Data Gathering

Sources of Data
A vast amount of historical data can be found in files such as:

- MS Word documents
- Emails
- Spreadsheets
- MS PowerPoints
- PDFs
- HTML
- and plaintext files

Public and Private Archives
CSV, JSON, and XML files use plaintext, a common format, and are compatible with a wide range of applications
The Web can be mined for data using a web scraping application
The IoT uses sensors create data
Sensors in smartphones, cars, airplanes, street lamps, and home appliances capture raw data

Open Data and Private Data

1. Open Data

The Open Knowledge Foundation describes Open Data as "any content, information or data that people are free to use, reuse, and redistribute without any legal, technological, or social restriction." 2. Private Data
Data related to an expectation of privacy and regulated by a particular country/government

Structured and Unstructured Data

1. Structured Data

Data entered and maintained in fixed fields within a file or record Easily entered, classified, queried, and analyzed Relational databases or spreadsheets 2. Unstructured Data Lacks organization
Raw data Photo contents, audio, video, web pages, blogs, books, journals, white papers, PowerPoint presentations, articles, email, wikis, word processing documents, and text in general

Example of gathering image data using webcam
Note: Run this snippet using local jupyter notebook

In [6]:
```
!pip install opencv-python
```

```
Requirement already satisfied: opencv-python in c:\users\eleazar\anaconda3\envs\cpe3
11_fernandez\lib\site-packages (4.11.0.86)
Requirement already satisfied: numpy>=1.21.2 in c:\users\eleazar\anaconda3\envs\cpe3
11_fernandez\lib\site-packages (from opencv-python) (2.2.4)
```

In [ ]:

```
import cv2
key = cv2. waitKey(1)
webcam = cv2.VideoCapture(0)
while True:
    try:
        check, frame = webcam.read()
        print(check) #prints true as long as the webcam is running
        print(frame) #prints matrix values of each framecd
        cv2.imshow("Capturing", frame)
        key = cv2.waitKey(1)
        if key == ord('s'):
            cv2.imwrite(filename='saved_img.jpg', img=frame)
            webcam.release()
            img_new = cv2.imread('saved_img.jpg', cv2.IMREAD_GRAYSCALE)
            img_new = cv2.imshow("Captured Image", img_new)
            cv2.waitKey(1650)
            cv2.destroyAllWindows()
            print("Processing image...")
            img_ = cv2.imread('saved_img.jpg', cv2.IMREAD_ANYCOLOR)
            print("Converting RGB image to grayscale...")
            gray = cv2.cvtColor(img_, cv2.COLOR_BGR2GRAY)
            print("Converted RGB image to grayscale...")
            print("Resizing image to 28x28 scale...")
            img_ = cv2.resize(gray,(28,28))
            print("Resized...")
            img_resized = cv2.imwrite(filename='saved_img-final.jpg', img=img_)
            print("Image saved!")
            break
        elif key == ord('q'):
            print("Turning off camera.")
            webcam.release()
            print("Camera off.")
            print("Program ended.")
            cv2.destroyAllWindows()
            break
    except(KeyboardInterrupt):
        print("Turning off camera.")
        webcam.release()
        print("Camera off.")
        print("Program ended.")
        cv2.destroyAllWindows()
        break
```

Example of gathering voice data using microphone

Note: Run the snippet of codes using local jupyter notebook

In [9]:

```
!pip3 install sounddevice
```

```
Requirement already satisfied: sounddevice in c:\users\eleazar\anaconda3\envs\cpe311
_fernandez\lib\site-packages (0.5.1)
Requirement already satisfied: CFFI>=1.0 in c:\users\eleazar\anaconda3\envs\cpe311_f
ernandez\lib\site-packages (from sounddevice) (1.17.1)
Requirement already satisfied: pycparser in c:\users\eleazar\anaconda3\envs\cpe311_f
ernandez\lib\site-packages (from CFFI>=1.0->sounddevice) (2.22)
```

In [10]:
```
!pip3 install wavio
```

```
Requirement already satisfied: wavio in c:\users\eleazar\anaconda3\envs\cpe311_ferna
ndez\lib\site-packages (0.0.9)
Requirement already satisfied: numpy>=1.19.0 in c:\users\eleazar\anaconda3\envs\cpe3
11_fernandez\lib\site-packages (from wavio) (2.2.4)
```

In [11]:
```
!pip3 install scipy
```

```
Requirement already satisfied: scipy in c:\users\eleazar\anaconda3\envs\cpe311_ferna
ndez\lib\site-packages (1.15.2)
Requirement already satisfied: numpy<2.5,>=1.23.5 in c:\users\eleazar\anaconda3\envs
\cpe311_fernandez\lib\site-packages (from scipy) (2.2.4)
```

In [12]:
```
# "!apt-get install libportaudio2" <- This code is for linux only

# Do the following in the anaconda prompt (download the modules):
# pip install pipwin
# pip install pyaudio
# pip install sounddevice
# pip install wavio
```

In [13]:
```
# import required libraries
import pyaudio
import sounddevice as sd
from scipy.io.wavfile import write
import wavio as wv

# Sampling frequency
freq = 44100

# Recording duration
duration = 5

# Start recorder with the given values
# of duration and sample frequency
recording = sd.rec(int(duration * freq),
 samplerate=freq, channels=1) # I changed the channels to 1, I only have one input

# Record audio for the given number of seconds
sd.wait()

# This will convert the NumPy array to an audio
# file with the given sampling frequency
write("recording0.wav", freq, recording)

# Convert the NumPy array to audio file
wv.write("recording1.wav", recording, freq, sampwidth=2)
```

In [14]:
```
from IPython.display import Audio # To output the recorded audio

# Play the recorded audio
Audio("recording0.wav")
```

Out[14]:

0:00 / 0:05

Web Scraping

Web scraping, web harvesting, or web data extraction is data scraping used for extracting data from websites. The web scraping software may directly access the World Wide Web using the Hypertext Transfer Protocol or a web browser. While web scraping can be done manually by a software user, the term typically refers to automated processes implemented using a bot or web crawler. It is a form of copying in which specific data is gathered and copied from the web, typically into a central local database or spreadsheet, for later retrieval or analysis.

Reference: https://en.wikipedia.org/wiki/Web_scraping

Image Scraping using BeautifulSoup and Request

In [17]:
```
!pip install bs4
```

```
Requirement already satisfied: bs4 in c:\users\eleazar\anaconda3\envs\cpe311_fernand
ez\lib\site-packages (0.0.2)
Requirement already satisfied: beautifulsoup4 in c:\users\eleazar\anaconda3\envs\cpe
311_fernandez\lib\site-packages (from bs4) (4.13.3)
Requirement already satisfied: soupsieve>1.2 in c:\users\eleazar\anaconda3\envs\cpe3
11_fernandez\lib\site-packages (from beautifulsoup4->bs4) (2.6)
Requirement already satisfied: typing-extensions>=4.0.0 in c:\users\eleazar\anaconda
3\envs\cpe311_fernandez\lib\site-packages (from beautifulsoup4->bs4) (4.13.2)
```

In [18]:
```
!pip install requests
```

```
Requirement already satisfied: requests in c:\users\eleazar\anaconda3\envs\cpe311_fe
rnandez\lib\site-packages (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\eleazar\anaconda
3\envs\cpe311_fernandez\lib\site-packages (from requests) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\eleazar\anaconda3\envs\cpe31
1_fernandez\lib\site-packages (from requests) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\eleazar\anaconda3\envs
\cpe311_fernandez\lib\site-packages (from requests) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\eleazar\anaconda3\envs
\cpe311_fernandez\lib\site-packages (from requests) (2025.1.31)
```

In [19]:
```
import requests
from bs4 import BeautifulSoup

def getdata(url):
    r = requests.get(url)
    return r.text
```

```
htmldata = getdata("https://www.google.com/")
soup = BeautifulSoup(htmldata, 'html.parser')
for item in soup.find_all('img'):
    print(item['src'])
```

```
/images/branding/googlelogo/1x/googlelogo_white_background_color_272x92dp.png
```

In [20]:
```
!pip install selenium
```

```
Requirement already satisfied: selenium in c:\users\eleazar\anaconda3\envs\cpe311_fe
rnandez\lib\site-packages (4.31.0)
Requirement already satisfied: urllib3<3,>=1.26 in c:\users\eleazar\anaconda3\envs\c
pe311_fernandez\lib\site-packages (from urllib3[socks]<3,>=1.26->selenium) (2.4.0)
Requirement already satisfied: trio~=0.17 in c:\users\eleazar\anaconda3\envs\cpe311_
fernandez\lib\site-packages (from selenium) (0.29.0)
Requirement already satisfied: trio-websocket~=0.9 in c:\users\eleazar\anaconda3\env
s\cpe311_fernandez\lib\site-packages (from selenium) (0.12.2)
Requirement already satisfied: certifi>=2021.10.8 in c:\users\eleazar\anaconda3\envs
\cpe311_fernandez\lib\site-packages (from selenium) (2025.1.31)
Requirement already satisfied: typing_extensions~=4.9 in c:\users\eleazar\anaconda3
\envs\cpe311_fernandez\lib\site-packages (from selenium) (4.13.2)
Requirement already satisfied: websocket-client~=1.8 in c:\users\eleazar\anaconda3\e
nvs\cpe311_fernandez\lib\site-packages (from selenium) (1.8.0)
Requirement already satisfied: attrs>=23.2.0 in c:\users\eleazar\anaconda3\envs\cpe3
11_fernandez\lib\site-packages (from trio~=0.17->selenium) (25.3.0)
Requirement already satisfied: sortedcontainers in c:\users\eleazar\anaconda3\envs\c
pe311_fernandez\lib\site-packages (from trio~=0.17->selenium) (2.4.0)
Requirement already satisfied: idna in c:\users\eleazar\anaconda3\envs\cpe311_fernan
dez\lib\site-packages (from trio~=0.17->selenium) (3.10)
Requirement already satisfied: outcome in c:\users\eleazar\anaconda3\envs\cpe311_fer
nandez\lib\site-packages (from trio~=0.17->selenium) (1.3.0.post0)
Requirement already satisfied: sniffio>=1.3.0 in c:\users\eleazar\anaconda3\envs\cpe
311_fernandez\lib\site-packages (from trio~=0.17->selenium) (1.3.1)
Requirement already satisfied: cffi>=1.14 in c:\users\eleazar\anaconda3\envs\cpe311_
fernandez\lib\site-packages (from trio~=0.17->selenium) (1.17.1)
Requirement already satisfied: wsproto>=0.14 in c:\users\eleazar\anaconda3\envs\cpe3
11_fernandez\lib\site-packages (from trio-websocket~=0.9->selenium) (1.2.0)
Requirement already satisfied: pysocks!=1.5.7,<2.0,>=1.5.6 in c:\users\eleazar\anaco
nda3\envs\cpe311_fernandez\lib\site-packages (from urllib3[socks]<3,>=1.26->seleniu
m) (1.7.1)
Requirement already satisfied: pycparser in c:\users\eleazar\anaconda3\envs\cpe311_f
ernandez\lib\site-packages (from cffi>=1.14->trio~=0.17->selenium) (2.22)
Requirement already satisfied: h11<1,>=0.9.0 in c:\users\eleazar\anaconda3\envs\cpe3
11_fernandez\lib\site-packages (from wsproto>=0.14->trio-websocket~=0.9->selenium)
(0.14.0)
```

Image Scraping using Selenium

Note: Run the snippet of code using local jupyter notebook

In [22]:
```
!pip install selenium
import sys
sys.path.insert(0,'/usr/lib/chromium-browser/chromedriver')

from selenium import webdriver
from selenium.webdriver.common.by import By
import time
```

```
import requests
import shutil
import os
import getpass
import urllib.request
import io
import time
from PIL import Image
user = getpass.getuser()
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument('--headless')
chrome_options.add_argument('--no-sandbox')
chrome_options.add_argument('--disable-dev-shm-usage')
driver = webdriver.Chrome()
def scroll_to_end(driver):
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
    time.sleep(5) # sleep_between_interactions

def getImageUrls(name,totalImgs,driver):
    search_url = "https://www.google.com/search?q=cat&tbm=isch&ved=2ahUKEwjNn_Gn7Yy
    driver.get(search_url)
    img_urls = set()
    img_count = 0
    results_start = 0

    while(img_count+results_start<totalImgs): # Extract actual images now
        scroll_to_end(driver)
        totalResults = driver.find_elements(By.CLASS_NAME,"Q4LuWd")
        print('total results:', len(totalResults))
        print(f"Found: {totalResults} search results. Extracting links from{results
        for img in totalResults[results_start:totalImgs]:
            img.click()
            time.sleep(5)
            image = driver.find_element(By.CLASS_NAME,'iPVvYb')
            img_urls.add(image.get_attribute('src'))
            print(img_urls)
            img_count=len(img_urls)
            print(img_count)
    return img_urls

def downloadImages(folder_path,file_name,url):
    try:
        image_content = requests.get(url).content
    except Exception as e:
        print(f"ERROR - COULD NOT DOWNLOAD {url} - {e}")
    try:
        image_file = io.BytesIO(image_content)
        image = Image.open(image_file).convert('RGB')
        file_path = os.path.join(folder_path, file_name)
        with open(file_path, 'wb') as f:
            image.save(f, "JPEG", quality=85)
        print(f"SAVED - {url} - AT: {file_path}")
    except Exception as e:
        print(f"ERROR - COULD NOT SAVE {url} - {e}")

def saveInDestFolder(searchNames,destDir,totalImgs,driver):
```

```
    for name in list(searchNames):
        path=os.path.join(destDir,name)
        if not os.path.isdir(path):
            os.mkdir(path)
        print('Current Path',path)
        totalLinks=getImageUrls(name,totalImgs,driver)
        print('totalLinks',totalLinks)

    if totalLinks is None:
        print('images not found for :',name)

    else:
        for i, link in enumerate(totalLinks):
            file_name = f"{i:150}.jpg"
            downloadImages(path,file_name,link)

searchNames=['cat']
destDir=f'C:\Users\Eleazar\Downloads'
totalImgs=5

saveInDestFolder(searchNames,destDir,totalImgs,driver)
```

```
  Cell In[22], line 81
    destDir=f'C:\Users\Eleazar\Downloads'
                                         ^
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position 2-3: truncated \UXXXXXXXX escape
```

Web Scraping of Movies Information using BeautifulSoup

We want to analyze the distributions of IMDB and Metacritic movie ratings to see if we find anything interesting. To do this, we'll first scrape data for over 2000 movies.

Identifying the URL structure

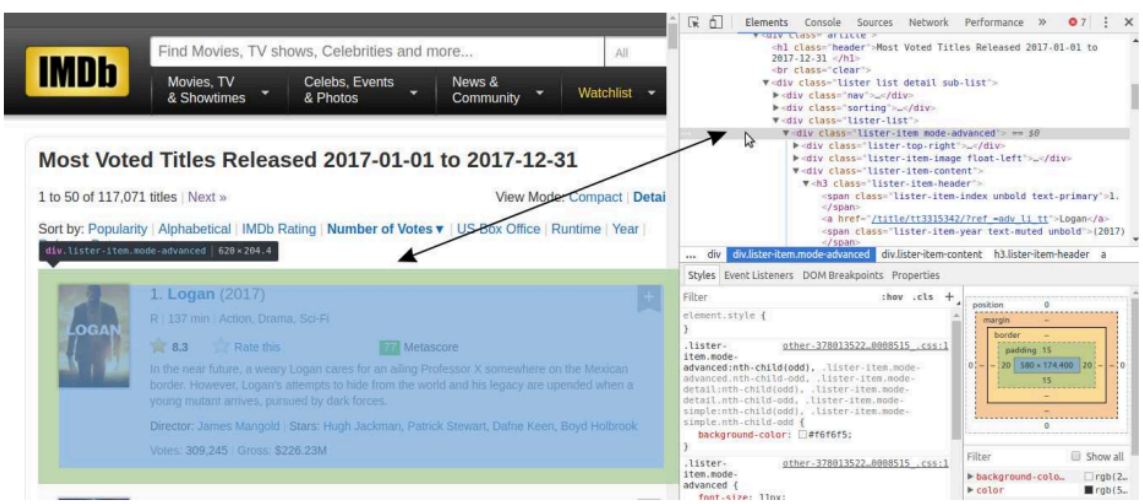
In the image above, you can see that the URL has several parameters after the question mark:

- release_date — Shows only the movies released in a specific year.
- sort — Sorts the movies on the page. sort=num_votes,desc translates to sort by number of votes in a descending order.
- page — Specifies the page number.
- ref_ — Takes us to the the next or the previous page. The reference is the page we are currently on. adv_nxt and adv_prv are two possible values. They translate to advance to the next page, and advance to the previous page, respectivel

In [23]:
```
from requests import get
url = 'https://www.imdb.com/search/title/?release_date=2017-01-01,2017-12-31&sort=n
agent = {"User-Agent": 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.3
response = get(url, headers = agent)
print(response)
print(response.text[:500])
```

```
<Response [200]>
<!DOCTYPE html><html lang="en-US" xmlns:og="http://opengraphprotocol.org/schema/" xm
lns:fb="http://www.facebook.com/2008/fbml"><head><meta charSet="utf-8"/><meta name
="viewport" content="width=device-width"/><script>if(typeof uet === 'function'){ uet
('bb', 'LoadTitle', {wb: 1}); }</script><script>window.addEventListener('load', (eve
nt) => {
        if (typeof window.csa !== 'undefined' && typeof window.csa === 'function') {
            var csaLatencyPlugin = window.csa('Content', {
```

Understanding the HTML structure of a single page



Using BeautifulSoup to parse the HTML content

To parse our HTML document and extract the 50 div containers, we'll use a Python module called BeautifulSoup, the most common web scraping module for Python.

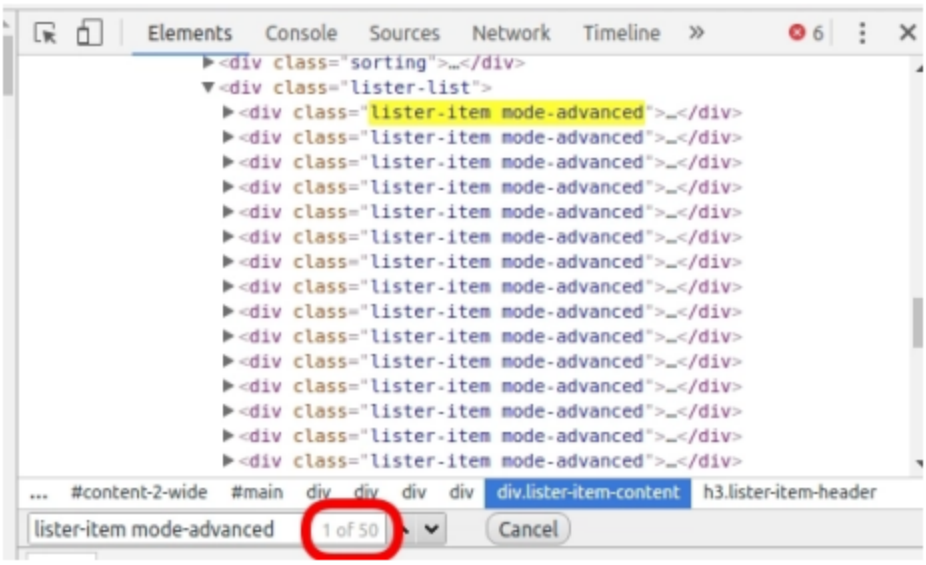
In the following code cell we will:

- Import the BeautifulSoup class creator from the package bs4.
- Parse response.text by creating a BeautifulSoup object, and assign this object to html_soup. The 'html.parser' argument indicates that we want to do the parsing using Python's built-in HTML parser.

In [25]:
```
from bs4 import BeautifulSoup
html_soup = BeautifulSoup(response.text, 'html.parser')
headers = {'Accept-Language': 'en-US,en;q=0.8'}
type(html_soup)
```

Out[25]:
```
bs4.BeautifulSoup
```

Before extracting the 50 div containers, we need to figure out what distinguishes them from other div elements on that page. Often, the distinctive mark resides in the class attribute. If you inspect the HTML lines of the containers of interest, you'll notice that the class attribute has two values: lister-item and mode-advanced. This combination is unique to these div containers. We can see that's true by doing a quick search (Ctrl + F). We have 50 such containers, so we expect to see only 50 matches:



Now let's use the find_all() method to extract all the div containers that have a class attribute of lister-item mode-advanced:

In [27]:
```
movie_containers = html_soup.find_all('li', class_ = 'ipc-metadata-list-summary-ite
print(type(movie_containers))
print(len(movie_containers))
```

```
<class 'bs4.element.ResultSet'>
25
```

find_all() returned a ResultSet object which is a list containing all the 50 divs we are interested in.

Now we'll select only the first container, and extract, by turn, each item of interest:

- The name of the movie.
- The year of release.
- The IMDB rating.
- The Metascore.
- The number of votes.

Extracting the data for a single movie

We can access the first container, which contains information about a single movie, by using list notation on movie_containers.

In [29]:
```
first_movie = movie_containers[0]
first_movie
```

Out[29]:


```
<li class="ipc-metadata-list-summary-item"><div class="ipc-metadata-list-summary-i
tem__c"><div class="ipc-metadata-list-summary-item__tc"><span aria-disabled="fals
e" class="ipc-metadata-list-summary-item__t ipc-btn--not-interactable"></span><div
class="sc-1c782bdc-1 jeRnfh dli-parent"><div class="sc-1c782bdc-0 kZFQUh"><div cla
ss="sc-ee514ad1-0 kYZRWL dli-poster-container"><div class="ipc-poster ipc-poster--
base ipc-poster--media-radius ipc-poster--dynamic-width ipc-sub-grid-item ipc-sub-
grid-item--span-2" role="group"><div aria-label="add to watchlist" class="ipc-watc
hlist-ribbon ipc-focusable ipc-watchlist-ribbon--s ipc-watchlist-ribbon--base ipc-
watchlist-ribbon--loading ipc-watchlist-ribbon--onImage ipc-poster__watchlist-ribb
on" data-testid="poster-watchlist-ribbon-add" role="button" tabindex="0"><svg clas
s="ipc-watchlist-ribbon__bg" height="34px" role="presentation" viewbox="0 0 24 34"
width="24px" xmlns="http://www.w3.org/2000/svg"><polygon class="ipc-watchlist-ribb
on__bg-ribbon" fill="#000000" points="24 0 0 0 0 32 12.2436611 26.2926049 24 31.77
28343"></polygon><polygon class="ipc-watchlist-ribbon__bg-hover" points="24 0 0 0
0 32 12.2436611 26.2926049 24 31.7728343"></polygon><polygon class="ipc-watchlist-
ribbon__bg-shadow" points="24 31.7728343 24 33.7728343 12.2436611 28.2926049 0 34
0 32 12.2436611 26.2926049"></polygon></svg><div class="ipc-watchlist-ribbon__ico
n" role="presentation"><svg class="ipc-loader ipc-loader--circle ipc-watchlist-rib
bon__loader" data-testid="watchlist-ribbon-loader" height="48px" role="presentatio
n" version="1.1" viewbox="0 0 48 48" width="48px" xmlns="http://www.w3.org/2000/sv
g"><g class="ipc-loader__container" fill="currentColor"><circle class="ipc-loader_
_circle ipc-loader__circle--one" cx="24" cy="9" r="4"></circle><circle class="ipc-
loader__circle ipc-loader__circle--two" cx="35" cy="14" r="4"></circle><circle cla
ss="ipc-loader__circle ipc-loader__circle--three" cx="39" cy="24" r="4"></circle><
circle class="ipc-loader__circle ipc-loader__circle--four" cx="35" cy="34" r="4">
</circle><circle class="ipc-loader__circle ipc-loader__circle--five" cx="24" cy="3
9" r="4"></circle><circle class="ipc-loader__circle ipc-loader__circle--six" cx="1
3" cy="34" r="4"></circle><circle class="ipc-loader__circle ipc-loader__circle--se
ven" cx="9" cy="24" r="4"></circle><circle class="ipc-loader__circle ipc-loader__c
ircle--eight" cx="13" cy="14" r="4"></circle></g></svg></div></div><div class="ipc
-media ipc-media--poster-27x40 ipc-image-media-ratio--poster-27x40 ipc-media--medi
a-radius ipc-media--base ipc-media--poster-m ipc-poster__poster-image ipc-media__i
mg" style="width:100%"><img alt="Hugh Jackman in Logan (2017)" class="ipc-image" l
oading="lazy" sizes="50vw, (min-width: 480px) 34vw, (min-width: 600px) 26vw, (min-
width: 1024px) 16vw, (min-width: 1280px) 16vw" src="https://m.media-amazon.com/ima
ges/M/MV5BM2JjODdkMGMtNmY2YS00OGM2LThiY2YtZGYyNzE4Nzc2ODA0XkEyXkFqcGc@._V1_QL75_UX
140_CR0,1,140,207_.jpg" srcset="https://m.media-amazon.com/images/M/MV5BM2JjODdkMG
MtNmY2YS00OGM2LThiY2YtZGYyNzE4Nzc2ODA0XkEyXkFqcGc@._V1_QL75_UX140_CR0,1,140,207_.j
pg 140w, https://m.media-amazon.com/images/M/MV5BM2JjODdkMGMtNmY2YS00OGM2LThiY2YtZ
GYyNzE4Nzc2ODA0XkEyXkFqcGc@._V1_QL75_UX210_CR0,2,210,311_.jpg 210w, https://m.medi
a-amazon.com/images/M/MV5BM2JjODdkMGMtNmY2YS00OGM2LThiY2YtZGYyNzE4Nzc2ODA0XkEyXkFq
cGc@._V1_QL75_UX280_CR0,3,280,414_.jpg 280w" width="140"/></div><a aria-label="Vie
w title page for Logan" class="ipc-lockup-overlay ipc-focusable" href="/title/tt33
15342/?ref_=sr_i_1"><div class="ipc-lockup-overlay__screen"></div></a></div></div>
<div class="sc-2bbfc9e9-0 jUYPWY"><div class="ipc-title ipc-title--base ipc-title-
-title ipc-title-link-no-icon ipc-title--on-textPrimary sc-495ef71a-2 cpvJUg dli-t
itle with-margin"><a class="ipc-title-link-wrapper" href="/title/tt3315342/?ref_=s
r_t_1" tabindex="0"><h3 class="ipc-title__text">1. Logan</h3></a></div><div class
="sc-2bbfc9e9-6 cZkKPy dli-title-metadata"><span class="sc-2bbfc9e9-7 jttFlJ dli-t
itle-metadata-item">2017</span><span class="sc-2bbfc9e9-7 jttFlJ dli-title-metadat
a-item">2h 17m</span><span class="sc-2bbfc9e9-7 jttFlJ dli-title-metadata-item">R-
16</span><span class="sc-2bbfc9e9-9 jMERWw"><span class="sc-b0901df4-0 bXIOoL meta
critic-score-box" style="background-color:#54A72A">77</span><span class="metacriti
c-score-label">Metascore</span></span></div><span class="sc-2bbfc9e9-2 cqpxjX"><di
v class="sc-ddff377f-0 eZxUuR sc-2bbfc9e9-3 kUwRam dli-ratings-container" data-tes
tid="ratingGroup--container"><span aria-label="IMDb rating: 8.1" class="ipc-rating
```

-star ipc-rating-star--base ipc-rating-star--imdb ratingGroup--imdb-rating" data-t
estid="ratingGroup--imdb-rating"><svg class="ipc-icon ipc-icon--star-inline" fill
="currentColor" height="24" role="presentation" viewbox="0 0 24 24" width="24" xml
ns="http://www.w3.org/2000/svg"><path d="M12 20.1l5.82 3.682c1.066.675 2.37-.322
2.09-1.584l-1.543-6.926 5.146-4.667c.94-.85.435-2.465-.799-2.567l-6.773-.602L13.2
9.89a1.38 1.38 0 0 0-2.581 0l-2.65 6.53-6.774.602C.052 8.126-.453 9.74.486 10.59l
5.147 4.666-1.542 6.926c-.28 1.262 1.023 2.26 2.09 1.585L12 20.099z"></path></svg>
8.1<span class="ipc-rating-star--vote
Count"> (<!-- -->886K<!-- -->)<button aria-label="Rate Logan" class
="ipc-rate-button sc-ddff377f-1 iggHKe ratingGroup--user-rating ipc-rate-button--u
nrated ipc-rate-button--base" data-testid="rate-button"><span class="ipc-rating-st
ar ipc-rating-star--base ipc-rating-star--rate"><svg class="ipc-icon ipc-icon--sta
r-border-inline" fill="currentColor" height="24" role="presentation" viewbox="0 0
24 24" width="24" xmlns="http://www.w3.org/2000/svg"><path d="M22.724 8.217l-6.786
-.587-2.65-6.22c-.477-1.133-2.103-1.133-2.58 0l-2.65 6.234-6.772.573c-1.234.098-1.
739 1.636-.8 2.446l5.146 4.446-1.542 6.598c-.28 1.202 1.023 2.153 2.09 1.51l5.818-
3.495 5.819 3.509c1.065.643 2.37-.308 2.089-1.51l-1.542-6.612 5.145-4.446c.94-.81.
45-2.348-.785-2.446zm-10.726 8.89l-5.272 3.174 1.402-5.983-4.655-4.026 6.141-.531
2.384-5.634 2.398 5.648 6.14.531-4.654 4.026 1.402 5.983-5.286-3.187z"></path></sv
g>Rate</button> <button ari
a-disabled="false" aria-label="Mark Logan as watched" aria-pressed="false" class
="ipc-btn ipc-btn--half-padding ipc-btn--left-align-content ipc-btn--default-heigh
t ipc-btn--core-base ipc-btn--theme-base ipc-btn--button-radius ipc-btn--on-accent
2 ipc-text-button sc-b8d371c-0 eazixf" data-testid="inline-watched-button-tt331534
2" tabindex="0"><svg class="ipc-icon ipc-icon--visibility ipc-btn__icon ipc-btn__i
con--pre watched-button--icon ipc-btn__icon--disable-margin" fill="currentColor" h
eight="24" role="presentation" viewbox="0 0 24 24" width="24" xmlns="http://www.w
3.org/2000/svg"><path d="M0 0h24v24H0V0z" fill="none"></path><path d="M12 6c3.79 0
7.17 2.13 8.82 5.5C19.17 14.87 15.79 17 12 17s-7.17-2.13-8.82-5.5C4.83 8.13 8.21 6
12 6m0-2C7 4 2.73 7.11 1 11.5 2.73 15.89 7 19 12 19s9.27-3.11 11-7.5C21.27 7.11 17
4 12 4zm0 5c1.38 0 2.5 1.12 2.5 2.5S13.38 14 12 14s-2.5-1.12-2.5-2.5S10.62 9 12 9m
0-2c-2.48 0-4.5 2.02-4.5 4.5S9.52 16 12 16s4.5-2.02 4.5-4.5S14.48 7 12 7z"></path>
</svg>Mark as watched</button> <div
class="sc-1c782bdc-2 cokvzD dli-post-element"><button aria-disabled="false" aria-l
abel="See more information about Logan" class="ipc-icon-button li-info-icon ipc-ic
on-button--base ipc-icon-button--onAccent2" tabindex="0" title="See more informati
on about Logan"><svg class="ipc-icon ipc-icon--info" fill="currentColor" height="2
4" role="presentation" viewbox="0 0 24 24" width="24" xmlns="http://www.w3.org/200
0/svg"><path d="M0 0h24v24H0V0z" fill="none"></path><path d="M11 7h2v2h-2zm0 4h2v6
h-2zm1-9C6.48 2 2 6.48 2 12s4.48 10 10 10 10-4.48 10-10S17.52 2 12 2zm0 18c-4.41 0
-8-3.59-8-8s3.59-8 8-8 8 3.59 8 8-3.59 8-8 8z"></path></svg></button> <
div class="sc-d49a611d-1 csbzhP"><div class="ipc-html-content ipc-html-content--ba
se sc-d49a611d-0 gKxuCN sttd-plot-container" role="presentation"><div class="ipc-h
tml-content-inner-div" role="presentation">In a future where mutants are nearly ex
tinct, an elderly and weary Logan leads a quiet life. But when Laura, a mutant chi
ld pursued by scientists, comes to him for help, he must get her to safety.

The name of the movie

```
▼<div class="lister-list">
  ▼<div class="lister-item mode-advanced">          ← 1st container
    ▶<div class="lister-top-right">…</div>          ← 1st div
    ▶<div class="lister-item-image float-left">…    ← 2nd div
    </div>
    ▼<div class="lister-item-content">              ← 3rd div
      ▼<h3 class="lister-item-header">              ← <h3>
        <span class="lister-item-index unbold text-
        primary">1.</span>
        <a href="/title/tt3315342/?ref_=adv_li_tt">  ← <a>
        Logan</a> == $0
        <span class="lister-item-year text-muted
        unbold">(2017)</span>
      </h3>
      ▶<p class="text-muted ">…</p>
      ▶<div class="ratings-bar">…</div>
      ▶<p class="text-muted">…</p>
      ▶<p class>…</p>
      ▶<p class="sort-num_votes-visible">…</p>
      </div>
    </div>
  ▶<div class="lister-item mode-advanced">…</div>
  ▶<div class="lister-item mode-advanced">…</div>
  ▶<div class="lister-item mode-advanced">…</div>
  ▶<div class="lister-item mode-advanced">…</div>
```

The other movie containers

In [31]:
```
first_name = first_movie.h3.text
first_name[3:]
```

Out[31]: 'Logan'

The year of the movie's release

In [34]:
```
first_year = movie_containers[0].find('span', class_ = "sc-2bbfc9e9-7 jttFlJ dli-ti
print(first_year)
```

```
<span class="sc-2bbfc9e9-7 jttFlJ dli-title-metadata-item">2017</span>
```

In [36]:
```
first_year = first_year.text
first_year
```

Out[36]: '2017'

The IMDB rating

In [39]:
```
first_imdb = movie_containers[0].find('span', class_ = "ipc-rating-star--rating")
first_imdb.text[:3]
```

Out[39]: '8.1'

The Metascore

In [42]:
```
first_mscore = movie_containers[0].find('span', class_ = 'sc-b0901df4-0 bXIOoL meta
first_mscore = first_mscore.text
print(first_mscore)
```

```
77
```

The number of votes

In [45]:
```
first_votes = movie_containers[0].find('span', class_ = 'ipc-rating-star--voteCount
first_votes.text[2:-1]
```

Out[45]:
```
'886K'
```

The script

In [48]:
```
# Initialize empty lists to store extracted movie details
names = []
years = []
imdb_ratings = []
metascores = []
votes = []

# Loop through each movie container to gather relevant data
for container in movie_containers:
    # Get the movie title, skipping the ranking number
    title = container.find('h3', class_="ipc-title__text").text[3:]
    names.append(title)

    # Extract the release year
    year = container.find('span', class_="sc-2bbfc9e9-7 jttFlJ dli-title-metadata-i
    years.append(year)

    # Extract IMDb rating
    rating = container.find('span', class_="ipc-rating-star ipc-rating-star--base i
    imdb_ratings.append(rating)

    # Extract the Metascore
    score = container.find('span', class_='sc-b0901df4-0 bcQdDJ metacritic-score-bo
    metascores.append(score.text if score else 0)

    # Get the number of votes, trimming extra characters
    vote_count = container.find('span', class_='ipc-rating-star--voteCount').text[2
    votes.append(vote_count)

print(len(names))
print(len(years))
print(len(imdb_ratings))
print(len(metascores))
print(len(votes))
```

```
25
25
25
25
25
```

In [50]:
```python
import pandas as pd
test_df = pd.DataFrame({'movie': names,
'year': years,
'imdb': imdb_ratings,
'metascore': metascores,
'votes': votes
})
print(test_df.info())
test_df
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   movie      25 non-null     object
 1   year       25 non-null     object
 2   imdb       25 non-null     object
 3   metascore  25 non-null     int64
 4   votes      25 non-null     object
dtypes: int64(1), object(4)
memory usage: 1.1+ KB
None
```

Out[50]:

	movie	year	imdb	metascore	votes
0	Logan	2017	8.1	0	886K
1	Thor: Ragnarok	2017	7.9	0	851K
2	Guardians of the Galaxy Vol. 2	2017	7.6	0	791K
3	Dunkirk	2017	7.8	0	770K
4	Get Out	2017	7.8	0	755K
5	Spider-Man: Homecoming	2017	7.4	0	754K
6	Wonder Woman	2017	7.3	0	718K
7	Blade Runner 2049	2017	8.0	0	710K
8	Star Wars: Episode VIII - The Last Jedi	2017	6.9	0	693K
9	Baby Driver	2017	7.5	0	647K
10	Coco	2017	8.4	0	646K
11	It	2017	7.3	0	641K
12	Three Billboards Outside Ebbing, Missouri	2017	8.1	0	581K
13	Money Heist	2017–2021	8.2	0	568K
14	John Wick: Chapter 2	2017	7.4	0	538K
15	Justice League	2017	6.0	0	491K
16	Dark	2017–2020	8.7	0	488K
17	Jumanji: Welcome to the Jungle	2017	7.0	0	476K
18	The Shape of Water	2017	7.3	0	460K
19	Kingsman: The Golden Circle	2017	6.7	0	384K
20	Ozark	2017–2022	8.5	0	368K
21	Mindhunter	2017–2019	8.6	0	365K
22	Pirates of the Caribbean: Salazar's Revenge	2017	6.5	0	364K
23	Kong: Skull Island	2017	6.7	0	362K
24	Lady Bird	2017	7.4	0	350K

The script for multiple pages

In [94]:

```
from time import time
from time import sleep
from requests import get
from random import randint
from IPython.core.display import clear_output
from bs4 import BeautifulSoup
```

```
from IPython.core.display import clear_output
pages = ['1','2','3','4','5']
years_url = ['2017', '2018', '2019', '2020']

# Redeclaring the lists to store data in
names = []
years = []
imdb_ratings = []
metascores = []
votes = []

# Preparing the monitoring of the loop
start_time = time()
requests = 0

# For every year in the interval 2000-2017
for year_url in years_url:

    # For every page in the interval 1-4
    for page in pages:

        # Make a get request
        url = f'https://www.imdb.com/search/title/?release_date={year_url}-01-01,{y
        agent = {"User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKi
        response = get(url,headers = agent)
        print(response.text[:500])
        #response = get('https://www.imdb.com/search/title?release_date=' + year_ur
        #'&sort=num_votes,desc&page=' + page, headers = headers)

        # Pause the loop
        sleep(5)

        # Monitor the requests
        requests += 1
        elapsed_time = time() - start_time
        print('Request:{}; Frequency: {} requests/s'.format(requests, requests/elap
        clear_output(wait = True)

        # Throw a warning for non-200 status codes
        if response.status_code != 200:
            print('Request: {}; Status code: {}'.format(requests, response.status_c

        # Break the loop if the number of requests is greater than expected
        if requests > 25:
            print('Number of requests was greater than expected.')
            break
        # Parse the content of the request with BeautifulSoup
        page_html = BeautifulSoup(response.text, 'html.parser')

        # Select all the 25 movie containers from a single page
        mv_containers = page_html.find_all('div', class_ = 'sc-ab6fa25a-3 bVYfLY dl

        # For every movie of these 25
        for container in mv_containers:
            # If the movie has a Metascore, then:
```

```
            if container.find('span', class_ = 'sc-b0901df4-0 bcQdDJ metacritic-sco
                # Scrape the name
                name = container.find('h3',class_='ipc-title__text').text[3:]
                names.append(name)

                # Scrape the year
                year = container.find('span', class_ = 'sc-b0691f29-8 ilsLEX dli-ti
                years.append(year)

                # Scrape the IMDB rating
                imdb = container.find('span', class_ = 'ipc-rating-star ipc-rating-
                imdb_ratings.append(imdb)

                # Scrape the Metascore
                m_score = container.find('span', class_ = 'sc-b0901df4-0 bcQdDJ met
                metascores.append(m_score)

                # Scrape the number of votes
                vote = container.find('span', class_ = 'ipc-rating-star--voteCount'
                votes.append(vote)
```

```
<!DOCTYPE html><html lang="en-US" xmlns:og="http://opengraphprotocol.org/schema/" xm
lns:fb="http://www.facebook.com/2008/fbml"><head><meta charSet="utf-8"/><meta name
="viewport" content="width=device-width"/><script>if(typeof uet === 'function'){ uet
('bb', 'LoadTitle', {wb: 1}); }</script><script>window.addEventListener('load', (eve
nt) => {
        if (typeof window.csa !== 'undefined' && typeof window.csa === 'function') {
            var csaLatencyPlugin = window.csa('Content', {

Request:20; Frequency: 0.14956094563676797 requests/s
```

In [56]:

```
movie_ratings = pd.DataFrame({'movie': names,
'year': years,
'imdb': imdb_ratings,
'metascore': metascores,
'votes': votes
})
print(movie_ratings.info())
movie_ratings.head(10)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 0 entries
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   movie      0 non-null      float64
 1   year       0 non-null      float64
 2   imdb       0 non-null      float64
 3   metascore  0 non-null      float64
 4   votes      0 non-null      float64
dtypes: float64(5)
memory usage: 132.0 bytes
None
```

Out[56]:

movie	year	imdb	metascore	votes

In [58]:
```
movie_ratings.tail(10)
```

Out[58]:

	movie	year	imdb	metascore	votes

In [60]:
```
movie_ratings.to_csv('/content/drive/My Drive/Colab Notebooks/Dataset/movie_ratings
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
Cell In[60], line 1
----> 1 movie_ratings.to_csv('/content/drive/My Drive/Colab Notebooks/Dataset/movie_
ratings.csv')

File ~\anaconda3\Lib\site-packages\pandas\util\_decorators.py:333, in deprecate_nonk
eyword_arguments.<locals>.decorate.<locals>.wrapper(*args, **kwargs)
    327 if len(args) > num_allow_args:
    328     warnings.warn(
    329         msg.format(arguments=_format_argument_list(allow_args)),
    330         FutureWarning,
    331         stacklevel=find_stack_level(),
    332     )
--> 333 return func(*args, **kwargs)

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:3967, in NDFrame.to_csv(se
lf, path_or_buf, sep, na_rep, float_format, columns, header, index, index_label, mod
e, encoding, compression, quoting, quotechar, lineterminator, chunksize, date_forma
t, doublequote, escapechar, decimal, errors, storage_options)
   3956 df = self if isinstance(self, ABCDataFrame) else self.to_frame()
   3958 formatter = DataFrameFormatter(
   3959     frame=df,
   3960     header=header,
   (...)
   3964     decimal=decimal,
   3965 )
-> 3967 return DataFrameRenderer(formatter).to_csv(
   3968     path_or_buf,
   3969     lineterminator=lineterminator,
   3970     sep=sep,
   3971     encoding=encoding,
   3972     errors=errors,
   3973     compression=compression,
   3974     quoting=quoting,
   3975     columns=columns,
   3976     index_label=index_label,
   3977     mode=mode,
   3978     chunksize=chunksize,
   3979     quotechar=quotechar,
   3980     date_format=date_format,
   3981     doublequote=doublequote,
   3982     escapechar=escapechar,
   3983     storage_options=storage_options,
   3984 )

File ~\anaconda3\Lib\site-packages\pandas\io\formats\format.py:1014, in DataFrameRen
derer.to_csv(self, path_or_buf, encoding, sep, columns, index_label, mode, compressi
on, quoting, quotechar, lineterminator, chunksize, date_format, doublequote, escapec
har, errors, storage_options)
    993     created_buffer = False
    995 csv_formatter = CSVFormatter(
    996     path_or_buf=path_or_buf,
    997     lineterminator=lineterminator,
   (...)
   1012     formatter=self.fmt,
```

```
   1013 )
-> 1014 csv_formatter.save()
   1016 if created_buffer:
   1017     assert isinstance(path_or_buf, StringIO)

File ~\anaconda3\Lib\site-packages\pandas\io\formats\csvs.py:251, in CSVFormatter.sa
ve(self)
    247 """
    248 Create the writer & save.
    249 """
    250 # apply compression and byte/text conversion
--> 251 with get_handle(
    252     self.filepath_or_buffer,
    253     self.mode,
    254     encoding=self.encoding,
    255     errors=self.errors,
    256     compression=self.compression,
    257     storage_options=self.storage_options,
    258 ) as handles:
    259     # Note: self.encoding is irrelevant here
    260     self.writer = csvlib.writer(
    261         handles.handle,
    262         lineterminator=self.lineterminator,
   (...)
    267         quotechar=self.quotechar,
    268     )
    270     self._save()

File ~\anaconda3\Lib\site-packages\pandas\io\common.py:749, in get_handle(path_or_bu
f, mode, encoding, compression, memory_map, is_text, errors, storage_options)
    747 # Only for write methods
    748 if "r" not in mode and is_path:
--> 749     check_parent_directory(str(handle))
    751 if compression:
    752     if compression != "zstd":
    753         # compression libraries do not like an explicit text-mode

File ~\anaconda3\Lib\site-packages\pandas\io\common.py:616, in check_parent_director
y(path)
    614 parent = Path(path).parent
    615 if not parent.is_dir():
--> 616     raise OSError(rf"Cannot save file into a non-existent directory: '{paren
t}'")

OSError: Cannot save file into a non-existent directory: '\content\drive\My Drive\Co
lab Notebooks\Dataset'
```

Data Preparation

- Collected data may not be compatible or formatted correctly
- Data must be prepared before it can be added to a data set
- Extract, Transform and Load (ETL), the process for collecting data from a variety of sources, transforming the data, and then loading the data into a database

Data preprocessing

Data Processing is a process of cleaning the raw data i.e. the data is collected in the real world and is converted to a clean data set. In other words, whenever the data is gathered from different sources it is collected in a raw format and this data isn't feasible for the analysis. Therefore, certain steps are executed to convert the data into a small clean data set, this part of the process is called as data preprocessing.

Most of the real-world data is messy, some of these types of data are:

1. Missing data: Missing data can be found when it is not continuously created or due to technical issues in the application (IOT system).
2. Noisy Data This type of data is also called outliners, this can occur due to human errors (human manually gathering the data) or some technical problem of the device at the time of collection of data.
3. Inconsistent data: This type of data might be collected due to human errors (mistakes with the name or values) or duplication of data.

These are some of the basic pre processing techniques that can be used to convert raw data.

1. Conversion of data: As we know that Machine Learning models can only handle numeric features, hence categorical and ordinal data must be somehow converted into numeric features.
2. Ignoring the missing values: Whenever we encounter missing data in the data set then we can remove the row or column of data depending on our need. This method is known to be efficient but it shouldn't be performed if there are a lot of missing values in the dataset.
3. Filling the missing values: Whenever we encounter missing data in the data set then we can fill the missing data manually, most commonly the mean, median or highest frequency value is used.
4. Machine learning: If we have some missing data then we can predict what data shall be present at the empty position by using the existing data.
5. Outliers detection: There are some error data that might be present in our data set that deviates drastically from other observations in a data set. [Example: human weight = 800

Kg; due to mistyping of extra 0]

Example of Data Preparation of movie_rating.csv

In [ ]:
```
movie_ratings['year'].unique()
```

In [ ]:
```
movie_ratings.dtypes
```

In [ ]:
```
movie_ratings['year'] = (movie_ratings.year.apply(lambda x:x.replace('(I)','')))
```

In [ ]:
```
movie_ratings['year'].unique()
```

In [66]:
```
movie_ratings['year'] = (movie_ratings.year.apply(lambda x:x.replace('(II)','')))
```

In [68]:
```
movie_ratings['year'] = (movie_ratings.year.apply(lambda x:x.replace('(III)','')))
```

In [70]:
```
movie_ratings['year'].unique()
```

Out[70]:
```
array([], dtype=float64)
```

In [72]:
```
movie_ratings['year'] = (movie_ratings.year.apply(lambda x:x.replace('(','')))
```

In [74]:
```
movie_ratings['year'].unique()
```

Out[74]:
```
array([], dtype=float64)
```

In [76]:
```
movie_ratings['year'] = (movie_ratings.year.apply(lambda x:x.replace(')','')))
```

In [78]:
```
movie_ratings['year'].unique()
```

Out[78]:
```
array([], dtype=float64)
```

In [80]:
```
movie_ratings['year'] = movie_ratings['year'].astype(int)
```

In [82]:
```
movie_ratings['year'].unique()
```

Out[82]:
```
array([], dtype=int32)
```

In [84]:
```
movie_ratings.dtypes
```

Out[84]:
```
movie        float64
year           int32
imdb         float64
metascore    float64
votes        float64
dtype: object
```

In [86]:
```
movie_ratings.head(10)
```

Out[86]:

movie	year	imdb	metascore	votes

In [88]:
```
movie_ratings.tail(10)
```

Out[88]:

movie	year	imdb	metascore	votes

In [90]:
```
movie_ratings
```

Out[90]:

movie	year	imdb	metascore	votes

In [ ]: