Correlation between stocks and sentiment analysis:

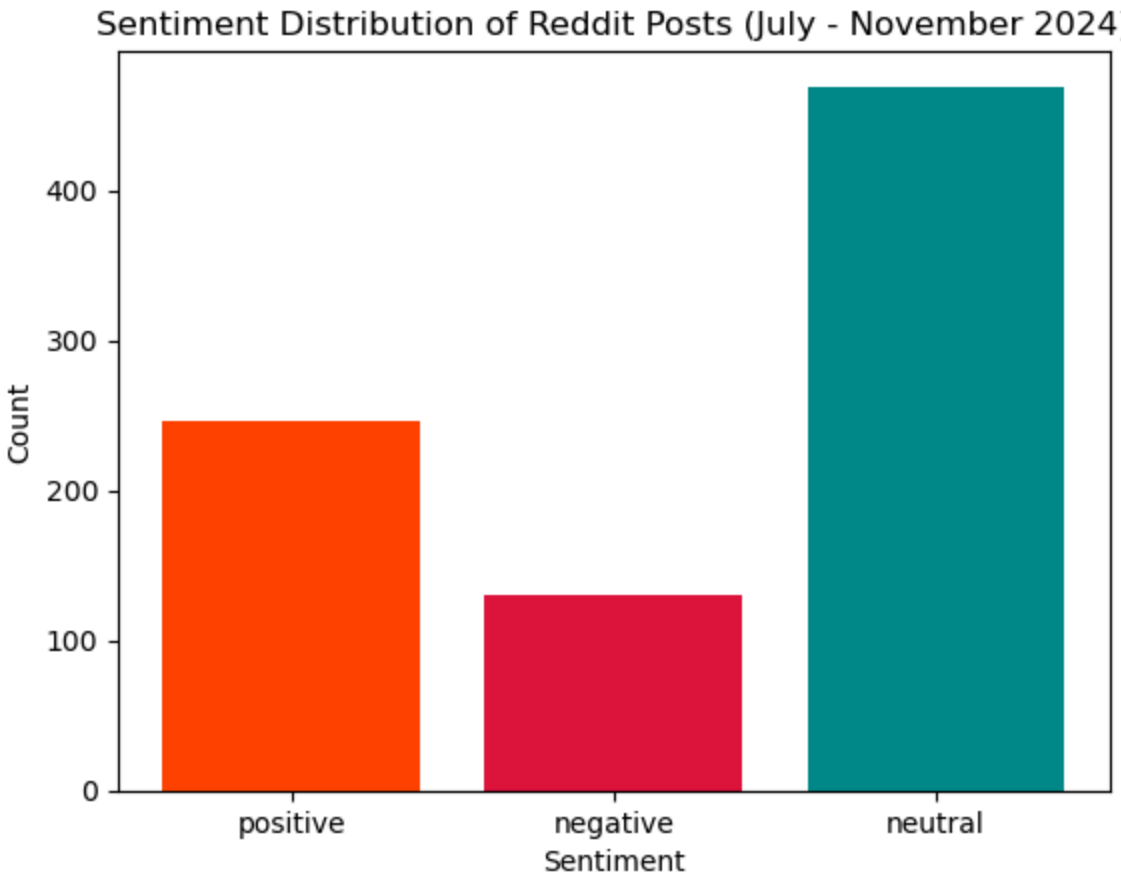
Analysis by month from TRUMP's Nomination for the elections till the election results.

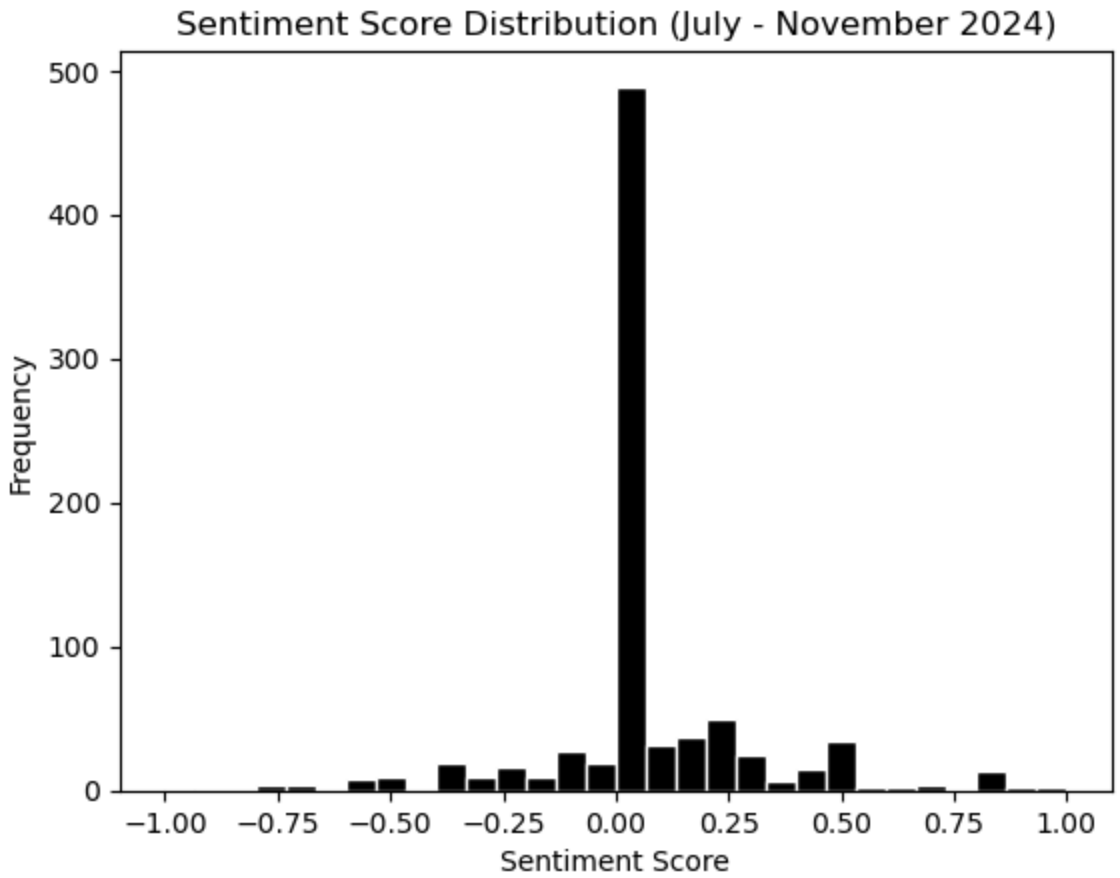
In [26]:

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

# Load Reddit data
with open("cleaned_healthcare_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 %H:%M:%S').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': 246, 'negative': 130, 'neutral': 46
9}
Average Sentiment Score: 0.05
```


Sentiment Distribution of Reddit Posts (July - November 2024)
Count
0
100
200
300
400
positive
negative
neutral
Sentiment



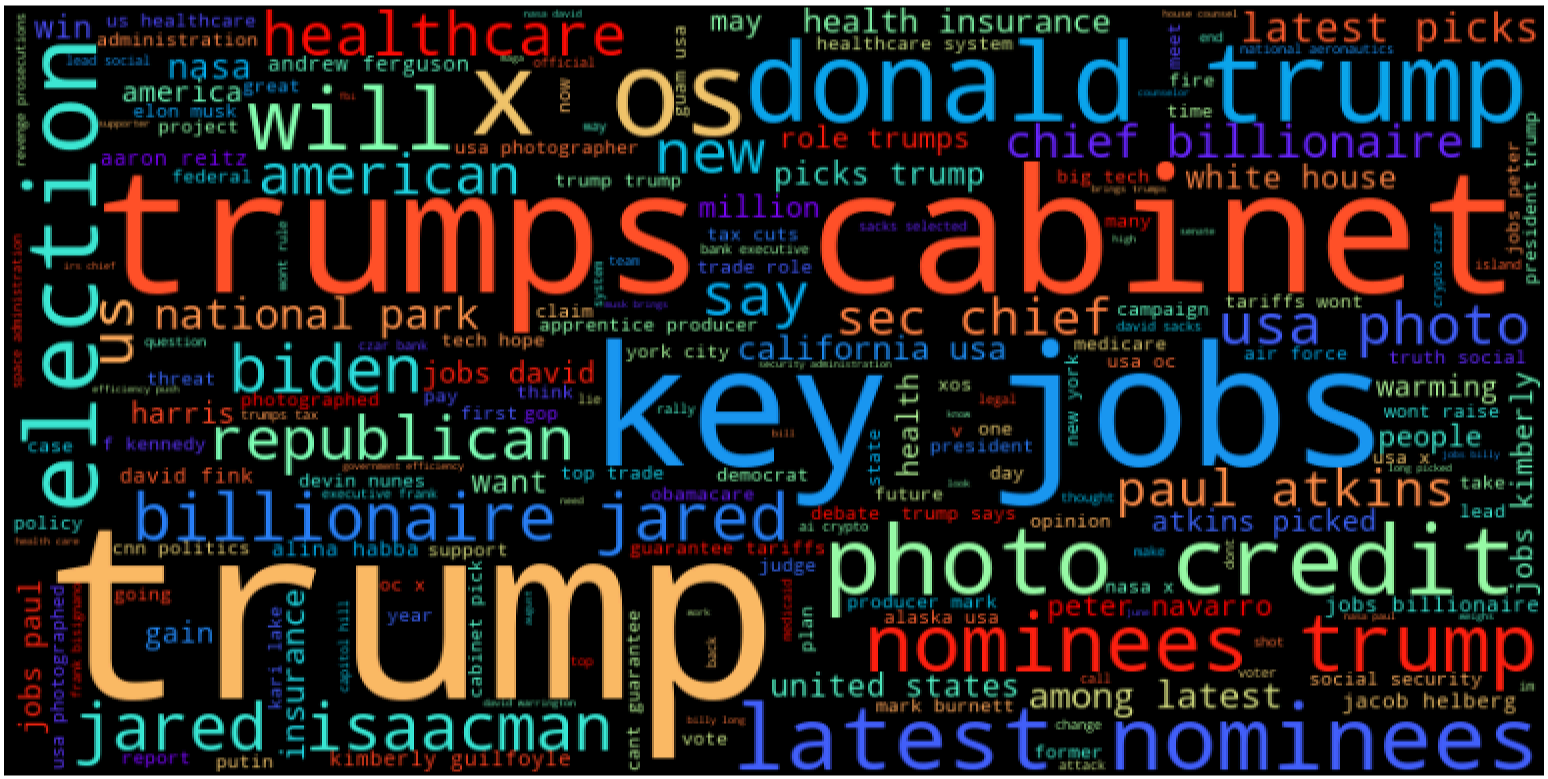
Sentiment Word Cloud

In [27]:

```
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_healthcare_API_Reddit_with_dates.json'
with open(file_path, 'r') as f:
    data = json.load(f)

# Define finance-related keywords related to Trump and the USA
healthcare_keywords = [
    'trump', 'healthcare', 'america first', 'trump again', 'maga', 'make america great again',
    'health reform', 'affordable care act', 'obamacare', 'medicare', 'medicaid',
    'pre-existing conditions', 'drug prices', 'pharmaceuticals', 'insurance',
    'trump administration', 'trumpcare', 'american healthcare', 'health crisis',
    'health policy', 'republicans', 'conservatives', 'tax reform', 'economy',
    'jobs', 'border security', 'immigration', 'american people', 'patriotism',
    'us economy', 'freedom', 'american values', 'national security',
    'usa', 'united states', 'election', 'president trump', 'white house'
]


# 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, healthcare_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
```

```python
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 & Healthcare Word Cloud", fontsize=16)
plt.show()
```

Trump & Healthcare Word Cloud



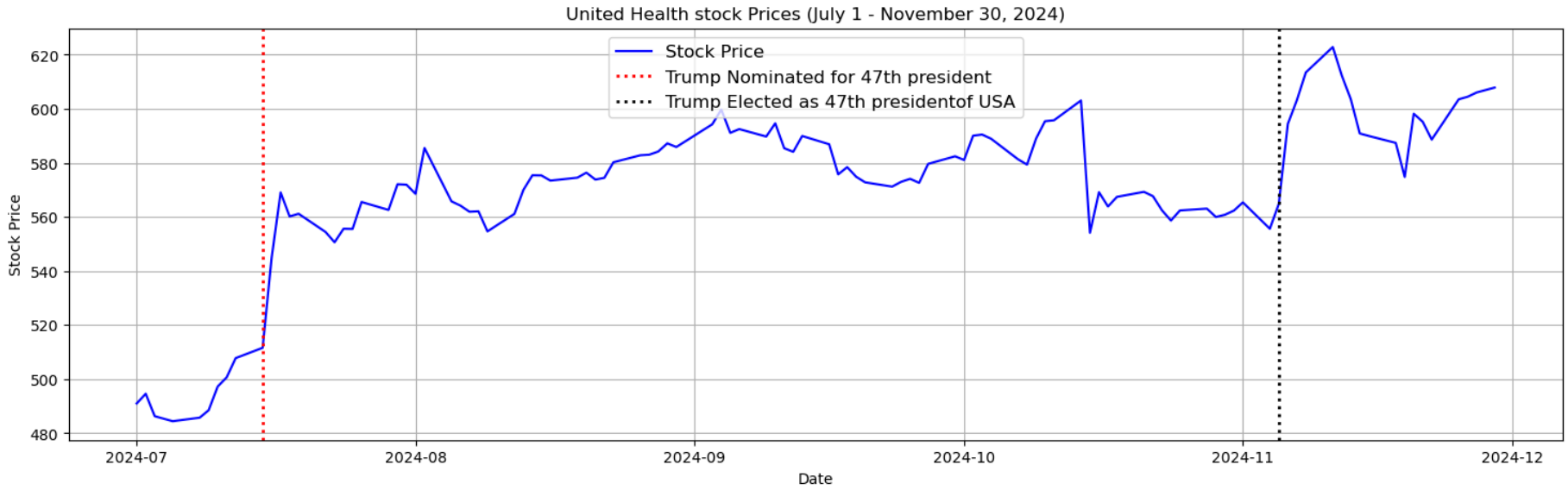
Stock Analysis

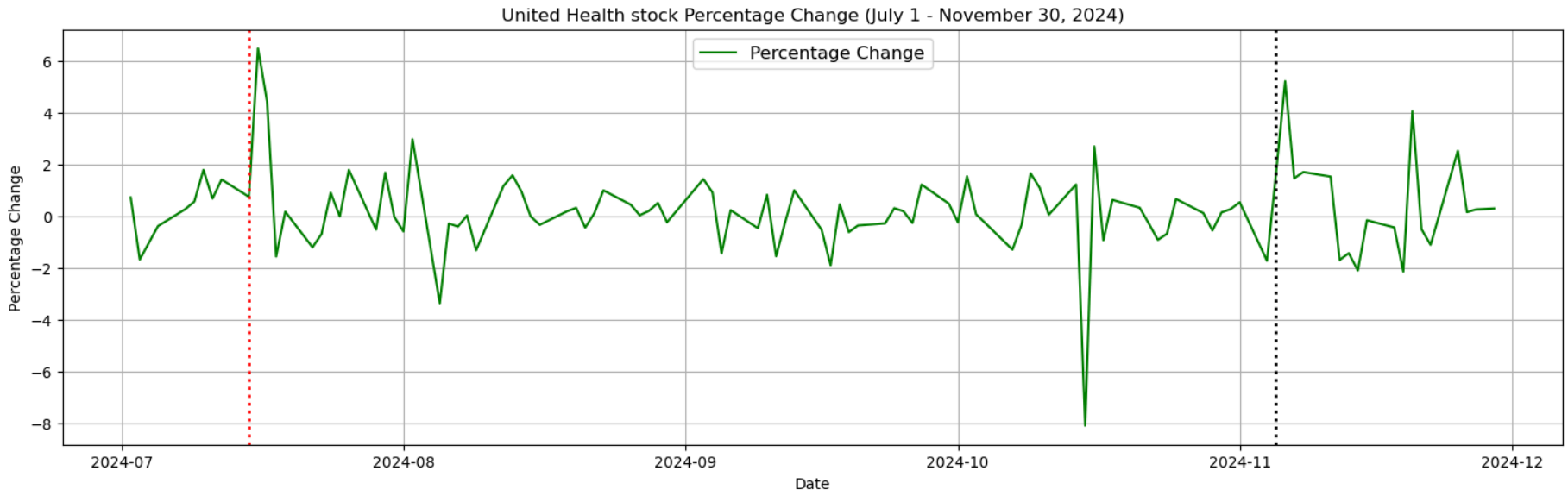
In [15]:

```
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("unitedhealth_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('United Health 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('United Health 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"United Health stock Prices Statistics (July 1 - November 30, 2024):")
print(stock_data_filtered[['Date', 'Close']].describe())
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_4232\2087267919.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
```





United Health stock Prices Statistics (July 1 - November 30, 2024):

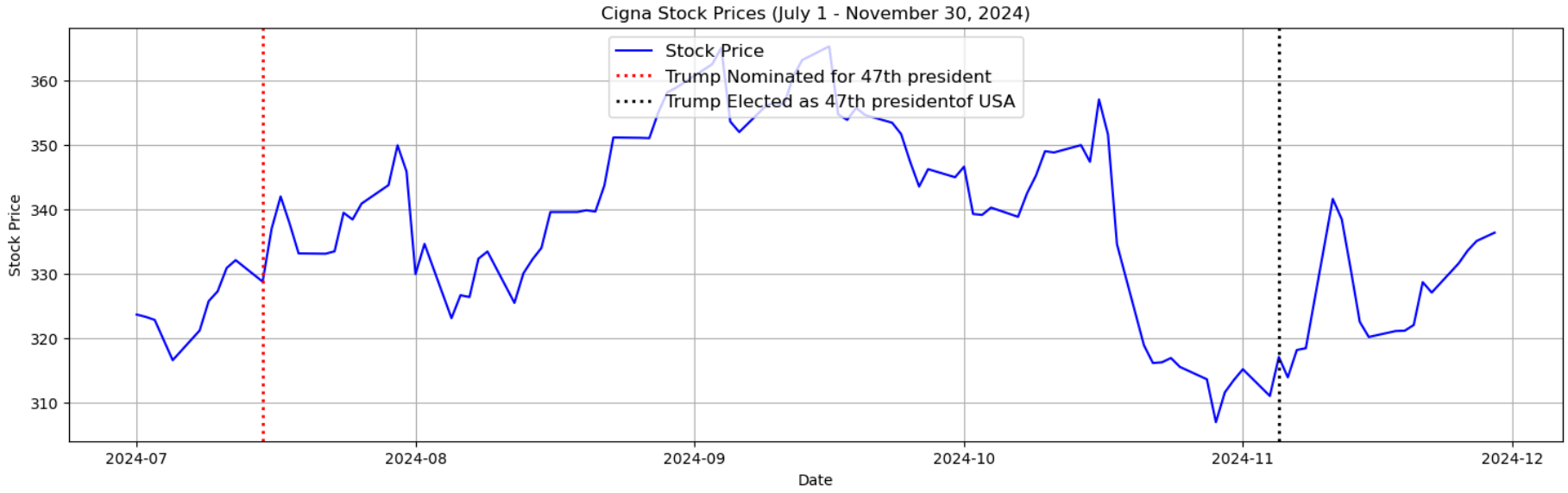
	Date	Close
count	107	107.000000
mean	2024-09-14 14:21:18.504673024	570.597258
min	2024-07-01 00:00:00	484.427673
25%	2024-08-07 12:00:00	562.373016
50%	2024-09-16 00:00:00	574.471741
75%	2024-10-22 12:00:00	588.951080
max	2024-11-29 00:00:00	622.861023
std	NaN	28.970819

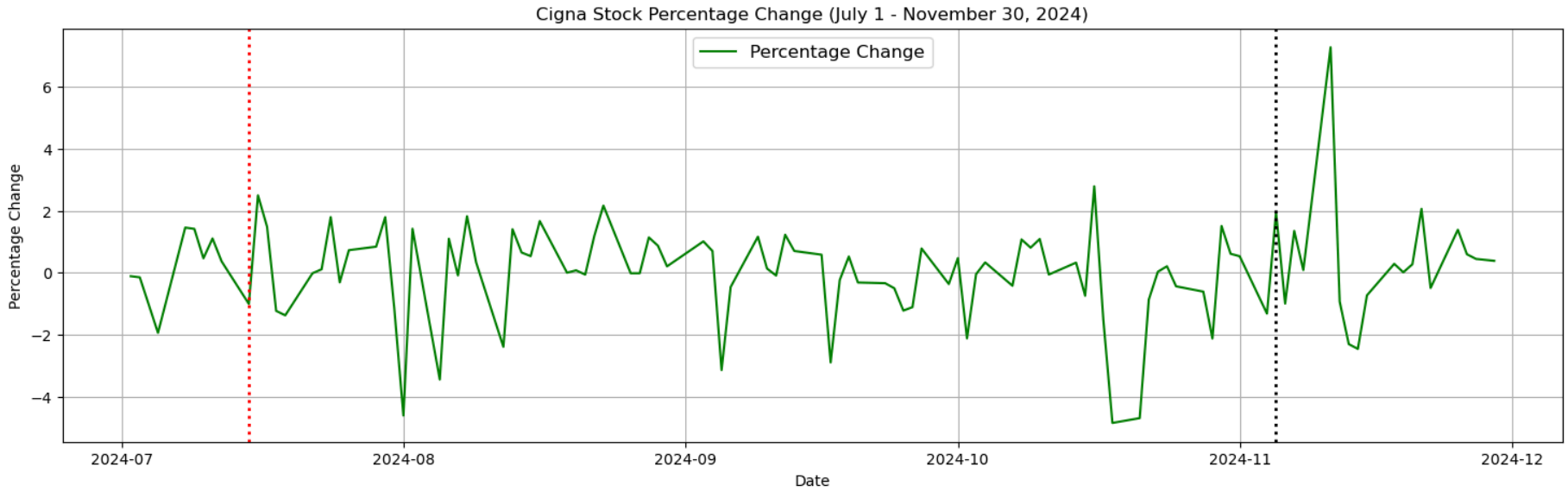
In [16]:

```
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("CIGNA_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('Cigna 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('Cigna 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"Cigna Stock Price Statistics (July 1 - November 30, 2024):")
print(stock_data_filtered[['Date', 'Close']].describe())
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_4232\101090693.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
```





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

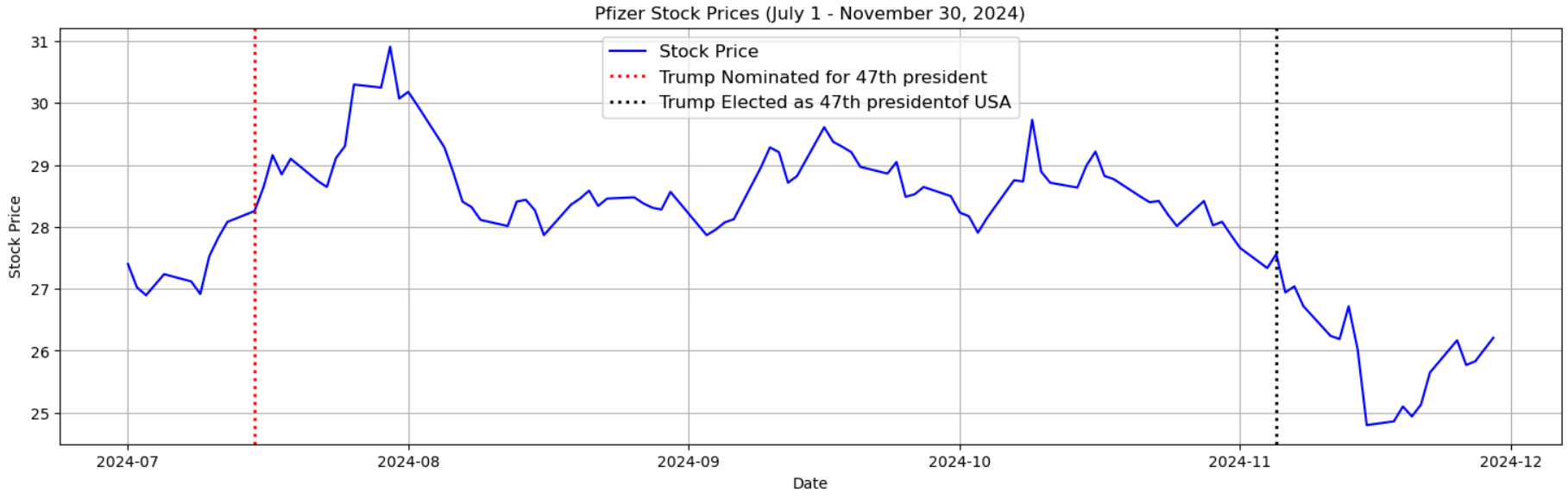
	Date	Close
count	107	107.000000
mean	2024-09-14 14:21:18.504673024	336.869725
min	2024-07-01 00:00:00	306.941498
25%	2024-08-07 12:00:00	325.610779
50%	2024-09-16 00:00:00	337.797546
75%	2024-10-22 12:00:00	348.935196
max	2024-11-29 00:00:00	365.316437
std	NaN	14.458979

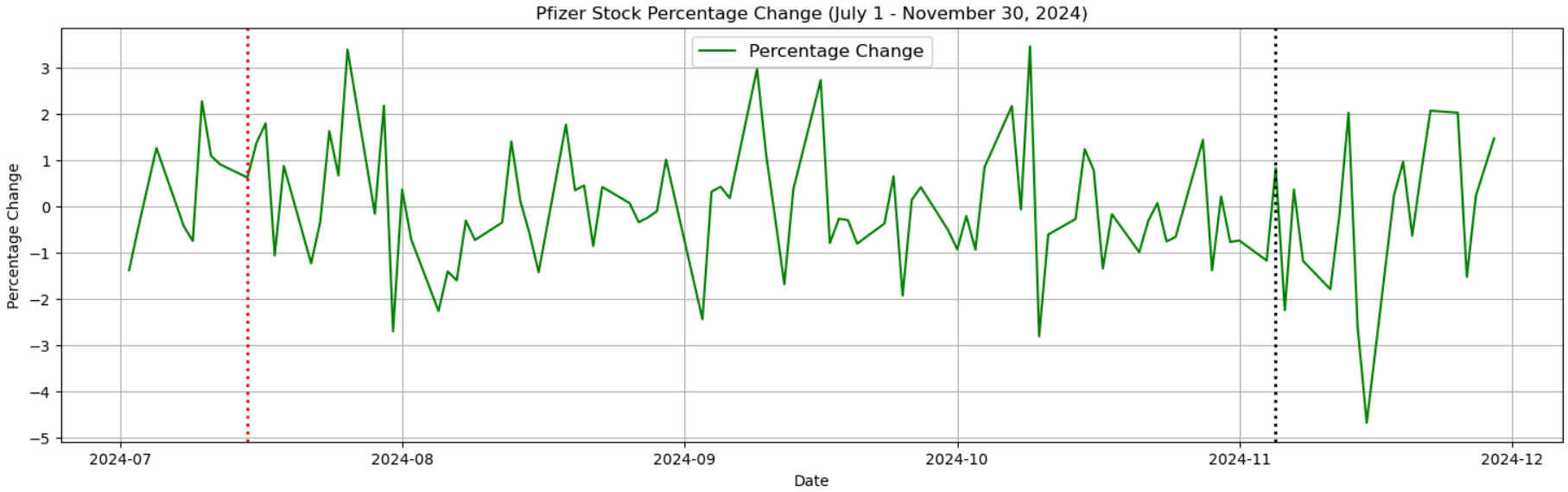
In [17]:

```
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("pfizer_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('Pfizer 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('Pfizer 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"Pfizer stocks Statistics (July 1 - November 30, 2024):")
print(stock_data_filtered[['Date', 'Close']].describe())
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_4232\3497305171.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
```





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

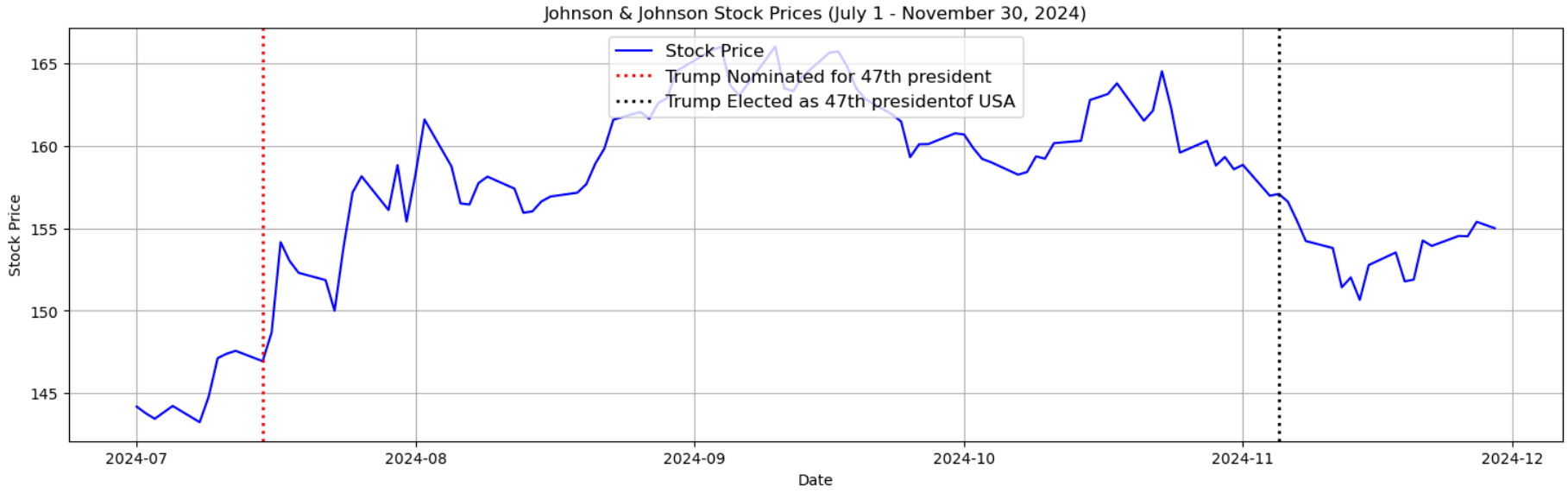
	Date	Close
count	107	107.000000
mean	2024-09-14 14:21:18.504673024	28.128800
min	2024-07-01 00:00:00	24.799999
25%	2024-08-07 12:00:00	27.744632
50%	2024-09-16 00:00:00	28.398893
75%	2024-10-22 12:00:00	28.835383
max	2024-11-29 00:00:00	30.909891
std	NaN	1.226560

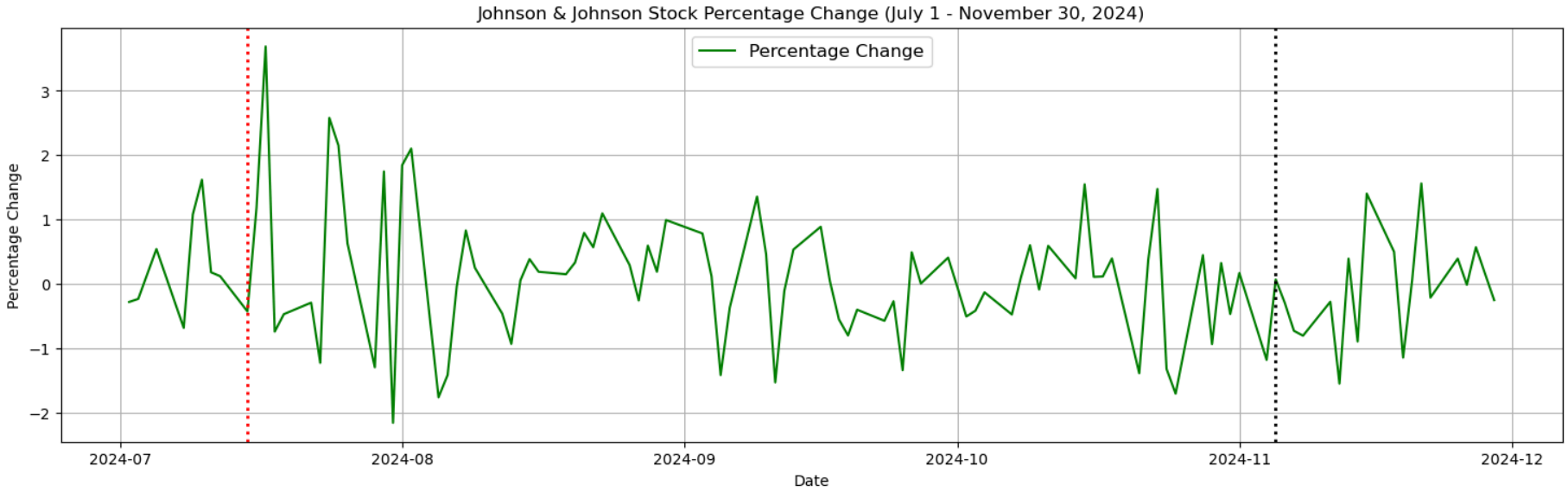
In [18]:

```
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("jnj_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('Johnson & Johnson 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('Johnson & Johnson 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"Johnson & Johnson Stock Price Statistics (July 1 - November 30, 2024):")
print(stock_data_filtered[['Date', 'Close']].describe())
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_4232\3307295894.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
```





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

	Date	Close
count	107	107.000000
mean	2024-09-14 14:21:18.504673024	157.461849
min	2024-07-01 00:00:00	143.234802
25%	2024-08-07 12:00:00	154.247345
50%	2024-09-16 00:00:00	158.418869
75%	2024-10-22 12:00:00	161.629875
max	2024-11-29 00:00:00	166.047668
std	NaN	5.616981

Comparative Analysis of Healthcare Employment: Trump vs. Biden

- Refrence : https://data.bls.gov/apps/industry-productivity-viewer/home.htm (https://data.bls.gov/apps/industry-productivity-viewer/home.htm)

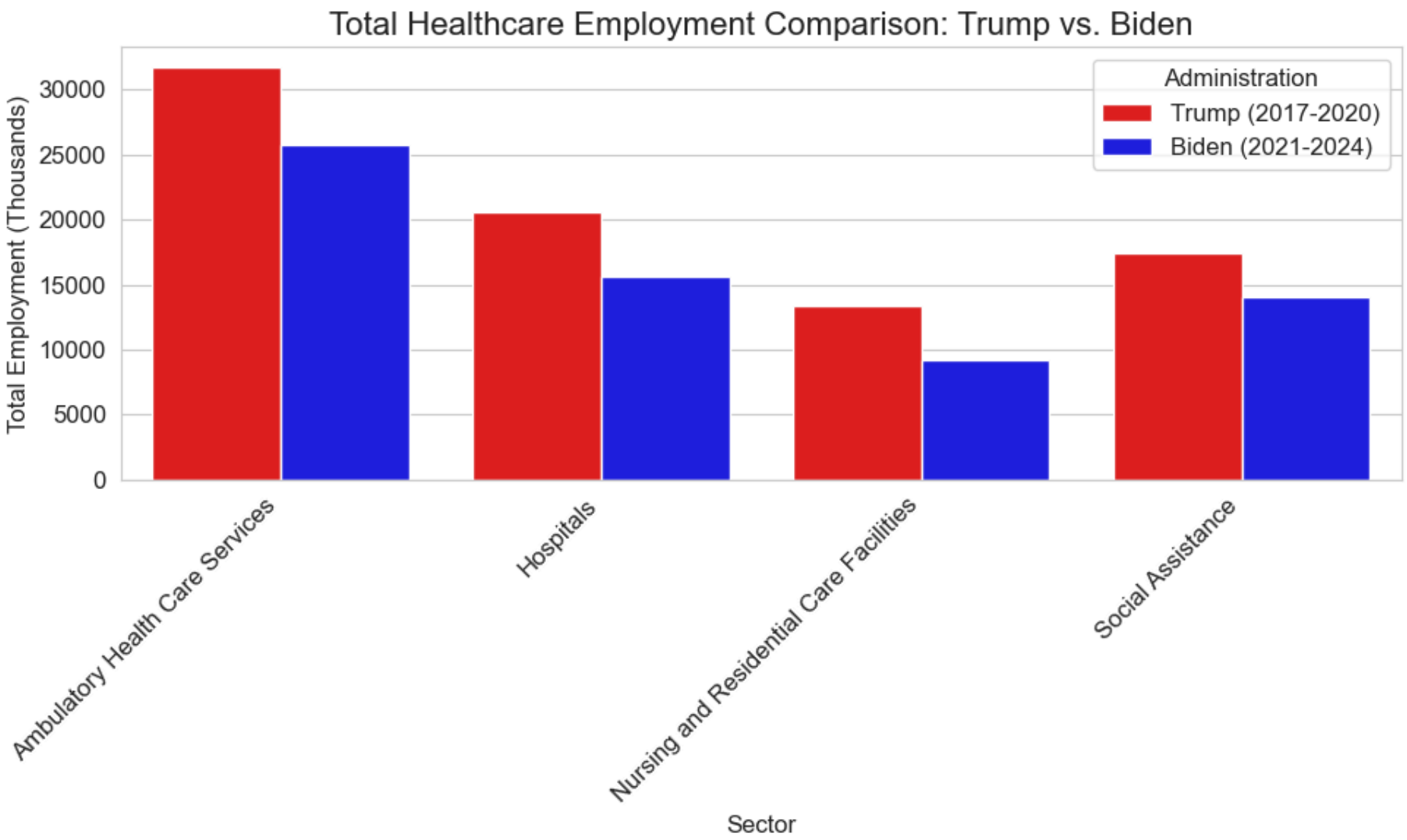
In [18]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Set a Seaborn style
sns.set_style("whitegrid")
plt.rcParams.update({'font.size': 12})  # Increase font size for better readability

# Load the dataset
df = pd.read_csv('2Cleaned_Healthcare_Data_2017_2024.csv')

# Print the exact column names to check for any issues
print("Column names in the dataset:", df.columns)

# If there are extra spaces or characters, you can clean the column names
df.columns = df.columns.str.strip()  # Strip extra spaces from column names

# Check again if everything is correct
print("Cleaned column names:", df.columns)

# Now you can proceed with the rest of the code.
```

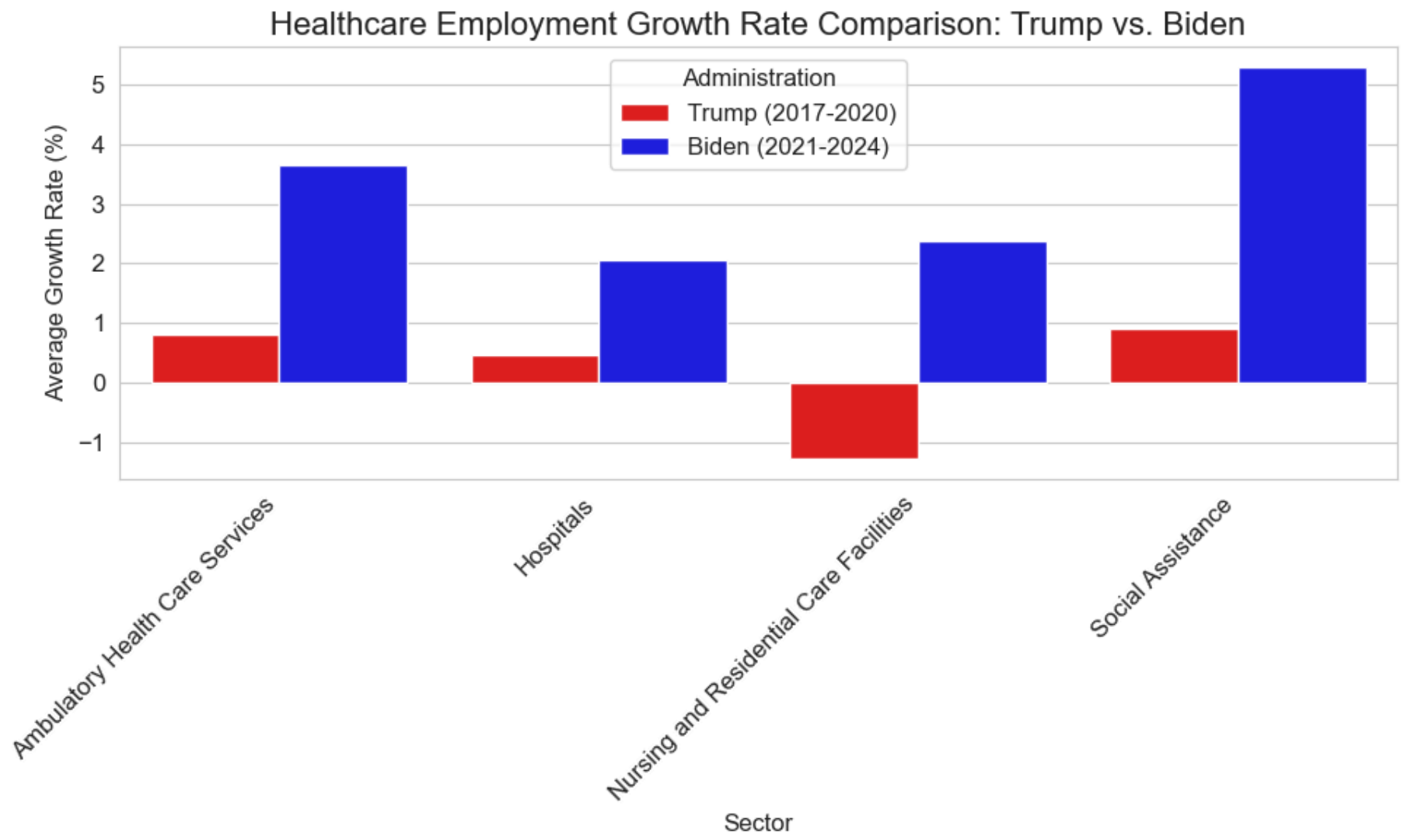
```
Column names in the dataset: Index(['Year', 'Ambulatory_healthcare_services_employment_in_thousands',
       'Hospitals_employment_in_thousands)  ',
       'Nursing_and_residential_care_facilities_employment_in_thousands)  ',
       'Social_assistance_employment_in_thousands '],
      dtype='object')
Cleaned column names: Index(['Year', 'Ambulatory_healthcare_services_employment_in_thousands',
       'Hospitals_employment_in_thousands)',
       'Nursing_and_residential_care_facilities_employment_in_thousands)',
       'Social_assistance_employment_in_thousands'],
      dtype='object')
```

In [21]:

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Set a Seaborn style
sns.set_style("whitegrid")
plt.rcParams.update({'font.size': 12})  # Increase font size for better readability

# Load the dataset
df = pd.read_csv('2Cleaned_Healthcare_Data_2017_2024.csv')

# Clean column names by removing unwanted characters (e.g., spaces and parentheses)
df.columns = df.columns.str.replace(r'[\)\s]', '', regex=True)

# Print the cleaned column names
#print("Cleaned column names:", df.columns)

# Create a DataFrame for the total employment comparison
df_comparison = pd.DataFrame({
    'Sector': ['Ambulatory Health Care Services', 'Hospitals',
               'Nursing and Residential Care Facilities', 'Social Assistance'],
    'Trump (2017-2020)': [
        df[df['Year'].between(2017, 2020)]['Ambulatory_healthcare_services_employment_in_thousands'].sum(),
        df[df['Year'].between(2017, 2020)]['Hospitals_employment_in_thousands'].sum(),
        df[df['Year'].between(2017, 2020)]['Nursing_and_residential_care_facilities_employment_in_thousands'].sum(),
        df[df['Year'].between(2017, 2020)]['Social_assistance_employment_in_thousands'].sum()
    ],
    'Biden (2021-2024)': [
        df[df['Year'].between(2021, 2024)]['Ambulatory_healthcare_services_employment_in_thousands'].sum(),
        df[df['Year'].between(2021, 2024)]['Hospitals_employment_in_thousands'].sum(),
        df[df['Year'].between(2021, 2024)]['Nursing_and_residential_care_facilities_employment_in_thousands'].sum(),
        df[df['Year'].between(2021, 2024)]['Social_assistance_employment_in_thousands'].sum()
    ]
})

# Melt the DataFrame for better plotting
df_comparison_melted = df_comparison.melt(id_vars="Sector",
                                          var_name="Administration",
                                          value_name="Total Employment")

# Define the custom colors for Trump and Biden
color_palette = {"Trump (2017-2020)": "red", "Biden (2021-2024)": "blue"}

# Plot total employment comparison with custom colors
plt.figure(figsize=(10, 6))
sns.barplot(data=df_comparison_melted, x="Sector", y="Total Employment", hue="Administration",
            palette=color_palette)
```

```
plt.title("Total Healthcare Employment Comparison: Trump vs. Biden", fontsize=16)
plt.ylabel("Total Employment (Thousands)")
plt.xlabel("Sector")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.legend(title="Administration")
plt.show()

# Create a DataFrame for the growth rate comparison
df_growth = pd.DataFrame({
    'Sector': ['Ambulatory Health Care Services', 'Hospitals',
               'Nursing and Residential Care Facilities', 'Social Assistance'],
    'Trump (2017-2020)': [
        df[df['Year'].between(2017, 2020)]['Ambulatory_healthcare_services_employment_in_thousands'].pct_change().mean(
        df[df['Year'].between(2017, 2020)]['Hospitals_employment_in_thousands'].pct_change().mean() * 100,
        df[df['Year'].between(2017, 2020)]['Nursing_and_residential_care_facilities_employment_in_thousands'].pct_chang
        df[df['Year'].between(2017, 2020)]['Social_assistance_employment_in_thousands'].pct_change().mean() * 100
    ],
    'Biden (2021-2024)': [
        df[df['Year'].between(2021, 2024)]['Ambulatory_healthcare_services_employment_in_thousands'].pct_change().mean(
        df[df['Year'].between(2021, 2024)]['Hospitals_employment_in_thousands'].pct_change().mean() * 100,
        df[df['Year'].between(2021, 2024)]['Nursing_and_residential_care_facilities_employment_in_thousands'].pct_chang
        df[df['Year'].between(2021, 2024)]['Social_assistance_employment_in_thousands'].pct_change().mean() * 100
    ]
})

# Melt the DataFrame for better plotting
df_growth_melted = df_growth.melt(id_vars="Sector",
                                  var_name="Administration",
                                  value_name="Growth Rate")

# Plot growth rate comparison with custom colors
plt.figure(figsize=(10, 6))
sns.barplot(data=df_growth_melted, x="Sector", y="Growth Rate", hue="Administration",
            palette=color_palette)
plt.title("Healthcare Employment Growth Rate Comparison: Trump vs. Biden", fontsize=16)
plt.ylabel("Average Growth Rate (%)")
plt.xlabel("Sector")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.legend(title="Administration")
plt.show()
```


Total Healthcare Employment Comparison: Trump vs. Biden
Administration
Trump (2017-2020)
Biden (2021-2024)
Total Employment (Thousands)
30000
25000
20000
15000
10000
5000
0
Ambulatory Health Care Services
Hospitals
Nursing and Residential Care Facilities
Social Assistance
Sector


Healthcare Employment Growth Rate Comparison: Trump vs. Biden
Administration
Trump (2017-2020)
Biden (2021-2024)
Average Growth Rate (%)
5
4
3
2
1
0
−1
Ambulatory Health Care Services
Hospitals
Nursing and Residential Care Facilities
Social Assistance
Sector

In [ ]: