GitHub Repo: https://github.com/kayleeboudrie/SI206Final.git

A. **The goals for your project, including what APIs/websites you planned to work with and what data you planned to gather (10 points)**
    a. Our goal was to examine whether public sentiment in the news and presidential approval ratings correlate during Donald Trump's presidency.
        i. We wanted to scrape presidential approvals from
            1. https://www.presidency.ucsb.edu/statistics/data/donald-j-trump-public-approval
        ii. Use the NewsAPI.AI to gather news articles about Trump from 100 days after the election.
            1. https://newsapi.ai/documentation?tab=searchArticles
        iii. Calculate sentiment scores and compare them to public approval ratings to see if media sentiment tracks with popularity.

B. **The goals that were achieved, including what APIs/websites you actually worked with and what data you did gather (10 points)**
    a. We successfully scraped the approval rating data from the UCSB Presidency Project.
    b. We used NewsAPI.AI to collect headlines and sentiment scores from news articles about Trump.
    c. We created a SQLite database to store both data sets.
    d. We visualized the approval ratings over time, using a red line with 144 daily data points from our dataset.

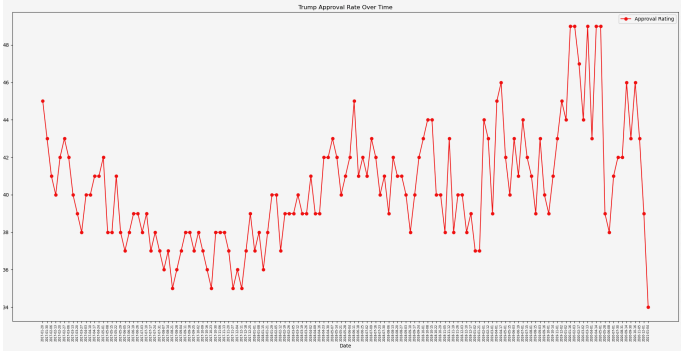
C. **The problems that you faced (10 points)**
    a. The API had very limited access for free accounts, so we had to upgrade and pay $90 of our own money to get access to all the data we needed.
    b. The original Gallup page we planned to scrape was not structured well for automated scraping. We pivoted to a new source (UCSB Presidency Project), which had a more consistent and scrappable layout.

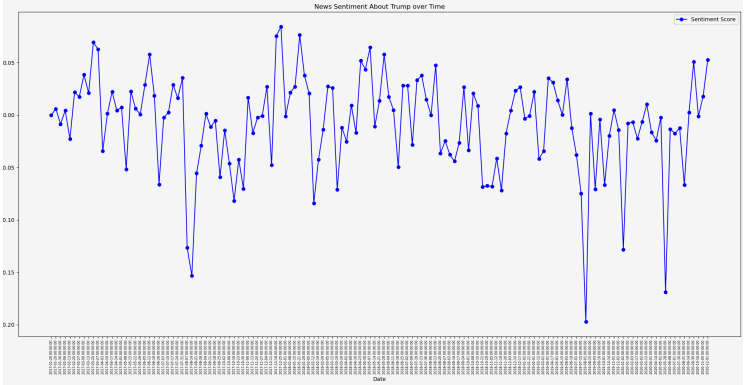
D. The calculations from the data in the database (i.e., a screenshot) (10 points)

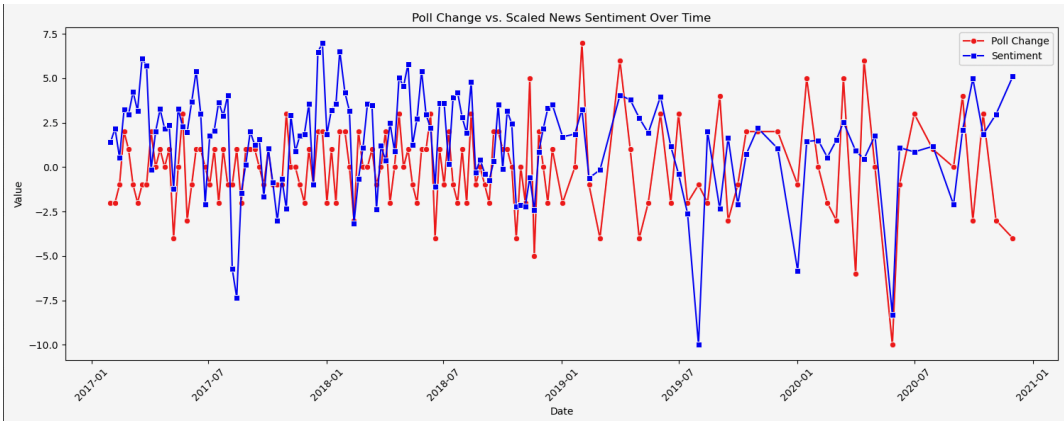
```
for (start_date, start_rating), (end_date, end_rating) in zip(rows, rows[1:]):
    diff = end_rating - start_rating
    changes.append((start_date, end_date, diff))
```

```
if sentiment_scores:
    avg_sentiment = sum(sentiment_scores) / len(sentiment_scores)
```

E. The visualization that you created (i.e., screenshot or image file) (10 points)







F. Instructions for running your code (10 points)

a. Clone the GitHub repository
b. Run the approval scraper script to gather the Gallup data
c. Run the command in the terminal: pip install eventregistry
d. Run the NewsAPI script
e. Run the main analysis and visualization file

G. Documentation for each function that you wrote. This includes describing the input and output for each function (20 points)

a. def create_tables(): Creates the SQLite table 'GallupApproval' in the database 'final_project.db'. Input: None. Output: None (creates table in database). If the table already exists, it will be dropped and recreated.

b. **def scrape_gallup(limit=1000):** Scrapes Donald Trump's public approval ratings from the UCSB Presidency Project website. Input: limit (optional integer, default is 1000) which limits the number of rows scraped. Output: A list of dictionaries with keys: 'date', 'President', and 'approval_rating'. Dates are cleaned and reformatted to 'YYYY-MM-DD'.

c. def store_gallup_in_db(records, batch_size=25): Stores scraped approval rating data into the 'GallupApproval' table in the SQLite database. Input: records (list of dictionaries from scrape_gallup) and batch_size (optional integer, default is 25). Output: None (inserts data into the database). Data is inserted in batches and duplicates are ignored.

d. def create_poll_changes_table(): Creates a new SQLite table 'PollChanges' in the database to store daily changes in approval ratings. Input: None. Output: None (creates table in the database). If the table already exists, it will be dropped and recreated.

e. **def store_poll_changes():** Calculates daily changes in approval rating based on data in the 'GallupApproval' table and stores them in the 'PollChanges' table. Input: None. Output: None (inserts change data into the database). Each row represents the difference in approval between two consecutive days.

f. **if __name__ == "__main__":** This is the main execution block that runs the entire process. It creates tables, scrapes Gallup approval data, stores the data in the database, and calculates daily approval changes. Input: None. Output: None (prints progress to the console).

g. def get_poll_date_ranges(): Retrieves all approval poll dates from the GallupApproval table in the database and converts them into datetime objects. Input: None. Output: A list of consecutive (start_date, end_date) tuples that represent the ranges between each pair of dates in the database. This is used to group news articles by polling periods.

h. **def create_tables():** Creates the SQLite table NewsSentiment in the 'final_project.db' database. Input: None. Output: None (creates table in database). If the table already exists, it will be dropped and recreated. This table will store sentiment analysis results for each polling range.

i. **def get_avg_sentiment_for_range(start_date, end_date):** Queries the NewsAPI.AI EventRegistry for English-language news articles about Donald Trump published in the United States between the given start_date and end_date. Input: start_date, end_date (both datetime objects). Output: The average sentiment score (float) of the retrieved articles. If no articles are found, returns 0.
j. **def get_sentiment_for_date_ranges(date_ranges):** Calculates the average sentiment score for each (start_date, end_date) date range. Input: date_ranges (list of date tuples from get_poll_date_ranges). Output: A list of dictionaries, each containing 'start_date', 'end_date', and 'avg_sentiment' keys.
k. **def store_sentiment_in_db(sentiment_data, batch_size=25):** Inserts the sentiment data for each date range into the NewsSentiment table in the database. Input: sentiment_data (list of dictionaries with sentiment scores), optional batch_size (default is 25). Output: None (inserts data into the database). Uses batching to reduce the number of database commits.
l. **if __name__ == "__main__":** Main execution block. It first gets polling date ranges, creates the NewsSentiment table, calculates average sentiment scores per date range, and stores that data in the database. Input: None. Output: Console messages showing progress and printed sentiment values for each range.
m. **def join_and_analyze():** Joins and analyzes data from the GallupApproval and NewsSentiment tables in the final_project.db database. It resamples the news sentiment data to daily frequency and merges it with the approval ratings data based on matching dates. Input: None. Output: A pandas DataFrame containing the merged data of approval ratings and news sentiment scores, saved as a CSV file for later use.
n. **def create_visualizations(df):** Creates a visualization (line plot) comparing Trump's approval rating over time. The plot displays the approval rating on the y-axis and the date on the x-axis, with sentiment scores overlayed in red. Input: df (a pandas DataFrame with merged approval and sentiment data). Output: A saved PNG image file (comparison_plot.png) containing the plot, and the plot displayed on screen.
o. **if __name__ == "__main__":** Main execution block. It calls join_and_analyze() to retrieve and merge the data and then calls create_visualizations() to generate the plot based on the merged data. Input: None. Output: A CSV file with the merged data and a visualization file showing the comparison of approval ratings over time.
p. **def join_and_scale(db_path="final_project.db"):** Joins the poll change data from the PollChanges table and sentiment score data from the NewsSentiment table in the final_project.db database. It then scales the sentiment scores to match the range of poll changes using linear mapping. The merged and scaled data is returned as a DataFrame. Input: db_path (optional, default "final_project.db") — the path to the database. Output: A pandas DataFrame with merged and scaled sentiment data.

q. **def create_combined_viz(df):** Creates a line plot to visualize the relationship between poll change and scaled sentiment over time. The poll changes are displayed in red and the sentiment scores in blue. Input: df — a pandas DataFrame containing merged poll change and scaled sentiment data. Output: A line plot saved as poll_change_vs_scaled_sentiment.png and displayed on the screen.

r. **def create_approval_viz():** Generates a line plot showing the approval rating of Donald Trump over time. It uses data from the GallupApproval table in the database, with the x-axis representing dates and the y-axis representing approval scores. Input: None. Output: A line plot saved as Approval.png and displayed on the screen.

s. **def create_sentiment_viz():** Generates a line plot showing the sentiment score about Donald Trump over time. It uses data from the NewsSentiment table in the database, with the x-axis representing dates and the y-axis representing sentiment scores. Input: None. Output: A line plot saved as Sentiment.png and displayed on the screen.

H. You must also clearly document all resources you used. The documentation should be of the following form (20 points)

Date	Issue Description	Location of Resource	Result (Did it solve the issue?)
4/8/25	We were having trouble with the repo the cloning it, and pulling it	Help from GSI	Yes - he was amazing!!
4/16/25	Having trouble making the data viz show and be correct.	ChatGPT	Not really, I kind of had to fix it on my own, but it pointed me in the right direction.
4/22/25	The database was not working, so I asked ChatGPT to help find the problem in our code.	ChatGPT	No, chat was no help in finding this problem, so I asked Ethan if he could help figure it out, and he did.