SI 206 Final Report
By: Esha Mungala, Linda Karti, Ben Li

Github link: https://github.com/eshaready/si206-final-project

1. The goals for your project including what APIs/websites you planned to work with and what data you planned to gather (10 points)

We planned to work with the New York Times Books API, gathering the best selling list for every week for the years we decided to focus on, and the number of NYT reviews for every book on the list; the Box Office Mojo website, scraping the movie with the highest box office gross earnings for every month for the years we decided to focus on; and the Weather Data API, gathering min/max temperature, precipitation amount, and average temperature for every day for the years we decided to focus on. Our aim was to observe whether there was a link between weather and the kind of entertainment people sought out (books vs. movies), the kinds of books or movies people sought out during different seasons, as well as the link between weather and people's tendencies towards going out to a movie theater.

2. The goals that were achieved including what APIs/websites you actually worked with and what data you did gather (10 points)

In our final project, we modified our plan a little. First of all, we settled on the years 2012-2022 to collect data for (in order to collect at least 100 rows of data). Then, from the New York Times (NYT) Books API, we decided to only collect bestselling list data for the first week of every month; this is because the NYT API rate limit is 5 requests per minute and 500 per day, and collecting weekly data for 10 years would quickly overwhelm that. Also, the NYT bestselling list has multiple lists embedded within it; we decided to pull the bestselling book from the Combined Print & E-book Fiction list, since that list covered all fiction books.

From the NYT API, we collected data for three tables. First, a Bestselling table, which had Date (of list published, which was needed for later requests), Year and Month (for ease of joining with other tables later), and ISBN (an integer unique key for a book), with the Date and ISBN combined serving as a primary key as this is a unique pair (since books can be bestselling for multiple weeks). Second, a Books table, which included all the information about a particular book present in the Bestselling table, including ISBN, Title, Author, Description, and Cover; the ISBN served as a primary key here, and was also the integer key linking the Books table to the Bestselling table. This organization ensured that there was no duplicate string data in the Bestselling table; any book that was bestselling for more than one month was represented in the database with only the ISBN integer. And thirdly, a Reviews table, which included information about the reviews of a book, with the columns of ISBN, Title, Review_URL, and Reviewer; this table ended up not being used in our final code incorporating all the APIs and website.

From the Box Office MOJO website, we scraped information about the top-grossing movie every month from 2012 to 2023. We collected data for one "Top Monthly Releases" table that included information about the month, the year, the movie title, and its gross box office value for every month of each year from 2012 to 2023. We scraped this information from the webpage that had listed the top monthly releases through each year. In other words, there was a January webpage with the top grossing movie in January for every year which made it much simpler to extract the data. (Please note that the month of January is represented by the number 1, the month of February by the number 2, etc.) We used the year and month columns to merge the data from all three data sources.

Finally, we collected weather data across the United States for one table: monthly_averages. This table included the average maximum temperature, average minimum temperature, average temperature, and average precipitation for all the months from 2012 to 2022. We also created year and month columns to join three tables into our combined database bell.db.

3. The problems that you faced (10 points)

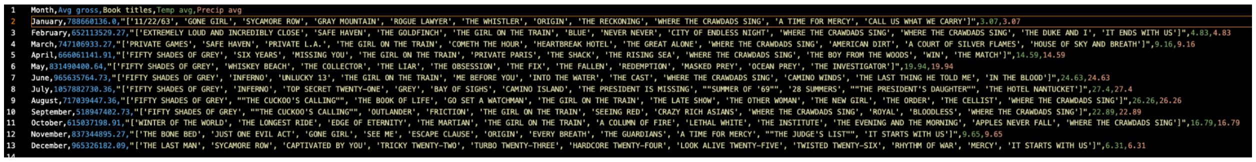
A major problem we faced with the NYT API was the rate limit; at one point I had to stop testing my code and gathering data because I ran into the rate limit, which was causing my code to crash. After I fixed that by adding a time.sleep() to delay my code and by changing my collection frequency from every week to every month, I still faced a smaller problem: the amount of time it took to gather code. Since I could only send 5 requests a minute. There were no significant issues with the box office data, I encountered a few problems with putting in too many requests, and was blocked for a short period of time, but that was quickly resolved. The issue we had with the weather API was also the rate limit. The website we pulled data from only allowed 1000 days of data per day, so we had to start pulling data ahead of time to make sure we could get all 11 years of data from 2012-2022.

When joining our data, a major problem we faced was making sure that the tables were easy to join with columns that corresponded to each other. We all had to go back and edit our tables so that the schema included a year and a month, so that we could join the data in every table for that; our sources originally all had different formats for the date (the bestselling list had YYYY-MM-DD, the weather data had YYYYMM, and the box office data had YYYY and month in words). After this fix, joining was easy and executed correctly.

Another problem we faced was that the books data and box office data was originally in a different database, and rerunning the files with a different database destination would have run into API request limits. To combat this, we attached the databases using the SQL ATTACH command and then copied the data from the table over into the group database to ensure that it was there and updated after the first run through.

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

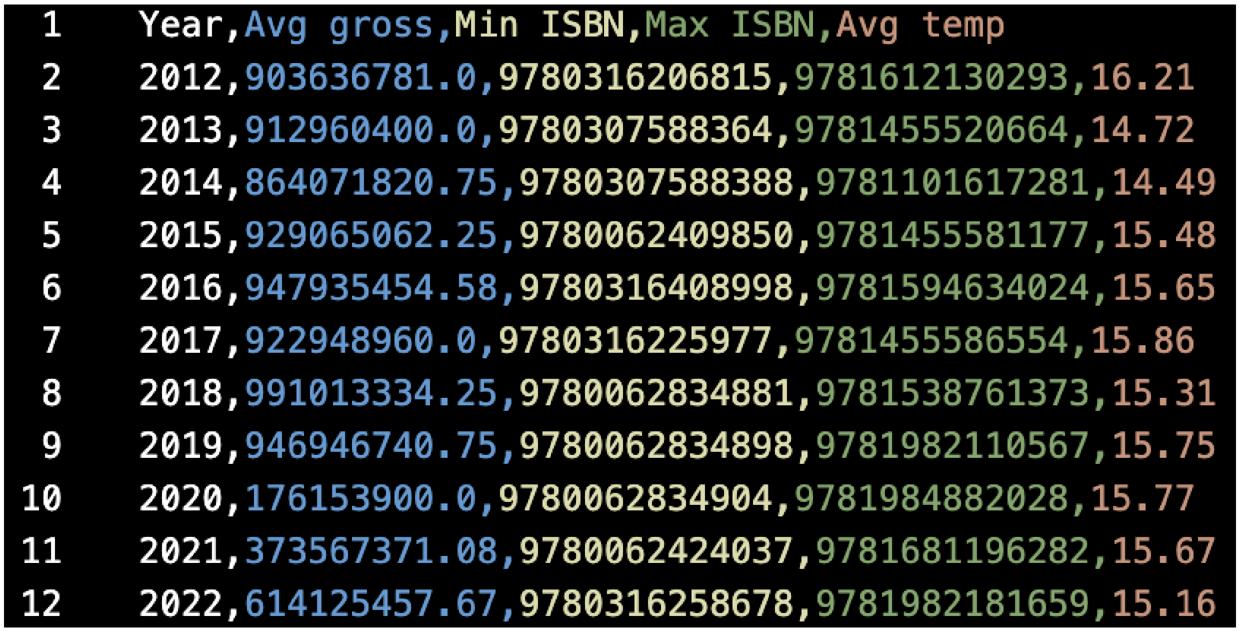
joined_data_monthly_calculations.csv



```
Month,Avg gross,Book titles,Temp avg,Precip avg
January,788660136.0,"['11/22/63', 'GONE GIRL', 'SYCAMORE ROW', 'GRAY MOUNTAIN', 'ROGUE LAWYER', 'THE WHISTLER', 'ORIGIN', 'THE RECKONING', 'WHERE THE CRAWDADS SING', 'A TIME FOR MERCY', 'CALL US WHAT WE CARRY']",3.07,3.07
February,652113529.27,"['EXTREMELY LOUD AND INCREDIBLY CLOSE', 'SAFE HAVEN', 'THE GOLDFINCH', 'THE GIRL ON THE TRAIN', 'BLUE', 'NEVER NEVER', 'CITY OF ENDLESS NIGHT', 'WHERE THE CRAWDADS SING', 'WHERE THE CRAWDADS SING', 'THE DUKE AND I', 'IT ENDS WITH US']",4.83,4.83
March,747106933.27,"['PRIVATE GAMES', 'SAFE HAVEN', 'PRIVATE L.A.', 'THE GIRL ON THE TRAIN', 'COMETH THE HOUR', 'HEARTBREAK HOTEL', 'THE GREAT ALONE', 'WHERE THE CRAWDADS SING', 'AMERICAN DIRT', 'A COURT OF SILVER FLAMES', 'HOUSE OF SKY AND BREATH']",9.16,9.16
April,666061141.91,"['FIFTY SHADES OF GREY', 'SIX YEARS', 'MISSING YOU', 'THE GIRL ON THE TRAIN', 'PRIVATE PARIS', 'THE SHACK', 'THE RISING SEA', 'WHERE THE CRAWDADS SING', 'THE BOY FROM THE WOODS', 'WIN', 'THE MATCH']",14.59,14.59
May,831490400.64,"['FIFTY SHADES OF GREY', 'WHISKEY BEACH', 'THE COLLECTOR', 'THE LIAR', 'THE OBSESSION', 'THE FIX', 'THE FALLEN', 'REDEMPTION', 'MASKED PREY', 'OCEAN PREY', 'THE INVESTIGATOR']",19.94,19.94
June,965635764.73,"['FIFTY SHADES OF GREY', 'INFERNO', 'UNLUCKY 13', 'THE GIRL ON THE TRAIN', 'ME BEFORE YOU', 'INTO THE WATER', 'THE CAST', 'WHERE THE CRAWDADS SING', 'CAMINO WINDS', 'THE LAST THING HE TOLD ME', 'IN THE BLOOD']",24.63,24.63
July,1057882730.36,"['FIFTY SHADES OF GREY', 'INFERNO', 'TOP SECRET TWENTY-ONE', 'GREY', 'BAY OF SIGHS', 'CAMINO ISLAND', 'THE PRESIDENT IS MISSING', ""SUMMER OF '69"", '28 SUMMERS', ""THE PRESIDENT'S DAUGHTER"", 'THE HOTEL NANTUCKET']",27.4,27.4
August,717039447.36,"['FIFTY SHADES OF GREY', ""THE CUCKOO'S CALLING"", 'THE BOOK OF LIFE', 'GO SET A WATCHMAN', 'THE GIRL ON THE TRAIN', 'THE LATE SHOW', 'THE OTHER WOMAN', 'THE NEW GIRL', 'THE ORDER', 'THE CELLIST', 'WHERE THE CRAWDADS SING']",26.26,26.26
September,518947402.73,"['FIFTY SHADES OF GREY', ""THE CUCKOO'S CALLING"", 'OUTLANDER', 'FRICTION', 'THE GIRL ON THE TRAIN', 'SEEING RED', 'CRAZY RICH ASIANS', 'WHERE THE CRAWDADS SING', 'ROYAL', 'BLOODLESS', 'WHERE THE CRAWDADS SING']",22.89,22.89
October,615037198.91,"['WINTER OF THE WORLD', 'THE LONGEST RIDE', 'EDGE OF ETERNITY', 'THE MARTIAN', 'THE GIRL ON THE TRAIN', 'A COLUMN OF FIRE', 'LETHAL WHITE', 'THE INSTITUTE', 'THE EVENING AND THE MORNING', 'APPLES NEVER FALL', 'WHERE THE CRAWDADS SING']",16.79,16.79
November,837344895.27,"['THE BONE BED', 'JUST ONE EVIL ACT', 'GONE GIRL', 'SEE ME', 'ESCAPE CLAUSE', 'ORIGIN', 'EVERY BREATH', 'THE GUARDIANS', 'A TIME FOR MERCY', ""THE JUDGE'S LIST"", 'IT STARTS WITH US']",9.65,9.65
December,965326182.09,"['THE LAST MAN', 'SYCAMORE ROW', 'CAPTIVATED BY YOU', 'TRICKY TWENTY-TWO', 'TURBO TWENTY-THREE', 'HARDCORE TWENTY-FOUR', 'LOOK ALIVE TWENTY-FIVE', 'TWISTED TWENTY-SIX', 'RHYTHM OF WAR', 'MERCY', 'IT STARTS WITH US']",6.31,6.31
```

First, we averaged data over every month, and for every month in the dataset we calculated the average box office gross, included a list of the top bestselling books over the years, and calculated the average temperature and precipitation.

joined_data_yearly_calculations.csv

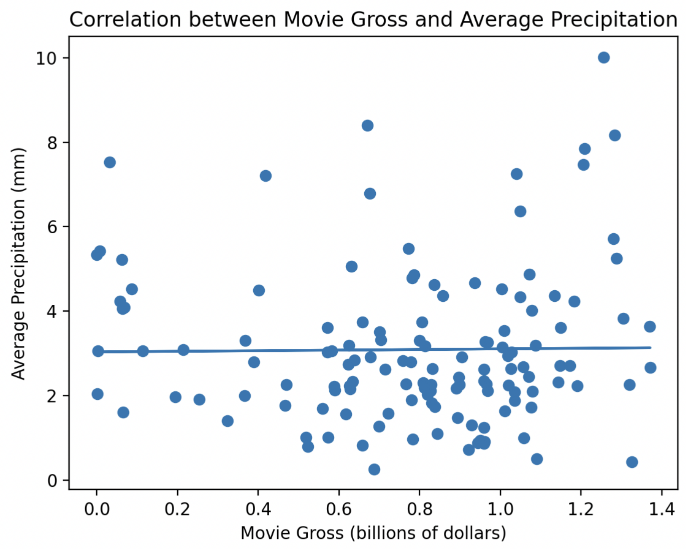


```
Year,Avg gross,Min ISBN,Max ISBN,Avg temp
2012,903636781.0,9780316206815,9781612130293,16.21
2013,912960400.0,9780307588364,9781455520664,14.72
2014,864071820.75,9780307588388,9781101617281,14.49
2015,929065062.25,9780062409850,9781455581177,15.48
2016,947935454.58,9780316408998,9781594634024,15.65
2017,922948960.0,9780316225977,9781455586554,15.86
2018,991013334.25,9780062834881,9781538761373,15.31
2019,946946740.75,9780062834898,9781982110567,15.75
2020,176153900.0,9780062834904,9781984882028,15.77
2021,373567371.08,9780062424037,9781681196282,15.67
2022,614125457.67,9780316258678,9781982181659,15.16
```

We also averaged data over every year, and for every year we calculated the average box office gross, the max and min ISBN (in hopes of maybe spotting a temporal pattern), and the average temperature.

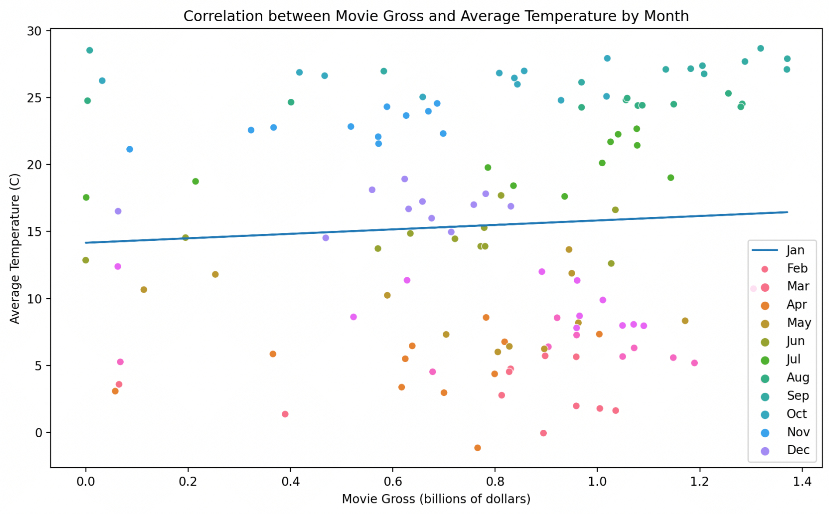
5. The visualization that you created (i.e. screenshot or image file) (10 points)

Visualization 1



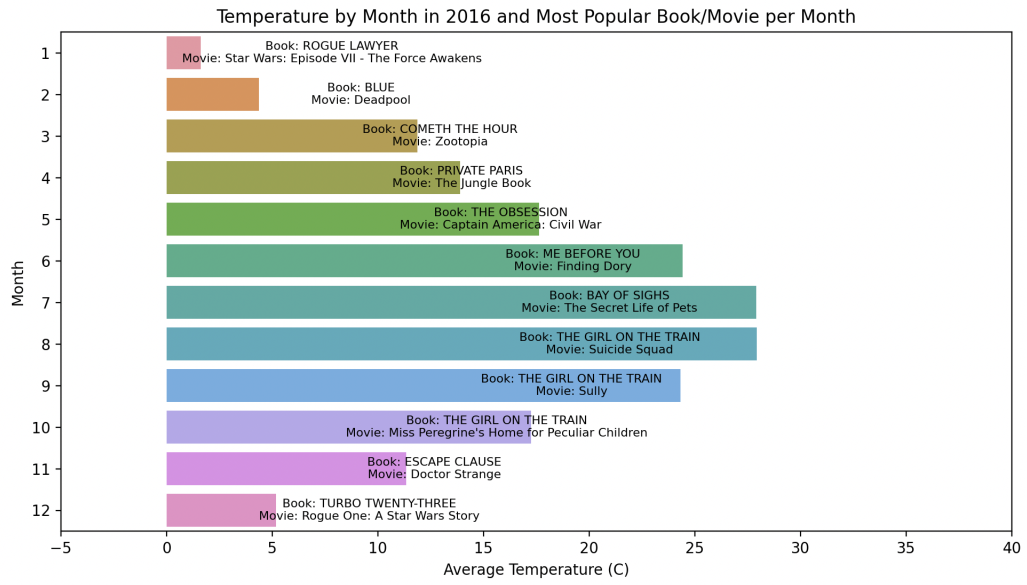
Above is our first visualization, which is a scatter plot of the monthly average precipitation for all 132 months from 2012-2022 compared to the highest movie box office gross for each month. There is little correlation between average monthly precipitation and highest monthly box office gross.

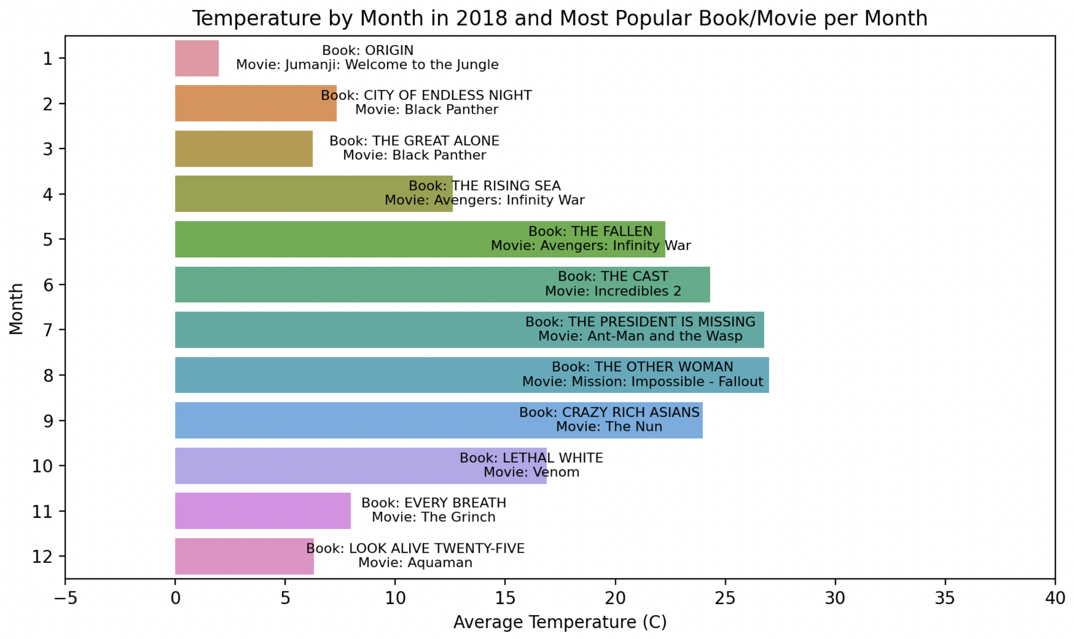
Visualization 2



Above is our second visualization, where we plotted movie gross against average temperature for every month in our dataset, and also color coded the scatterplot by month; we then added a line of best-fit. In this visual, the line of best fit is still pretty flat, but there is a slight slope upwards, showing a slight correlation between higher movie gross and higher temperature. It can also be seen from the data point hues that the movies out in summer that do the best at the box office do as a group do better than the movies out in winter that do the best at the box office.

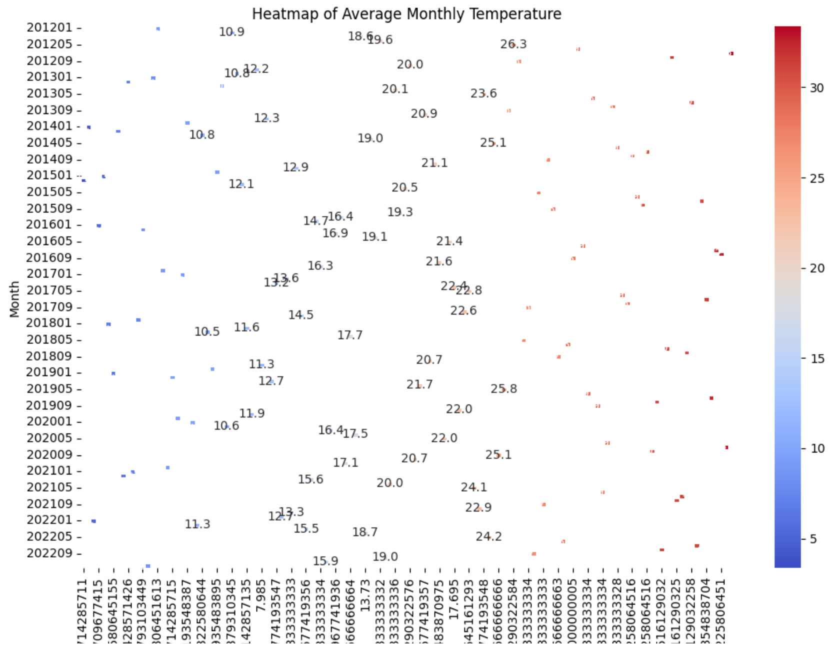
Visualization 3





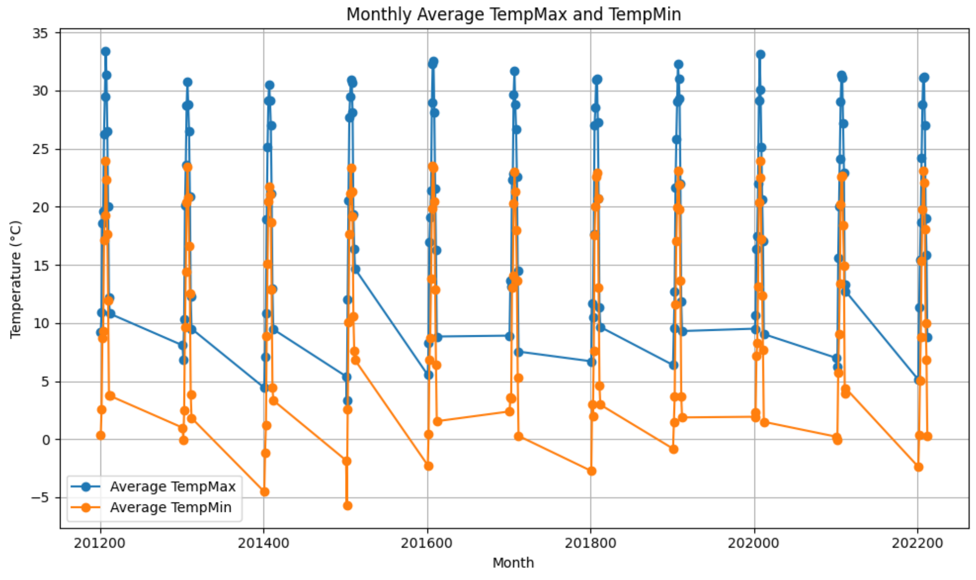
Above are our third visualizations (you can change the year variable in the file to get the visual for the different years). This visualization depicts a bat graph of average temperature by month, along with the most popular (by NYT bestselling or box office gross measures) book and movie for that month, for the year mentioned in the title.

Extra visualization 1



Above is our first extra credit visualization, which is a heatmap of the average monthly temperatures for all the months in our data set.

Extra visualization 2



Above is our second extra credit visualization, which is a line plot of the average monthly temperature vs the average monthly temperature over 2012-2022.

6. Instructions for running your code (10 points)

Linda: Box office instructions
After running the script, it will prompt you to enter the name of a month (in lowercase, full month name). Enter the desired month, and the script will scrape and store the box office data for that month for each year from Box Office Mojo. The script will then retrieve and add the top-grossing titles for that month from 2012 to 2022 from the website to the database. This limits one run of the file to adding only 10 pieces of data to the table (top grossing titles for that month for every year from 2012-2022).

Esha: Book instructions
When my file is run, a prompt will appear asking the user whether they want to add to the Bestsellers, Books, or Reviews table, which they can choose by pressing a number. If they choose to add to the Bestsellers table, it will prompt them to enter a year; then, the file will add data pulled from the API to the Bestsellers table for each month in that year. This limits one run of the file to adding only 12 pieces of data to the Bestsellers table. If they choose to add to the Books table, the file will automatically run through the book ISBNs currently in the Bestsellers table, and request the API to add extra information for up to 25 of them if they're not already present into the Books table; this limits the rows added to 25. If they choose to add to the Reviews table, the file will do a very similar thing, except it will request the reviews API branch and add the appropriate information about up to 25 of the ISBNs in the Bestsellers table to the Reviews table instead. This prompt for input runs in a loop until the user types q for quit.

Ben: Weather instructions
To get the weather data for each year, run get_weather_data.py and enter a year. Then, the file will request the API and then write output for that year to a specific csv file with a name corresponding to that year. To obtain the monthly averages data for temperature, max temperature, minimum temperature, and precipitation, run average_by_month_csv.py. To add the monthly_averages table to the database bell.db, run the file monthly_average_table.py 11 times. This is because the file only inserts 1 year of data into the table each run.

Joined data instructions:
Simply run the file; no user input is necessary.

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

Filename	Function	Input	Output	Description
get_weather_data.py	get_weather_data	None	None	Pulls weather data across the entire United States for each year from 2012-2022
average_by_month_csv.py	average_by_month	None	None	Creates an average_by_month csv file
monthly_average_table.py	write_monthly_data_to_db			Reads monthly average data from averaged_by_month.csv and writes to the monthly averages table in the database. Run 11 times to get all 11 years
tempmax_min_line_graph_month.py	No function	N/A	N/A	Plots a line plot for average temp min and temp max for all 132 months
heatmap_by_month.py	No function	N/A	N/A	Plots a heatmap of average maximum monthly temperatures and the month
average_precip_box_office_graph.py	No function	N/A	N/A	Plots a scatter plot with a line of best fit for average precipitation and box office earning
esha_books.py	create_database	N/A	N/A	Creates the tables for this API if they don’t already exist.
esha_books.py	insert_bestsellers_data	year, conn, cur	N/A	Takes a connection and cursor object and a year for which to insert

				monthly data into the Bestsellers table from the NYT books bestsellers list API.
esha_books.py	insert_books_data	conn, cur	N/A	Takes a connection and cursor object and inserts up to 25 rows into the Books table, using ISBNs in the Bestsellers table to query the NYT books bestsellers list API.
esha_books.py	insert_reviews_data	conn, cur	N/A	Takes a connection and cursor object and inserts up to 25 rows into the Reviews table, using ISBNs in the Bestsellers table to query the NYT reviews API.
group_part.py	attach_databases	N/A	N/A	Attaches the data from the separate box office and books database into the overall database.
group_part.py	join_data	year (Year on which to join data)	data (list of dictionaries with the joined data)	Given a year on which to join the data, runs a join statement joining all tables together, and returns the data formatted into a list of dictionaries (where the dictionary keys are

				the column names, and the items in the list are the rows in the joined data)
group_part.py	calculate_data	data	N/A	Given the data returned from join_data, runs the calculations on it and outputs the results to two CSV files.
average_temp_box_office_graph.py	No function	N/A	N/A	Outputs Visualization 2: a scatterplot of average temperature versus box office gross, colored by month, with a line of best fit.
monthly_temperature_media_graph.py	No function	N/A	N/A	Outputs Visualization 3: With one variable that is variable/settable by the user (the year to graph), outputs a bar graph of average temperature by month with the most popular book and movie per month printed on the bars.
Linda_office_edited.py	get_box_office_page	url	Parsed HTML content	The get_box_office_page function fetches the HTML content of a specified URL (intended to be a

				box office page) and parses it. If successful, it outputs the parsed HTML content; otherwise, it raises an exception if the page fails to load.
Linda_office_edited.py	get_box_office_info	Tr_tag, month	Returns the year, month, movie title, and gross revenue	The get_box_office_info function extracts and returns the year, month, movie title, and gross revenue from a given table row (tr_tag) of a web page. It returns None for the title or gross if either of these details is not found in the row.
Linda_office_edited.py	scrape_info	url, month	N/A	The scrape_info function scrapes box office data for a specified month from a given URL and inserts this data into an SQLite database. It ensures that duplicate entries are not added to the database.
Linda_office_edited.py	month_string_to_int	month	Returns month's numerical representation	The month_string_to_int function converts the name of a month given as a string into its corresponding

				numerical representation (e.g., "January" to 1, "February" to 2, etc.).

8. 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
12/7/2023	Include line of best fit to average precipitation and temperature graph	https://www.statology.org/line-of-best-fit-python/	Fixed issue
11/29/2023	Needed help with parameter setting when plotting graphs	https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html	Fixed issue
11/24/2023	Each API had its own requirements and ways of obtaining data	https://www.visualcrossing.com/weather/weather-data-services	Fixed issue
12/2/2023	Needed help adding bar labels to the media graph	https://matplotlib.org/stable/gallery/lines_bars_and_markers/bar_label_demo.html	Fixed issue
12/2/2023	Needed help making a horizontal bar plot in seaborn	https://stackoverflow.com/questions/54017422/plot-horizontal-bar-plot-with-seaborn	Fixed issue
11/30/23	Got an error code when sending API requests, had to find the API request limit	https://developer.nytimes.com/faq	Fixed issue
12/1/23	Needed help getting started with matplotlib in general	https://matplotlib.org/stable/gallery/index.html#subplots-axes-and-figures	Fixed issue