How to Clean Your Data in Python

https://towardsdatascience.com/how-to-clean-your-data-in-python-8f178638b98d/

https://gist.github.com/huongngo-8?page=2

A detailed guide on how to clean your data to kickstart your personal projects

Huong Ngo

Jul 30, 2022

13 min read

Share

Photo by Towfiqu barbhuiya on Unsplash

When I participated in my college’s directed reading program (a mini-research program where undergrad students get mentored by grad students), *I had only taken 2 statistics in R courses*. While these classes taught me a lot about how to manipulate data, create data visualizations, and extract analyses, working on my first personal project in the program made me **realize I had never worked with "messy data"**. Those courses involved pre-cleaned and processed datasets but **didn’t teach students how to clean datasets** which **creates a barrier to starting on personal projects**. Hence, I hope that this article serves as a starting point for you to **learn how to clean your data efficiently** to **kickstart your personal projects**.

For this article, I’ll be working with the **Netflix TV Shows and Movies Dataset** which features many inconsistencies and missing data

Table of Contents

Step 1: Look into your data

Before even performing any cleaning or manipulation of your dataset, you should take a glimpse at your data to understand **what variables you're working with, how the values are structured based on the column they're in, and maybe you could have a rough idea of the inconsistencies that you'll need to address or they'll be cumbersome in the analysis phase.** Here, you might also be able to eliminate certain columns that you won't need depending on the analysis you want to do.

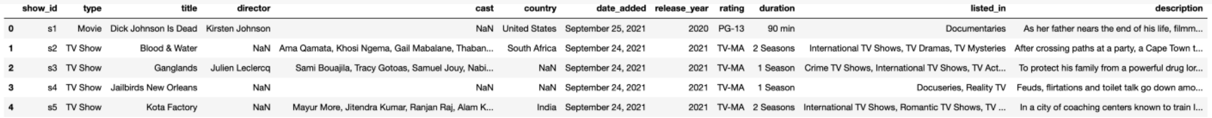
1. Print the first few rows of your dataset

Here, I printed the first 7 rows of my dataset, but you can print 5 or 10. I recommend keeping it to anything less than 10 or else it’ll be too overwhelming for what you’re currently trying to do–a quick glimpse of the dataset.

```
# importing dataset

netflix_titles = pd.read_csv("/Users/huongngo/Desktop/PERSONAL PROJECTS/zuckflix_meta/Data/netflix_titles.csv")

# printing the first 5 rows of dataset

netflix_titles.head()
```

view raw

code.py hosted with ❤ by GitHub



	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...	To protect his family from a powerful drug lor...
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Season	Docuseries, Reality TV	Feuds, flirtations and toilet talk go down amo...
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV ...	In a city of coaching centers known to train l...

Doing this will give you a good idea of what data types you might be dealing with, what columns you need to perform transformations or cleaning, and other data you might be able to extract.

Before we look at this more closely, let’s perform the next step.

2. Save the variables to a list

You want to do this to have **easy access to the different columns of the dataset**, especially when you want to perform the same transformations to different subsets of columns.

```
# getting the columns of the
dataset
```

```
columns =
list(netflix_titles.columns)
```

```
columns
```

```
"""
```

```
Output:
```

```
['show_id',
```

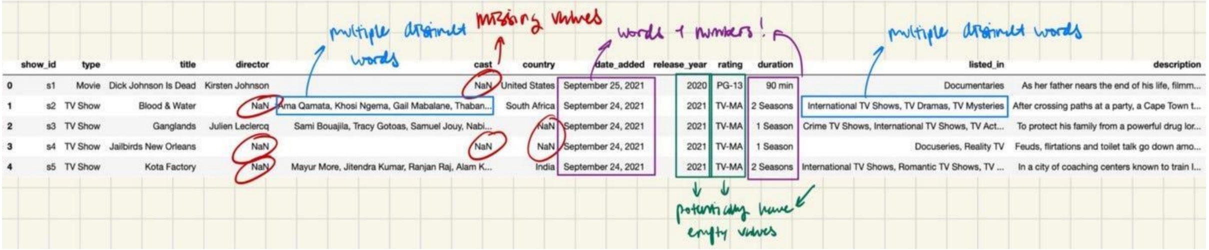
```
'type',
'title',

'director',

'cast',

'country',

'date_added',

'release_year',

'rating',
'duration',

'listed_in',

'description']

"""
```

view raw

code_1.py hosted with ❤ by GitHub

3. Note down potential issues you will have to address in each column.

To stay organized, note the issues you see in your dataset (by taking a glimpse of your dataset like in Step 1).



	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...	To protect his family from a powerful drug lor...
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Season	Docuseries, Reality TV	Feuds, flirtations and toilet talk go down amo...
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV ...	In a city of coaching centers known to train I...

This picture above represents what I can see just from glimpsing at the dataset and is something that **you should think about when you're looking at your dataset**. Here are a few things that stand out to me:

- There are some columns with missing values. This could cause a lot of problems for analysis and plotting if not addressed and resolved early in the process.
- There are columns with words and numbers, such as `date_added` and `duration`. This can be a problem if we want to **make time-series graphs** by the date, or **other plots to explore duration's relationship with other variables.**
- There are 2 columns with multiple distinct words joined together by a comma. This is an issue if we want to make **plots exploring the distribution of `listed_in` (genre) or the actors on Netflix.**
- Other columns could potentially have missing values. The next step looks at the way to **check which columns have missing values** and **how much missing data they have.**

Step 2: Look at the proportion of missing data

```
# examining missing values

print("Missing values distribution: ")

print(netflix_titles.isnull().mean())

print("")

"""
```

Output:

```
Missing values distribution:

show_id         0.000000

type            0.000000

title           0.000000

director        0.299080

cast            0.093675
```

```
country         0.094357

date_added      0.001135

release_year    0.000000

rating          0.000454

duration        0.000341

listed_in       0.000000

description     0.000000
```

```
dtype: float64

"""
```

view raw

code_2.py hosted with ❤ by GitHub

From this code chunk, you can easily look at the distribution of missing values in the dataset to get a good idea of which columns you’ll need to work with to resolve the missing values issue.

From the output, these are insights you can gather:

- `director` column has the highest percentage of missing data ~ 30%
- `cast` and `country` column also has a considerable percentage of missing data ~ 9%
- `date_added, rating` and `duration` don’t have that much missing data ~ 0% – 0.1%
- Fortunately, most other columns are not empty.

Your next question is probably, **how do I deal with these columns with missing values?**

There are a few ways to deal with it:

1. Drop the column completely. If the column isn't that important to your analysis, just drop it.
2. Keep the column. In this case, because **the `director, cast` and `country` columns are quite important to my analysis**, I will keep them.
3. **Imputation – the process of replacing missing data with substituted values.** Here, it is not possible to do so because most of the data are string values and not numerical values. However, I will be writing an article that talks more about imputation in detail, why and when it should be used, and how you can use it in R and Python with the help of some packages.

Before I continue, I will bring up the issue of missing values **across rows.**

In some cases, you might want to examine the **distribution of missing values across all the rows of your dataset** (given that your dataset doesn't have a large number of observations/rows). From here, you can **choose from the choices above depending on how important the rows are to your analysis**. For instance, your dataset contains recorded data of something that is changing over time. Even though a row can contain missing values, you might not want to eliminate it because there is important time information you want to retain.

Let’s continue to step 3 before I show you how to deal with the NaN values even after keeping the columns.

Step 3: Check the data type of each column

```
# check datatype in each column

print("Column datatypes: ")

print(netflix_titles.dtypes)
```

```
"""

Output:

Column datatypes:

show_id         object

type            object

title           object
```

```
director        object

cast            object

country         object

date_added      object

release_year     int64

rating          object

duration        object
```

```
listed_in       object

description     object

dtype: object

"""
```

view raw

code_3.py hosted with ❤ by GitHub

Here, you can see that all the columns have `object` as their datatype aside from `release_year`. In pandas, object means either string or mixed type (numerical and non-numerical type mixed). And from our dataset, you’ll be able to tell which columns are strictly string and mixed type.

Step 4: If you have columns of strings, check for trailing whitespaces

After we know which data types we are dealing with, let’s make sure we remove any trailing characters and whitespace using `strip`.

```
# getting all the columns with string/mixed type values

str_cols = list(netflix_titles.columns)

str_cols.remove('release_year')
```

```
# removing leading and trailing characters from columns with
str type

for i in str_cols:

   netflix_titles[i] = netflix_titles[i].str.strip()
```

view raw

code_4.py hosted with ❤ by GitHub

Step 5: Dealing with Missing Values (NaN Values)

Referring back to the columns of missing values, let’s take a look at the columns: `director, cast, country, date_added, rating, duration`. We can segment these columns by whether they are a string or mixed type.

String: `director, cast, country, rating` (here, it’s a string and not mixed because the numerical values won’t have any meaning if separated)

Mixed: `date_added, duration`

`NaN` means Not a Number in pandas. It is a special floating-point value that is different from `NoneType` in Python. `NaN` values can be annoying to work with, especially when you want to filter them out for plots or analysis. To make our lives easier, let’s **replace these NaN values with something else.**

For string type values, we can replace `NaN` values with "" or "None" or any string that can indicate to you that there isn’t any value in that entry. Here, I chose to replace it with "" using the `fillna` function. Because it’s not an in-place function, I reassigned the changed values to the column in the dataset.

```
# names of the columns
```

```
columns = ['director', 'cast', 'country', 'rating', 'date_added']

# looping through the columns to fill the entries with NaN values with ""

for column in columns:

    netflix_titles[column] = netflix_titles[column].fillna("")
```

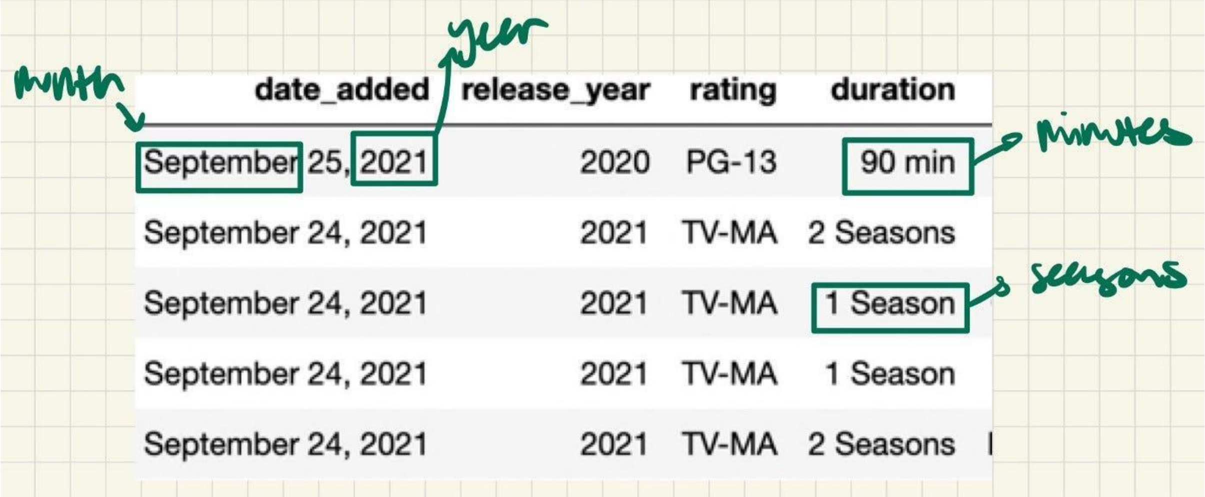
view raw

code_5.py hosted with ❤ by GitHub

Here, you must have noticed that I left out the duration column. This is because we'll be doing something with that column later down the road.

Step 6: See if there are any other variables that you can obtain by extracting them from other variables

For mixed-type values, before we tackle the missing value issue, let's see if we can extract any data to make our analysis richer or process easier.



date_added	release_year	rating	duration
September 25, 2021	2020	PG-13	90 min
September 24, 2021	2021	TV-MA	2 Seasons
September 24, 2021	2021	TV-MA	1 Season
September 24, 2021	2021	TV-MA	1 Season
September 24, 2021	2021	TV-MA	2 Seasons

month
year
minutes
seasons

Looking at `date_added`, we can see that it contains the month, date, and year that the film/show was added. Instead of having all this information in one column, why not try to separate them? That way, we can choose to isolate how month or year interacts with the other variables instead of looking at `date_added` where its granularity will make it difficult for any trend to be discovered.

Below, I’ve written code to not only separate the information into 2 other columns but also filtered out the rows with `NaN` values and replaced them with 0, just like what was done before with "".

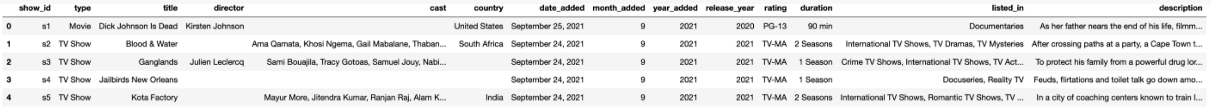
```
# examining rows with null values for date_added column

rows = []

for i in range(len(netflix_titles)):

   if netflix_titles['date_added'].iloc[i] == "":
        rows.append(i)

# examine those rows to confirm null state

netflix_titles.loc[rows, :]

# extracting months added and years added

month_added = []

year_added = []
```

```
for i in range(len(netflix_titles)):

    # replacing NaN values with 0

    if i in rows:

        month_added.append(0)

        year_added.append(0)

    else:

        date = netflix_titles['date_added'].iloc[i].split(" ")
        month_added.append(date[0])

        year_added.append(int(date[2]))



# turning month names into month numbers

for i, month in enumerate(month_added):

    if month != 0:

        datetime_obj = datetime.strptime(month, "%B")
        month_number = datetime_obj.month

        month_added[i] = month_number


# checking all months

print(set(month_added))

print(set(year_added))
```

```
# inserting the month and year columns into the dataset

netflix_titles.insert(7, "month_added", month_added, allow_duplicates = True)

netflix_titles.insert(8, "year_added", year_added, allow_duplicates =
True)netflix_titles.head()
```

view raw

code_6.py hosted with ❤ by GitHub



	show_id	type	title	director	cast	country	date_added	month_added	year_added	release_year	rating	duration	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson		United States	September 25, 2021	9	2021	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water		Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	9	2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...		September 24, 2021	9	2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...	To protect his family from a powerful drug lor...
3	s4	TV Show	Jailbirds New Orleans				September 24, 2021	9	2021	2021	TV-MA	1 Season	Docuseries, Reality TV	Feuds, flirtations and toilet talk go down amo...
4	s5	TV Show	Kota Factory		Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	9	2021	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV ...	In a city of coaching centers known to train l...

Now, the new dataset contains the `month_added` and `year_added` columns. This will allow us to do some trend analysis later.

Looking at `duration`, on top of it being a mixed type, there are also 2 different time units in this column. This is a problem because we are dealing with 2 different types of content that are measured differently for time. Thus, making graphs for `duration`will be quite difficult to interpret if we keep them as it is. The good thing is that there are many ways to deal with this issue. The way I've chosen to deal with it is by **separating the type of content into 2 different datasets and naturally, the duration column will just be numerical and just have 1 type of time unit.** This way, you can **easily and clearly plot using the values.**

```
# separating original dataset to tv show and movie dataset respectively

shows = []

films = []
```

```
# looping through the dataset to identify rows that are TV shows and films

for i in range(len(netflix_titles)):

    if netflix_titles['type'].iloc[i] == "TV Show":

        shows.append(i)

    else:

        films.append(i)
```

```
# grouping rows that are TV shows

netflix_shows = netflix_titles.loc[shows, :]

#grouping rows that are films

netflix_films = netflix_titles.loc[films, :]

# reseting the index of the new datasets
```

```
netflix_shows = netflix_shows.set_index([pd.Index(range(0,
len(netflix_shows)))])
```

```
netflix_films = netflix_films.set_index([pd.Index(range(0,
len(netflix_films)))])
```

view raw

code_7.py hosted with ❤ by GitHub

Because the `duration` column has both strings and numbers, **I’ll also have to create a function to extract the number from that column so that it can be inserted into the columns of the 2 new datasets.**

```
# get length of movie or number of seasons of show
```

```
def getDuration(data):
```

```
count = 0

durations = []

for value in data:

    # filling in missing values

    if type(value) is float:

        durations.append(0)

    else:
            values = value.split(" ")

            durations.append(int(values[0]))

    return durations




# inserting new duration type column for shows (renamed column)

netflix_shows.insert(11, 'seasons',
getDuration(netflix_shows['duration']))

netflix_shows = netflix_shows.drop(['duration'], axis = 1)
```

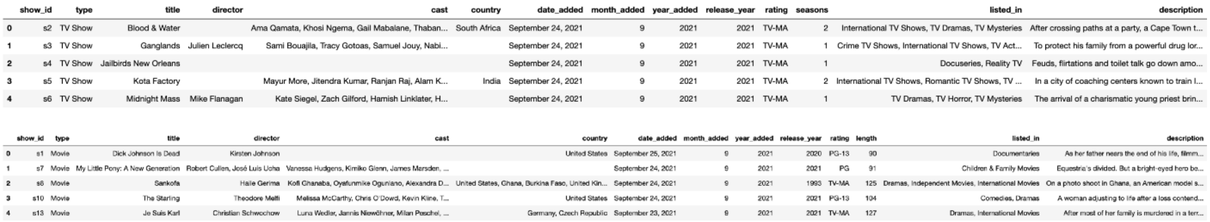
```
netflix_shows.head()

# inserting new duration type column for films (renamed column)

netflix_films.insert(11, 'length',
getDuration(netflix_films['duration']))

netflix_films = netflix_films.drop(['duration'], axis = 1)

netflix_films.head()
```

view raw

code_8.py hosted with ❤ by GitHub

Step 7: Check the unique values of columns

Beyond potentially missing values, there could be corrupted values that you can run into once you perform analysis. To check this, we can check for unique values for some of the columns. Let's refer to the first 5 rows of the datasets as our starting point.



	show_id	type	title	director	cast	country	date_added	month_added	year_added	release_year	rating	seasons	listed_in	description
0	s2	TV Show	Blood & Water		Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	9	2021	2021	TV-MA	2	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...
1	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...		September 24, 2021	9	2021	2021	TV-MA	1	Crime TV Shows, International TV Shows, TV Act...	To protect his family from a powerful drug lor...
2	s4	TV Show	Jailbirds New Orleans				September 24, 2021	9	2021	2021	TV-MA	1	Docuseries, Reality TV	Feuds, flirtations and toilet talk go down amo...
3	s5	TV Show	Kota Factory		Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	9	2021	2021	TV-MA	2	International TV Shows, Romantic TV Shows, TV ...	In a city of coaching centers known to train I...
4	s6	TV Show	Midnight Mass	Mike Flanagan	Kate Siegel, Zach Gilford, Hamish Linklater, H...		September 24, 2021	9	2021	2021	TV-MA	1	TV Dramas, TV Horror, TV Mysteries	The arrival of a charismatic young priest brin...

	show_id	type	title	director	cast	country	date_added	month_added	year_added	release_year	rating	length	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson		United States	September 25, 2021	9	2021	2020	PG-13	90	Documentaries	As her father nears the end of his life, filmm...
1	s7	Movie	My Little Pony: A New Generation	Robert Cullen, José Luis Ucha	Vanessa Hudgens, Kimiko Glenn, James Marsden, ...		September 24, 2021	9	2021	2021	PG	91	Children & Family Movies	Equestria's divided. But a bright-eyed hero be...
2	s8	Movie	Sankofa	Haile Gerima	Kofi Ghanaba, Oyafunmike Ogunlano, Alexandra D...	United States, Ghana, Burkina Faso, United Kin...	September 24, 2021	9	2021	1993	TV-MA	125	Dramas, Independent Movies, International Movies	On a photo shoot in Ghana, an American model s...
3	s10	Movie	The Starling	Theodore Melfi	Melissa McCarthy, Chris O'Dowd, Kevin Kline, T...	United States	September 24, 2021	9	2021	2021	PG-13	104	Comedies, Dramas	A woman adjusting to life after a loss contend...
4	s13	Movie	Je Suis Karl	Christian Schwochow	Luna Wedler, Jannis Niewöhner, Milan Peschel, ...	Germany, Czech Republic	September 23, 2021	9	2021	2021	TV-MA	127	Dramas, International Movies	After most of her family is murdered in a terr...

It might not be strategic to check the unique values of all the columns, especially the title, director, and cast as there could be a large number of unique values to examine. Instead, let's focus on a list of potential unique values that could be easier and more important to check given that it could be more insightful for future analysis. From a glimpse at the datasets, the columns `country, rating, listed_in` are probably the ones of interest. Let's examine the rating column first as that seems to be the least complicated one to deal with.

You can easily obtain the unique values of a column like rating using Python’s built-in function, `unique`. Let’s try that!

```
# getting the unique ratings for films

netflix_films['rating'].unique()

"""

Output:

array(['PG-13', 'PG', 'TV-MA', 'TV-PG', 'TV-14', 'TV-Y', 'R', 'TV-
G',
      'TV-Y7', 'G', 'NC-17', '74 min', '84 min', '66 min', 'NR',
'',

      'TV-Y7-FV', 'UR'], dtype=object)

"""

# getting the unique ratings for shows

netflix_shows['rating'].unique()
```

```
"""

Output:

array(['TV-MA', 'TV-14', 'TV-Y7', 'TV-PG', 'TV-Y', 'TV-G', 'R',
'NR', '',

      'TV-Y7-FV'], dtype=object)

"""
```

view raw

code_9.py hosted with ❤ by GitHub

This seems interesting. Why are there 74 min, 84 min, and 66 min in the unique types of rating for films? And why are there UR (Unrated) and NR (Not Rated)? Aren't they supposed to mean the same thing? Let’s investigate this further by extracting the rows that have these weird entries.

```
# printing more details of the rows that have incorrect ratings

incorrect_ratings = ['74 min', '84 min', '66 min']

for i in range(len(netflix_films)):

    if netflix_films['rating'].iloc[i] in incorrect_ratings:

        print(netflix_films.iloc[i])

        print("")
```

```
"""

Output:

show_id
s5542

type
Movie

title                                              Louis
C.K. 2017
```

```
director
Louis C.K.




cast
Louis C.K.




country                                            United
States




date_added                                         April
4, 2017




month_added
4




year_added
2017
```

```
release_year
2017

rating
74 min

length
0

listed_in
Movies

description     Louis C.K. muses on religion, eternal love,
gi...

Name: 3562, dtype: object
```

show_id

s5795

type

Movie

title Louis C.K.:

Hilarious

director

Louis C.K.

cast

Louis C.K.

```
country                                             United
States




date_added                                     September
16, 2016




month_added
9




year_added
2016




release_year
2010




rating
84 min
```

```
length
0

listed_in
Movies

description     Emmy-winning comedy writer Louis C.K.
brings h...

Name: 3738, dtype: object

show_id
s5814
```

type

Movie

title Louis C.K.: Live at the Comedy Store

director

Louis C.K.

cast

Louis C.K.

country United States

date_added August 15, 2016

```
month_added
8

year_added
2016

release_year
2015

rating
66 min

length
0

listed_in
Movies
```

```
description     The comic puts his trademark
hilarious/thought...


Name: 3747, dtype: object


"""
```

view raw

code_10.py hosted with ❤ by GitHub

Using this code chunk, we can see that 3 distinct rows contain this weird rating and that it actually belongs to the length column. We can also see the row number where the issue is located which will be useful to use for fixing the entries.

After some quick Googling, we can proceed to fix these entries by moving the "wrong ratings" (actually duration) to the length column and entering the right ratings.

```
# getting the row indices

index = [3562, 3738, 3747]

# fixing the entries

for i in index:

   split_value = netflix_films['rating'].iloc[i].split(" ")

   length = split_value[0]
```

```
    netflix_films['duration'].iloc[i] = length

    netflix_films['rating'].iloc[i] = "NR"



# double checking the entries again

for i in index:

    print(netflix_films.iloc[i])
```

view raw

code_11.py hosted with ❤ by GitHub

For the UR and NR values in the rating column, we should keep the consistency where NR is used in the `netflix_shows` dataset and change UR values to NR.

```
# fixing the entries

for i in range(len(netflix_films)):

    if netflix_films['rating'].iloc[i] == "UR":

        netflix_films['rating'].iloc[i] = "NR"



# double checking
```

```
netflix_films['rating'].unique()

"""

array(['PG-13', 'PG', 'TV-MA', 'TV-PG', 'TV-14', 'TV-Y', 'R',
'TV-G',

      'TV-Y7', 'G', 'NC-17', 'NR', '', 'TV-Y7-FV'],
dtype=object)

"""
```

view raw

code_12.py hosted with ❤ by GitHub

Now that we’ve cleaned up the `rating` column, let’s look at the country and `listed_in` column. By now, you must have realized that it’s not as easy as the `rating` column to extract unique values. This is because the values in those columns are words joined together by commas, making it more difficult to extract the set of words and then find unique words from that set.

How we’re going to get around this issue is by implementing a unique function for this special case.

To start, let’s think about what data structure can give us unique values easily. If you guessed sets, you’re right! Given its ability to **store unique elements of the same type in sorted order,** it’s a fitting data structure for what we want to do.

Then, to extract those words that are joined by commas, we can use the `split` function to split up the string by the comma.

```
# function to get unique values of a
column
```

```
def getUnique(data):
    unique_values = set()
    for value in data:
        if type(value) is float:
            unique_values.add(None)
        else:
            values = value.split(", ")
            for i in values:

                unique_values.add(i)

    return list(unique_values)
```

view raw

code_13.py hosted with ❤ by GitHub

After using the function, we can easily obtain the unique values for the `country` and `listed_in` columns.

```
# getting unique country names
```

```
unique_countries =
getUnique(netflix_titles['country'])

unique_countries

"""

Output:

['',

'Czech Republic',
'Armenia',

'Belgium',

'Mozambique',

'East Germany'*,

'West Germany'*,

'Soviet Union'*,

'Burkina Faso', etc.] (shortened for article)
```

```
"""
```

view raw

code_14.py hosted with ❤ by GitHub

Next, let’s examine the list of unique countries to see if there are any inconsistencies or mistakes. By doing so and with a little bit of Googling, we can see there are some issues with this list:

- There’s both the Soviet Union and Russia
- There’s both the West/East Germany and Germany

We can easily fix this with a few modifications to the dataset.

```
# converting soviet union to russia and east/west germany to
germany
```

```
for i in range(len(netflix_titles)):

   if type(netflix_titles['country'].iloc[i]) is not float:

       countries = netflix_titles['country'].iloc[i].split(", ")

       for j in range(len(countries)):

           if "Germany" in countries[j]:

               countries[j] = "Germany"

           elif "Soviet Union" in countries[j]:
```

```
                    countries[j] = "Russia"
```

```
            netflix_titles['country'].iloc[i] = ",
    ".join(countries)
```

view raw

code_15.py hosted with ❤ by GitHub

As for the list of genres, we can see that there are some genres we might not want or need to include. Thus, we can easily remove it from the dataset to make our analysis less confounding.

```
# getting unique film genres
```

```
unique_genres_films =
getUnique(netflix_films['listed_in'])
```

```
unique_genres_films
```

```
"""
```

```
Output:

['International Movies',

'Children & Family Movies',

'LGBTQ Movies',
'Classic Movies',

'Action & Adventure',

'Stand-Up Comedy',

'Sports Movies',

'Documentaries',

'Movies'*,

'Music & Musicals',
'Romantic Movies',

'Anime Features',

'Comedies',

'Independent Movies',

'Dramas',

'Thrillers',

'Cult Movies',
'Faith & Spirituality',

'Horror Movies',

'Sci-Fi & Fantasy']

"""

# getting unique show genres

unique_genres_shows =
getUnique(netflix_shows['listed_in'])
```

```
unique_genres_shows
```

```
"""
```

```
Output:

['TV Dramas',

'TV Comedies',

'TV Action & Adventure',
'TV Mysteries',

'Romantic TV Shows',

"Kids' TV",

'TV Horror',

'International TV Shows',

'TV Sci-Fi & Fantasy',

'Korean TV Shows',
'Spanish-Language TV Shows',

'Science & Nature TV',

'Crime TV Shows',

'TV Shows'*,

'Classic & Cult TV',

'Teen TV Shows',

'TV Thrillers',
'Stand-Up Comedy & Talk Shows',

'Docuseries',

'Reality TV',

'British TV Shows',

'Anime Series']

"""
```

view raw

code_16.py hosted with ❤ by GitHub

In both the TV shows and films dataset, there is a "TV Shows" and "Movies" genre. Technically, this isn’t a genre but could be a label of the type of content. To confirm this, we should print out the counts of these "genres" appearing in the respective datasets.

The hypothesis is that if these "genres" appear in all the rows of the datasets, it means that they’re simply labels. Otherwise, we’ll have to investigate further as to what those "genres" represent.

```
# checking for TV shows

# replace netflix_shows with netflix_films to check for movies

count = 0

index = []
```

```python
for i, value in enumerate(netflix_shows['listed_in']):

    genres = value.split(", ")

    if "TV Shows" in genres:

        count += 1

        index.append(i)

print("count %s" %count)

print("index %s" %index)
```

```
"""

Output:

TV shows:

count 16

index [59, 110, 272, 286, 452, 599, 991, 1432, 1548, 1808, 1840, 2107, 2160,
2190, 2465, 2559]
```

```
Movies:

count 57

index [197, 310, 456, 457, 458, 476, 477, 1906, 1938, 1941, 2146, 2165, 2621,
2711, 2758, 2862, 2863, 2867, 3036, 3137, 3138, 3139, 3140, 3141, 3142, 3225,
3226, 3228, 3232, 3517, 3562, 3652, 3694, 3722, 3738, 3747, 3789, 3824, 3883,
4271, 4273, 4543, 4544, 4784, 4910, 4911, 5006, 5178, 5259, 5290, 5292, 5293,
5295, 5476, 5477, 5478, 6092]

"""
```

view raw

code_17.py hosted with ❤ by GitHub

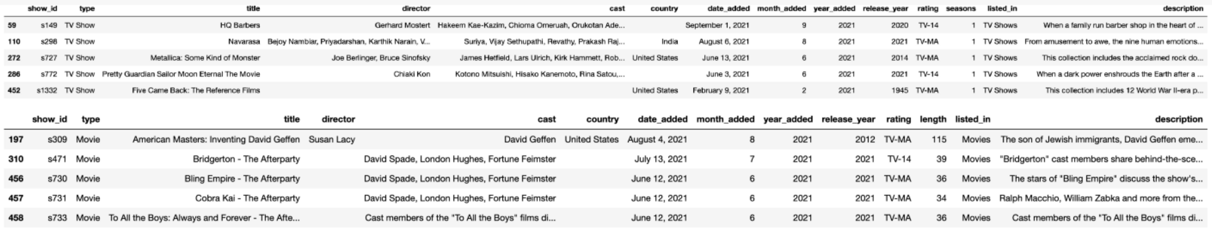
As the count of the "genres" is less than the size of the datasets, let’s use the output of the code to examine the rows.

Since I’ve written the code to specifically output the row indices in a list, we can easily use that list and the `iloc` function to get a view of the rows.

```
# printing the first 5 rows of all rows that have TV Shows as its genre

netflix_shows.iloc[index[0:5]]

# printing the first 5 rows of all rows that have Movies as its genre

netflix_films.iloc[index[0:5]]
```

view raw

code_18.py hosted with ❤ by GitHub



	show_id	type	title	director	cast	country	date_added	month_added	year_added	release_year	rating	seasons	listed_in	description
59	s149	TV Show	HQ Barbers	Gerhard Mostert	Hakeem Kae-Kazim, Chioma Omeruah, Orukotan Ade...		September 1, 2021	9	2021	2020	TV-14	1	TV Shows	When a family run barber shop in the heart of ...
110	s298	TV Show	Navarasa	Bejoy Nambiar, Priyadarshan, Karthik Narain, V...	Suriya, Vijay Sethupathi, Revathy, Prakash Raj...	India	August 6, 2021	8	2021	2021	TV-MA	1	TV Shows	From amusement to awe, the nine human emotions...
272	s727	TV Show	Metallica: Some Kind of Monster	Joe Berlinger, Bruce Sinofsky	James Hetfield, Lars Ulrich, Kirk Hammett, Rob...	United States	June 13, 2021	6	2021	2014	TV-MA	1	TV Shows	This collection includes the acclaimed rock do...
286	s772	TV Show	Pretty Guardian Sailor Moon Eternal The Movie	Chiaki Kon	Kotono Mitsuishi, Hisako Kanemoto, Rina Satou,...		June 3, 2021	6	2021	2021	TV-14	1	TV Shows	When a dark power enshrouds the Earth after a ...
452	s1332	TV Show	Five Came Back: The Reference Films			United States	February 9, 2021	2	2021	1945	TV-MA	1	TV Shows	This collection includes 12 World War II-era p...

	show_id	type	title	director	cast	country	date_added	month_added	year_added	release_year	rating	length	listed_in	description
197	s309	Movie	American Masters: Inventing David Geffen	Susan Lacy	David Geffen	United States	August 4, 2021	8	2021	2012	TV-MA	115	Movies	The son of Jewish immigrants, David Geffen eme...
310	s471	Movie	Bridgerton - The Afterparty		David Spade, London Hughes, Fortune Feimster		July 13, 2021	7	2021	2021	TV-14	39	Movies	"Bridgerton" cast members share behind-the-sce...
456	s730	Movie	Bling Empire - The Afterparty		David Spade, London Hughes, Fortune Feimster		June 12, 2021	6	2021	2021	TV-MA	36	Movies	The stars of "Bling Empire" discuss the show's...
457	s731	Movie	Cobra Kai - The Afterparty		David Spade, London Hughes, Fortune Feimster		June 12, 2021	6	2021	2021	TV-MA	34	Movies	Ralph Macchio, William Zabka and more from the...
458	s733	Movie	To All the Boys: Always and Forever - The Afte...		Cast members of the "To All the Boys" films di...		June 12, 2021	6	2021	2021	TV-MA	36	Movies	Cast members of the "To All the Boys" films di...

Taking a look at the rows, it is now obvious that the "TV Shows" and "Movies" genre was used to signify that these content didn’t have a genre in the first place. Now that we understand what this meant, we can either choose to exclude or include it in our analysis. Here, I’ve chosen to include it because it doesn’t affect my analysis.

Although this step is tedious, it is also quite important as it allows us to find the issues in our dataset that are hidden away at a first glance.

Step 8: Join the cleaned datasets together to create another dataset [Optional]

This step is optional, but in the case that you'd want the cleaned TV shows and movies dataset in one place, you should **concatenate** them.

And that's it! You've successfully cleaned this dataset. Keep in mind that everyone has their methodology of Data Cleaning, and a lot of it is just from putting in the effort to understand your dataset. However, I hope that this article has helped you understand why data scientists spend 80% of their time cleaning their datasets. **In all seriousness, this article highlights the importance of data cleaning and more importantly, the need for a good data cleaning methodology which will help you keep your work organized which will help if you need to go back to it during the analysis process**. You can check out the full notebook here.