1. Scrape the details of most viewed videos on YouTube from Wikipedia. Url = https://en.wikipedia.org/wiki/List_of_most-viewed_YouTube_videos (https://en.wikipedia.org/wiki/List_of_most-viewed_YouTube_videos) You need to find following details: A) Rank

B) Name C) Artist D) Upload date E) Views

In [1]:
```
import requests
from bs4 import BeautifulSoup

# URL of the Wikipedia page to scrape
url = "https://en.wikipedia.org/wiki/List_of_most-viewed_YouTube_videos"

# Sending a request to the webpage
response = requests.get(url)

# Parsing the HTML content
soup = BeautifulSoup(response.content, 'html.parser')

# Finding the table that contains the most viewed YouTube videos
table = soup.find('table', {'class': 'wikitable'})

# Initializing lists to hold the scraped data
ranks = []
names = []
artists = []
upload_dates = []
views = []

# Extracting data from the table rows
for row in table.find_all('tr')[1:]:  # Skipping the header row
    columns = row.find_all('td')
    if len(columns) >= 5:
        ranks.append(columns[0].text.strip())
        names.append(columns[1].text.strip())
        artists.append(columns[2].text.strip())
        upload_dates.append(columns[4].text.strip())
        views.append(columns[3].text.strip())

# Creating a dictionary to hold the scraped data
data = {
    'Rank': ranks,
    'Name': names,
    'Artist': artists,
    'Upload date': upload_dates,
    'Views': views
}

# Printing the data
for i in range(len(ranks)):
    print(f"Rank: {ranks[i]}, Name: {names[i]}, Artist: {artists[i]}, Upload d
```

```
Rank: "Baby Shark Dance"[7], Name: Pinkfong Baby Shark - Kids' Songs & Storie
s, Artist: 14.82, Upload date: [A], Views: June 17, 2016
Rank: "Despacito"[10], Name: Luis Fonsi, Artist: 8.49, Upload date: [B], View
s: January 12, 2017
Rank: "Johny Johny Yes Papa"[18], Name: LooLoo Kids - Nursery Rhymes and Chil
dren's Songs, Artist: 6.94, Upload date: , Views: October 8, 2016
Rank: "Bath Song"[19], Name: Cocomelon - Nursery Rhymes, Artist: 6.79, Upload
date: , Views: May 2, 2018
Rank: "Wheels on the Bus"[20], Name: Cocomelon - Nursery Rhymes, Artist: 6.3
4, Upload date: , Views: May 24, 2018
Rank: "See You Again"[21], Name: Wiz Khalifa, Artist: 6.33, Upload date: [C],
Views: April 6, 2015
Rank: "Shape of You"[26], Name: Ed Sheeran, Artist: 6.30, Upload date: [D], V
iews: January 30, 2017
Rank: "Phonics Song with Two Words"[29], Name: ChuChu TV Nursery Rhymes & Kid
s Songs, Artist: 5.90, Upload date: , Views: March 6, 2014
Rank: "Uptown Funk"[30], Name: Mark Ronson, Artist: 5.28, Upload date: , View
s: November 19, 2014
Rank: "Gangnam Style"[31], Name: Psy, Artist: 5.22, Upload date: [E], Views:
July 15, 2012
Rank: "Learning Colors – Colorful Eggs on a Farm"[36], Name: Miroshka TV, Art
ist: 5.15, Upload date: , Views: February 27, 2018
Rank: "Dame Tu Cosita"[37], Name: Ultra Records, Artist: 4.72, Upload date: ,
Views: April 5, 2018
Rank: "Axel F"[38], Name: Crazy Frog, Artist: 4.66, Upload date: , Views: Jun
e 16, 2009
Rank: "Masha and the Bear – Recipe for Disaster"[39], Name: Get Movies, Artis
t: 4.59, Upload date: , Views: January 31, 2012
Rank: "Baa Baa Black Sheep"[40], Name: Cocomelon - Nursery Rhymes, Artist: 4.
14, Upload date: , Views: June 25, 2018
Rank: "Lakdi Ki Kathi"[41], Name: Jingle Toons, Artist: 4.11, Upload date: ,
Views: June 14, 2018
Rank: "Sugar"[42], Name: Maroon 5, Artist: 4.08, Upload date: , Views: Januar
y 14, 2015
Rank: "Counting Stars"[43], Name: OneRepublic, Artist: 4.05, Upload date: , V
iews: May 31, 2013
Rank: "Roar"[44], Name: Katy Perry, Artist: 4.03, Upload date: , Views: Septe
mber 5, 2013
Rank: "Waka Waka (This Time for Africa)"[45], Name: Shakira, Artist: 3.99, Up
load date: , Views: June 4, 2010
Rank: "Shree Hanuman Chalisa"[46], Name: T-Series Bhakti Sagar, Artist: 3.94,
Upload date: , Views: May 10, 2011
Rank: "Humpty the train on a fruits ride"[47], Name: Kiddiestv Hindi - Nurser
y Rhymes & Kids Songs, Artist: 3.87, Upload date: , Views: January 26, 2018
Rank: "Sorry"[48], Name: Justin Bieber, Artist: 3.83, Upload date: , Views: O
ctober 22, 2015
Rank: "Thinking Out Loud"[49], Name: Ed Sheeran, Artist: 3.79, Upload date: ,
Views: October 7, 2014
Rank: "Perfect"[50], Name: Ed Sheeran, Artist: 3.77, Upload date: , Views: No
vember 9, 2017
Rank: "Dark Horse"[51], Name: Katy Perry, Artist: 3.76, Upload date: , Views:
February 20, 2014
Rank: "Let Her Go"[52], Name: Passenger, Artist: 3.69, Upload date: , Views:
July 25, 2012
Rank: "Faded"[53], Name: Alan Walker, Artist: 3.66, Upload date: , Views: Dec
ember 3, 2015
Rank: "Girls Like You"[54], Name: Maroon 5, Artist: 3.64, Upload date: , View
```

```
s: May 31, 2018
Rank: "Lean On"[55], Name: Major Lazer Official, Artist: 3.64, Upload date: ,
Views: March 22, 2015
```

2. Scrape the details team India’s international fixtures from bcci.tv.

Url = https://www.bcci.tv/ (https://www.bcci.tv/). You need to find following details: A) Series B) Place C) Date D) Time Note: - From bcci.tv home page you have reach to the international fixture page through code.

In [3]:
```
import requests
from bs4 import BeautifulSoup

# Step 1: Send a request to the BCCI homepage
homepage_url = 'https://www.bcci.tv/'
homepage_response = requests.get(homepage_url)

# Step 2: Parse the homepage to find the link to the international fixtures pag
homepage_soup = BeautifulSoup(homepage_response.content, 'html.parser')
fixture_page_link = homepage_soup.find('a', string='International Fixtures')['

# Step 3: Send a request to the international fixtures page
fixture_page_url = homepage_url.rstrip('/') + fixture_page_link
fixture_page_response = requests.get(fixture_page_url)

# Step 4: Parse the international fixtures page
fixture_page_soup = BeautifulSoup(fixture_page_response.content, 'html.parser'
fixtures = fixture_page_soup.find_all('div', class_='fixture-item')

# Step 5: Extract the required details from the fixtures
series_list = []
places_list = []
dates_list = []
times_list = []

for fixture in fixtures:
    series = fixture.find('div', class_='fixture-item__title').text.strip()
    place = fixture.find('span', class_='fixture-item__place').text.strip()
    datetime = fixture.find('div', class_='fixture-item__datetime').text.strip
    date, time = datetime.split('|')

    series_list.append(series)
    places_list.append(place)
    dates_list.append(date.strip())
    times_list.append(time.strip())

# Step 6: Print the extracted details
for i in range(len(series_list)):
    print(f"Series: {series_list[i]}, Place: {places_list[i]}, Date: {dates_li
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[3], line 10
      8 # Step 2: Parse the homepage to find the link to the international fi
xtures page
      9 homepage_soup = BeautifulSoup(homepage_response.content, 'html.parse
r')
---> 10 fixture_page_link = homepage_soup.find('a', string='International Fix
tures')['href']
     12 # Step 3: Send a request to the international fixtures page
     13 fixture_page_url = homepage_url.rstrip('/') + fixture_page_link

TypeError: 'NoneType' object is not subscriptable
```

Q.3 Scrape the details of State-wise GDP of India from statisticstime.com.

Url = http://statisticstimes.com/ (http://statisticstimes.com/) You have to find following details: A) Rank B) State C) GSDP(18-19)- at current prices D) GSDP(19-20)- at current prices E) Share(18-19) F) GDP($ billion) Note: - From statisticstimes home page you have to reach to economy page through code.

In [5]:
```
import requests
from bs4 import BeautifulSoup

# Base URL of Statistics Times
base_url = "http://statisticstimes.com/"

# Function to get the full URL
def get_full_url(path):
    return base_url + path

# Step 1: Load the Statistics Times homepage
response = requests.get(base_url)
soup = BeautifulSoup(response.content, 'html.parser')

# Step 2: Find the link to the economy page
economy_link = None
for a in soup.find_all('a'):
    if 'economy' in a['href']:
        economy_link = a['href']
        break

# Step 3: Navigate to the economy page
economy_url = get_full_url(economy_link)
response = requests.get(economy_url)
soup = BeautifulSoup(response.content, 'html.parser')

# Step 4: Find the link to the State-wise GDP page
gdp_link = None
for a in soup.find_all('a'):
    if 'india/indian-states-gdp.php' in a['href']:
        gdp_link = a['href']
        break

# Step 5: Navigate to the State-wise GDP page
gdp_url = get_full_url(gdp_link)
response = requests.get(gdp_url)
soup = BeautifulSoup(response.content, 'html.parser')

# Step 6: Extract the required details from the table
table = soup.find('table', {'id': 'table_id'})
rows = table.find_all('tr')

# Initializing lists to hold the scraped data
ranks = []
states = []
gsdp_18_19 = []
gsdp_19_20 = []
shares_18_19 = []
gdp_billions = []

# Loop through each row and extract the details
for row in rows[1:]:  # Skipping the header row
    columns = row.find_all('td')
    ranks.append(columns[0].text.strip())
    states.append(columns[1].text.strip())
    gsdp_18_19.append(columns[2].text.strip())
    gsdp_19_20.append(columns[3].text.strip())
```

```
    shares_18_19.append(columns[4].text.strip())
    gdp_billions.append(columns[5].text.strip())

# Creating a dictionary to hold the scraped data
data = {
    'Rank': ranks,
    'State': states,
    'GSDP(18-19) at current prices': gsdp_18_19,
    'GSDP(19-20) at current prices': gsdp_19_20,
    'Share(18-19)': shares_18_19,
    'GDP($ billion)': gdp_billions
}

# Printing the data
for i in range(len(ranks)):
    print(f"Rank: {ranks[i]}, State: {states[i]}, GSDP(18-19): {gsdp_18_19[i]}
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[5], line 35
     32         break
     34 # Step 5: Navigate to the State-wise GDP page
---> 35 gdp_url = get_full_url(gdp_link)
     36 response = requests.get(gdp_url)
     37 soup = BeautifulSoup(response.content, 'html.parser')

Cell In[5], line 9, in get_full_url(path)
      8 def get_full_url(path):
----> 9     return base_url + path

TypeError: can only concatenate str (not "NoneType") to str
```

4. Scrape the details of trending repositories on Github.com.

Url = https://github.com/ (https://github.com/) You have to find the following details: A) Repository title B) Repository description C) Contributors count D) Language used Note: - From the home page you have to click on the trending option from Explore menu through code.

In [1]:
```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time
from bs4 import BeautifulSoup

# Set up the Selenium WebDriver (make sure you have downloaded the WebDriver ex
driver = webdriver.Chrome()

# Open GitHub homepage
driver.get("https://github.com/")

# Wait for the Explore menu to be clickable and click on it
explore_menu = WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.XPATH, '//*[@aria-label="Explore"]'))
)
explore_menu.click()

# Wait for the Trending option to be visible and click on it
trending_option = WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Trending"))
)
trending_option.click()

# Wait for the Trending page to load
time.sleep(3)

# Parse the page with BeautifulSoup
soup = BeautifulSoup(driver.page_source, 'html.parser')

# Find all trending repositories
repositories = soup.find_all('article', class_='Box-row')

# Extract the required details
for repo in repositories:
    title_tag = repo.find('h1', class_='h3 lh-condensed')
    title = title_tag.text.strip() if title_tag else "N/A"

    description_tag = repo.find('p', class_='col-9 color-fg-muted my-1 pr-4')
    description = description_tag.text.strip() if description_tag else "No des

    language_tag = repo.find('span', itemprop='programmingLanguage')
    language = language_tag.text.strip() if language_tag else "No language spe

    contributors_tag = repo.find_all('a', class_='Link--muted d-inline-block m
    contributors = len(contributors_tag)

    print(f"Repository Title: {title}")
    print(f"Description: {description}")
    print(f"Language: {language}")
    print(f"Contributors: {contributors}")
    print("-" * 40)

# Close the browser
```

```
driver.quit()
```

```
---------------------------------------------------------------------------
TimeoutException                              Traceback (most recent call last)
Cell In[1], line 17
     14 driver.get("https://github.com/")
     16 # Wait for the Explore menu to be clickable and click on it
---> 17 explore_menu = WebDriverWait(driver, 10).until(
     18     EC.element_to_be_clickable((By.XPATH, '//*[@aria-label="Explor
e"]'))
     19 )
     20 explore_menu.click()
     22 # Wait for the Trending option to be visible and click on it

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\selenium\web
driver\support\wait.py:95, in WebDriverWait.until(self, method, message)
     93     if time.monotonic() > end_time:
     94         break
---> 95 raise TimeoutException(message, screen, stacktrace)

TimeoutException: Message:
Stacktrace:
        GetHandleVerifier [0x00007FF64FAA9632+30946]
        (No symbol) [0x00007FF64FA5E3C9]
        (No symbol) [0x00007FF64F956FDA]
        (No symbol) [0x00007FF64F9A822C]
        (No symbol) [0x00007FF64F9A850C]
        (No symbol) [0x00007FF64F9EDCB7]
        (No symbol) [0x00007FF64F9CCAAF]
        (No symbol) [0x00007FF64F9EB041]
        (No symbol) [0x00007FF64F9CC813]
        (No symbol) [0x00007FF64F99A6E5]
        (No symbol) [0x00007FF64F99B021]
        GetHandleVerifier [0x00007FF64FBDF83D+1301229]
        GetHandleVerifier [0x00007FF64FBEBDB7+1351783]
        GetHandleVerifier [0x00007FF64FBE2A03+1313971]
        GetHandleVerifier [0x00007FF64FADDD06+245686]
        (No symbol) [0x00007FF64FA6758F]
        (No symbol) [0x00007FF64FA63804]
        (No symbol) [0x00007FF64FA63992]
        (No symbol) [0x00007FF64FA5A3EF]
        BaseThreadInitThunk [0x00007FFFDA117C24+20]
        RtlUserThreadStart [0x00007FFFDA26D4D1+33]
```

5. Scrape the details of top 100 songs on billiboard.com. Url = https:/www.billboard.com/ (http://www.billboard.com/) You have to find the following details:

A) Song name B) Artist name C) Last week rank D) Peak rank E) Weeks on board Note: - From the home page you have to click on the charts option then hot 100-page link through code.

In [2]:
```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from bs4 import BeautifulSoup
import time

# Initialize the Selenium WebDriver
driver = webdriver.Chrome()  # Make sure to have the ChromeDriver in your PATH

# Open Billboard homepage
driver.get("https://www.billboard.com/")

# Wait for the Charts menu to be clickable and click on it
charts_menu = WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Charts"))
)
charts_menu.click()

# Wait for the Hot 100 page link to be visible and click on it
hot_100_link = WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Hot 100"))
)
hot_100_link.click()

# Wait for the Hot 100 page to load completely
time.sleep(5)

# Parse the page with BeautifulSoup
soup = BeautifulSoup(driver.page_source, 'html.parser')

# Find all songs in the Hot 100 list
songs = soup.find_all('li', class_='o-chart-results-list__item')

# Extract the required details
for song in songs:
    song_name_tag = song.find('h3', id="title-of-a-story")
    song_name = song_name_tag.get_text(strip=True) if song_name_tag else "N/A"

    artist_name_tag = song.find('span', class_='c-label')
    artist_name = artist_name_tag.get_text(strip=True) if artist_name_tag else

    last_week_rank_tag = song.find('span', class_='c-label', text='Last Week')
    last_week_rank = last_week_rank_tag.find_next('span').get_text(strip=True)

    peak_rank_tag = song.find('span', class_='c-label', text='Peak Position')
    peak_rank = peak_rank_tag.find_next('span').get_text(strip=True) if peak_r

    weeks_on_board_tag = song.find('span', class_='c-label', text='Weeks on Ch
    weeks_on_board = weeks_on_board_tag.find_next('span').get_text(strip=True)

    print(f"Song Name: {song_name}")
    print(f"Artist Name: {artist_name}")
    print(f"Last Week Rank: {last_week_rank}")
    print(f"Peak Rank: {peak_rank}")
    print(f"Weeks on Board: {weeks_on_board}")
    print("-" * 40)
```

```
# Close the browser
driver.quit()
```

```
---------------------------------------------------------------------------
ElementClickInterceptedException          Traceback (most recent call last)
Cell In[2], line 18
     14 # Wait for the Charts menu to be clickable and click on it
     15 charts_menu = WebDriverWait(driver, 10).until(
     16     EC.element_to_be_clickable((By.LINK_TEXT, "Charts"))
     17 )
---> 18 charts_menu.click()
     20 # Wait for the Hot 100 page link to be visible and click on it
     21 hot_100_link = WebDriverWait(driver, 10).until(
     22     EC.element_to_be_clickable((By.LINK_TEXT, "Hot 100"))
     23 )

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\selenium\web
driver\remote\webelement.py:93, in WebElement.click(self)
     91 def click(self) -> None:
     92     """Clicks the element."""
---> 93     self._execute(Command.CLICK_ELEMENT)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\selenium\web
driver\remote\webelement.py:394, in WebElement._execute(self, command, param
s)
    392     params = {}
    393 params["id"] = self._id
--> 394 return self._parent.execute(command, params)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\selenium\web
driver\remote\webdriver.py:344, in WebDriver.execute(self, driver_command, pa
rams)
    342 response = self.command_executor.execute(driver_command, params)
    343 if response:
--> 344     self.error_handler.check_response(response)
    345     response["value"] = self._unwrap_value(response.get("value", Non
e))
    346     return response

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\selenium\web
driver\remote\errorhandler.py:229, in ErrorHandler.check_response(self, respo
nse)
    227         alert_text = value["alert"].get("text")
    228     raise exception_class(message, screen, stacktrace, alert_text)  #
type: ignore[call-arg]  # mypy is not smart enough here
--> 229 raise exception_class(message, screen, stacktrace)

ElementClickInterceptedException: Message: element click intercepted: Element
is not clickable at point (639, 10934)
  (Session info: chrome=127.0.6533.72)
Stacktrace:
        GetHandleVerifier [0x00007FF64FAA9632+30946]
        (No symbol) [0x00007FF64FA5E3C9]
        (No symbol) [0x00007FF64F956FDA]
        (No symbol) [0x00007FF64F9AFEEE]
        (No symbol) [0x00007FF64F9AD962]
        (No symbol) [0x00007FF64F9AAE7B]
        (No symbol) [0x00007FF64F9AA095]
        (No symbol) [0x00007FF64F99C271]
        (No symbol) [0x00007FF64F9CCA6A]
```

```
(No symbol) [0x00007FF64F99BBB6]
(No symbol) [0x00007FF64F9CCC80]
(No symbol) [0x00007FF64F9EB041]
(No symbol) [0x00007FF64F9CC813]
(No symbol) [0x00007FF64F99A6E5]
(No symbol) [0x00007FF64F99B021]
GetHandleVerifier [0x00007FF64FBDF83D+1301229]
GetHandleVerifier [0x00007FF64FBEBDB7+1351783]
GetHandleVerifier [0x00007FF64FBE2A03+1313971]
GetHandleVerifier [0x00007FF64FADDD06+245686]
(No symbol) [0x00007FF64FA6758F]
(No symbol) [0x00007FF64FA63804]
(No symbol) [0x00007FF64FA63992]
(No symbol) [0x00007FF64FA5A3EF]
BaseThreadInitThunk [0x00007FFFDA117C24+20]
RtlUserThreadStart [0x00007FFFDA26D4D1+33]
```

6. Scrape the details of Highest selling novels.

A) Book name B) Author name C) Volumes sold D) Publisher E) Genre Url - https://www.theguardian.com/news/datablog/2012/aug/09/best-selling-books-all-time-fifty-shades-grey-compare (https://www.theguardian.com/news/datablog/2012/aug/09/best-selling-books-all-time-fifty-shades-grey-compare)

In [3]:
```
import requests
from bs4 import BeautifulSoup

# URL of the page to scrape
url = 'https://www.theguardian.com/news/datablog/2012/aug/09/best-selling-books

# Fetch the HTML content of the page
response = requests.get(url)
soup = BeautifulSoup(response.content, 'html.parser')

# Extract the table containing the book details
table = soup.find('table')

# Extract the table headers
headers = [header.text.strip() for header in table.find_all('th')]

# Extract the table rows
rows = table.find_all('tr')

# Loop through each row and extract the details
for row in rows[1:]:  # Skip the header row
    cells = row.find_all('td')
    if len(cells) == 5:
        book_name = cells[0].text.strip()
        author_name = cells[1].text.strip()
        volumes_sold = cells[2].text.strip()
        publisher = cells[3].text.strip()
        genre = cells[4].text.strip()

        print(f"Book Name: {book_name}")
        print(f"Author Name: {author_name}")
        print(f"Volumes Sold: {volumes_sold}")
        print(f"Publisher: {publisher}")
        print(f"Genre: {genre}")
        print("-" * 40)
```

7. Scrape the details most watched tv series of all time from imdb.com.

Url = https://www.imdb.com/list/ls095964455/ (https://www.imdb.com/list/ls095964455/) You have to find the following details: A) Name B) Year span C) Genre D) Run time E) Ratings F) Votes

In [5]:
```
import requests
from bs4 import BeautifulSoup

# URL of the IMDb page to scrape
url = 'https://www.imdb.com/list/ls095964455/'

# Fetch the HTML content of the page
response = requests.get(url)
soup = BeautifulSoup(response.content, 'html.parser')

# Find the container with all the TV series
containers = soup.find_all('div', class_='lister-item mode-detail')

# Loop through each container and extract the required details
for container in containers:
    # Extract the name of the TV series
    name_tag = container.find('h3', class_='lister-item-header')
    name = name_tag.find('a').text.strip()

    # Extract the year span of the TV series
    year_span = name_tag.find('span', class_='lister-item-year').text.strip()

    # Extract the genre of the TV series
    genre_tag = container.find('span', class_='genre')
    genre = genre_tag.text.strip() if genre_tag else "N/A"

    # Extract the run time of the TV series
    run_time_tag = container.find('span', class_='runtime')
    run_time = run_time_tag.text.strip() if run_time_tag else "N/A"

    # Extract the ratings of the TV series
    ratings_tag = container.find('span', class_='ipl-rating-star__rating')
    ratings = ratings_tag.text.strip() if ratings_tag else "N/A"

    # Extract the votes of the TV series
    votes_tag = container.find('span', attrs={'name': 'nv'})
    votes = votes_tag.text.strip() if votes_tag else "N/A"

    print(f"Name: {name}")
    print(f"Year Span: {year_span}")
    print(f"Genre: {genre}")
    print(f"Run Time: {run_time}")
    print(f"Ratings: {ratings}")
    print(f"Votes: {votes}")
    print("-" * 40)
```

8. Details of Datasets from UCI machine learning repositories.

Url = https://archive.ics.uci.edu/ (https://archive.ics.uci.edu/) You have to find the following details: A) Dataset name B) Data type C) Task D) Attribute type E) No of instances F) No of attribute G) Year Note: - from the home page you have to go to the Show All Dataset page through code.

In [6]:
```
import requests
from bs4 import BeautifulSoup

# URL of the UCI Machine Learning Repository homepage
url = 'https://archive.ics.uci.edu/'

# Fetch the HTML content of the homepage
response = requests.get(url)
soup = BeautifulSoup(response.content, 'html.parser')

# Find the link to the "Show All Dataset" page
show_all_datasets_link = soup.find('a', text='View ALL Data Sets')['href']
all_datasets_url = url + show_all_datasets_link

# Fetch the HTML content of the "Show All Dataset" page
response = requests.get(all_datasets_url)
soup = BeautifulSoup(response.content, 'html.parser')

# Find the table containing the dataset details
table = soup.find('table', {'border': '1'})

# Loop through each row in the table (excluding the header row) and extract the
for row in table.find_all('tr')[1:]:
    cells = row.find_all('td')
    if len(cells) == 7:
        dataset_name = cells[0].text.strip()
        data_type = cells[1].text.strip()
        task = cells[2].text.strip()
        attribute_type = cells[3].text.strip()
        no_of_instances = cells[4].text.strip()
        no_of_attributes = cells[5].text.strip()
        year = cells[6].text.strip()

        print(f"Dataset Name: {dataset_name}")
        print(f"Data Type: {data_type}")
        print(f"Task: {task}")
        print(f"Attribute Type: {attribute_type}")
        print(f"No of Instances: {no_of_instances}")
        print(f"No of Attributes: {no_of_attributes}")
        print(f"Year: {year}")
        print("-" * 40)
```

```
C:\Users\99Minds-1\AppData\Local\Temp\ipykernel_12244\2416075011.py:12: Depre
cationWarning: The 'text' argument to find()-type methods is deprecated. Use
'string' instead.
  show_all_datasets_link = soup.find('a', text='View ALL Data Sets')['href']
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[6], line 12
      9 soup = BeautifulSoup(response.content, 'html.parser')
     11 # Find the link to the "Show All Dataset" page
---> 12 show_all_datasets_link = soup.find('a', text='View ALL Data Sets')['h
ref']
     13 all_datasets_url = url + show_all_datasets_link
     15 # Fetch the HTML content of the "Show All Dataset" page

TypeError: 'NoneType' object is not subscriptable
```

In [ ]: