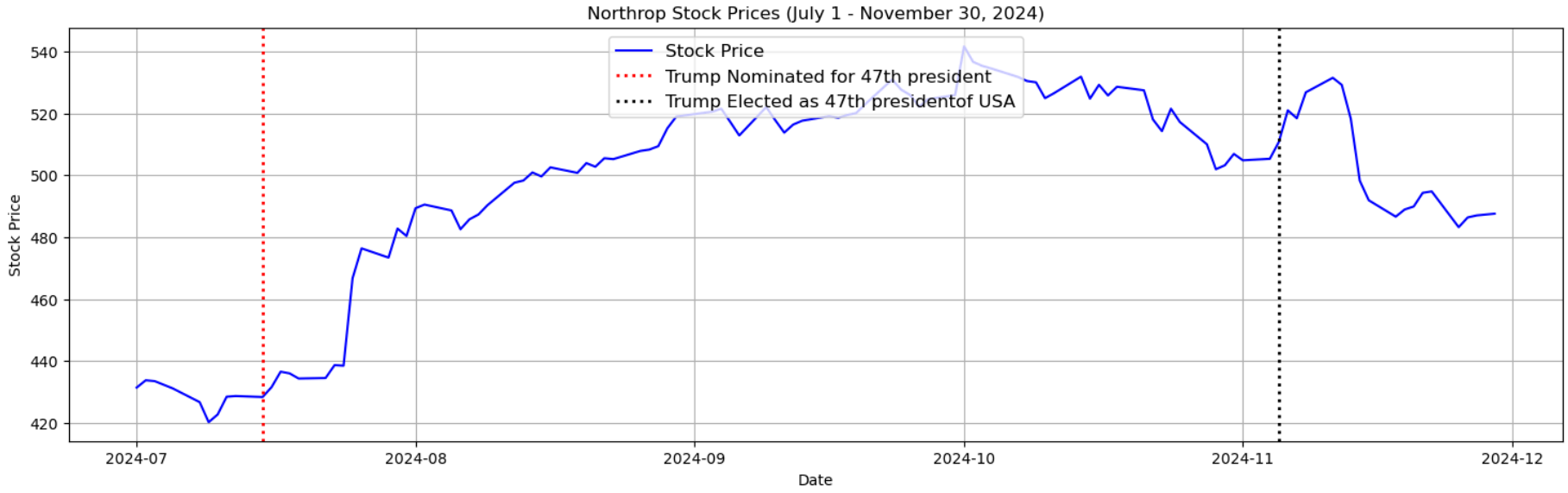
	Date	Close
count	107	107.000000
mean	2024-09-14 14:21:18.504673024	294.488180
min	2024-07-01 00:00:00	278.169342
25%	2024-08-07 12:00:00	288.907578
50%	2024-09-16 00:00:00	294.182159
75%	2024-10-22 12:00:00	301.109756
max	2024-11-29 00:00:00	314.029999
std	NaN	8.736067

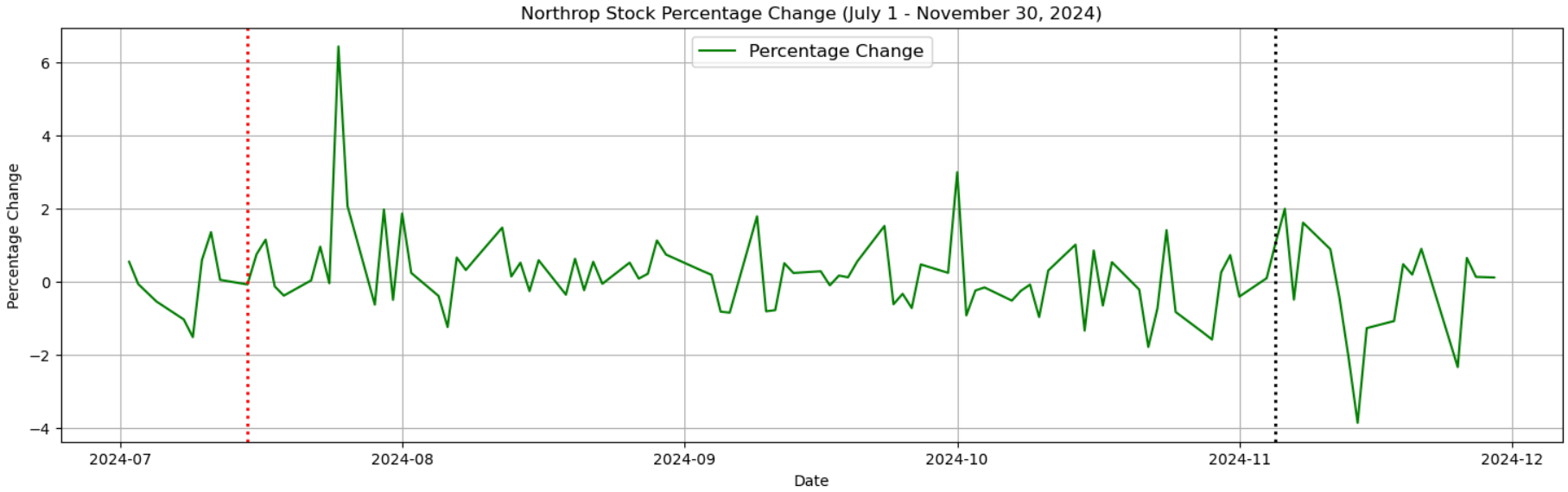
In [10]:

```
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime

# Load stock data for JPMorgan (Assuming you have already loaded this)
stock_data = pd.read_csv("northrop_stock_data.csv")  # Replace with your actual file path

# Ensure Date column is properly converted to datetime
stock_data['Date'] = pd.to_datetime(stock_data['Date'], errors='coerce')

# Check for invalid dates and drop them
if stock_data['Date'].isnull().any():
    print("Some rows have invalid dates and will be dropped.")
stock_data = stock_data.dropna(subset=['Date'])

# Remove timezone information to make the Date column timezone-naive
stock_data['Date'] = stock_data['Date'].apply(lambda x: x.replace(tzinfo=None) if x.tzinfo else x)

# Define the date range for filtering
start_date = datetime(2024, 7, 1)
end_date = datetime(2024, 11, 30)

# Filter stock data within the date range
stock_data_filtered = stock_data[
    (stock_data['Date'] >= start_date) & (stock_data['Date'] <= end_date)
]

# Calculate percentage change in stock prices
stock_data_filtered['Pct_Change'] = stock_data_filtered['Close'].pct_change() * 100

# Plot stock prices
plt.figure(figsize=(18, 5))
plt.plot(stock_data_filtered['Date'], stock_data_filtered['Close'], label='Stock Price', color='blue')
plt.title('Northrop Stock Prices (July 1 - November 30, 2024)')
plt.xlabel('Date')
plt.ylabel('Stock Price')

# Add dotted vertical lines on July 15 and November 5
plt.axvline(datetime(2024, 7, 15), color='red', linestyle=':', linewidth=2, label='Trump Nominated for 47th president')
plt.axvline(datetime(2024, 11, 5), color='black', linestyle=':', linewidth=2, label='Trump Elected as 47th presidentof U



# Add legends for the main plot and the vertical lines
plt.legend(loc='upper center', fontsize=12, frameon=True)

# Show the grid and the plot
```

```
plt.grid(True)
plt.show()

# Plot percentage change in stock price
plt.figure(figsize=(18, 5))
plt.plot(stock_data_filtered['Date'], stock_data_filtered['Pct_Change'], label='Percentage Change', color='green')
plt.title('Northrop Stock Percentage Change (July 1 - November 30, 2024)')
plt.xlabel('Date')
plt.ylabel('Percentage Change')

# Add dotted vertical lines on July 15 and November 5
plt.axvline(datetime(2024, 7, 15), color='red', linestyle=':', linewidth=2,)
plt.axvline(datetime(2024, 11, 5), color='black', linestyle=':', linewidth=2)

plt.legend(loc='upper center', fontsize=12)
plt.grid(True)
plt.show()

# Display stock price statistics
print(f"Northrop Stock Price Statistics (July 1 - November 30, 2024):")
print(stock_data_filtered[['Date', 'Close']].describe())
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_8952\269465852.py:29: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  stock_data_filtered['Pct_Change'] = stock_data_filtered['Close'].pct_change() * 100
```





Northrop Stock Price Statistics (July 1 - November 30, 2024):

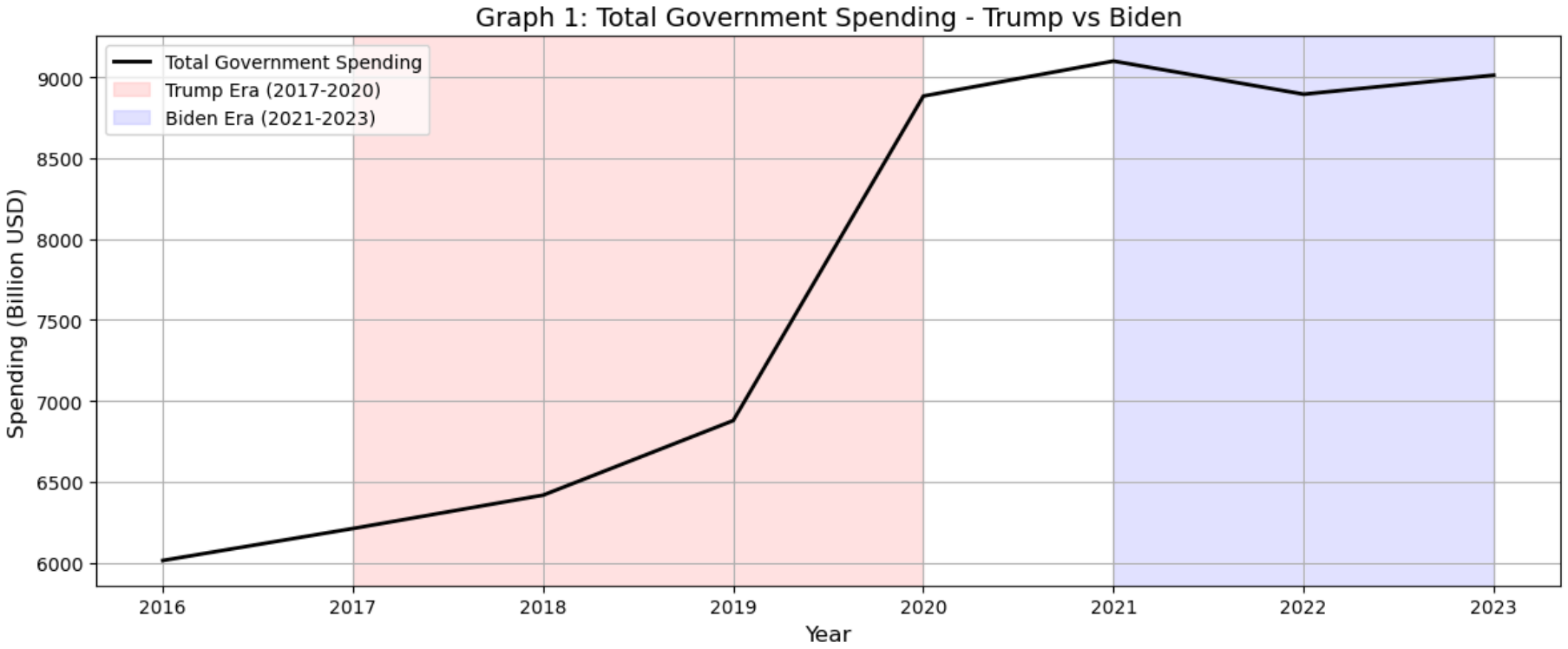
	Date	Close
count	107	107.000000
mean	2024-09-14 14:21:18.504673024	497.474547
min	2024-07-01 00:00:00	420.286041
25%	2024-08-07 12:00:00	486.823242
50%	2024-09-16 00:00:00	505.487854
75%	2024-10-22 12:00:00	521.476807
max	2024-11-29 00:00:00	541.591858
std	NaN	32.871479

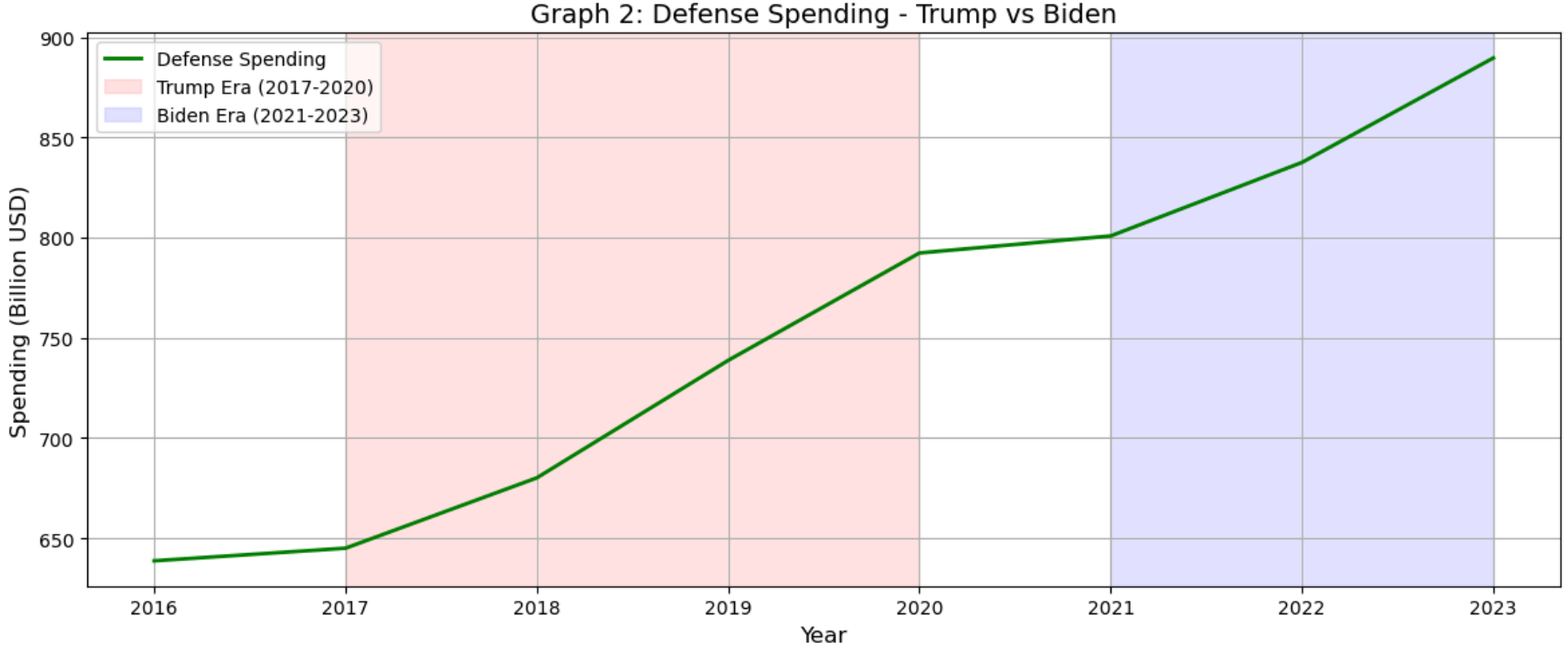
Government and Defense Spending: A Comparative Analysis of Trump vs Biden Era (2016-2023)

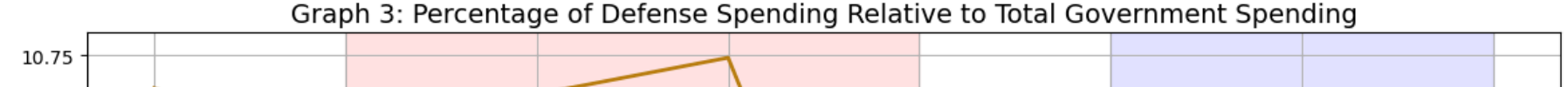
Reference : https://www.whitehouse.gov/omb/budget/historical-tables/ (https://www.whitehouse.gov/omb/budget/historical-tables/)

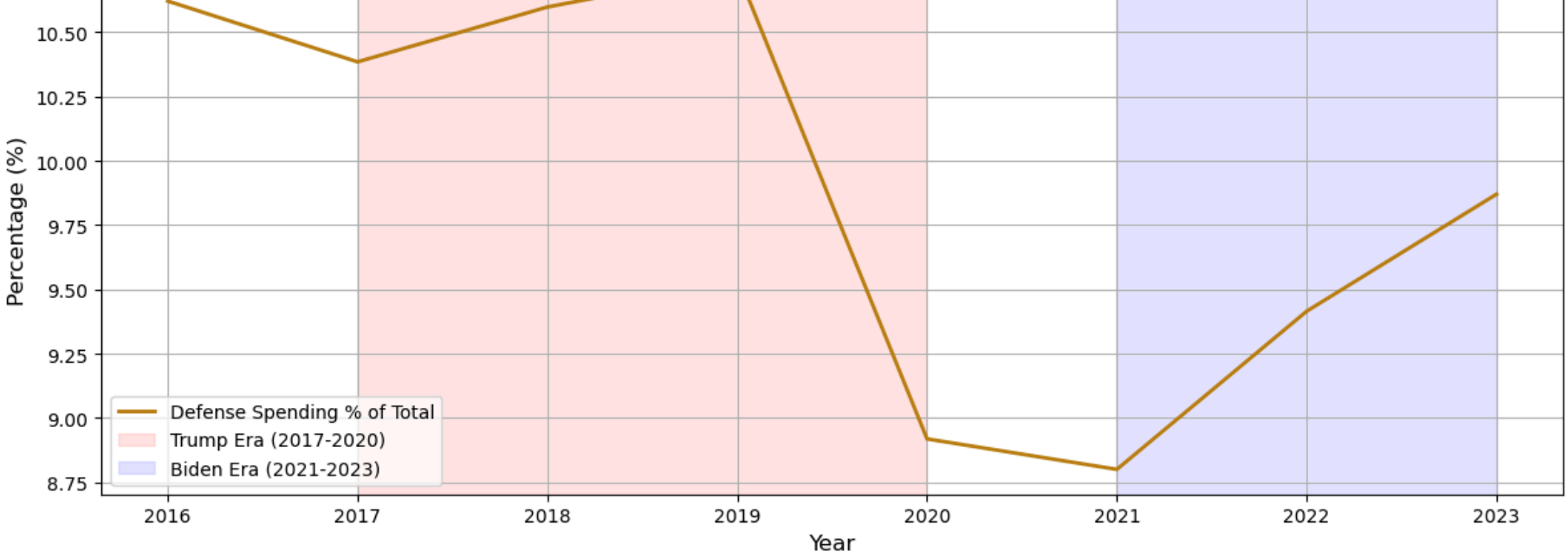
In [15]:

```
import matplotlib.pyplot as plt
import numpy as np

# Data for the years, total government spending, and defense spending (including 2016)
years = [2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023]
total_spending = [6013.30, 6210.90, 6417.10, 6878.00, 8883.80, 9100.00, 8895.30, 9012.90]
defense_spending = [638.7, 645, 680.1, 738.7, 792.3, 800.8, 837.5, 889.6]

# Calculate percentage of defense spending relative to total spending
defense_percentage = [(defense / total) * 100 for defense, total in zip(defense_spending, total_spending)]

# Create the figure with subplots
fig, axs = plt.subplots(3, 1, figsize=(12, 15))

# Graph 1: Total Government Spending - Trump vs Biden
axs[0].plot(years, total_spending, label="Total Government Spending", color='black', linestyle='-', linewidth=2)
axs[0].axvspan(2017, 2020, color='red', alpha=0.1, label="Trump Era (2017-2020)")
axs[0].axvspan(2021, 2023, color='blue', alpha=0.1, label="Biden Era (2021-2023)")
axs[0].set_xlabel("Year", fontsize=12)
axs[0].set_ylabel("Spending (Billion USD)", fontsize=12)
axs[0].set_title("Graph 1: Total Government Spending - Trump vs Biden", fontsize=14)
axs[0].legend(loc="upper left")
axs[0].grid(True)

# Graph 2: Defense Spending - Trump vs Biden
axs[1].plot(years, defense_spending, label="Defense Spending", color='green', linestyle='-', linewidth=2)
axs[1].axvspan(2017, 2020, color='red', alpha=0.1, label="Trump Era (2017-2020)")
axs[1].axvspan(2021, 2023, color='blue', alpha=0.1, label="Biden Era (2021-2023)")
axs[1].set_xlabel("Year", fontsize=12)
axs[1].set_ylabel("Spending (Billion USD)", fontsize=12)
axs[1].set_title("Graph 2: Defense Spending - Trump vs Biden", fontsize=14)
axs[1].legend(loc="upper left")
axs[1].grid(True)

# Graph 3: Percentage of Defense Spending Relative to Total Government Spending
axs[2].plot(years, defense_percentage, label="Defense Spending % of Total", color='#b97d10', linestyle='-', linewidth=2
axs[2].axvspan(2017, 2020, color='red', alpha=0.1, label="Trump Era (2017-2020)")
axs[2].axvspan(2021, 2023, color='blue', alpha=0.1, label="Biden Era (2021-2023)")
axs[2].set_xlabel("Year", fontsize=12)
axs[2].set_ylabel("Percentage (%)", fontsize=12)
axs[2].set_title("Graph 3: Percentage of Defense Spending Relative to Total Government Spending", fontsize=14)
axs[2].legend(loc="lower left")
axs[2].grid(True)

# Adjust layout and show the plot
plt.tight_layout()
plt.show()
```


Graph 1: Total Government Spending - Trump vs Biden
Total Government Spending
Trump Era (2017-2020)
Biden Era (2021-2023)
Spending (Billion USD)
9000
8500
8000
7500
7000
6500
6000
2016
2017
2018
2019
2020
2021
2022
2023
Year


Graph 2: Defense Spending - Trump vs Biden
Defense Spending
Trump Era (2017-2020)
Biden Era (2021-2023)
Spending (Billion USD)
900
850
800
750
700
650
2016
2017
2018
2019
2020
2021
2022
2023
Year


Graph 3: Percentage of Defense Spending Relative to Total Government Spending
10.75



Trends over Time:

You can see that during the Trump administration (2017-2020), military spending fluctuated around 9%, peaking in 2019 at 9.55%, then slightly decreasing in 2020 and 2021, and finally increasing again in 2022. Under Biden, the military expenditure percentage seems to have decreased slightly in 2020 and 2021, but remained relatively stable around 9% in 2022 and 2023.

Comparing Policies:

Under Trump, you might analyze policies such as increased military spending, the establishment of the Space Force, and his focus on expanding the military. The peak in military expenditure in 2019 can be linked to the administration's focus on defense modernization. Under Biden, the emphasis on shifting focus to China as a primary national security threat and the Afghanistan withdrawal can be tied to the lower expenditure in the first years. The increase in 2022 could reflect the need for a stronger military presence in response to global tensions, including the Russia-Ukraine war.