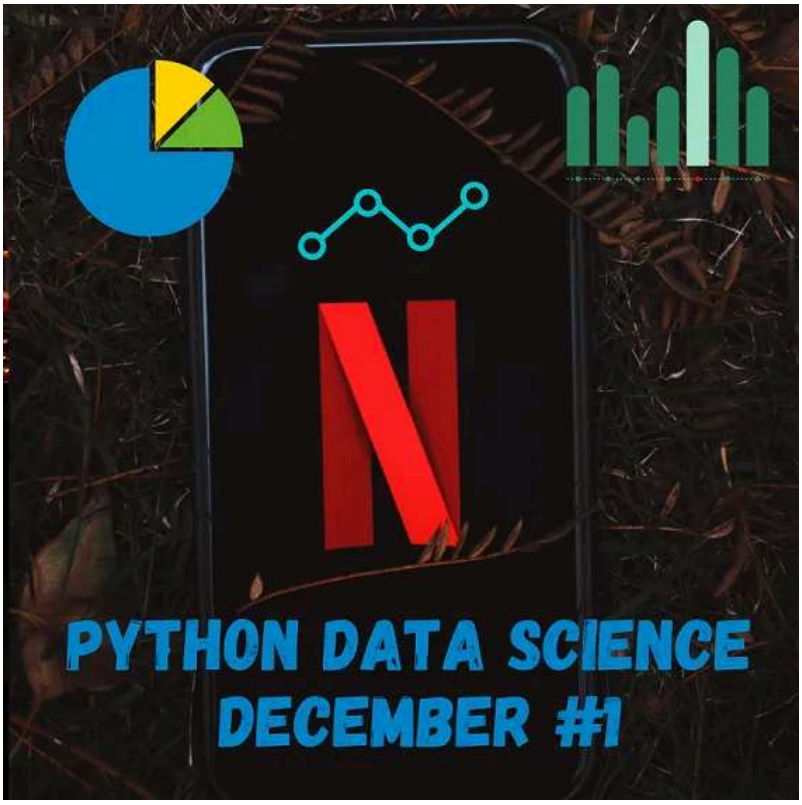
Member-only story

# Python — Analyze Your Own Netflix Data

## Build Your Own Dataset Using Netflix Data & Python Pandas

Techletters · Follow
Published in Python Point · 11 min read · Dec 1, 2022






Analyze Netflix Data using Python. Image by the author.

Are you looking for datasets to start a new data science project? While it's not always easy to find relevant data, why not start with your own? Welcome to Python Data Science December #1.

Netflix lets you download your complete watching history and you can build cool data science projects on top of it. I'll guide you through

- getting the raw data from Netflix
- cleaning & transforming the data
- visualizing the data.

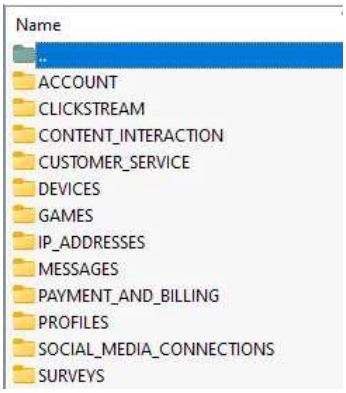
We will make use of Python Pandas and Matplotlib. At the end of this story, you will have finished a simple but full-blown data science project that you can add to your portfolio.

If you do not use Netflix, I will share my sampled & anonymized watching history with you. You can find it at the end of the story, in the chapter *Summary & Resources.*

## 📓 Getting The Raw Data

Netflix allows you to request your own data for download.

- Navigate to Netflix's *get my info* page & request your data
- After sending the request, you get an email that you need to confirm
- After a while, you get another mail and the download is ready (for me it took not more than 1 day)
- You will get a .zip file with the following folder structure




The folder structure of your Netflix data (Image by the author).

- Unzip it and navigate to ***CONTENT_INTERACTION*** to open the file ***ViewingActivity.csv*** which contains the full list of your viewing history.

Alright, time to get our hands dirty.

## 🔍 Examine the data

I am using Python Jupyter notebook to examine the data, but you can also just take a regular Python script. To get a first impression of the data, let's install & import Pandas and read the file into a Pandas DataFrame.

```
import pandas as pd
df = pd.read_csv('ViewingActivity.csv')
```

First, let us understand how many rows & columns we have using...

```
df.shape      #returns the number of rows and columns
```

```
> (9273, 10)
```

... and have a closer look at the first rows of the dataset.

```
df.head()     #returns the first 5 rows
```

| | Profile Name | Start Time | Duration | Attributes | Title | Supplemental Video Type | Device Type | Bookmark | Latest Bookmark | Country |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Mom & Dad | 2022-09-27 21:16:43 | 00:00:13 | Autoplayed: user action: None; | Die Strafe Gottes (Clip): Die Strafe Gottes | HOOK | Samsung CE 2020 Nike-L UHD TV Smart TV | 00:00:13 | 00:00:13 | DE (Germany) |
| 1 | Mom & Dad | 2022-09-27 19:00:02 | 02:09:10 | Autoplayed: user action: User_Interaction; | A Beautiful Mind – Genie und Wahnsinn | NaN | Samsung CE 2020 Nike-L UHD TV Smart TV | 02:09:11 | 02:09:11 | DE (Germany) |
| 2 | Mom & Dad | 2022-09-27 18:58:40 | 00:00:06 | Autoplayed: user action: None; | A Jazzman's Blues (Clip): A Jazzman's Blues | HOOK | Samsung CE 2020 Nike-L UHD TV Smart TV | 00:00:06 | 00:00:06 | DE (Germany) |
| 3 | Mom & Dad | 2022-09-26 18:35:22 | 01:41:17 | NaN | Lou | NaN | Samsung CE 2020 Nike-L UHD TV Smart TV | 01:41:19 | 01:41:19 | DE (Germany) |
| 4 | Mom & Dad | 2022-09-26 18:35:07 | 00:00:13 | Autoplayed: user action: None; | Lou (Clip 5): Lou | HOOK | Samsung CE 2020 Nike-L UHD TV Smart TV | 00:00:13 | 00:00:13 | DE (Germany) |

Additionally, let's get some random sample rows for a further first understanding of the data.

```
df.sample(n=10)     #returns a random sample of 10 rows
```

| | Profile Name | Start Time | Duration | Attributes | Title | Supplemental Video Type | Device Type | Bookmark | Latest Bookmark | Country |
|---|---|---|---|---|---|---|---|---|---|---|
| 3524 | Code & Dogs | 2019-09-26 18:06:27 | 00:35:30 | NaN | Marvel's Jessica Jones: Staffel 3: Ich wünscht... | NaN | Internet Explorer (Cadmium) | 00:51:23 | 00:51:23 | DE (Germany) |
| 5484 | Kids | 2021-09-01 09:45:49 | 00:11:57 | NaN | Ninjago: Aufstieg der Schlangen: Familienbande... | NaN | Sony CE Sony Android TV 2020 M5 Smart TV | 00:21:48 | Not latest view | DE (Germany) |
| 3414 | Code & Dogs | 2020-02-25 15:25:18 | 00:02:26 | NaN | The Big Bang Theory: Staffel 5: Ein guter Kerl... | NaN | PC | 00:02:25 | 00:02:25 | DE (Germany) |
| 8999 | Brother & Wife | 2019-07-26 20:18:38 | 00:00:04 | Autoplayed: user action: None; | Haus des Geldes: Teil 2: Folge 2 (Folge 2) | NaN | Sony Sony Android TV 2015 Smart TV | 00:00:10 | Not latest view | DE (Germany) |
| 8097 | Brother & Wife | 2021-05-12 19:42:34 | 00:06:14 | Autoplayed: user action: User_Interaction; | Narcos: Staffel 2: Unser Mann in Madrid (Folge 3) | NaN | Samsung CE 2019 Muse-L UHD TV Smart TV | 00:41:10 | Not latest view | DE (Germany) |
| 4883 | Kids | 2022-06-05 17:40:50 | 00:20:58 | NaN | Ninjago: Aufstieg der Schlangen: Familienbande... | NaN | Sony CE Sony Android TV 2020 M5 Smart TV | 00:21:43 | Not latest view | DE (Germany) |
| 4949 | Kids | 2022-03-05 19:58:24 | 00:01:01 | Autoplayed: user action: User_Interaction; | Pokémon – Der Film: Du bist dran! | NaN | FireTV 4K Stick 2018 | 00:01:01 | Not latest view | DE (Germany) |
| 2977 | Code & Dogs | 2020-06-06 18:58:57 | 00:55:42 | NaN | Ozark: Staffel 1: Kaffee, schwarz (Folge 9) | NaN | Chrome PC (Cadmium) | 00:55:45 | 00:55:45 | DE (Germany) |
| 4747 | Kids | 2022-07-17 14:32:21 | 00:00:32 | Autoplayed: user action: None; | Ninjago: Aufstieg der Schlangen: Der Nindroid ... | NaN | Sony CE Sony Android TV 2020 M5 Smart TV | 00:04:42 | Not latest view | DE (Germany) |
| 4048 | Code & Dogs | 2018-06-24 22:54:14 | 00:13:09 | Autoplayed: user action: None; | The Big Bang Theory: Staffel 10: Die Charlie-B... | NaN | Internet Explorer (Cadmium) | 00:13:09 | Not latest view | DE (Germany) |

Alright, what do we know at this point already, after running just a couple of easy commands?

- we have 9273 rows of data structured into 10 columns
- *'Profile Name'* — it seems multiple persons/profiles are sharing the Netflix account
- *'Device Type'* — it seems the same profiles using Netflix with different devices & browsers
- **'Title'** — For series, the title contains the name of the series, but also the season and episode
- *'Start Time' & 'Duration'* — We know when & how long someone watched something

Let's take a closer look at the 'Profile Name'.

```
df["Profile Name"].unique()

> array(['Mom & Dad',
>        'Friends',
>        'Code & Dogs',
>        'Kids',
>        'Brother & Wife'],dtype=object)
```

It seems we have four Profiles that are using the Netflix account:

- Code & Dogs — me and my wife
- Mom & Dad — my wife's mom & dad
- Brother & Wife — my wife's brother & wife
- Friends — friends of us

- Kids — my wife's brother does have a child, so probably this Profile is most used by them

Let's do the same for the 'Device Type' column.

```
df["Device Type"].unique()
```

```
> array([
'Samsung CE 2020 Nike-L UHD TV Smart TV',
'FireTV 4K Stick 2018',
'DefaultWidevineAndroidTablets',
'Apple iPad Pro 12.9 in 5th Gen (Wi-Fi/Cell) iPad',
'Sony CE Sony Android TV 2020 M5 Smart TV',
'Chrome PC (Cadmium)',
...
'Sony 2012 Blu-ray Players']
```

It seems a whole bunch of different devices & browsers are used to watch Netflix.

After having a first impression of the data, the most important topic is: **What questions do we want to answer?**

- Which Profile watched the most (time)?
- Which Profile has the most watching activities/interactions?
- What is the average watching time (per Profile)?
- What devices are used by which Profile? And which device is used the most?

You can imagine there are much more questions that could be explored. I am already curious about your ideas & research questions for your own or my Netflix data. In the chapter '**Further ideas**' down below I collected more example questions & community ideas for a shared dataset.

## ❎ Transform the data

Before we can work with the data in more detail, we need to understand the underlying data types.

```
df.dtypes
```

```
>Profile Name               object
>Start Time                 object
>Duration                   object
>Attributes                 object
>Title                      object
>Supplemental Video Type    object
>Device Type                object
>Bookmark                   object
>Latest Bookmark            object
>Country                    object
```

We can see that all columns store the data in the *object* data type. Especially if we want to work with times, dates & durations (e.g. to understand overall viewing times) we need to transform it.

```
df['Start Time'] = pd.to_datetime(df['Start Time'], utc=True)
df['Duration'] = pd.to_timedelta(df['Duration'])
df.dtypes
```

```
>Profile Name                            object
>Start Time                 datetime64[ns, UTC]
>Duration                       timedelta64[ns]
>Attributes                              object
>Title                                   object
>Supplemental Video Type                 object
>Device Type                             object
>Bookmark                                object
>Latest Bookmark                         object
>Country                                 object
```

Now both columns ‘Start Time’ and ‘Duration’ have an accurate format that we can work with. Let’s take another sample of the data to see how the data representation has changed.

```
df.sample(n=10)     #returns a random sample of 10 rows
```

| | Profile Name | Start Time | Duration | Attributes | Title | Supplemental Video Type | Device Type | Bookmark | Latest Bookmark | Country |
|---|---|---|---|---|---|---|---|---|---|---|
| 6172 | Kids | 2021-03-08 16:35:27+00:00 | 0 days 00:14:31 | NaN | Miraculous – Geschichten von Ladybug und Cat N... | NaN | Sony Sony Android TV 2015 Smart TV | 00:21:13 | Not latest view | DE (Germany) |
| 331 | Mom & Dad | 2022-03-24 20:30:46+00:00 | 0 days 00:00:05 | Autoplayed: user action: None; | Operation Schwarze Krabbe (Clip): Operation Sc... | HOOK | Samsung CE 2020 Nike-L UHD TV Smart TV | 00:00:05 | Not latest view | DE (Germany) |
| 7005 | Kids | 2020-07-30 12:34:19+00:00 | 0 days 00:10:55 | Autoplayed: user action: Unspecified; | Dinotrux: Staffel 1: Die Grube (Folge 5) | NaN | Sony Sony Android TV 2015 Smart TV | 00:23:25 | Not latest view | DE (Germany) |
| 6289 | Kids | 2021-02-07 16:59:08+00:00 | 0 days 00:17:11 | NaN | Dragons – Die jungen Drachenretter: Staffel 1:... | NaN | Sony Sony Android TV 2015 Smart TV | 00:22:49 | Not latest view | DE (Germany) |
| 3709 | Code & Dogs | 2019-03-04 20:30:02+00:00 | 0 days 00:35:04 | Autoplayed: user action: Unspecified; | Aufräumen mit Marie Kondo: Staffel 1: Ein leer... | NaN | Internet Explorer (Cadmium) | 00:35:11 | 00:35:11 | DE (Germany) |

## 📊 Visualize the data

Now it's time to answer our questions. The best way to find answers is to visualize the data.

- **Which Profile has the most watching activities/interactions?**

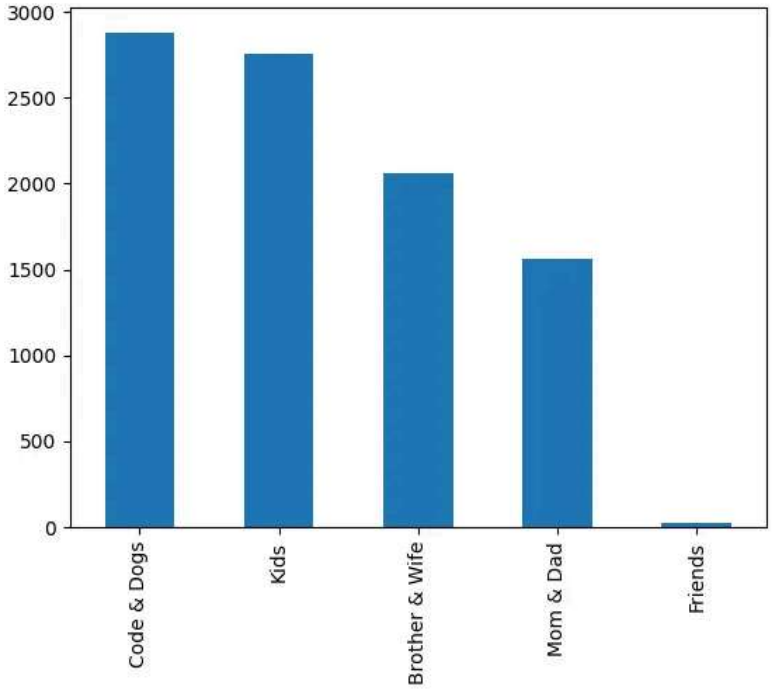
We can again make use of the *DataFrame.value_counts()* method to count the number of row occurrences for each Profile.

```
df['Profile Name'].value_counts()
>Code & Dogs       2881
>Kids              2754
>Brother & Wife    2057
>Mom & Dad         1559
>Friends             22
>Name: Profile Name, dtype: int64
```

It seems me & my wife (*'Code & Dogs'*) are having the most viewing interactions, followed by the *'Kids'* profile and *'Brother & Wife'*. We can not make any statement about how representative this order is. It might be that my wife's brother watched some stuff forgetting to switch back from the Kids Profile.

My wife's Mom & Dad missed the podium and made the fourth place and our friends only had 22 viewing interactions. Using *matplotlib* it is really easy to visualize this.

```
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
df['Profile Name'].value_counts().plot(kind='bar')
plt.show()
```




- **Which Profile watched the most (time)?**

Let's first check the overall viewing duration for all Profiles.

```
df['Duration'].sum()
```

```
> Timedelta('124 days 00:57:25')
```

To set the overall viewing duration of >124 days into some context. The data was logged for 5 1/2 years from May 2017 to September 2022.

```
df.sort_values('Start Time')
```

```
> 2017-05-25 20:13:40+00:00
...
> 2022-09-28 14:06:55+00:00
```

We can easily summarize the viewing duration for each profile.

```
df.loc[df['Profile Name']=='Code & Dogs','Duration'].sum()
> Timedelta('47 days 21:26:40')

df.loc[df['Profile Name']=='Mom & Dad','Duration'].sum()
> Timedelta('16 days 02:00:31'

df.loc[df['Profile Name']=='Kids','Duration'].sum()
> Timedelta('23 days 03:40:55')

df.loc[df['Profile Name']=='Brother & Wife','Duration'].sum()
> Timedelta('34 days 09:36:07')

df.loc[df['Profile Name']=='Friends','Duration'].sum()
> Timedelta('0 days 11:03:42')
```

We can see that we get the same order again. It seems logical that the Profiles with more watching activities also have a higher overall watching duration. But it could be that some Profiles only watch short series, more often and others only watch long movies. We will analyze this with the next question (‘Average viewing times per Profile’).

The visualization is not that easy, because a Timedelta is nothing that can be plotted using *matplotlib*. We need to run another transformation, using *NumPy's astype* method to transform it into seconds.

```
df.loc[df['Profile Name']=='Code & Dogs','Duration'].astype('timedelta64[s]').su
```

```
> 4138000.0
```

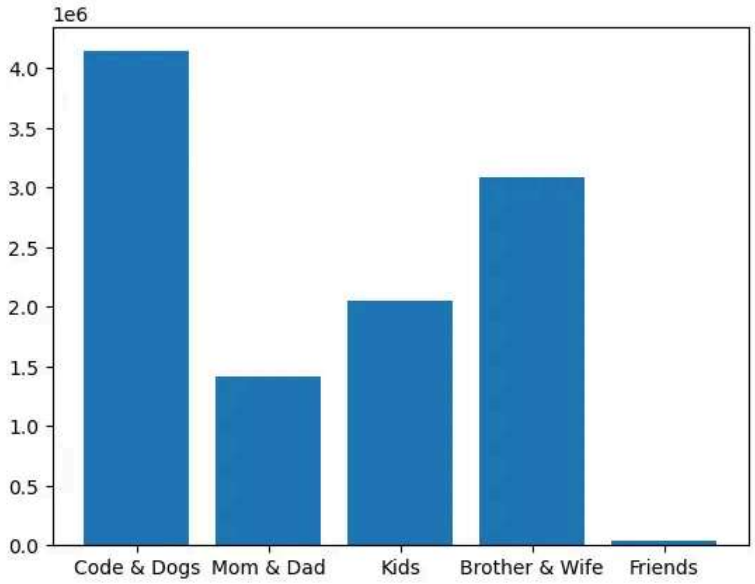
Let’s create a helper dict *viewTime* to store the overall view time per Profile in seconds.

```
viewTime = {}
viewTime.update({"Code & Dogs": df.loc[df['Profile Name']=='Code & Dogs','Durati
viewTime.update({"Mom & Dad": df.loc[df['Profile Name']=='Mom & Dad','Duration']
viewTime.update({"Kids": df.loc[df['Profile Name']=='Kids','Duration'].astype('t
viewTime.update({"Brother & Wife": df.loc[df['Profile Name']=='Brother & Wife','
viewTime.update({"Friends": df.loc[df['Profile Name']=='Friends','Duration'].ast
viewTime
```

```
>{'Code & Dogs': 4138000.0,
> 'Mom & Dad': 1411573.0,
> 'Kids': 2050125.0,
> 'Brother & Wife': 3077525.0,
> 'Friends': 39822.0}
```

Finally, we can plot it and have a look at the chart.

```
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
plt.bar(*zip(*viewTime.items()))
plt.show()
```




- **What is the average watching time (per Profile)?**

To analyze the average watching time per Profile, we must divide the overall watching duration by the number of views/interactions. Remember the value_counts() operation to get the overall view interactions per Profile.

```
df['Profile Name'].value_counts()
```

```
>Code & Dogs       2881
>Kids              2754
>Brother & Wife    2057
>Mom & Dad         1559
>Friends             22
```

Now we can divide the overall watching duration per Profile by these values.

```
df.loc[df['Profile Name']=='Code & Dogs','Duration'].sum()/2881
> Timedelta('0 days 00:23:56.306837903')
```

```
df.loc[df['Profile Name']=='Kids','Duration'].sum()/2754
> Timedelta('0 days 00:12:24.417211328')

df.loc[df['Profile Name']=='Brother & Wife','Duration'].sum()/2057
> Timedelta('0 days 00:24:56.122994652')

df.loc[df['Profile Name']=='Mom & Dad','Duration'].sum()/1559
> Timedelta('0 days 00:15:05.434894162')

df.loc[df['Profile Name']=='Friends','Duration'].sum()/22
>  Timedelta('0 days 00:30:10.090909090')
```

Somehow these values seem strange. The average viewing time per interaction is < 30 minutes for all Profiles.

- Point 1: The viewing time is only displayed per title. And the title for the series shows the name, season & episode. This means: if you are doing a series viewing marathon, it is split into several viewing interactions with short durations in the dataset.
- Point 2: After analyzing the 'Duration' column it became obvious that there are some unrealistic short durations logged with < 15 seconds. It seems that teasers, trailers & hooks (if you hover with the mouse over a title on Netflix, there sometimes is a short hook) are also logged in the data. Let's filter these out.

Let's have a look at the values of column '*Supplemental Video Type*'.

```
df['Supplemental Video Type'].value_counts()
```

```
> HOOK               886
> TRAILER            367
> TEASER_TRAILER      74
> RECAP               10
> PROMOTIONAL          5
> BUMPER               1
```

It seems we have overall 1343 short interactions, that we want to filter out. The remaining 7930 of the overall 9273 rows are empty (NaN) and represent actual watched series & movies. So let's only keep the rows with NaN in Supplement Video Type.

```
df.loc[df['Supplemental Video Type'].isnull()]
```

And check the numbers again. First, we take a look at the value counts again...

```
df['Profile Name'].value_counts()
```

```
> Kids              2693
> Code & Dogs       2614
> Brother & Wife    1774
> Mom & Dad          831
> Friends             18
```

... and divide the remaining viewing sum by them.

We can see that the average viewing times are slightly increased for most Profiles (e.g. for 'Code & Dogs' it got increased from 23:56 to 26:20 minutes). The most dramatic increase happened for 'Mom & Dad' and nearly doubled from 15:05 to 28:07 minutes. Perhaps they simply couldn't decide what to watch and instead browsed through trailers & hooks a lot.

```
df.loc[df['Profile Name']=='Code & Dogs','Duration'].sum()/2614
> Timedelta('0 days 00:26:20.112471308')

df.loc[df['Profile Name']=='Kids','Duration'].sum()/2693
> Timedelta('0 days 00:12:40.924248050')

df.loc[df['Profile Name']=='Brother & Wife','Duration'].sum()/1774
> Timedelta('0 days 00:28:51.166290868')

df.loc[df['Profile Name']=='Mom & Dad','Duration'].sum()/831
> Timedelta('0 days 00:28:07.525872442')

df.loc[df['Profile Name']=='Friends','Duration'].sum()/18
> Timedelta('0 days 00:36:52.333333333')
```

- **What devices are used by which Profile? And which device is used the most?**

Let's take a look at the different devices used by the Profiles to watch Netflix... and there are really a lot. Some were used really often while one device was used only once. I can see that even Internet Explorer was used. Hopefully not by me 😄.

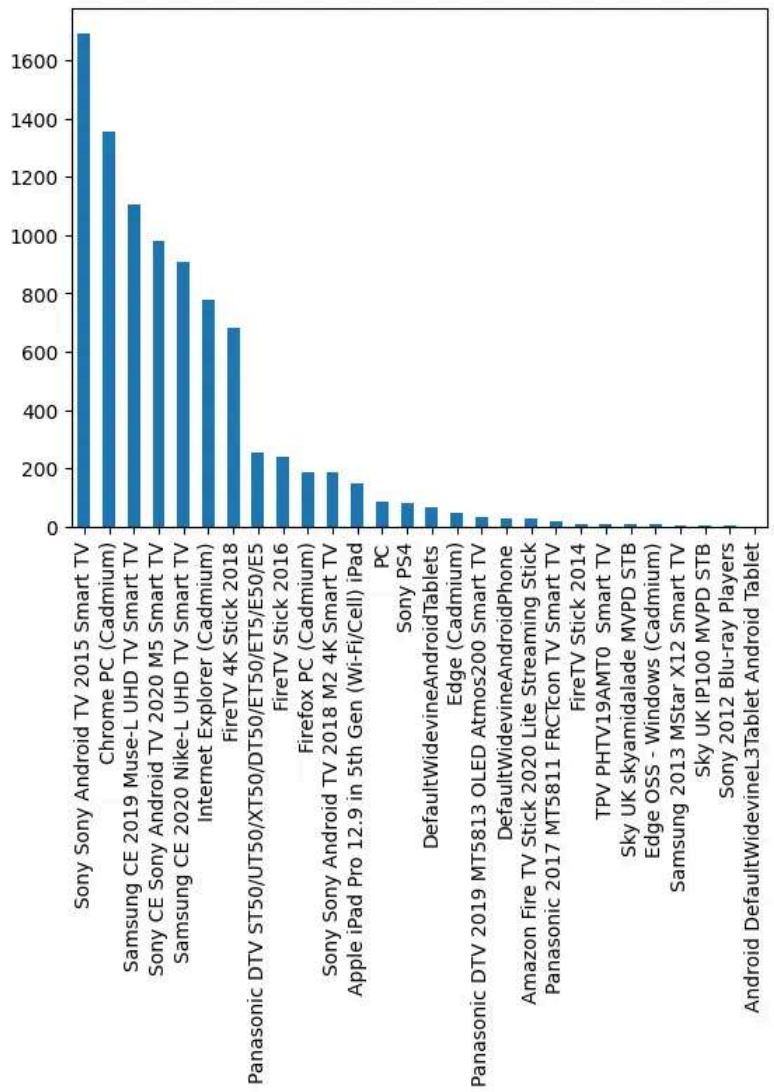
```
df['Device Type'].value_counts()
```

```
> Sony Sony Android TV 2015 Smart TV                  1692
> Chrome PC (Cadmium)                                 1357
```

```
> Samsung CE 2019 Muse-L UHD TV Smart TV                1103
> Sony CE Sony Android TV 2020 M5 Smart TV               980
> Samsung CE 2020 Nike-L UHD TV Smart TV                 910
> Internet Explorer (Cadmium)                            776
> FireTV 4K Stick 2018                                   681
> Panasonic DTV ST50/UT50/XT50/DT50/ET50/ET5/E50/E5      252
> FireTV Stick 2016                                      241
> Firefox PC (Cadmium)                                   188
> Sony Sony Android TV 2018 M2 4K Smart TV               186
> Apple iPad Pro 12.9 in 5th Gen (Wi-Fi/Cell) iPad       149
> PC                                                      85
> Sony PS4                                                79
> [...]                                                 [...]
> Edge OSS - Windows (Cadmium)                            8
> Samsung 2013 MStar X12 Smart TV                         6
> Sky UK IP100 MVPD STB                                   6
> Sony 2012 Blu-ray Players                               6
> Android DefaultWidevineL3Tablet Android Tablet          1
```

We can again quickly plot this.

```
df['Device Type'].value_counts().plot(kind='bar')
plt.show()
```



I want to know more about my own devices, so let us filter them based on Profile '*Code & Dogs*'.

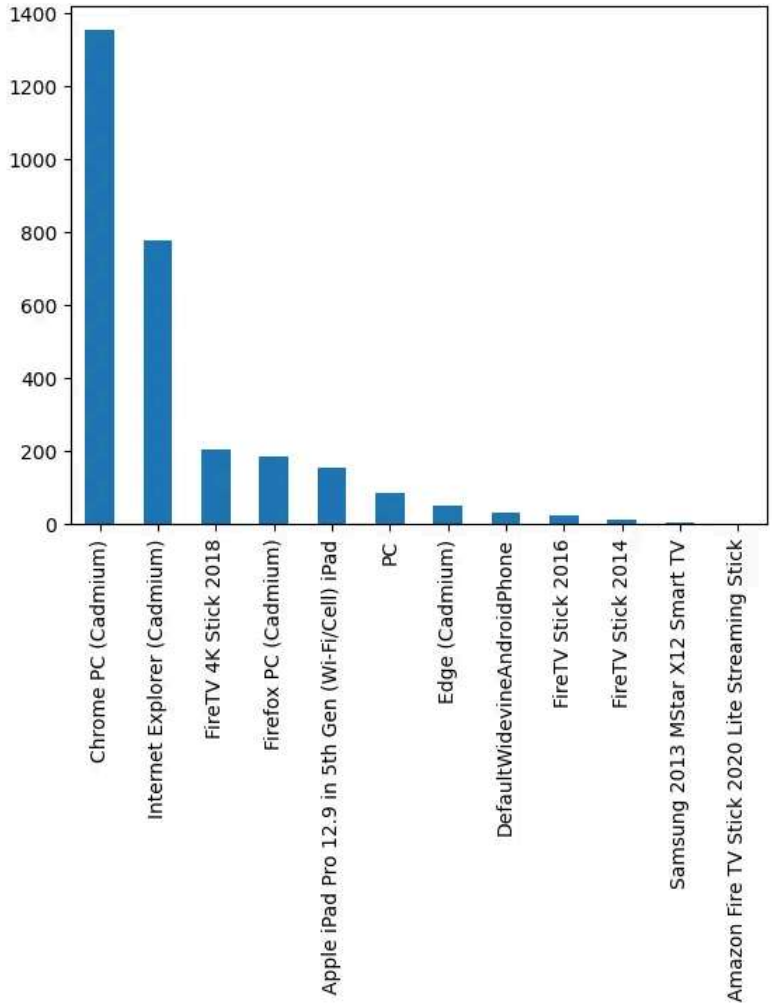
```
df=df.loc[df['Profile Name'] == 'Code & Dogs']
```

On no, indeed Internet Explorer was used by us. I blame it on my wife 😀

```
df['Device Type'].value_counts()
> Chrome PC (Cadmium)                                  1353
> Internet Explorer (Cadmium)                           776
> FireTV 4K Stick 2018                                  204
> Firefox PC (Cadmium)                                  184
> Apple iPad Pro 12.9 in 5th Gen (Wi-Fi/Cell) iPad      155
> PC                                                     85
> Edge (Cadmium)                                         50
> DefaultWidevineAndroidPhone                            30
> FireTV Stick 2016                                      25
> FireTV Stick 2014                                      11
> Samsung 2013 MStar X12 Smart TV                         6
> Amazon Fire TV Stick 2020 Lite Streaming Stick          2
```

Alright, time for the last chart of the day.

```
df['Device Type'].value_counts().plot(kind='bar')
plt.show()
```




## 💡 Further Ideas

I have much more ideas on further questions that we can answer with the data.

- What was the most popular/watched title?
- Was there any title watched by all Profiles?
- Can we recommend a title for one Profile based on the common watching history of other Profiles?
- ...

I am already curious about your ideas. **Perhaps you can also download your Netflix viewing history and we can build a community dataset to further analyze interesting things.**

## 📓 Summary & Resources

**This was story #1 of the Python Data Science December**. We analyzed my own Netflix viewing history and tried to answer some interesting questions about it.

If you want to follow along with all my stories & **support me**, you can **register on Medium**. If something is unclear or you need help, just drop a comment. I will answer it for sure.

You can find the whole Python code together with a sample dataset (167 rows) for free on GitHub. As it is my very personal data, I decided to offer the full anonymized dataset (9273 rows) only against a small donation on Patreon or Gumroad.

- ✔ **GitHub** (for free) — full code & sample dataset (167 rows)
- ✔ **Gumroad** (1$ one-time purchase) — full dataset (9273 rows)
- ✔ **Patreon** (3$ / month for regular & advanced content) — full dataset (9273 rows)

Programming Python Coding Data Science Technology

15

Written by Techletters

Follow

1.1K Followers · Editor for Python Point

#ai #datascience #salesforce #crypto #kafka #python — Follow me: https://techletters.medium.com/membership

More from Techletters and Python Point

Techletters in Python Point

Techletters in Python Point

### MQTT Beginners Guide

The IOT protocol explained with Python examples

6 min read · Aug 30, 2020

258 6

### MQTT and Kafka

How to combine two complementary technologies

13 min read · Jan 8, 2021

595 2


Techletters in Python Point

### Python— Sankey Diagrams

Visualize Data Flows Using Sankey Diagrams With Python

8 min read · Dec 8, 2022

8 1

Techletters in Tech Force

### How To Publish Platform Events

Publish Salesforce Platform Events with different technologies

8 min read · Feb 4, 2022

10 1

See all from Techletters

See all from Python Point

## Recommended from Medium

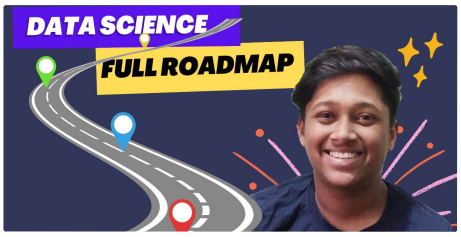
Gabe Araujo, M.Sc. in Level Up Coding

### 🧬 How I Used Python to Make Everyday Tasks Easier

Hey there! As a busy person with a lot on my plate, I'm always looking for ways to make m...

8 min read · May 1

425 1




Amit Timalsina

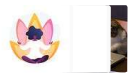
### How I became Machine Learning engineer at 18 years old | Full...

At 18 years old, I landed my dream job as a Machine Learning Engineer, a role many...

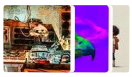
8 min read · Mar 13

267 5

Lists



**Stories to Help You Grow as a Software Developer**
19 stories · 61 saves



**What is ChatGPT?**
9 stories · 57 saves

**Leadership**
30 stories · 24 saves

**Stories to Help You Grow as a Designer**
11 stories · 43 saves

Sofia Pinto in Data analytics at Nesta

## All you need to get started with Twitter API v2 using Python

Do you want to start collecting Twitter data, but don't know where to start? Are you a...

12 min read · Dec 1, 2022

24 1

Shubhankar Goje

## Spotify Data Analysis through Power BI

On the selected data set, we had to do exploratory data analysis for a statistics...

7 min read · Dec 20, 2022

17 4

Abubakarumoru

## ADVENTURE WORKS DATA 2019: EXPLORATORY ANALYSIS ON...

INTRODUCTION

10 min read · May 12

81 1

Ethan Duong

## Hotel Booking project— Exploratory Data Analysis

Conducting EDA using Python. Visualization made by Tableau.

9 min read · Feb 8

24 1

See more recommendations