
# *ANALYSING NETFLIX DATA*
# *TECH WEEK 2021*

# *Table Of Contents*

# *Analysing Netflix Data*

How much time have you spent watching movies on Netflix? Want to know some cool analytics from your Netflix data?

At the end of the workshop, you would be able to analyse your own Netflix data and find out which show you watched the most, how much time you spent watching each show and which days of the week you spend the most on Netflix (ps. there's a bonus section to find your favourite genres with the help of TMDb API!)

## PART 1 Preparations

### Installing Libraries

For this you can use any code editor/ IDE, but I will be using VScode. We have to first install the following library https://pandas.pydata.org/docs/getting_started/index.html

In Command Prompt:

```
pip install pandas
pip install matplotlib
```

### Why use Python instead of Excel?

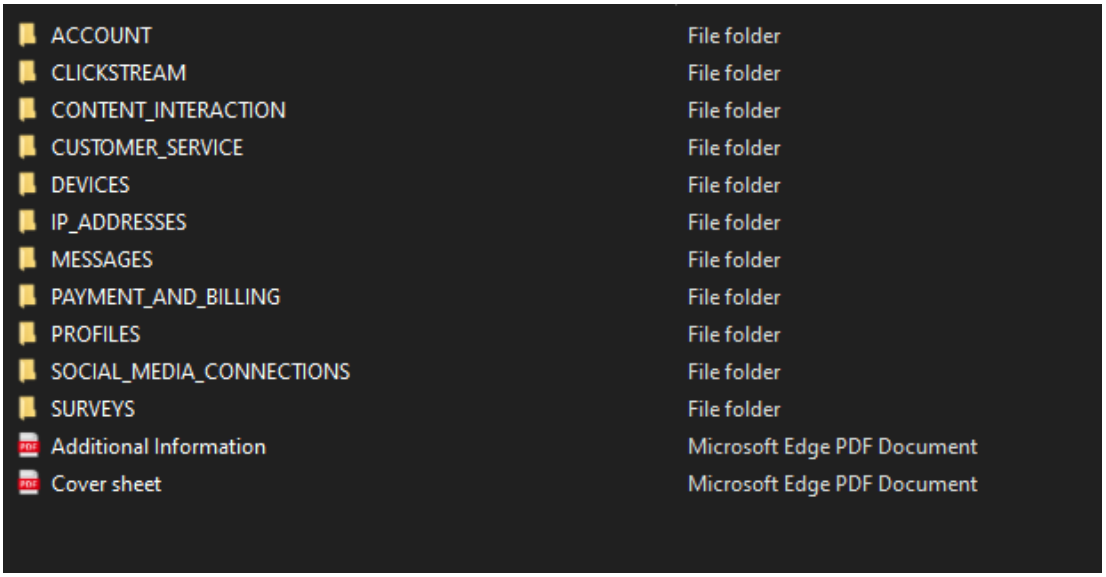
Depending on how much Netflix you watch and how long you've had the service, you might be able to use Excel or some other spreadsheet software to analyze your data but it would be much simpler with python and its libraries, which will allow you to more easily analyse the data.

If you plan on using our sample data, you can skip the preparations!

### Downloading your own Netflix Data

First download your own netflix data https://www.netflix.com/account/getmyinfo. They might take anywhere from a few days to 30 days to send the data (got mine in a few days!). Data will be sent to the email connected to your Netflix account (act fast as it might expire!). If you don't want to use your own Netflix data, we have sample data for you. Head over to our github and download the sample data. https://github.com/nyp-lit/techweek2021

Next, take a look at the zip file netflix sent. Do unzip it first.




Since we want to find out information about how much time we spend watching netflix, navigate to the `CONTENT_INTERACTION` folder and then to the `ViewingActivity.csv` file. Now create a new folder and label it anything you'd like. Paste the `ViewingActivity.csv` file into the folder you just created and add a python file where we'll add the code.

That's all the preparation you would need! Time to code!

## *PART 2 Understanding the code - Reading CSV and finding Longest Durations*

### 1. Import the libraries we downloaded initially

```
import pandas as pd
```

### 2. Reading CSV file

- We use pandas read_csv to create a dataframe from the data in ViewingActivity.csv.
- View documentation here if you want to know more: https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html

```
# read csv file
df = pd.read_csv('SampleViewingAcitivty.csv')
```

### 3. Printing a list of all columns

- `df` refers to the dataframe we defined earlier from reading the csv file. Refer to the output below.
- By using .columns we are getting the title of the columns and using list() means me are transforming the data to a list

```
# see the list of columns we have
print(list(df.columns))

# Output : ['Profile Name', 'Start Time', 'Duration', 'Attributes',
'Title', 'Supplemental Video Type', 'Device Type', 'Bookmark', 'Latest
Bookmark', 'Country']
```

If the data in ViewingActivity.csv has other names (meaning the account is shared), you will have to take only the rows of data that are yours. `print(df)` to see what's left in your dataframe.

The first `df[ ]` means you're looking into the dataframe and then you want to search specifically for the column profile name. `.str.contains` means that you will return only those rows that contain the string "Jolene". You can read more about `.str.contains` https://pandas.pydata.org/docs/reference/api/pandas.Series.str.contains.html. `df=` means you would re-assign it to dataframe, resulting in a new dataframe with only your data.

```
#get only my data
df = df[df["Profile Name"].str.contains("Jolene")]
print(df)
```

### 4. Find the longest session on Netflix

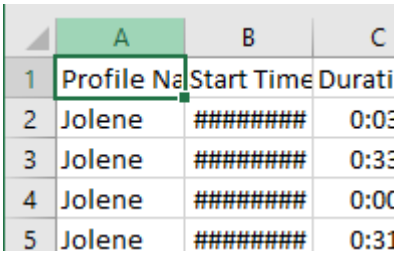
Let's try coding our first function to find out the longest session we spent on Netflix and what the movie was.
`df["Duration"].max()` returns the maximum duration that you spent on Netflix. With that duration you can look for the index of it in the dataframe. You will need the index to find the whole role of data in order to get the title of the movie

```
# longest session on netflix and what movie it was
def longestSession():
    print("Longest Session:", df["Duration"].max())
    longsess = df["Duration"].max()
    # df.index returns the number corresponding to the row (the axis
labels) and we only return the index when the duration is equal to the
max duration
    longsess_idx = df.index[df["Duration"] == longsess][0]
    # gets back entire row of data for that session
    longsess = df.iloc[[longsess_idx]]
```

```
    print("You watched", longsess["Title"].values[0], "for",
longsess["Duration"].values[0])

longestSession()
```

### 5. Removing unnecessary column data from your dataframe

- `df.drop()` drops columns, we can reassign it to df.
- Specifying the axis tells pandas to drop labels from the index of 1 instead of from 0 since the row count of an excel sheet starts from 1 instead of 0
- 

| | A | B | C |
|---|---|---|---|
| 1 | Profile Na | Start Time | Durati |
| 2 | Jolene | ######## | 0:03 |
| 3 | Jolene | ######## | 0:33 |
| 4 | Jolene | ######## | 0:00 |
| 5 | Jolene | ######## | 0:31 |

```
# we want profile name duration and title so we'll drop the rest
df = df.drop(["Profile Name", 'Attributes', 'Supplemental Video Type',
              'Device Type', 'Bookmark', 'Latest Bookmark', 'Country'],
axis=1)
print(df.head())
```

### 6. Converting time data types

We have to parse the data and change the formats into datetime format so that we would be able to add time. Read more about to_datetime here: https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html

```
# convert to the right format so that we can add it tgt
df['Duration'] = pd.to_timedelta(df['Duration'])
df['Start Time'] = pd.to_datetime(df['Start Time'])
```

### 7. Finding Total Time spent on Netflix

- Get all the times in the column for Duration with `df["Duration"]`
- We will shape the data and check on the shape of it using `.shape`
- Since Netflix also records the time less than 1 minute that could be autoplay, we will remove the data for those with less than a minute, making our data more accurate
- Print the shape again and you would be able to see the difference

```
def totalNetflix():
    totalDuration = df["Duration"]
    # print(totalDuration.shape)
    totalDuration = totalDuration[(totalDuration > '0 days 00:01:00')]
    # print(totalDuration.shape)
```

```
    print("total duration on netflix",totalDuration.sum())

totalNetflix()
```

# *PART 3 Understanding the code - Plotting Bar Graphs*

### 1. Import matplotlib library

- which we would use to display data in a bar graph

```
from matplotlib import pyplot as plt
```

### 2. What are the days you spend most watching Netflix?

- `df["weekday"]` is used to add a new column to the dataframe
- `.dt.weekday` returns day of the week (Example: monday = 0, tuesday = 1) See Documentation: https://pandas.pydata.org/docs/reference/api/pandas.Series.dt.weekday.html#pandas.Series.dt.weekday
- `.dt.hour` returns hour (Example 01:00:00)
- `pd.Categorical` used to set categories in bar graph

```
def whenWatch():
    pd.options.mode.chained_assignment = None  # default='warn' use
this to remove warnings

    df['weekday'] = df['Start Time'].dt.weekday
    df['hour'] = df['Start Time'].dt.hour

    # print(df['weekday'])

    # set categories -> in this case its the days
    df['weekday'] = pd.Categorical(df['weekday'],
categories=[0,1,2,3,4,5,6], ordered=True)
    # count how many rows
    watchByDay = df['weekday'].value_counts()
    # make monday first, ordered by date to fit in categories
    watchByDay = watchByDay.sort_index()
    watchByDay.plot(kind='bar', figsize=(20,10),title='Netflix Shows
Watched By Day')
    plt.show()

whenWatch()
```

# *Bonus Section: APIs*

### 1. Import requests

To use APIs we would need to send requests, import requests.

```
import requests
```

### 2. What is an API?

API refers to Application Programming Interface, it allows two applications to communicate with each other. For example, each time you check the weather with your phone, you're using a weather API. In this case, our API returns data of the movie according to what we query for.

### 3. About IMDb API and how we are using it

- Using https://rapidapi.com/apidojo/api/imdb8 IMDB api to get genre of the movies
- Can be used to find your most watched genre
- This function would cause the next function to take quite a while to run due to the API, we would have to call the API for every movie
- The first API call in the function takes the title of the movie and then finds the movie details. However the movie details does not include the genres, but it does return important info like title id.
- So, we will need to call the API again to get the overview details.
- We will pass the title id we got from the last API call into the second API call to get the genres for the movie.
- PS: the movie from the first API call might not always be accurate as we are taking into account only the first result of the query

```
#finding genres with movie title
def findGenreAPI(title):
    url = "https://imdb8.p.rapidapi.com/title/find"

    querystring = {"q": title}

    headers = {
        'x-rapidapi-host': "imdb8.p.rapidapi.com",
        'x-rapidapi-key':
"2614095f10msh111b259710e3f7dp1279d3jsn21efd84301bf"
        }

    response = requests.request("GET", url, headers=headers,
params=querystring)

    movieCode = response.json()
    # might not be accurate cuz we only care abt the first response
```

```
    try:
        movieCode = movieCode['results'][0]['id']
        movieCode = movieCode[7:]
        # print(movieCode)

        url = "https://imdb8.p.rapidapi.com/title/get-overview-details"

        querystring = {"const":movieCode,"currentCountry":"US"}

        headers = {
            'x-rapidapi-host': "imdb8.p.rapidapi.com",
            'x-rapidapi-key':
"2614095f10msh111b259710e3f7dp1279d3jsn21efd84301bf"
            }

        response = requests.request("GET", url, headers=headers,
params=querystring)

        response = response.json()
        movieGenres = response['genres']
        return movieGenres
        # print(movieGenres)
    except:
        return 'no data found'

# findGenreAPI('Trailer')
# findGenreAPI('Strangers from hell')
# uncomment the 2 lines above to test ur code, the first one will fail
while the second will not
```

### 4. Find the genres using API function above

Find the genres of the movies in your dataframe using the function above.

- Running the function below will return you a list of all the genres of the movies watched
- `list(df['Title'])` returns a list of all titles
- Unfortunately the list of titles would have duplicates/ information we don't need to know like the season and also some weird thing at the end of the movie title
- These extra information would make the findGenreAPI() fail as it searches by movie name (it wouldn't return ['results'] if no results)
- So we needa clean the data by removing everything after ":" and "_"
- Now we can return a list with no duplicates and weird information :D

```
# find genres for the movies we watched
```

```
def findGenres():
    movies = list(df['Title'])
    genres = []
    no_duplicate_movielist = []

    for movie in movies:
        # print(movie.find(':'))
        if movie.find(':') != -1:
            movie = movie[:movie.find(':')]

        if movie.find('_') != -1:
            movie = movie[:movie.find('_')]

        no_duplicate_movielist.append(movie)

    no_duplicate_movielist =
list(dict.fromkeys(no_duplicate_movielist))

    print(no_duplicate_movielist)

    for movie in no_duplicate_movielist:
        genres.append(findGenre(movie))

    print(genres)

findGenre()
#remember to comment this out if you alr got your genres
```

### 5. Find out your top 3 genres!

Find out the most popular genres in the movies you've watched (basically your favourite genres!)

- With the list of genres you got from the last function, instead of having to run the function again you can copy and paste the output of the last function and store it in genres (allows you to keep running the code without having to use the last function which takes quite a while cuz of API calls).

```
# fill in list according to the output from the function above, js copy
paste
genres = [['Drama', 'Family', 'Sci-Fi'], ['Adventure', 'Drama',
'Horror', 'Thriller'], ['Adventure', 'Drama', 'Horror', 'Thriller'],
['Crime', 'Drama', 'Mystery', 'Thriller'], ['Comedy', 'Romance'],
['Action', 'Drama', 'Mystery', 'Thriller'], 'no data found', ['Drama',
'Mystery'], ['Crime', 'Horror', 'Mystery'], ['Drama', 'Horror',
```

```
'Mystery', 'Thriller'], ['Action', 'Drama', 'Horror', 'Sci-Fi'],
['Comedy', 'Drama', 'Music', 'Romance'], ['Mystery', 'Thriller'],
['Action', 'Adventure', 'Thriller'], ['Action', 'Crime', 'Drama',
'Sci-Fi', 'Thriller'], ['Crime', 'Drama', 'Thriller'], ['Action',
'Crime', 'Thriller'], ['Horror', 'Mystery'], 'no data found',
['Action', 'Adventure', 'Drama', 'Fantasy', 'Sci-Fi'], ['Horror',
'Mystery', 'Thriller'], ['Action', 'Drama', 'Horror', 'Thriller'], 'no
data found', ['Drama', 'Romance'], ['Action', 'Horror', 'Thriller'],
['Comedy', 'Horror'], ['Drama', 'Romance'], ['Drama', 'History',
'Romance'], ['Crime', 'Mystery', 'News'], ['Comedy'], ['Comedy',
'Drama'], ['Comedy', 'Drama', 'Romance'], ['Action', 'Adventure',
'Mystery', 'Sci-Fi'], ['Drama', 'Horror', 'Mystery'], ['Drama',
'Thriller'], ['Adventure', 'Drama', 'Fantasy', 'Romance'], ['Drama',
'Fantasy', 'Romance'], ['Mystery', 'Thriller'], ['Biography', 'Crime',
'Drama', 'History', 'Mystery', 'Thriller'], ['Horror', 'Thriller'],
['Game-Show', 'Reality-TV'], ['Crime', 'Drama', 'Thriller'],
['Comedy'], ['Comedy', 'Romance'], ['Action', 'Comedy', 'Romance'],
['Crime', 'Drama', 'Mystery', 'Thriller'], ['Action', 'Comedy',
'Crime', 'Thriller'], ['Fantasy', 'Horror', 'Thriller'], ['Horror',
'Mystery', 'Thriller'], ['Drama', 'Horror', 'Thriller'], ['Horror',
'Mystery', 'Thriller'], ['Drama', 'Horror', 'Mystery', 'Thriller'],
['Horror', 'Mystery', 'Thriller'], ['Horror', 'Thriller'], ['Drama',
'Horror', 'Thriller'], ['Drama'], ['Action', 'Comedy', 'Fantasy'],
['Horror', 'Mystery', 'Thriller'], ['Crime', 'Drama', 'Thriller'],
['Comedy', 'Family'], ['Mystery', 'Thriller'], ['Action', 'Adventure',
'Crime', 'Drama', 'Thriller'], ['Drama', 'Horror', 'Mystery',
'Thriller'], ['Horror', 'Thriller'], ['Crime', 'Horror', 'Thriller'],
['Crime', 'Drama', 'Thriller'], ['Horror', 'Sci-Fi', 'Thriller'],
['Action', 'Crime', 'Drama', 'Mystery', 'Thriller']]
```

The list returned has many levels (like nested list) so we will flatten it in order to count the number of times repeated for each genre. So let's create a flatten function.

```
164
165  listoflists = [1, 2, 3, 4, 2, [2,3]]
166
167  for i in listoflists:
168      if isinstance(i,list):
169          print('yes')
170      else:
171          print('no')
172
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
no
no
no
no
no
yes
```

Understanding what `isinstance(i,list)` does, running the code above returns “yes” only when there's a nested list.

```
def flatten(listoflists):
    rt = []
    for i in listoflists:
        if isinstance(i,list):
           # if i is an instance of the list (checking if its a value
or another list)
            # if its another list den flatten
            rt.extend(flatten(i))
        else:
            # else append
            rt.append(i)
    return rt
```

Now for the final function to pull the last 2 functions together.

- With the genres list we got, we can flatten it using the function we created above
- We can now use the Counter function and get the top 3 most common words
- Learn more about what Counter can do https://www.guru99.com/python-counter-collections-example.html

```
from collections import Counter

def favouriteGenres(genres):
    genreList = flatten(genres)

    c = Counter(genreList)
    print(c.most_common(3))

favouriteGenres(genres)
```