CS210 PROJECT

Introduction to Data Science

Eren Yamak
31283

19.01.2024

Introduction

The purpose of this project is to test the hypothesis that my Netflix watch time is correlated with the weather conditions. The project utilizes EDA and ML techniques to quantify and analyze my (and my parents') Netflix data from January 2019 to December 2023 and at last compare my seasonal viewing data. My hypotheses are:

Null Hypothesis (H_0): I spend more time on Netflix during cold weather.

Alternative Hypothesis (H_1): There is no correlation between the weather and my watch-time habits.

The data, provided by Netflix themselves, includes My Family Account's:

- Billing Information
- Viewing Activity
- Notifications Received from Spotify
- Lists
- Ratings and several other qualifications to enrich my documentation.

The project also delves into correlations between the above-given data and external data that may have affected them. These external data will be examined further within the scope of the project.

Analysis of the Project

This part of the report will delve into the EDA and the Data Preprocessing part of the project code.

This code segment is part of a comprehensive analysis of a user's Netflix activity. It starts by importing various datasets into pandas DataFrames, a popular Python library for data manipulation and analysis. Each dataset represents a different aspect of the user's interaction with Netflix.

- **Viewing Activity:** The first dataset, loaded from 'ViewingActivity.csv', contains records of the user's viewing history on Netflix. The head() function displays the first few rows, providing a quick overview of the data structure, including titles watched, timestamps, and other relevant details.

- **Ratings:** The next dataset, from 'Ratings.csv', details the ratings given by the user to various shows or movies. This is crucial for understanding user preferences and can be used to recommend similar content.

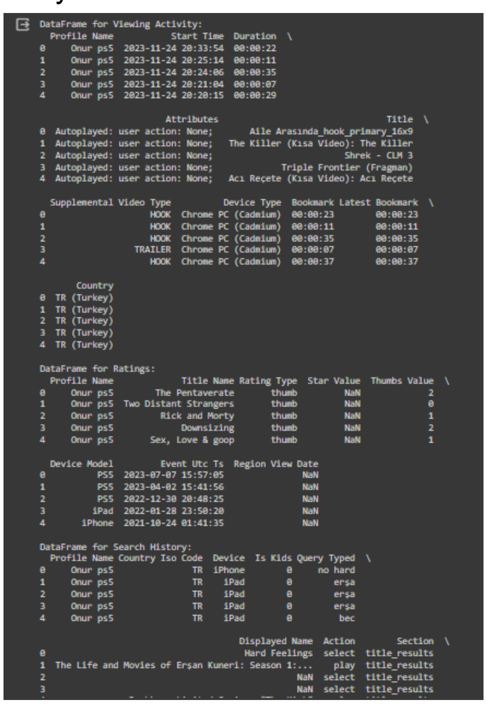
- **Search History:** The 'SearchHistory.csv' file holds data about the searches conducted by the user within the Netflix platform, giving insights into what the user might be interested in watching.

- **Messages from Netflix:** This dataset ('MessagesSentByNetflix.csv') likely contains data regarding communications or notifications sent by Netflix to the user, such as recommendations, updates, or alerts.

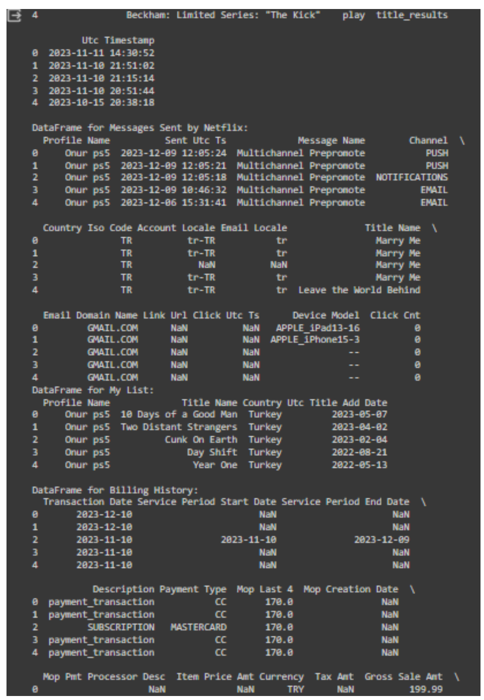
- **My List**: The 'MyList.csv' file includes information about the titles the user has added to their personal 'My List' on Netflix. This reflects the user's interests and intentions to watch certain titles.

- **Billing History:** Lastly, 'BillingHistory.csv' provides financial transaction data related to the user's subscription, including payment dates and amounts.

After loading these datasets, the code employs descriptive statistics and visualizations (like histograms and correlation matrices) to offer deeper insights. Below, each descriptive statistic is analyzed.



```
DataFrame for Viewing Activity:
  Profile Name           Start Time  Duration  \
0     Onur ps5  2023-11-24 20:33:54  00:00:22
1     Onur ps5  2023-11-24 20:25:14  00:00:11
2     Onur ps5  2023-11-24 20:24:06  00:00:35
3     Onur ps5  2023-11-24 20:21:04  00:00:07
4     Onur ps5  2023-11-24 20:20:15  00:00:29

                    Attributes                             Title  \
0  Autoplayed: user action: None;      Aile Arasında_hook_primary_16x9
1  Autoplayed: user action: None;   The Killer (Kısa Video): The Killer
2  Autoplayed: user action: None;                        Shrek - CLM 3
3  Autoplayed: user action: None;             Triple Frontier (Fragman)
4  Autoplayed: user action: None;   Acı Reçete (Kısa Video): Acı Reçete

  Supplemental Video Type          Device Type  Bookmark Latest Bookmark  \
0                    HOOK  Chrome PC (Cadmium)  00:00:23        00:00:23
1                    HOOK  Chrome PC (Cadmium)  00:00:11        00:00:11
2                    HOOK  Chrome PC (Cadmium)  00:00:35        00:00:35
3                 TRAILER  Chrome PC (Cadmium)  00:00:07        00:00:07
4                    HOOK  Chrome PC (Cadmium)  00:00:37        00:00:37

       Country
0  TR (Turkey)
1  TR (Turkey)
2  TR (Turkey)
3  TR (Turkey)
4  TR (Turkey)

DataFrame for Ratings:
  Profile Name            Title Name Rating Type  Star Value  Thumbs Value  \
0     Onur ps5       The Pentaverate       thumb         NaN             2
1     Onur ps5  Two Distant Strangers       thumb         NaN             0
2     Onur ps5         Rick and Morty       thumb         NaN             1
3     Onur ps5             Downsizing       thumb         NaN             2
4     Onur ps5      Sex, Love & goop       thumb         NaN             1

  Device Model         Event Utc Ts  Region View Date
0          PS5  2023-07-07 15:57:05               NaN
1          PS5  2023-04-02 15:41:56               NaN
2          PS5  2022-12-30 20:48:25               NaN
3         iPad  2022-01-28 23:50:20               NaN
4       iPhone  2021-10-24 01:41:35               NaN

DataFrame for Search History:
  Profile Name Country Iso Code  Device  Is Kids Query Typed  \
0     Onur ps5               TR  iPhone        0     no hard
1     Onur ps5               TR    iPad        0        erşa
2     Onur ps5               TR    iPad        0        erşa
3     Onur ps5               TR    iPad        0        erşa
4     Onur ps5               TR    iPad        0         bec

                                  Displayed Name  Action        Section  \
0                                  Hard Feelings  select  title_results
1  The Life and Movies of Erşan Kuneri: Season 1:...    play  title_results
2                                            NaN  select  title_results
3                                            NaN  select  title_results
```



```
4            Beckham: Limited Series: "The Kick"    play  title_results

         Utc Timestamp
0  2023-11-11 14:30:52
1  2023-11-10 21:51:02
2  2023-11-10 21:15:14
3  2023-11-10 20:51:44
4  2023-10-15 20:38:18

DataFrame for Messages Sent by Netflix:
  Profile Name          Sent Utc Ts            Message Name        Channel  \
0     Onur ps5  2023-12-09 12:05:24  Multichannel Prepromote           PUSH
1     Onur ps5  2023-12-09 12:05:21  Multichannel Prepromote           PUSH
2     Onur ps5  2023-12-09 12:05:18  Multichannel Prepromote  NOTIFICATIONS
3     Onur ps5  2023-12-09 10:46:32  Multichannel Prepromote          EMAIL
4     Onur ps5  2023-12-06 15:31:41  Multichannel Prepromote          EMAIL

  Country Iso Code Account Locale Email Locale             Title Name  \
0               TR          tr-TR           tr               Marry Me
1               TR          tr-TR           tr               Marry Me
2               TR            NaN          NaN               Marry Me
3               TR          tr-TR           tr               Marry Me
4               TR          tr-TR           tr  Leave the World Behind

  Email Domain Name Link Url Click Utc Ts      Device Model  Click Cnt
0         GMAIL.COM      NaN          NaN   APPLE_iPad13-16          0
1         GMAIL.COM      NaN          NaN  APPLE_iPhone15-3          0
2         GMAIL.COM      NaN          NaN                --          0
3         GMAIL.COM      NaN          NaN                --          0
4         GMAIL.COM      NaN          NaN                --          0
DataFrame for My List:
  Profile Name            Title Name Country Utc Title Add Date
0     Onur ps5  10 Days of a Good Man  Turkey         2023-05-07
1     Onur ps5  Two Distant Strangers  Turkey         2023-04-02
2     Onur ps5          Cunk On Earth  Turkey         2023-02-04
3     Onur ps5              Day Shift  Turkey         2022-08-21
4     Onur ps5               Year One  Turkey         2022-05-13

DataFrame for Billing History:
  Transaction Date Service Period Start Date Service Period End Date  \
0       2023-12-10                       NaN                     NaN
1       2023-12-10                       NaN                     NaN
2       2023-11-10                2023-11-10              2023-12-09
3       2023-11-10                       NaN                     NaN
4       2023-11-10                       NaN                     NaN

          Description Payment Type  Mop Last 4  Mop Creation Date  \
0  payment_transaction           CC       170.0                NaN
1  payment_transaction           CC       170.0                NaN
2         SUBSCRIPTION   MASTERCARD       170.0                NaN
3  payment_transaction           CC       170.0                NaN
4  payment_transaction           CC       170.0                NaN

  Mop Pmt Processor Desc  Item Price Amt Currency  Tax Amt  Gross Sale Amt  \
0                    NaN             NaN      TRY      NaN          199.99
```

```
  Mop Pmt Processor Desc  Item Price Amt Currency  Tax Amt  Gross Sale Amt  \
0                    NaN             NaN      TRY      NaN          199.99
1                    NaN             NaN      TRY      NaN          199.99
2             PAYMENTECH          199.99      TRY    33.33          199.99
3                    NaN             NaN      TRY      NaN          199.99
4                    NaN             NaN      TRY      NaN          199.99

  Pmt Txn Type Pmt Status Final Invoice Result Country Next Billing Date
0         SALE        NEW                  NaN      TR               NaN
1         SALE   APPROVED                  NaN      TR               NaN
2         SALE    PENDING              SETTLED      TR        2023-12-10
3         SALE    PENDING                  NaN      TR               NaN
4         SALE        NEW                  NaN      TR               NaN
Descriptive Statistics for My List:
       Profile Name Title Name Country Utc Title Add Date
count           473        473         473            473
unique            5        401           1            249
top      thanks bro     Maniac      Turkey     2022-09-03
freq            259          4         473             12
```

(Due to extreme amount of data, screenshots are divided for readability.)

- **Descriptive Statistics for Each DataFrame:** The code uses the describe() method on each DataFrame. This method generates descriptive statistics that summarize the central tendency, dispersion, and shape of the dataset's distribution, excluding NaN values. It typically provides the count, mean, standard deviation, minimum, quartiles, and maximum for numeric columns. For non-numeric columns, it includes count, unique, top, and frequency of the top element. These statistics are essential for understanding the basic characteristics of the data without delving into complex analysis.

  For instance, in the 'My List' DataFrame, the describe(include='all') function provides insights into not just the numerical aspects but all columns, including categorical data like movie titles or genres. This can reveal the most common genre or the average number of movies in a specific category that the user adds to their list.

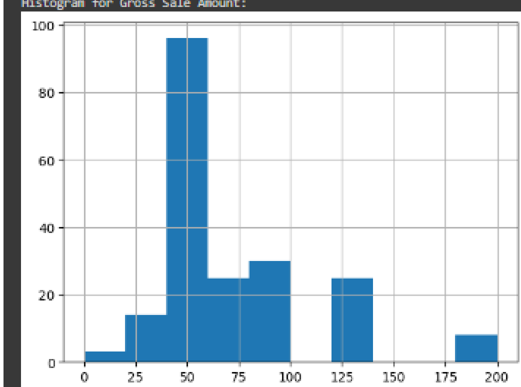
- **Histogram for 'Gross Sale Amount':** The code includes a histogram for the 'Gross Sale Amount' from the billing history. Histograms are graphical representations of data distributions. In this case, it shows the frequency distribution of the amount spent by the user on Netflix. This can reveal spending patterns, like the most common transaction amounts or how spending varies over time.

- **Correlation Matrix for Billing History:** Finally, the code generates a correlation matrix specifically for the billing history data. A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables. The values range from -1 to 1. A value close to 1 implies a strong positive correlation (as one variable increases, so does the other), a value close to -1 implies a strong negative correlation (as one increases, the other decreases), and a value around 0 implies no correlation. This matrix is visually represented using a heatmap from the seaborn library, making it easier to identify any significant relationships between different billing aspects, such as the relationship between the gross sale amount and other numerical factors in the billing history.

Together, these descriptive statistical tools give a comprehensive snapshot of the user's interaction with Netflix. They help in identifying trends, patterns, and anomalies in the data, which are crucial for any further detailed analysis or predictive modeling.

```
Descriptive Statistics for My List:
       Profile Name Title Name Country Utc Title Add Date
count           473        473     473               473
unique            5        401       1               249
top      thanks bro     Maniac  Turkey        2022-09-03
freq            259          4     473                12

Descriptive Statistics for Billing History:
        Mop Last 4  Mop Creation Date  Item Price Amt     Tax Amt  \
count   197.000000                0.0      101.000000  101.000000
mean   4965.258883                NaN       73.554356   11.447228
std    1991.696618                NaN       37.269670    6.379667
min     170.000000                NaN       39.990000    0.000000
25%    5324.000000                NaN       41.990000    6.410000
50%    5527.000000                NaN       54.990000    8.390000
75%    5527.000000                NaN       93.990000   14.340000
max    8990.000000                NaN      199.990000   33.330000

       Gross Sale Amt
count      201.000000
mean        74.054826
std         39.899481
min          0.000000
25%         41.990000
50%         54.990000
75%         93.990000
max        199.990000

Descriptive Statistics for Viewing Activity:
       Profile Name           Start Time  Duration  \
count         18924                18924     18924
unique            5                18893      3774
top      thanks bro  2022-08-15 20:35:27  00:00:04
freq           5726                    2       315

                        Attributes       Title Supplemental Video Type  \
count                         2349       18924                    1170
unique                           5        7636                       5
top     Autoplayed: user action: None;  Sadakatsiz                    HOOK
freq                          1789          27                     838

          Device Type  Bookmark  Latest Bookmark      Country
count           18924     18924            18924        18924
unique             18      4521             3482            3
top     iPad Air 2 WiFi  00:00:05  Not latest view  TR (Turkey)
freq             5420       194             8350        18922

Descriptive Statistics for Ratings:
       Star Value  Thumbs Value  Region View Date
count         0.0     84.000000               0.0
mean          NaN      1.523810               NaN
std           NaN      0.735931               NaN
min           NaN      0.000000               NaN
25%           NaN      1.000000               NaN
50%           NaN      2.000000               NaN
75%           NaN      2.000000               NaN
```

```
Descriptive Statistics for Search History:
        Is Kids
count    1647.0
mean        0.0
std         0.0
min         0.0
25%         0.0
50%         0.0
75%         0.0
max         0.0

Histogram for Gross Sale Amount:
```




```
Correlation Matrix for Billing History:
<ipython-input-47-ca05bcf9e841>:66: FutureWarning: The default value of numeric_only in D
  sns.heatmap(billing_history_data.corr(), annot=True)
```

Billing Information

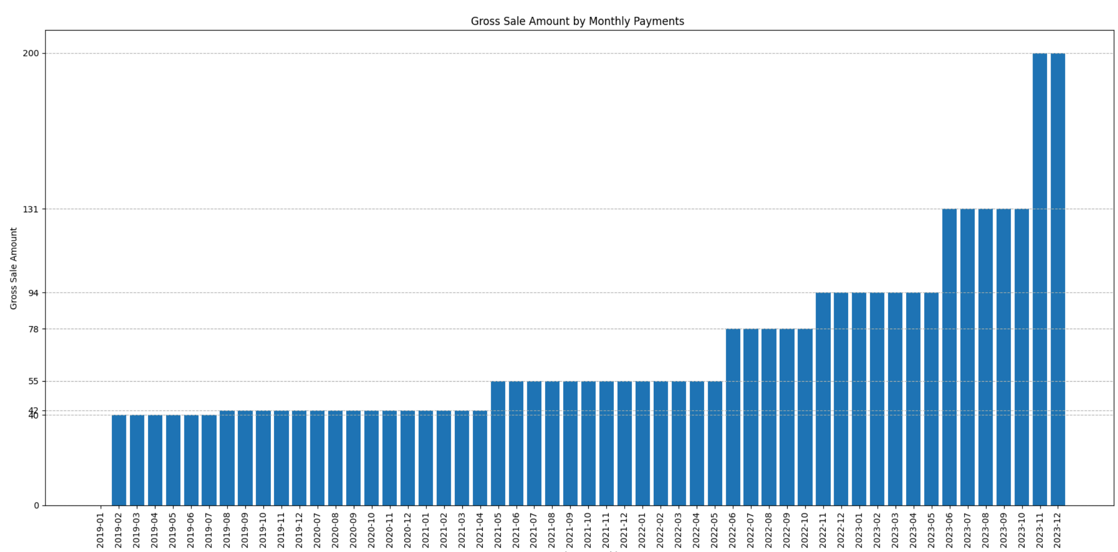
In this part of the code, credit card payments since the beginning of 2019 are analyzed. Initially, the code loads the billing data from 'BillingHistory.csv' into a DataFrame called billing_data. This DataFrame includes various details of billing transactions such as dates, payment types, and amounts. The transaction dates are then formatted into pandas' datetime format for precise date-related operations.

The main analysis begins by filtering the dataset to select only those transactions where the payment type is 'Credit Card' (CC) and the payment status is 'APPROVED'. Although the actual raw data is not shared, to emphasize, normally the data also contains 'PENDING' payment status, which indicates that a payment is delayed from its initial time, 'CANCELLED' which is for credit cards that are cancelled and therefore cannot be used and 'DECLINED' for payments that were declined by the bank due to insufficient funds etc. The chosen subset of data is stored in a new DataFrame, cc_approved_payments. To facilitate a monthly summary of expenditures, a 'Year-Month' column is created by extracting the year and month from the 'Transaction Date' and converting it into a period format.

The next step involves summarizing the total amount spent each month. The code accomplishes this by grouping the data by the new 'Year-Month' column and summing up the 'Gross Sale Amt' for each group. This process effectively consolidates the credit card payments into monthly totals. The result is a new DataFrame, cc_approved_payment_summary_df, which is then displayed. This DataFrame neatly presents each month and year, along with the total amount of approved credit card payments for that period.

	Year-Month	Gross Sale Amt
0	2019-01	0.00
1	2019-02	39.99
2	2019-03	39.99
3	2019-04	39.99
4	2019-05	39.99
5	2019-06	39.99
6	2019-07	39.99
7	2019-08	41.99
8	2019-09	41.99
9	2019-10	41.99
10	2019-11	41.99
11	2019-12	41.99
12	2020-07	41.99
13	2020-08	41.99
14	2020-09	41.99
15	2020-10	41.99
16	2020-11	41.99
17	2020-12	41.99
18	2021-01	41.99
19	2021-02	41.99
20	2021-03	41.99
21	2021-04	41.99
22	2021-05	54.99
23	2021-06	54.99
24	2021-07	54.99
25	2021-08	54.99
26	2021-09	54.99
27	2021-10	54.99
28	2021-11	54.99
29	2021-12	54.99
30	2022-01	54.99
31	2022-02	54.99
32	2022-03	54.99
33	2022-04	54.99
34	2022-05	54.99
35	2022-06	77.99
36	2022-07	77.99
37	2022-08	77.99
38	2022-09	77.99
39	2022-10	77.99
40	2022-11	93.99
41	2022-12	93.99
42	2023-01	93.99
43	2023-02	93.99
44	2023-03	93.99
45	2023-04	93.99
46	2023-05	93.99
47	2023-06	130.99
48	2023-07	130.99
49	2023-08	130.99
50	2023-09	130.99
51	2023-10	130.99
52	2023-11	199.99
53	2023-12	199.99

Later, the left DataFrame of the approved is converted into a bar chart that has 'Gross Sale Amount' as y-axis and Date (in Year-Month) format as x-axis. Below, the screenshot of this plot is given:



Although initially I thought that this was conclusive for the billing information; in later stages of the code, something popped up inside my head. I thought that the billing data and the inflation rate of Turkey could have been correlated. So, I thought of a plot that represents both. But for this, I needed the inflation data of the country. Hence, I extracted the data from the Central Bank of the Turkish Republic (Merkez Bankası) on Consumer Price Index (TÜFE). Below left, the data I used from the Central Bank is given.

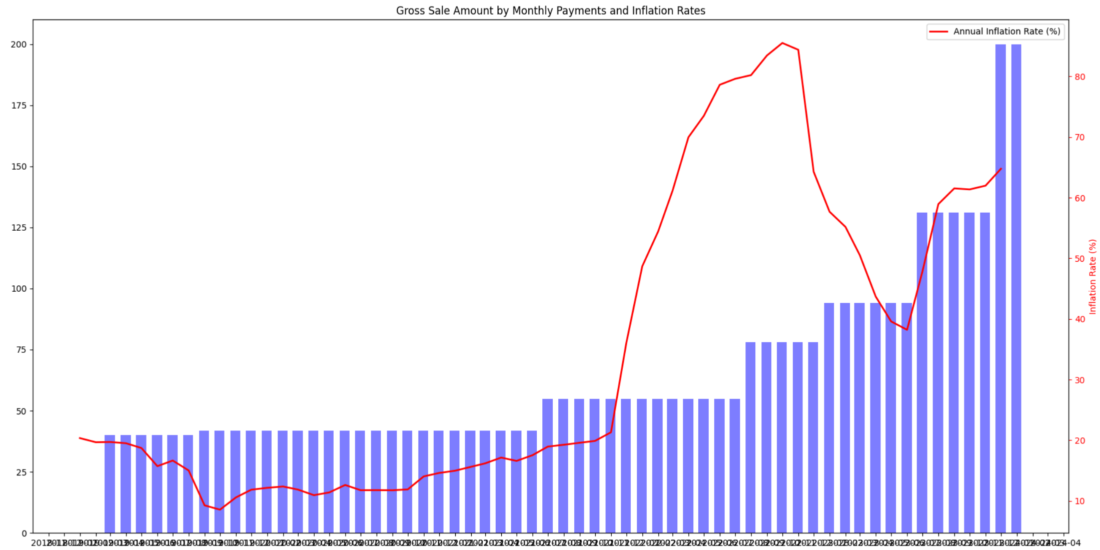
	TÜFE (Yıllık % Değişim)
12-2023	64.77
11-2023	61.98
10-2023	61.36
09-2023	61.53
08-2023	58.94
07-2023	47.83
06-2023	38.21
05-2023	39.59
04-2023	43.68
03-2023	50.51
02-2023	55.18
01-2023	57.68
12-2022	64.27
11-2022	84.39
10-2022	85.51
09-2022	83.45
08-2022	80.21
07-2022	79.60
06-2022	78.62
05-2022	73.50
04-2022	69.97
03-2022	61.14
02-2022	54.44
01-2022	48.69
12-2021	36.08
11-2021	21.31
10-2021	19.89
09-2021	19.58
08-2021	19.25
07-2021	18.95
06-2021	17.53
05-2021	16.59
04-2021	17.14
03-2021	16.19
02-2021	15.61
01-2021	14.97
12-2020	14.60
11-2020	14.03
10-2020	11.89
09-2020	11.75
08-2020	11.77
07-2020	11.76
06-2020	12.62
05-2020	11.39
04-2020	10.94
03-2020	11.86
02-2020	12.37
01-2020	12.15
12-2019	11.84
11-2019	10.56
10-2019	8.55
09-2019	9.26
08-2019	15.01
07-2019	16.65
06-2019	15.72
05-2019	18.71
04-2019	19.50
03-2019	19.71
02-2019	19.67
01-2019	20.35

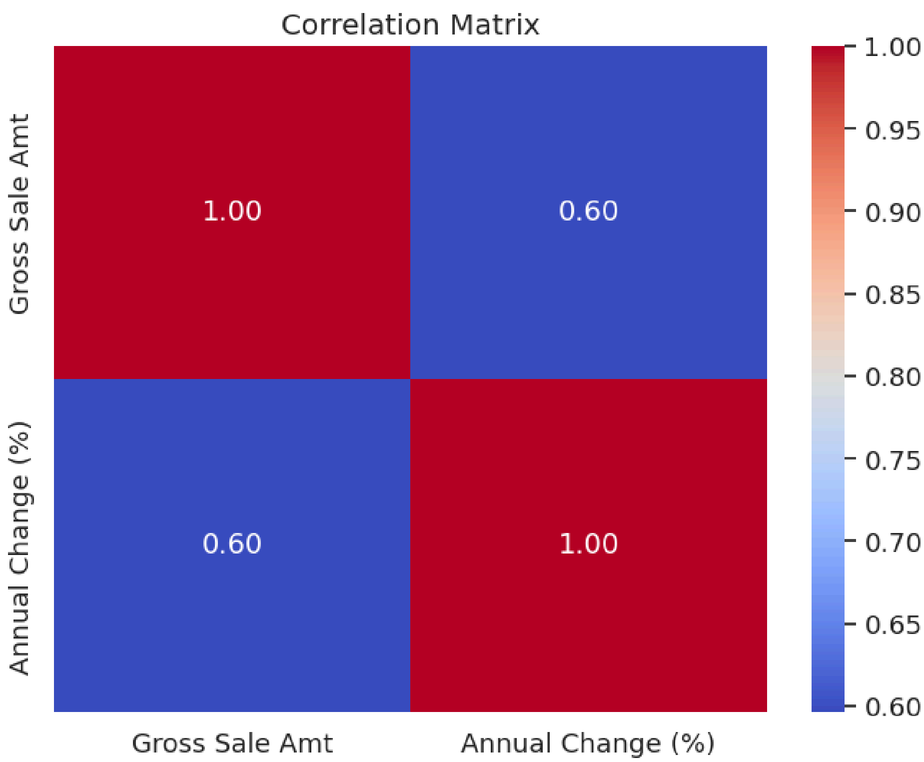
By juxtaposition, the code creates a dual-perspective plot.The billing data, represented as a blue bar chart, shows Netflix payments over time, while the inflation data is overlaid as a red line graph, indicating annual inflation rates.

Key to this visualization is the use of a dual-axis plot: one axis for the billing amounts and another for the inflation rates. This setup allows for an effective comparison between Netflix's revenue trends and broader economic conditions represented by inflation.
The plot is carefully designed for clarity and ease of interpretation. Dates are formatted in a 'Year-Month' style and rotated for better readability. The distinct color schemes (blue for billing, red for inflation) and the inclusion of legends aid in distinguishing between the two datasets.

The data seemed to be increasing in positive direction with the inflation. So, to further justify my gut feelings, I also created a correlation matrix between the inflation rates and the billing information. It begins by resampling the inflation data to a monthly frequency, providing a more granular view of inflation trends. The resulting dataset, inflation_df_resampled, is merged with the billing data (billing_df) based on the 'Year-Month' column, ensuring alignment by date. The code then computes a correlation matrix, quantifying the linear relationship between 'Gross Sale Amt' (Netflix's billing amounts) and 'Annual Change (%)' (inflation rates).

To visualize the correlation, a heatmap is created using seaborn (sns). This heatmap annotates correlation values, with color coding indicating the strength and direction of correlations. A positive value suggests a positive correlation, while a negative value implies a negative correlation. In this specific case, the correlation matrix output indicates a moderately positive correlation of approximately 0.596638 between billing amounts and inflation rates. This data-driven insight helps in understanding how economic conditions may impact Netflix's revenue trends.


Gross Sale Amount by Monthly Payments and Inflation Rates
Annual Inflation Rate (%)
Inflation Rate (%)
0
25
50
75
100
125
150
175
200
10
20
30
40
50
60
70
80


Correlation Matrix
Gross Sale Amt
Annual Change (%)
1.00
0.60
0.60
1.00
Gross Sale Amt
Annual Change (%)
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60

Viewing Activity

The viewing activity part of the code is seperated into two parts: one for me and one for the entire family.

In the first part, which is mine, my watch-time is analyzed. The code begins by loading a CSV file into a pandas DataFrame, which contains data about each user's viewing history. The 'Start Time' column is converted to a datetime object and set as the index of the DataFrame. The DataFrame is then filtered to only include viewing activity from the profile named 'eren'. Unnecessary columns are dropped from the DataFrame for simplicity. At this point, the DataFrame only consists of rows that have "Profile Name" value as 'eren'.

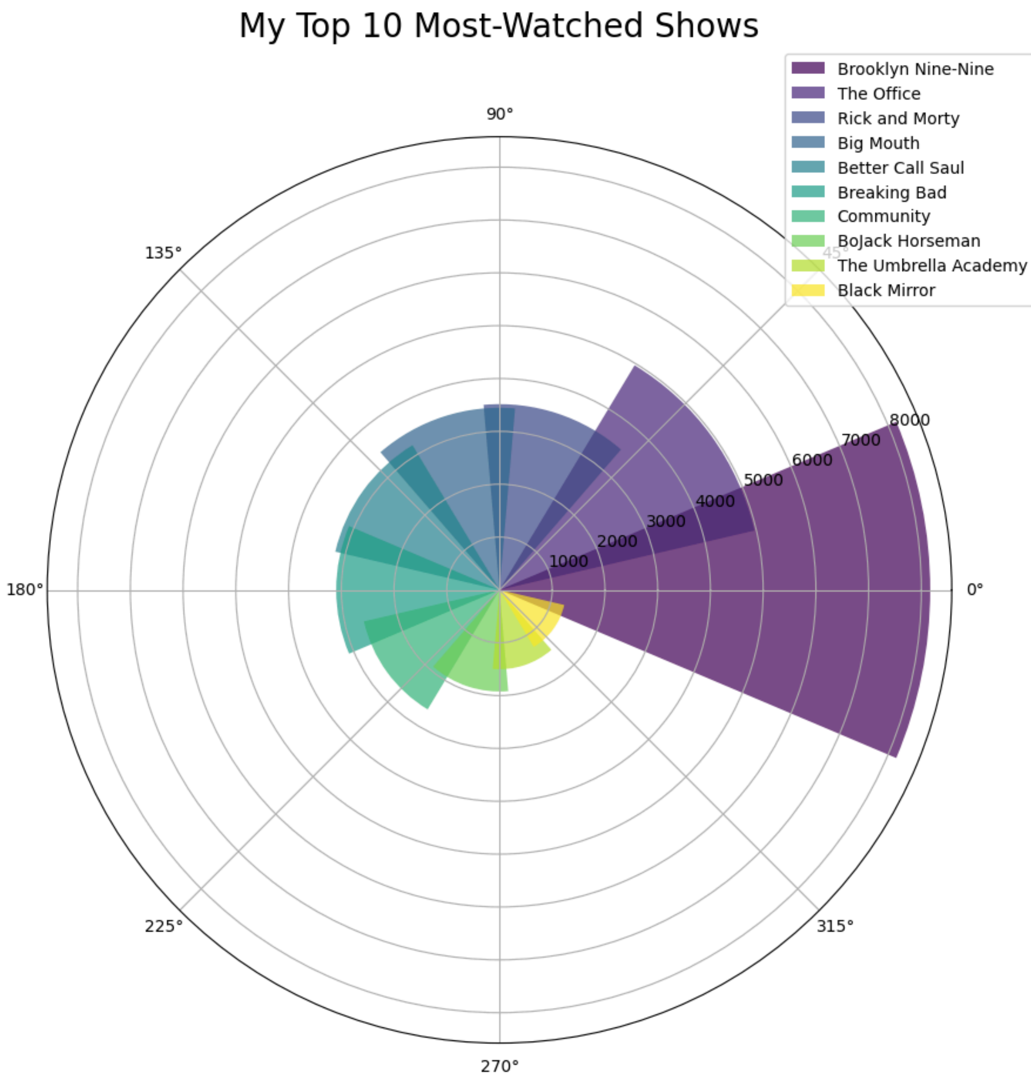
The 'Duration' column, which represents the length of each viewing session, is converted to a timedelta object. A new column, 'Show Identifier', is created by extracting the part of the title before the colon. This is done under the assumption that the part of the title before the colon is a common identifier for each show.

The DataFrame is then grouped by 'Show Identifier', and the 'Duration' of each group is summed to find the total viewing duration for each show. The show with the maximum total duration is identified as the most-watched show. The total duration is then converted from a timedelta object to a more human-readable format (days, hours, minutes, and seconds).

According to the output of the code, the most-watched show is 'Brooklyn Nine-Nine', with a total viewing duration of 5 days, 16 hours, 7 minutes, and 46 seconds, or equivalently, 8167 minutes. This actually made sense to me. Even though I finished the series first back in 2022, I still watch it from time to time.

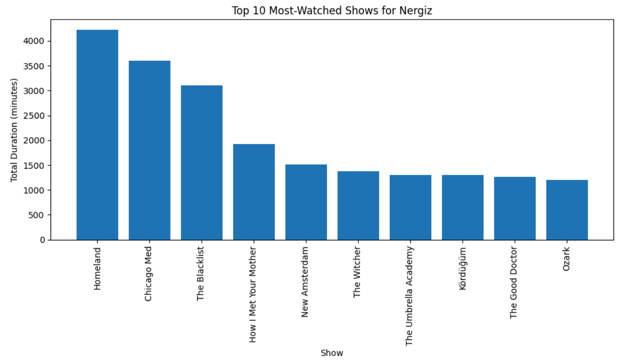
In the second part of "my personal watch time information" code, I wanted to see my top 10 most watched series. This code also follows the same logic with the first one but uses "iterrows" and iterates over the shows with maximum duration value. I also wanted to visualize this more aesthetically, so I used a polar plot along with the bar chart.

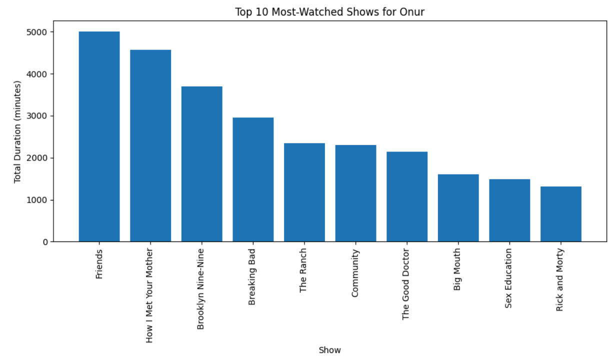
```
1. Brooklyn Nine-Nine: 8167.77 minutes
2. The Office: 4962.28 minutes
3. Rick and Morty: 3514.82 minutes
4. Big Mouth: 3446.72 minutes
5. Better Call Saul: 3195.25 minutes
6. Breaking Bad: 3096.33 minutes
7. Community: 2639.28 minutes
8. BoJack Horseman: 1921.93 minutes
9. The Umbrella Academy: 1500.20 minutes
10. Black Mirror: 1260.47 minutes
```

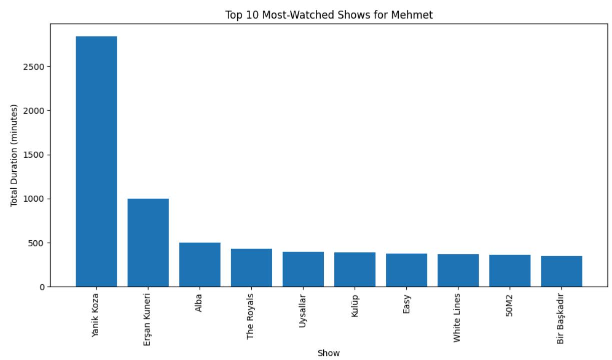


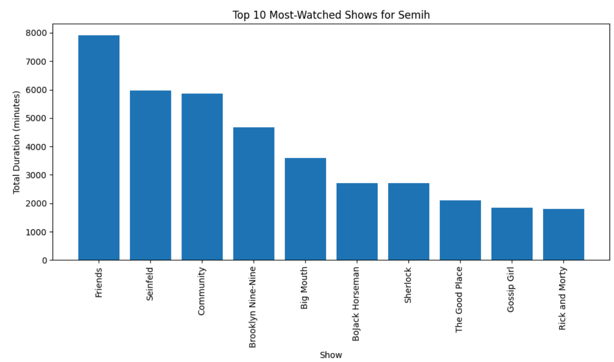
In the second part of the “Viewing Activity”, I did the same for my “Netflix family”; my mother Nergiz, my father Mehmet, my brother Onur and my close friend that I gave my account to years ago, Semih. The logics behind the procurement of the most-watched series and most watched 10 series are pretty much the same with the above explained code. Hence, I won’t delve into it in great detail. But, I just want to point out that after this point, I’ll always be changing the Netflix Names of them. As my brothers Netflix profile name is “Onur ps5” and Semih’s profile name is “thanks bro”, this would clarify the project in most ways.

```
Most-watched show for Nergiz:
Show Identifier: Homeland
Duration: 2 days, 22 hours, 17 minutes, and 51 seconds


Most-watched show for Onur:
Show Identifier: Friends
Duration: 3 days, 11 hours, 29 minutes, and 18 seconds


Most-watched show for Mehmet:
Show Identifier: Yanik Koza
Duration: 1 days, 23 hours, 20 minutes, and 43 seconds


Most-watched show for Semih:
Show Identifier: Friends
Duration: 5 days, 11 hours, 59 minutes, and 49 seconds
```

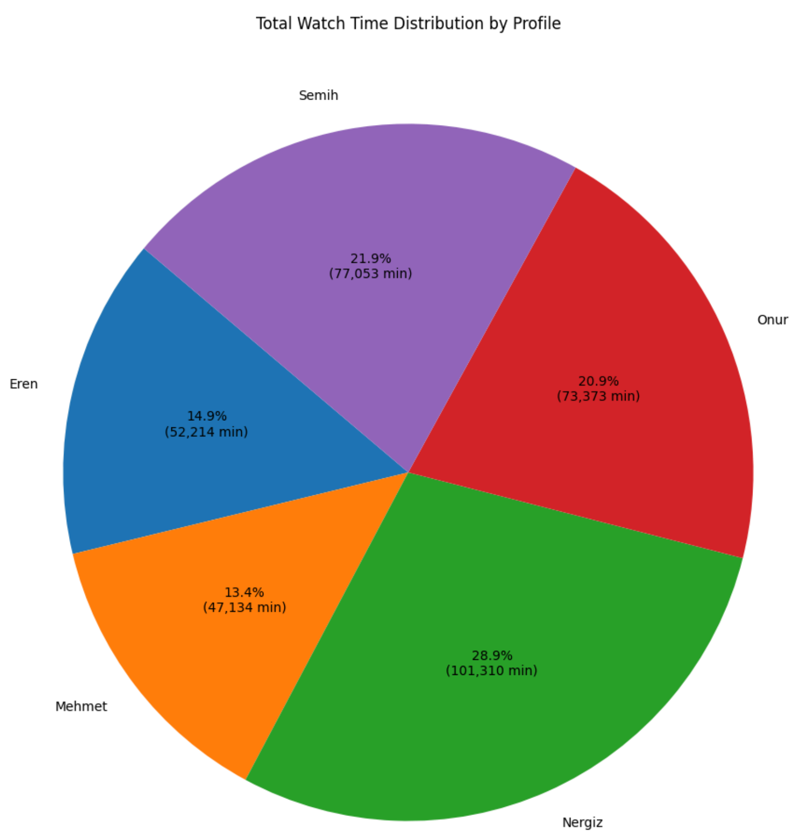








Lastly, at the end of the Viewing Activity part, I compared everybody's watching times in a pie chart. This felt senseful, as the person with the highest watch time would have had the biggest slice of the pie.



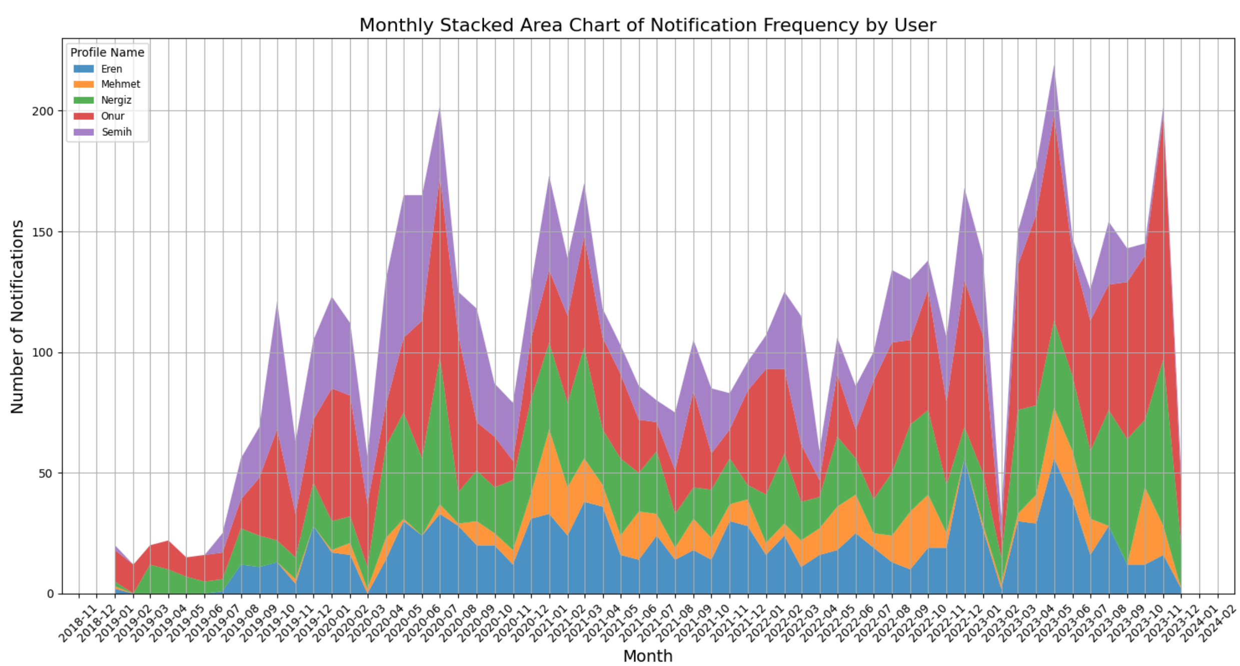
According to the plot, my mom has actually spent most time while watching, which is 101.310 minutes, almost 71 days nonstop. Seeing this felt hilarious, as she would always tell us that we are spending so much time watching Netflix. I, on the other hand, came on 4th place with 52.214 minutes, almost 36 days nonstop. I felt that my watch-time would have been substantially more, as I always binge watch several series during the summers. (Spoiler: I was wrong.)

Netflix Notifications per User and Their Frequencies

For this part, I wanted to see how many notifications Netflix sent per user. Although this information itself is not really needed, I believe that it is a crucial part for the correlations that it may birth.
This code is enclosed in a try-except block for error handling. After loading the CSV file and replacing the names as I previously explained, the value_counts() method is applied to the 'Profile Name' column to count the frequency of each unique value, essentially counting how many messages each profile sent. If any error occurs during these operations, such as a file not being found or a syntax error, the exception is caught in the except block, and a descriptive error message is assigned to notifications_count_updated. Finally, the code displays the top five profile names with the highest message counts using the .head() method, showing 'Onur' at the top with 2137 messages.

```
Onur      2137
Nergiz    1407
Semih     1267
Eren      1120
Mehmet     475
Name: Profile Name, dtype: int64
```

The visualization code first converts the 'Sent Utc Ts' column of the messages_df DataFrame into a datetime format. This is crucial for time-based analysis as it allows the code to manipulate and group data based on dates and times.
Next, it groups the data by both the date and the profile name, counting the number of occurrences (i.e., how many messages were sent each day by each user). This step is essential for understanding daily usage patterns per user.
The code then further groups this data by month and profile name, summing up these daily counts to get a total count of messages for each month for each user. This monthly grouping provides a clearer view of longer-term trends in message frequency.
Afterwards, the code pivots the data to prepare it for visualization. In this pivoted format, each row represents a month, and each column represents a different user, with the values being the count of messages sent in that month.



Rating Analysis and Correlation Between Rating Number and Rating Average

Now, in this section of the code, I used the “Rating.csv” file to obtain each user’s rating count and rating average. The code focuses on a column named 'Thumbs Value', containing the ratings. These ratings are converted to numeric values, with any non-numeric entries being replaced by NaN (which we have also done in HW2). Rows with these NaN values are then removed. My main objective with this was to clean the data further and conduct my analysis with valid, numerical values only.

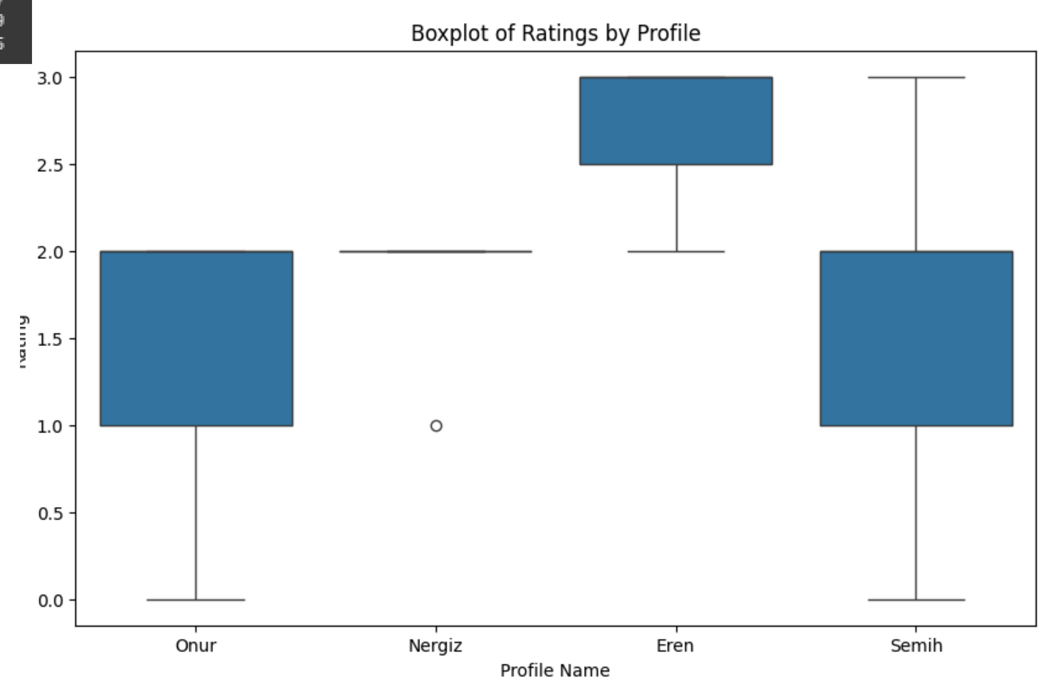
The code then calculates two key pieces of information:

- **Average Rating Per User:** It groups the data by 'Profile Name' and calculates the mean (average) of the 'Thumbs Value' for each user. This average rating is indicative of how each user generally rates content.

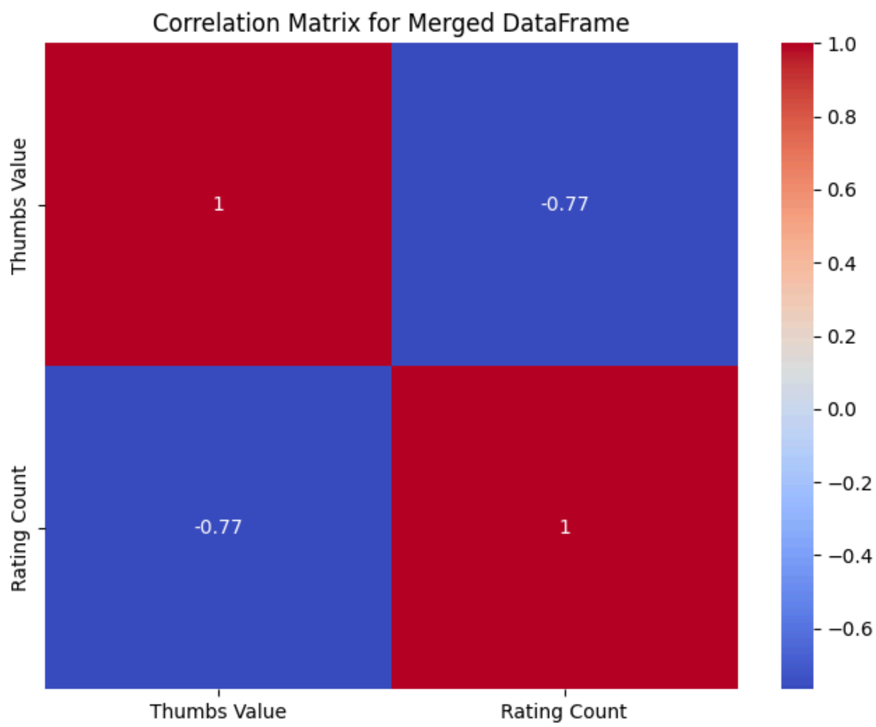
- **Count of Ratings Per User:** It also counts the number of ratings made by each user. This count provides insight into how active each user is in terms of rating content.

After obtaining the values, I’ve plotted a box plot to show the variance of ratings for each user and the maximums, minimums and averages. The box plot suggested that my mother keeps on giving the same rating to each series’, which is 2. This data also showed me that I’m light hearted when it comes to rating series.

```
  Profile Name  Thumbs Value
0         Eren      2.666667
1       Nergiz      1.857143
2         Onur      1.631579
3        Semih      1.381818
  Profile Name  Rating Count
0         Eren             3
1       Nergiz             7
2         Onur            19
3        Semih            55
```



But, the data also showed me something interesting. People with more ratings tend to have a lower rating average. So, to test this, I also calculated the correlation coeffient between the Rating Count and Rating Average and later created a correlation matrix between them. The correlation coefficient yielded a solution of -0.77, which indicated that there is a strong negative correlation between them and that people with higher rating counts (ie. people who do rate more often) have a higher chance of rating a lower score than the average watcher.



Search Behavior

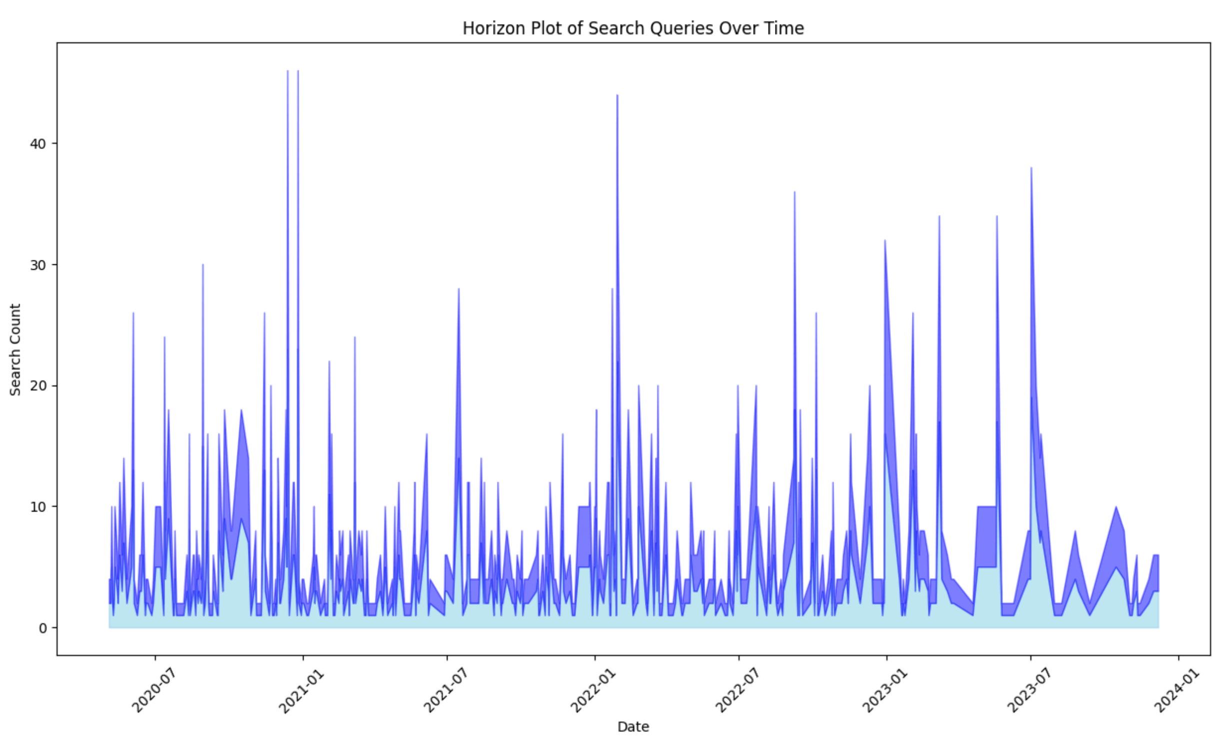
Upon visiting the file "Search History", I decided the find the most searched words. I decided to create a word cloud for this and to provide a graphical representation of the most common terms in the search history, helping to quickly grasp key themes or topics that are frequently searched for. The code prepares the text for the word cloud by combining all the search queries from the 'Query Typed' column of the "Search History" file into a single string. Any missing values in the queries are dropped to ensure that only actual text is included.

Then, a WordCloud object is created. This object is configured to generate a word cloud image with specified dimensions (800 pixels wide and 400 pixels high) and a white background. The WordCloud object takes the combined string of search queries and processes it to create the word cloud, where the most frequently occurring words appear larger than those less common.

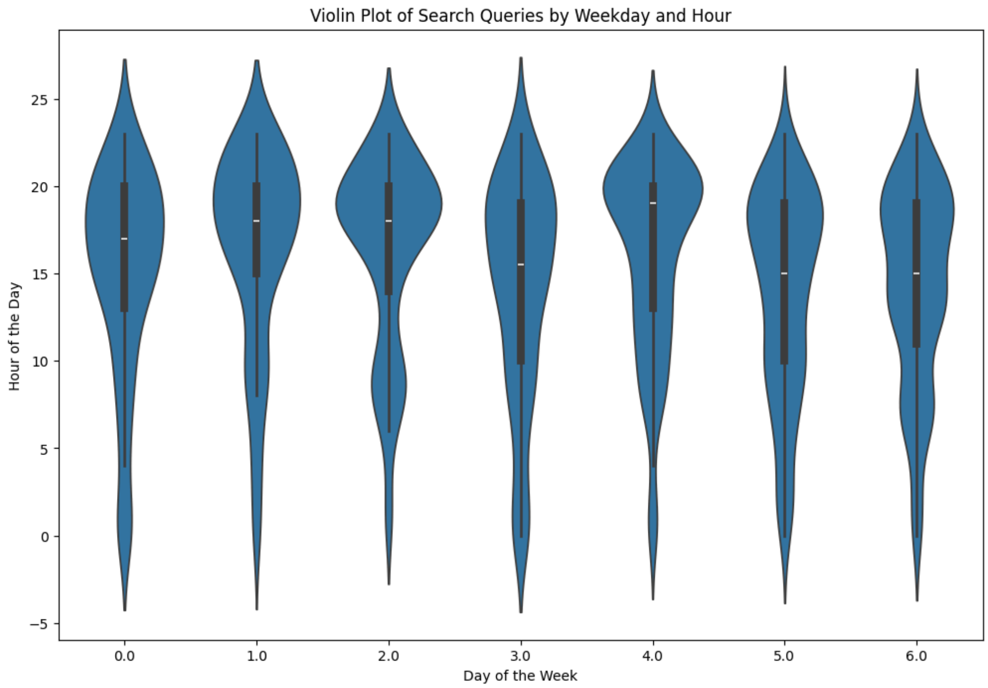
Although I wanted to make it cloud-shaped, it required additional libraries and photos that I could not use.

Later, I created a horizon plot to display the frequency of search queries over time using a dataset of search history. A horizon plot is a type of graph used to display time-series data, like search counts over days, in a compact and efficient way.

The code starts by grouping the data in search_history_df by the date part of the 'Utc Timestamp' column. It then counts the number of entries (search queries) for each date. This results in daily_counts, a series where each date is associated with the number of searches conducted on that day.



After plotting the results for search count, I also wanted to see average daily search queries in a weekly basis. To do this, I used a violin plot.



Correlation Analysis Between Ratings and Search Behavior

Here, I wanted to see the Correlation Analysis between Ratings and Search Behavior. The code is designed to explore whether there's a link between how often something is searched and how it's rated by users. It does this by combining two sets of data: one that tracks what users search for and another that records how they rate things.

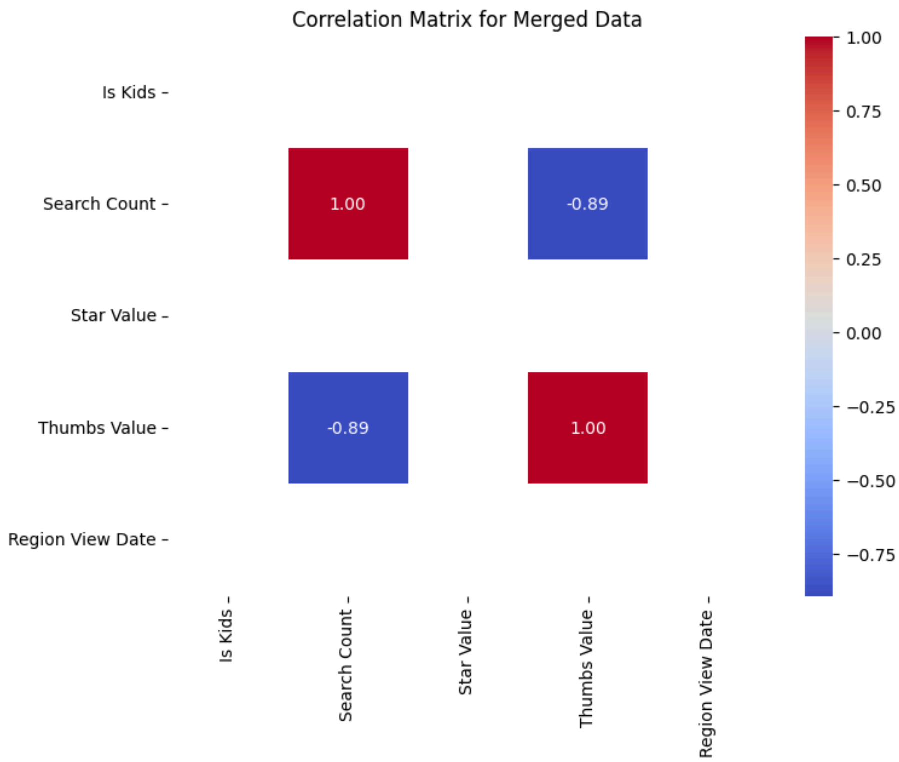
First, the code looks at the search data and counts how many times each item is searched for. It then adds this count to the search data, so now, for every item searched, there's a number showing how popular that search was.

Next, it does something similar with the rating data, which has information on how users rated different items.

Then, the code brings these two datasets together, matching items based on their names and who searched or rated them. This creates a new dataset that shows both the search count and ratings for each item.

Finally, the code calculates a correlation value. This value is like a score that tells us if there's a connection between how often an item is searched for and its ratings. If the score is high, it means items that are searched for a lot tend to have similar types of ratings (either mostly good or mostly bad). If the score is around zero, it means there's no clear pattern between how often items are searched for and how they are rated.

In my case, the correlation coefficient was -0.89 which is a strong negative correlation. This suggests that users who search for a show more often tend to give it a lower rating. Conversely, shows that are searched for less often tend to receive a higher rating.



Correlation Analysis Between Ratings and Watch Time

Now I also wanted to find the connection between how long people watch something and how they rate it. The code does this by looking at two kinds of data: one that shows how long items were watched and another that shows how these items were rated.

First, it loads the data about viewing activity. This data includes information about how long each item was watched, but in a format like hours:minutes:seconds. The code changes this into just seconds, making it easier to work with.

Then, it loads the ratings data, which is about how users rated different items.

Next, the code combines these two datasets. It matches each item in the viewing data with the same item in the ratings data, so long as they were watched and rated by the same user. This way, for each item, there's information on both how long it was watched and how it was rated.

After that, the code calculates a correlation value. This is a number that tells us if there's a link between the length of time an item was watched and its rating.

In my case, the correlation value was 0.38, which suggests a weak positive relationship between the two.

This means that users who watch a show for longer durations tend to give it a slightly higher rating (as indicated by the ‘Thumbs Value’). However, because the correlation is weak, there are likely other factors at play influencing both watch time and ratings.

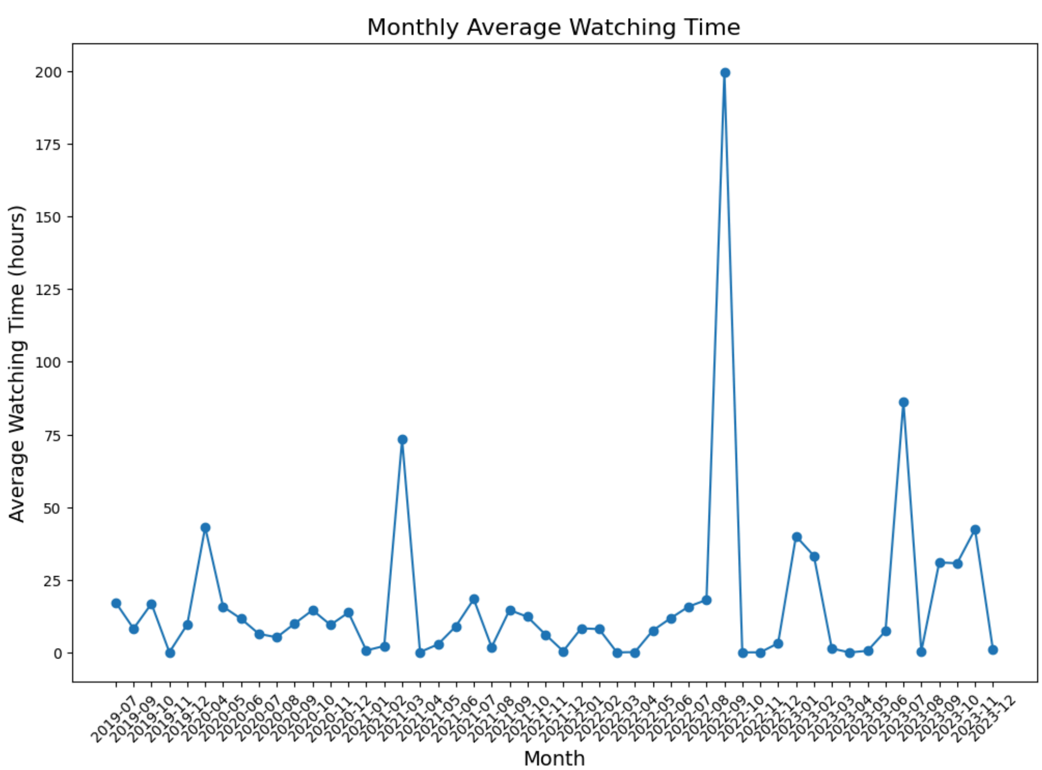
Monthly Average Watching Time vs Weather Conditions (Hypothesis)

After a long time of data exploration, I’m back again with a (simple) bar chart. This code is designed to figure out how much time I spend watching shows or movies each month. It takes a closer look at my viewing habits over time.

First, it organizes my viewing data by months and years. So instead of having a long list with exact dates and times, it just focuses on which month and year each viewing happened.

Next, it adds up all the time I spent watching shows or movies for each month. This tells us the total amount of time I was watching something in, say, January, February, and so on between 2019 and 2023. Below, the screenshot of the data and the line chart for the data is given:

YearMonth	
2019-07	17.197778
2019-09	8.198611
2019-10	16.788056
2019-11	0.054167
2019-12	9.641667
2020-04	43.085000
2020-05	15.740556
2020-06	11.601944
2020-07	6.394722
2020-08	5.206667
2020-09	9.955833
2020-10	14.570000
2020-11	9.466389
2020-12	13.751111
2021-01	0.684444
2021-02	2.223611
2021-03	73.271667
2021-04	0.138889
2021-05	2.778889
2021-06	8.991667
2021-07	18.336389
2021-08	1.843333
2021-09	14.591667
2021-10	12.273889
2021-11	6.188611
2021-12	0.440000
2022-01	8.299444
2022-02	8.039167
2022-03	0.009722
2022-04	0.121944
2022-05	7.538056
2022-06	11.780278
2022-07	15.774722
2022-08	18.082778
2022-09	199.682778
2022-10	0.025000
2022-11	0.036111
2022-12	3.162778
2023-01	40.019167
2023-02	33.239444
2023-03	1.354444
2023-04	0.003889
2023-05	0.566111
2023-06	7.381389
2023-07	86.336389
2023-08	0.355000
2023-09	30.957500
2023-10	30.697500
2023-11	42.383056
2023-12	0.974167

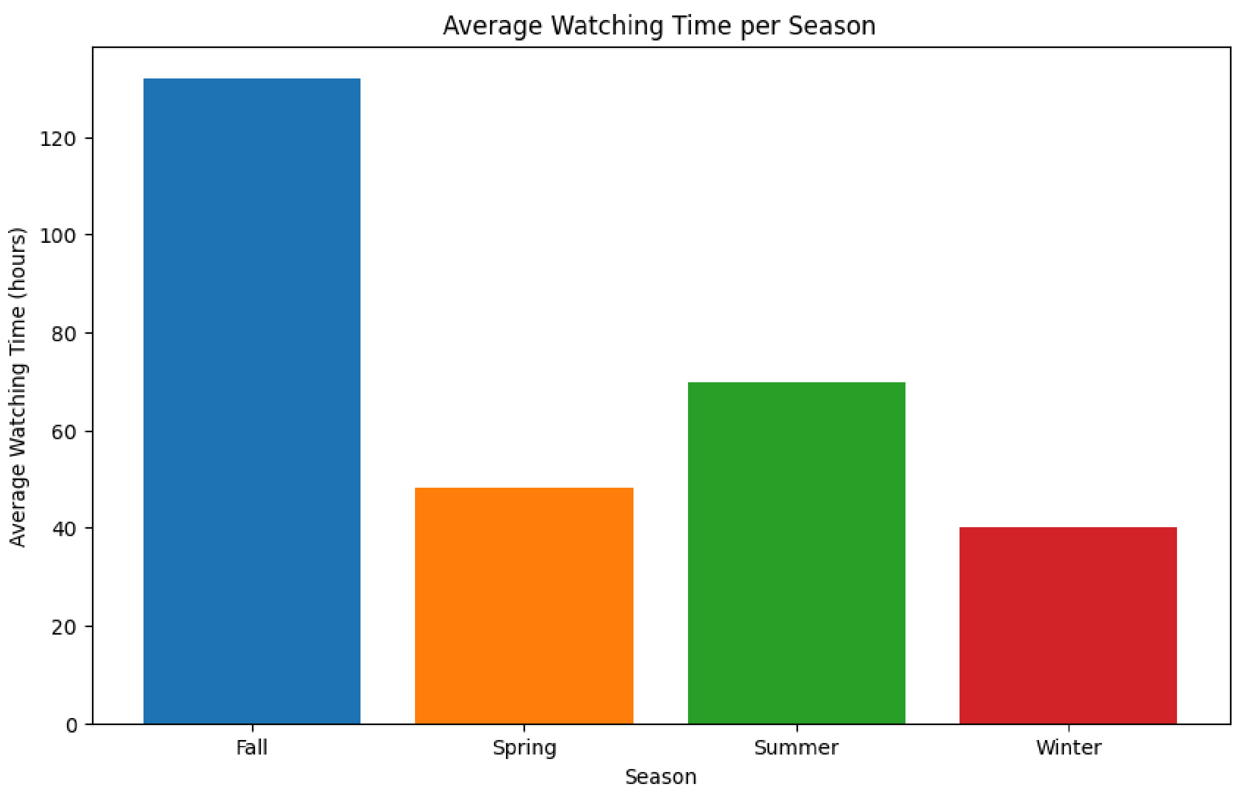


Later, I calculated my seasonal watching to see the affect of seasons (Fall- Spring - Summer and Winter) on my watching habits. The code starts by loading my viewing activity from the CSV file, 'ViewingActivity.csv'. It then transforms the 'Start Time' column into a datetime format, which is a way of handling dates and times that makes them easier to work with. Focusing specifically on my profile, the code filters the data to only include my viewing sessions. It also converts the 'Duration' field from a string to a time format, making it easier to calculate total viewing times.

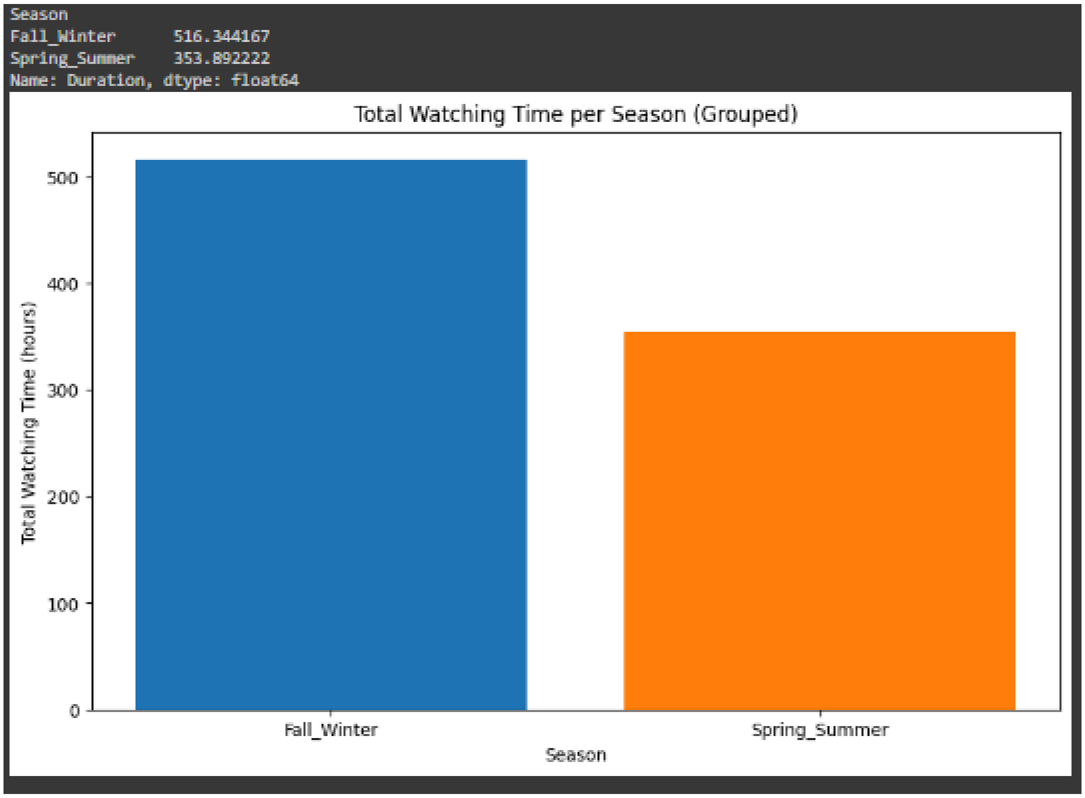
To understand my viewing patterns in different seasons, the code includes a function that categorizes months into seasons: December through February as Winter, March to May as Spring, June to August as Summer, and September to November as Fall. This function is applied to the viewing data to tag each viewing session with the appropriate season.

Next, the code groups this data by season and sums up the total viewing duration for each season. It then calculates the average number of hours spent watching content per month for each season. Below, there is a screenshot of the output values and the bar chart representing it.

```
Season
Fall      131.956389
Spring     48.203056
Summer     69.761019
Winter     40.158333
```



Lastly, I merged the seasons as Cold (Winter and Fall) and Hot (Spring and Summer) and visualized my watching habits to have a last look at my hypothesis. Below, there is a screen shot the values and the bar chart visualization.



Now, let's remember the hypotheses.

Null Hypothesis (H_0): I spend more time on Netflix during cold weather.

Alternative Hypothesis (H_1): There is no correlation between the weather and my watch-time habits.

The data backs my claim because I actually do spend more time on Netflix on colder weather. Although most of that time is in Fall and not on Winter, this does make sense. The school on the months of Fall (September, October and November) are relatively easier, as the exams are generally not on these periods. And also, because it is also colder than before, staying at home in this free time, watching Netflix seems normal.

Hence, I did not reject my Null Hypothesis.