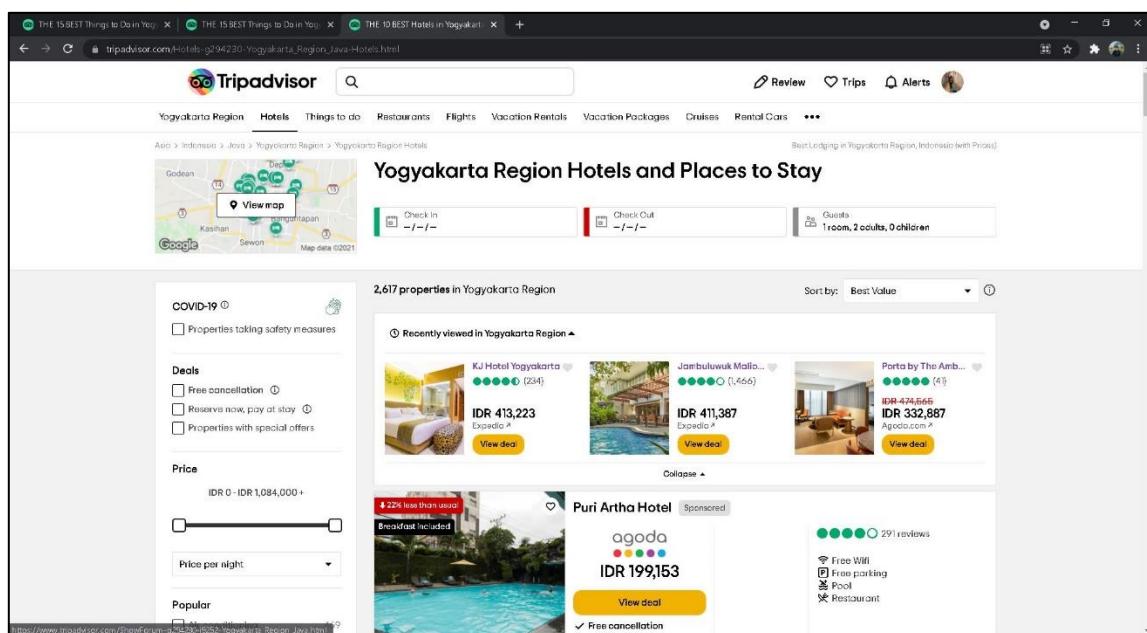


## Modul 3 (Scraping Tripadvisor dan IMDB Movie)

### Scraping Tripadvisor

Kali ini kita akan melakukan *scraping* situs Tripadvisor untuk mendapatkan data-data seperti daftar hotel yang ada di Yogyakarta dan kegiatan atau wisata yang ada di Yogyakarta. Scraping ini akan menggunakan *library* python BeautifulSoup. Yang pertama akan dilakukan adalah melakukan scraping daftar hotel yang ada di Yogyakarta.

Yang pertama adalah, buka situs <https://www.tripadvisor.com> kemudian pilih Hotel dan masukkan tempat “Yogyakarta Region”, atau dapat menggunakan link berikut [https://www.tripadvisor.com/Hotels-g294230-Yogyakarta\\_Region\\_Java-Hotels.html](https://www.tripadvisor.com/Hotels-g294230-Yogyakarta_Region_Java-Hotels.html). Maka tampilan web tripadvisor adalah seperti gambar 1.0 di bawah ini.



Gambar 1. 1 tampilan website tripadvisor

Kemudian buka IDE, dapat menggunakan *Google Collab* atau menggunakan *Visual Studio Code*. Di sini saya menggunakan *Visual Studio Code*. Kemudian setelah itu *import* beberapa *library* yang akan digunakan seperti *requests*, dan *BeautifulSoup*.

```
import requests  
from bs4 import BeautifulSoup  
import csv
```

kemudian, buatlah variabel yang berfungsi untuk menampung situs url, *requests*, objek soup, dan list kosong untuk menampung data. Di sini saya menggunakan varianel

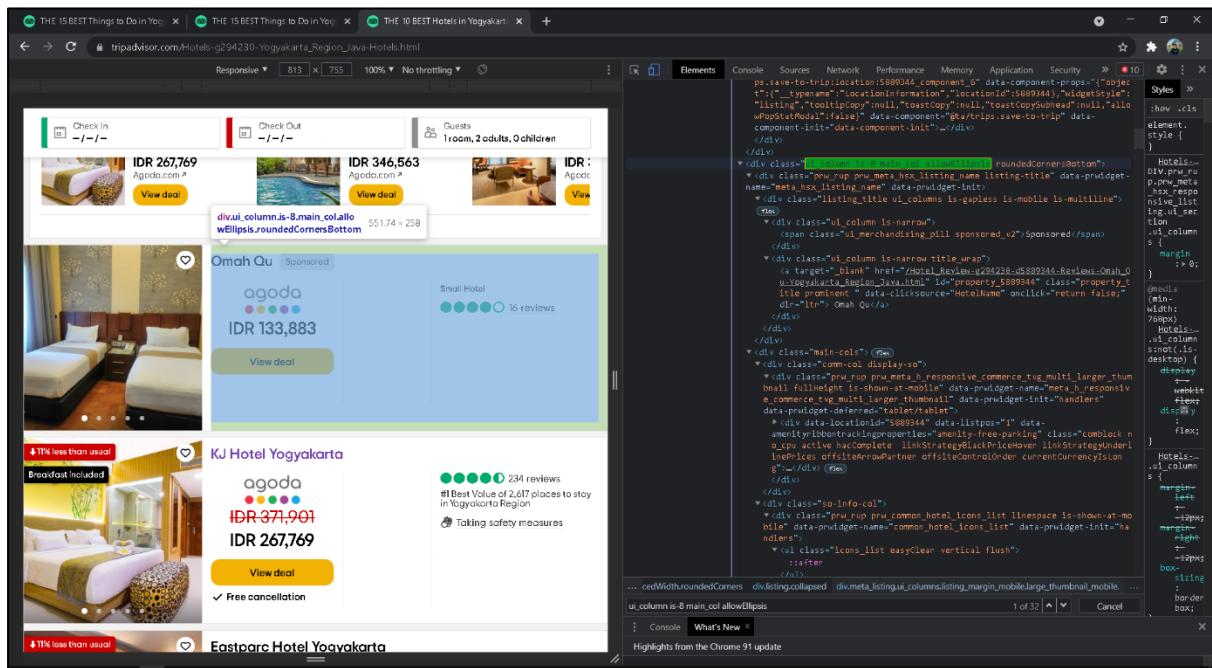
- url = untuk menampung url
- req = untuk menampung *requests*
- soup = untuk menampung objek soup

- `data_hotel = list kosong untuk menampung data`

dan kemudian tuliskan kode seperti di bawah ini:

```
url = "https://www.tripadvisor.com/Hotels-g294230-Yogyakarta_Region_Java-Hotels.html"

req = requests.get(url)
soup = BeautifulSoup(req.text, "html.parser")
data_hotel = []
```

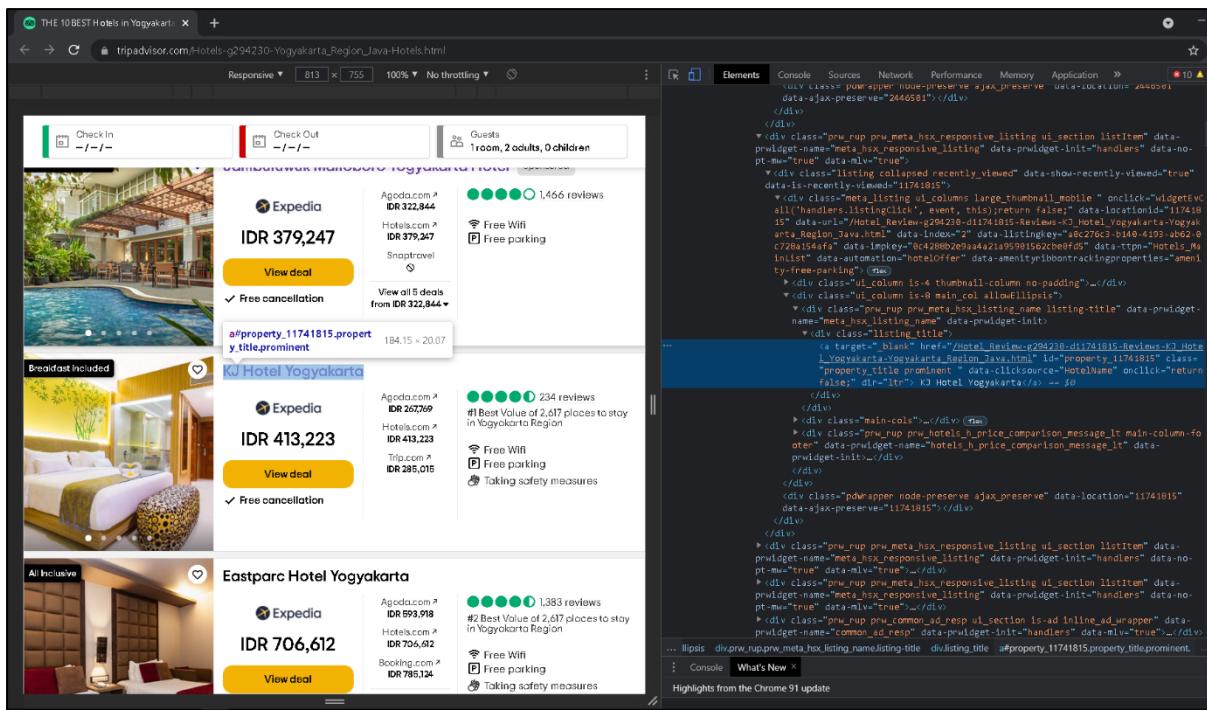


Gambar 1. 2 *inspect element interface website*

kemudian, buka website dan klik kanan *inspect elemen*, arahkan pada bagian nama hotel. Lihat pada div tersebut pada *inspect elemen* (sesuai gambar di atas), catat nama class dari div tersebut yang nantinya akan kita gunakan pada perulangan. sesuai dengan *inspect elemen* tersebut, maka kita memilih div dengan class “`ui_column is-8 main_col allowEllipsis`”. Sehingga dapat dituliskan seperti kode di bawah ini.

```
for item in soup.findAll("div", "ui_column is-8 main_col allowEllipsis"):
```

di dalam for tersebut kita akan mengambil beberapa data diantaranya adalah nama dan harga, dan jumlah review yang ada pada hotel tersebut. seperti yang sudah kita lakukan, caranya adalah melakukan *inspect elemen* dan arahkan pada nama hotel. Seperti gambar di bawah ini.



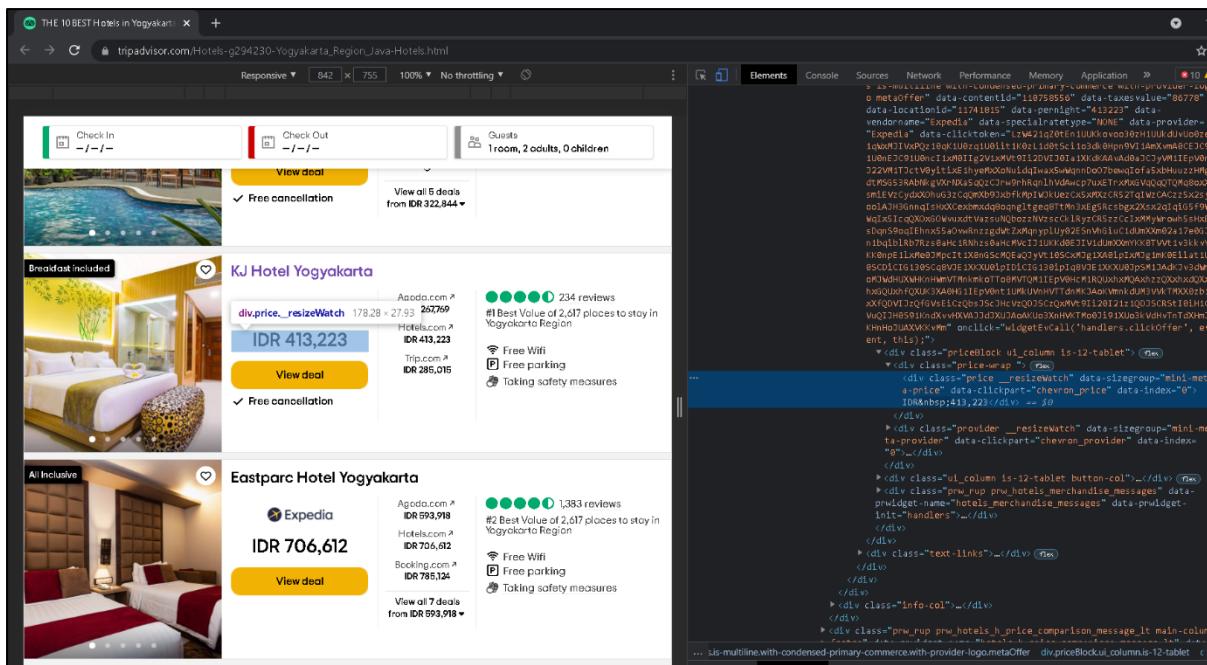
Gambar 1.3 inspect elemen nama hotel

Sesuai dengan gambar di atas, maka untuk mendapatkan nama hotel elemen yang kita pilih adalah a yang memiliki class adalah "property\_title\_prominent". Maka kita dapat menuliskan kode untuk mencari nama hotel adalah sebagai berikut.

```
for item in soup.findAll("div", "ui_column is-8 main_col allowEllipsis"):
    nama = item.find("a", "property_title_prominent").text.replace("@", "").strip()
```

setelah menuliskan find, elemen, dan class, kita menuliskan .text untuk langsung mendapatkan string dari elemen tersebut. Sedangkan .replace digunakan untuk mengganti karakter @ dengan none. Karena saat ditampilkan nama hotel memiliki spasi di awal nama, maka kita menggunakan tambahan .strip() untuk menghapus spasi di awal nama tersebut.

setelah nama adalah harga, untuk mencari harga sama seperti di atas kita mencari menggunakan inspect elemen dan mengarahkan ke bagian harga. Seperti gambar 1.4 di bawah ini.



Gambar 1. 4 inspect elemen mencari harga hotel

Sesuai dengan yang kita cari, maka harga terdapat pada elemen div yang memiliki atribut "data-sizegroup" berupa "mini-meta-price". Oleh karena itu kode yang kita tulis adalah seperti di bawah ini. Replace digunakan untuk menghilangkan karakter "IDR" dan karakter koma ",," sehingga nantinya data harga mudah diolah.

```
harga = item.find("div", {"data-sizegroup": "mini-meta-price"}).text.replace("IDR", "").replace(",","")
```

kemudian mencari jumlah review, sama seperti hal di atas, kita melakukan inspect elemen dan mengarahkan pada bagian angka jumlah review. Maka elemen yang menampung konten jumlah review adalah a dengan class yaitu "review\_count".

```
jumlah_review = item.find("a", "review_count").text.replace("reviews", "").replace("review", "")
```

Setelah itu, kita memasukkan semua data yang sudah kita cari di dalam satu iterasi tersebut ke dalam variabel yang kita beri nama data\_hotel dengan menggunakan perintah append([nama, harga, jumlah\_review]). Sehingga kode yang ditulis haruslah seperti di bawah ini.

```
for item in soup.findAll("div", "ui_column is-8 main_col allowEllipsis"):
    nama = item.find("a", "property_title prominent").text.replace("@", "").strip()
    harga = item.find("div", {"data-sizegroup": "mini-meta-price"}).text.replace("IDR", "").replace(",","")
    jumlah_review = item.find("a", "review_count").text.replace("reviews", "").replace("review", "")
    data_hotel.append([nama, harga, jumlah_review])
```

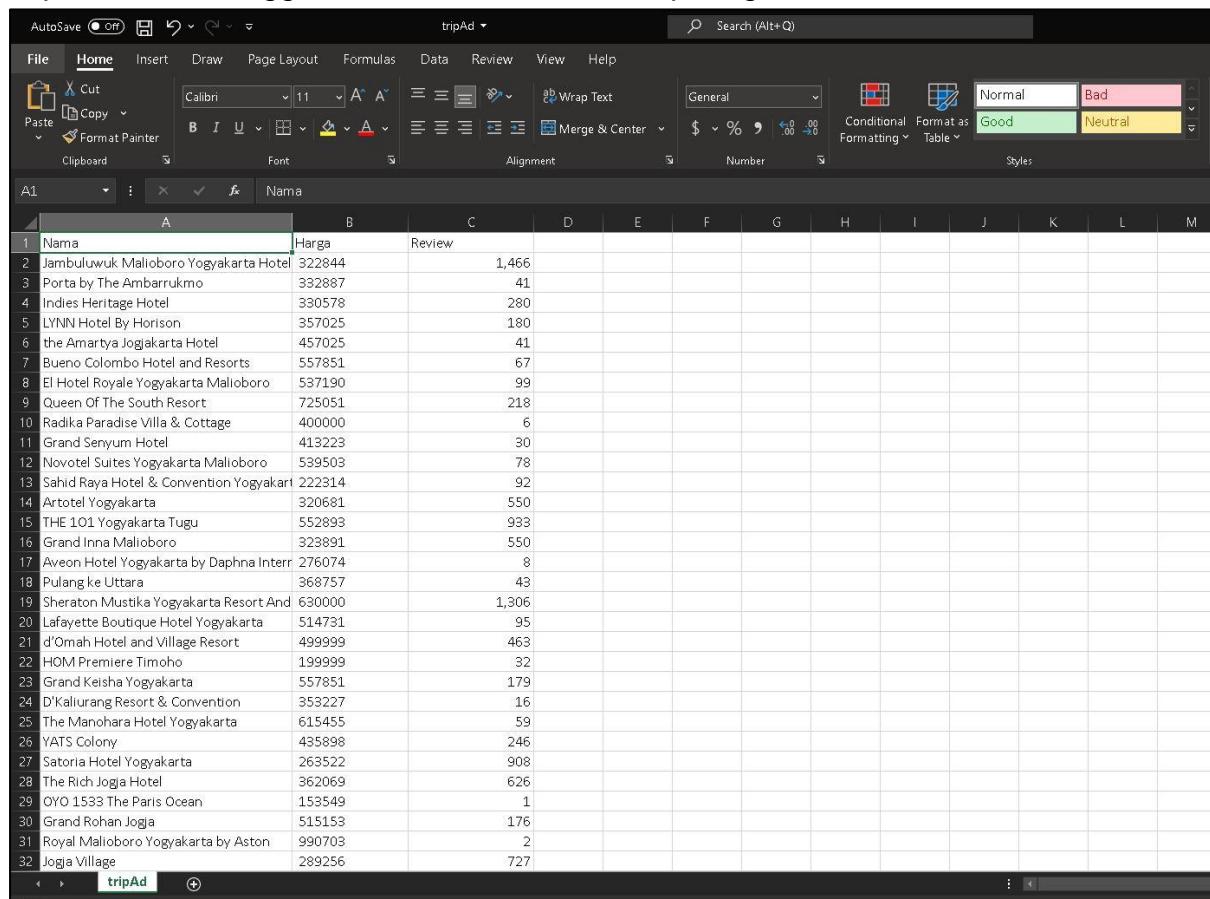
Setelah itu kita bisa menyimpan data hasil *scraping* tersebut ke dalam file csv dengan menggunakan *library* csv dan menggunakan perintah writerow. Sehingga dapat ditulis seperti kode di bawah ini.

```
# save to csv
kolom = ["Nama", "Harga", "Review"]
writer = csv.writer(open("tripAd.csv", "w", newline=""))
writer.writerow(kolom)

for d in data_hotel:
    writer.writerow(d)

print(data_hotel)
```

Pertama adalah membuat variabel dengan nama kolom untuk menampung nama kolom apa saja yang akan dibuat. Setelah itu menuliskan writer(open("nama\_file.csv")) seperti kode id atas untuk membuat file baru. Kemudian perintah writerow digunakan untuk menuliskan data pada file csv yang sudah dibuat. Maka file csv yang sudah berisi data *scraping* sudah berhasil dibuat dan dapat dibuka menggunakan Microsoft Excel seperti gambar di bawah ini.



	Nama	Harga	Review
1	Jambuluwuk Malioboro Yogyakarta Hotel	322844	1,466
2	Porta by The Ambarrukmo	332887	41
3	Indies Heritage Hotel	330578	280
5	LYNN Hotel By Horison	357025	180
6	the Amartya Jogjakarta Hotel	457025	41
7	Bueno Colombo Hotel and Resorts	557851	67
8	EI Hotel Royale Yogyakarta Malioboro	537190	99
9	Queen Of The South Resort	725051	218
10	Radika Paradise Villa & Cottage	400000	6
11	Grand Senyum Hotel	413223	30
12	Novotel Suites Yogyakarta Malioboro	539503	78
13	Sahid Raya Hotel & Convention Yogyakart	222314	92
14	Artotel Yogyakarta	320681	550
15	THE 101 Yogyakarta Tugu	552893	933
16	Grand Inna Malioboro	323891	550
17	Aveon Hotel Yogyakarta by Daphna Intern	276074	8
18	Pulang ke Uttara	368757	43
19	Sheraton Mustika Yogyakarta Resort And	630000	1,306
20	Lafayette Boutique Hotel Yogyakarta	514731	95
21	d'Omah Hotel and Village Resort	499999	463
22	HOM Premiere Timoho	199999	32
23	Grand Keisha Yogyakarta	557851	179
24	D'Kalurang Resort & Convention	353227	16
25	The Manohara Hotel Yogyakarta	615455	59
26	YATS Colony	435898	246
27	Satoria Hotel Yogyakarta	263522	908
28	The Rich Jogja Hotel	362069	626
29	OYO 1533 The Paris Ocean	158549	1
30	Grand Rohan Jogja	515153	176
31	Royal Malioboro Yogyakarta by Aston	990703	2
32	Jogja Village	289256	727

Gambar 1.5 File csv hasil scraping hotel

Keseluruhan kode untuk melakukan scraping hotel adalah sebagai berikut.

```
import requests
from bs4 import BeautifulSoup
import csv

url = "https://www.tripadvisor.com/Hotels-g294230-Yogyakarta_Region_Java-Hotels.html"

req = requests.get(url)
soup = BeautifulSoup(req.text, "html.parser")
data_hotel = []

for item in soup.findAll("div", "ui_column is-8 main_col allowEllipsis"):
    nama = item.find("a", "property_title prominent").text.replace(
        "@", "").strip()
    harga = item.find(
        "div", {"data-sizegroup": "mini-meta-price"}).text.replace("IDR",
    "").replace(","," ")
    jumlah_review = item.find("a", "review_count").text.replace(
        "reviews", "").replace("review", "")
    data_hotel.append([nama, harga, jumlah_review])
    print(f"Harga: {harga}\n")

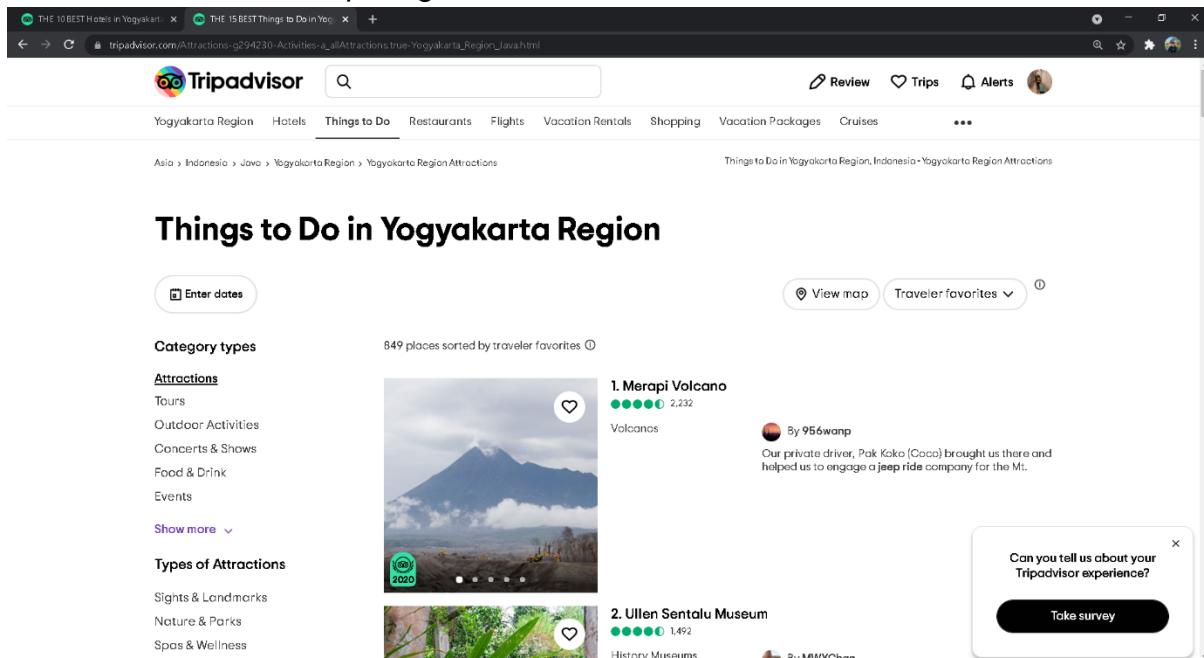
# save to csv
kolom = ["Nama", "Harga", "Review"]
writer = csv.writer(open("tripAd.csv", "w", newline=""))
writer.writerow(kolom)

for d in data_hotel:
    writer.writerow(d)

print(data_hotel)
```

## Scraping Things To Do

Selanjutkan kita akan melakukan *scraping* pada bagian *things to do*. Caranya hampir sama dengan saat kita melakukan scraping daftar hotel, namun yang membedakannya tentunya url yang akan kita *scraping* dan komponen apa saja yang akan kita cari. Pertama buka situs tripadvisor dan arahkan pada bagian *things to do* atau dapat melalui link berikut [https://www.tripadvisor.com/Attractions-g294230-Activities-a\\_allAttractions.true-Yogyakarta\\_Region\\_Java.html](https://www.tripadvisor.com/Attractions-g294230-Activities-a_allAttractions.true-Yogyakarta_Region_Java.html). Maka tampilan dari situs tersebut adalah seperti gambar 1.6 di bawah ini.



Gambar 1. 6 Tampilan tripadvisor bagian things to do

Kemudian, buka IDE yang digunakan untuk menulis kode, di sini saya menggunakan Visual Studio Code. Kita akan mengimport beberapa library yang akan digunakan seperti requests, BeautifulSoup, dan csv. Kode yang ditulis dapat dilihat di bawah ini.

```
import requests
import urllib
import requests
from bs4 import BeautifulSoup
import csv
```

setelah itu, buat variabel yang digunakan untuk menampung link yang akan kita scraping, di sini saya menamai variabel tersebut dengan nama url. Kemudian buat beberapa variabel lainnya yang digunakan untuk menampung requests dan objek soup yang dibuat, serta list kosong yang nantinya akan digunakan untuk menampung data yang discraping. Saya menggunakan nama berikut untuk variabelnya.

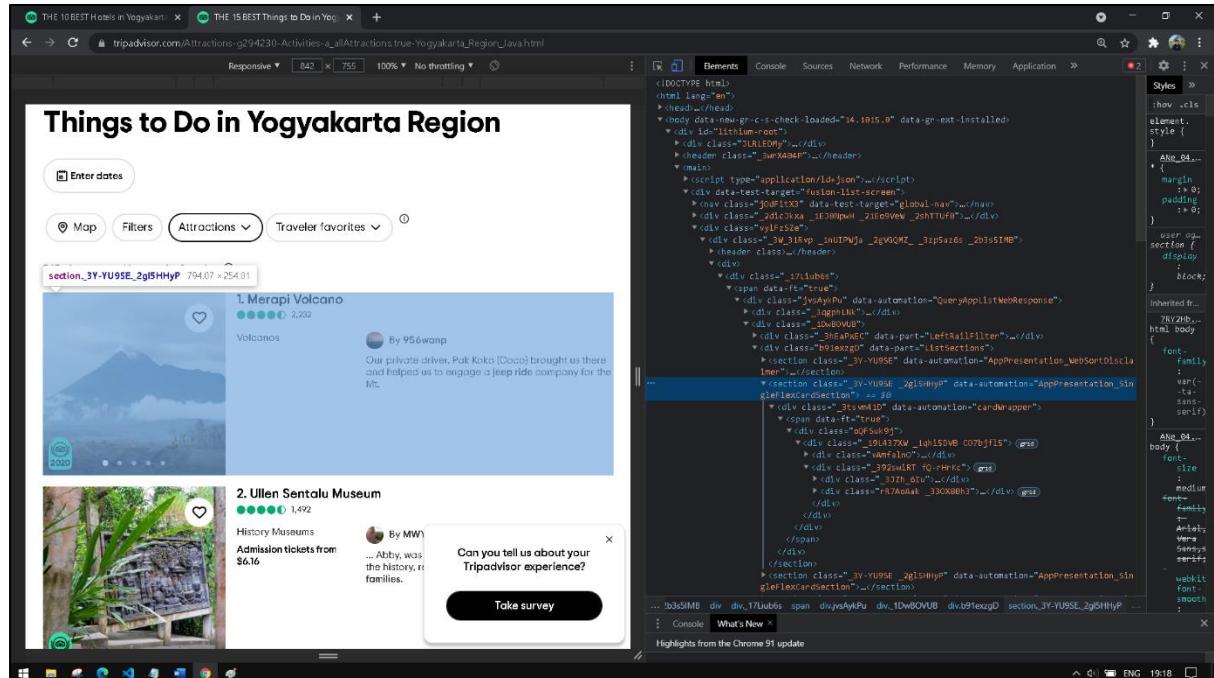
- url = untuk menampung link
- req = untuk menampung requests
- soup = untuk menampung objek soup
- data\_things\_to\_do = untuk menampung data hasil *scraping*

sehingga kode yang ditulis adalah sebagai berikut.

```
url = "https://www.tripadvisor.com/Attractions-g294230-Activities-a_allAttractions.true-Yogyakarta_Region_Java.html"

req = requests.get(url)
soup = BeautifulSoup(req.content, "html.parser")
data_things_to_do = []
```

kemudian kita buka situs yang akan discraping dan klik kanan inspect elemen, arahkan pada div yang menampung beberapa informasi seperti nama, deskripsi singkat, jumlah reviews yang ada, serta cuplikan review yang ditampilkan. Lihat pada gambar 1.7 di bawah ini.

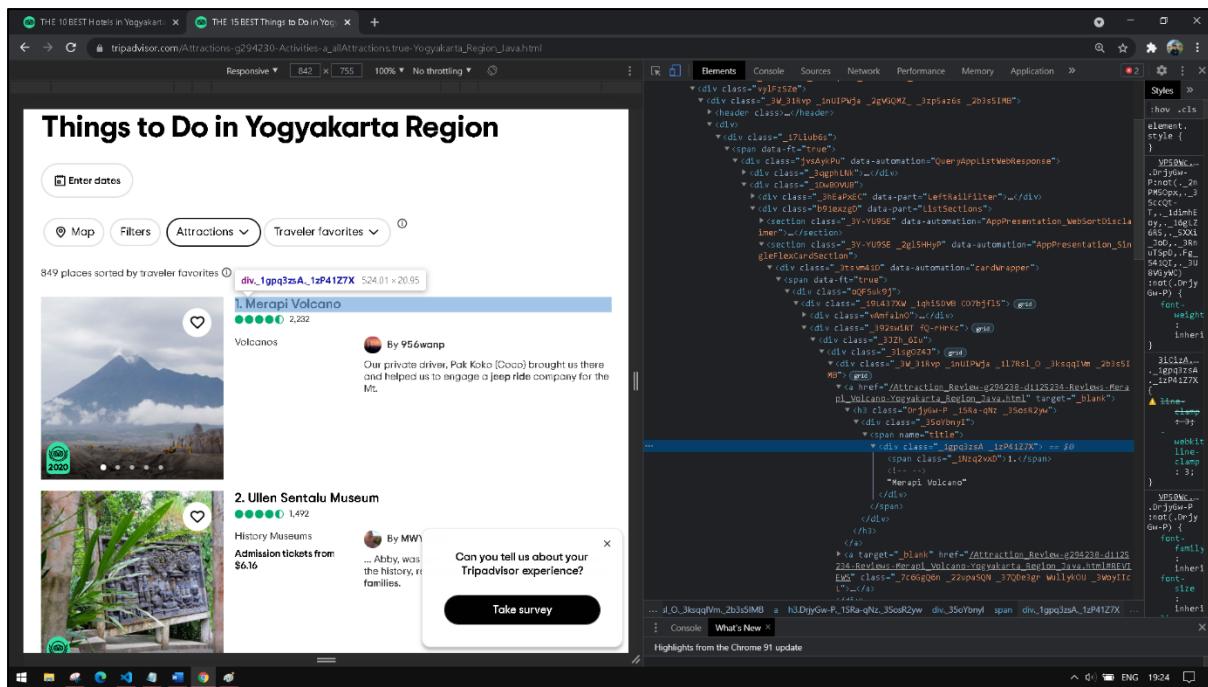


Gambar 1.7 *inspect elemen pada div yang menampung konten*

Setelah mengetahui elemen dan class yang menampung konten tersebut, maka kita bisa melakukan perulangan untuk setiap konten yang ditemukan dengan menggunakan perintah findAll, seperti kode yang dapat dilihat di bawah ini.

```
for item in soup.findAll("section", {"class": "_3Y-YU9SE _2g15HHyP"}):
```

Kemudian kita akan mencari beberapa informasi seperti nama tempat wisata, jumlah reviews yang ada, deskripsi singkat yang ditulis, serta cuplikan reviews dari salah satu pengunjung yang ditampilkan pada halaman web tersebut. Untuk mencari nama adalah dengan melakukan *inspect elemen* dan mengarahkan pada tampilan nama, sehingga akan diketahui elemen dan atribut yang dimiliki dari konten nama tersebut. Seperti yang ditampilkan pada gambar 1.8 di bawah ini.



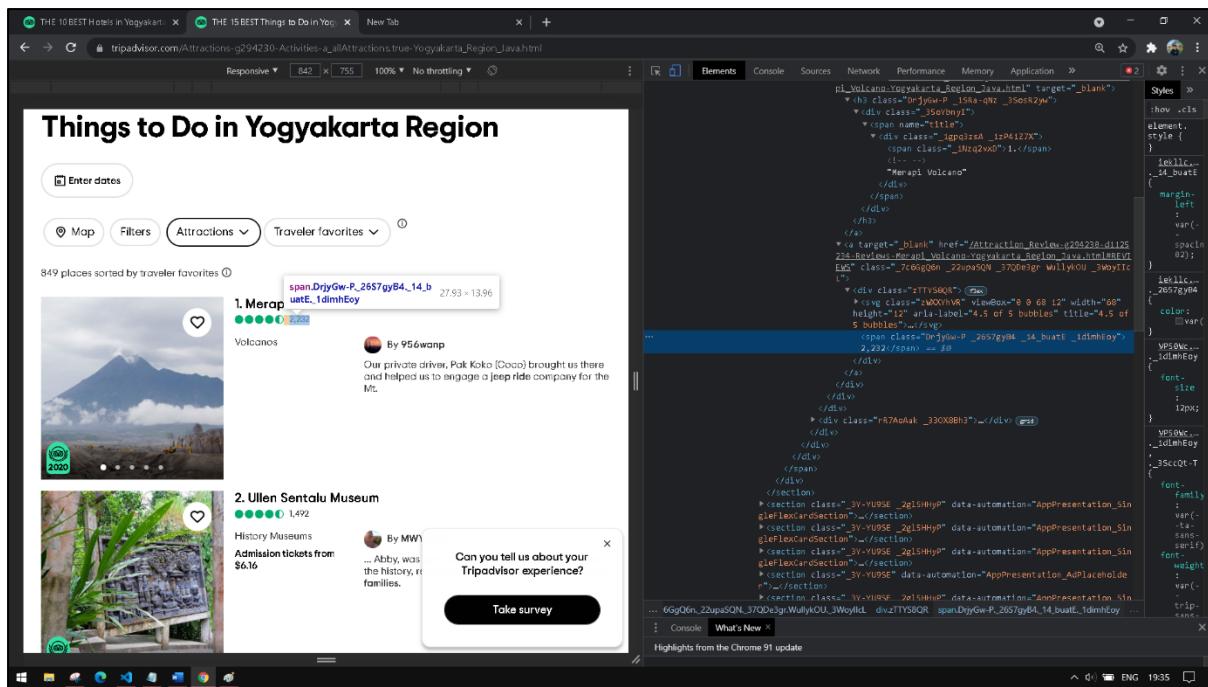
Gambar 1.8 Inspect elemen konten nama tempat wisata

Setelah mengetahui elemen dan atribut yang membentuk nama tersebut, maka kita dapat langsung menuliskan kode `.find()` untuk mencari elemen tersebut dan menggunakan `.text` untuk langsung mengambil string dari elemen tersebut. Kode dapat dilihat di bawah ini.

```
for item in soup.findAll("section", {"class": "_3Y-YU9SE _2g15HHyP"}):
    nama = item.find("div", {"class": "_1gpq3zsA _1zP41Z7X"}).text
```

kemudian setelah mencari nama, kita mencari informasi jumlah reviews yang ada. Cara yang dilakukan sama yaitu dengan melakukan *inspect elemen* untuk menemukan elemen yang membentuk konten jumlah reviews, apabila sudah ketemu maka kita dapat menggunakan perintah `.find()` dan diakhiri `.text` untuk mengambil konten yang kita cari tersebut, misalnya mengambil jumlah reviews.

Cara yang sama juga dilakukan untuk mencari elemen selanjutnya yang ingin dicari, seperti deskripsi tempat wisata dan cuplikan review dari salah satu pengunjung yang merupakan user tripadvisor. Gambar dan kode yang digunakan dapat dilihat di bawah ini.



Gambar 1.9 Inspect element jumlah reviews tempat wisata

Kode yang ditulis adalah sebagai berikut.

```
for item in soup.findAll("section", {"class": "_3Y-YU9SE _2gl5HHyP"}):
    nama = item.find("div", {"class": "1gpq3zsA _1zP41Z7X"}).text
    review = item.find(
        "span", {"class": "DrjyGw-P _26S7gyB4 _14_buatE
_1dimhEoy"}).text.replace(", ", "")
    try:
        deskripsi = item.find(
            "div", {"class": "DrjyGw-P _26S7gyB4 _3SccQt-T"}).text
    except:
        deskripsi = ""
    try:
        single_review = item.find("div", {"class": "2AdNKb-q"}).find(
            "div", {"class": "DrjyGw-P _26S7gyB4 _3SccQt-T"}).text.replace("\n", " ")
    except:
        single_review = ""

    data_things_to_do.append([nama, deskripsi, single_review, review])
```

Pada kode di atas, saya menggunakan try dan except apabila data yang ingin dicari ternyata tidak ada (*ada satu iterasi yang tidak memiliki deskripsi*), sehingga tidak menyebabkan error dan variabel deskripsi akan diisi none atau string kosong.

Kemudian untuk menyimpan data hasil scraping ke file csv, kita menggunakan kode seperti di bawah ini.

```
kolom = ["Nama", "Deskripsi", "Review", "Jumlah_Review"]
writer = csv.writer(open("tripAd_things_to_do.csv", "w", newline=""))
writer.writerow(kolom)
```

```

for data in data_things_to_do:
    writer.writerow(data)

print(data_things_to_do)

```

Sehingga keseluruhan kode yang digunakan untuk *scraping* bagian *things to do* adalah seperti berikut.

```

import requests
import urllib
import requests
from bs4 import BeautifulSoup
import csv

url = "https://www.tripadvisor.com/Attractions-g294230-Activities-
a_allAttractions.true-Yogyakarta_Region_Java.html"

req = requests.get(url)
soup = BeautifulSoup(req.content, "html.parser")
data_things_to_do = []

print(req)

for item in soup.findAll("section", {"class": "_3Y-YU9SE _2gl5HHyP"}):
    nama = item.find("div", {"class": "_1gpq3zsA _1zP41Z7X"}).text
    review = item.find(
        "span", {"class": "DrjyGw-P _26S7gyB4 _14_buatE
_1dimhEoy"}).text.replace(", ", "")
    try:
        deskripsi = item.find(
            "div", {"class": "DrjyGw-P _26S7gyB4 _3SccQt-T"}).text
    except:
        deskripsi = ""
    try:
        single_review = item.find("div", {"class": "_2AdNKb-q"}).find(
            "div", {"class": "DrjyGw-P _26S7gyB4 _3SccQt-T"}).text.replace("\n", "")
    except:
        single_review = ""

    data_things_to_do.append([nama, deskripsi, single_review, review])

# save to csv
kolom = ["Nama", "Deskripsi", "Review", "Jumlah_Review"]
writer = csv.writer(open("tripAd_things_to_do.csv", "w", newline=""))
writer.writerow(kolom)

for data in data_things_to_do:
    writer.writerow(data)

print(data_things_to_do)

```

Sesuai dengan kode di atas hasil scraping akan disimpan ke dalam file dengan nama “tripAd\_things\_to\_do.csv” pada direktori yang sama dengan file kode yang dibuat. Dan kita dapat membuka file csv tersebut dengan menggunakan Microsoft Excel seperti yang ditampilkan pada gambar di bawah ini.

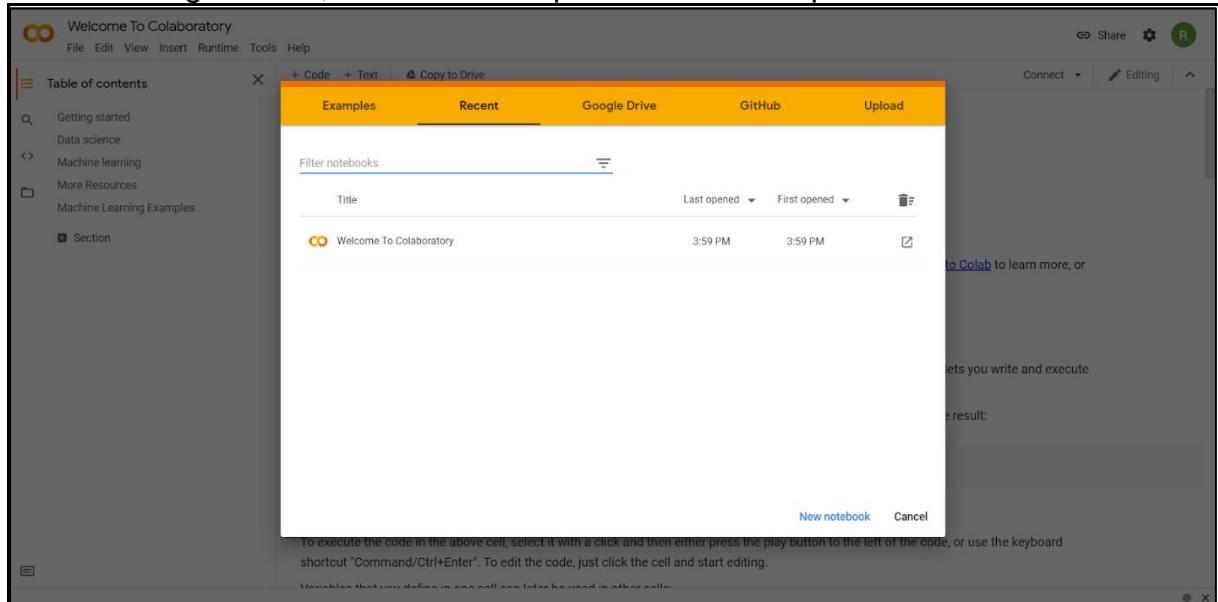
A	B	C	D
1. Merapi Volcano	Volcanos	Our private driver, Pak Keko (Coco) brought us there	2232
2. Ullen Sentalu Museum	History Museums	... Abby was very informative and shared a lot about	1492
3. Malioboro Road	Flea & Street Markets	... dofis along the road, also many traditional famous	2809
4. Ramayana Ballet at Prambanan	Ballets	The stage is set at the Prambanan Temple which is ill	786
5. Pinus Penger Nature Tourism	Nature & Wildlife Areas • Forests	Very nice place to rest.	159
6. Sewu Temple	Ancient Ruins • Points of Interest & Landmarks	Sewu Temple is a must see to all visitors.	851
7. Teman Pintar Science Park	Amusement & Theme Parks	Cheap,fun and interesting educational park	465
8. Garjuran Church	Churches & Cathedrals	You can visit this all for free and can also attend a mi	186
9. Yogyakarta Palace	Historic Sites • Points of Interest & Landmarks	We joined a funny lady tour guide who shared intere	2824
10. Yogyakarta Monument	Monuments & Statues	It was part of Imagine linear which connects in one li	376
11. Plaosan Temple	Ancient Ruins • Points of Interest & Landmarks	The complex consists of twin main temple buildings	669
12. Wisata Seribu Batu Songgo Langit	Nature & Wildlife Areas • Parks	... matter is so personal, seeing his work in a persona	64
13. Bukit Pangguk Kediriung	Lookouts • Observation Decks & Towers	... 36	36
14. Jogokaryan Mosque	Religious Sites	4	4
15. UGM Campus Mosque	Religious Sites	11	11
16. Jogja Bay Waterpark	Water Parks	You can meet Lifeguard staff in every spot and they r	291
17. Pule Payung	Lookouts • Observation Decks & Towers	29	29
18. Affandi Museum	Art Museums	... matter is so personal, seeing his work in a persona	276
19. Sosrowijayan Street	Points of Interest & Landmarks	9	9
20. De Mata Tridi Eye 3D Museum	Specialty Museums • Amusement & Theme Parks	309	309
21. Warung Boto	Historic Sites • Ancient Ruins	9	9
22. Beringharjo Market	Flea & Street Markets	452	452
23. Water Castle (Taman Sari)	Points of Interest & Landmarks	The historical background is really interesting and als	3259
24. Wediombo Beach	Beaches	I always spend my time in this beach because this be	141
25. Pindul Cave	Caverns & Caves	Look out for bats in the cave.	662
26. Ijo Temple	Ancient Ruins • Points of Interest & Landmarks	290	290
27. Ratu Boko Temple	Ancient Ruins • Religious Sites	The sunset view is like a portal gate to another world	1385
28. Sonobudoyo Museum	Specialty Museums	... a flashback to me. Only 3k for domestic entrance fr	190
29. Beidai Pine Peak	Hiking Trails • Forests	50	50
30. Gembira Loka Zoo	Zoos	The best place in town very very good place! advice	411
31.			
32.			
33.			
34.			

Gambar 1. 10 File csv hasil scraping

## Scraping IMDB

Selanjutnya kita akan mencoba untuk *scrapping* situs lainnya agar kita tidak hanya terpaku untuk melakukan *scraping* pada satu situs saja. Situs selanjutnya yang akan kita scraping adalah situs IMDB. Situs ini merupakan situs yang menampilkan daftar film-film layar lebar, mulai dari judul film sampai pendapatan kotor film tersebut. Sama seperti sebelumnya, kita akan menggunakan *library* python Beautiful Soup. Namun, kali ini kita akan mencoba melakukan *scraping* menggunakan Google Colaboratory. Google Colaboratory atau biasa disebut Google Colab merupakan sebuah *notebook* python buatan Google yang dapat diakses tanpa harus meng-*install* aplikasinya di komputer karena kita hanya perlu membukanya menggunakan browser. Selain itu, Google Colab memungkinkan kita untuk melakukan kolaborasi dengan teman atau rekan kerja kita karena Google Colab mempunyai fitur untuk membagikan *link* dari file yang sedang kita kerjakan ke teman atau rekan kerja. Namun seperti aplikasi buatan Google lainnya, kita dapat mengakses Google Colab ini ketika kita mempunyai jaringan internet.

Mari kita mulai kegiatan *scraping* kita. Pertama, buka situs <https://colab.research.google.com/> untuk mengakses Google Colab. Setelah masuk ke situs Google Colab, kita akan ditampilkan halaman seperti berikut:



Gambar 2. 1 Tampilan awal halaman google colab

Setelah itu, kita akan membuat sebuah *notebook* baru. Klik tulisan “[New notebook](#)” dan kita akan ditampilkan halaman *notebook* baru seperti berikut:

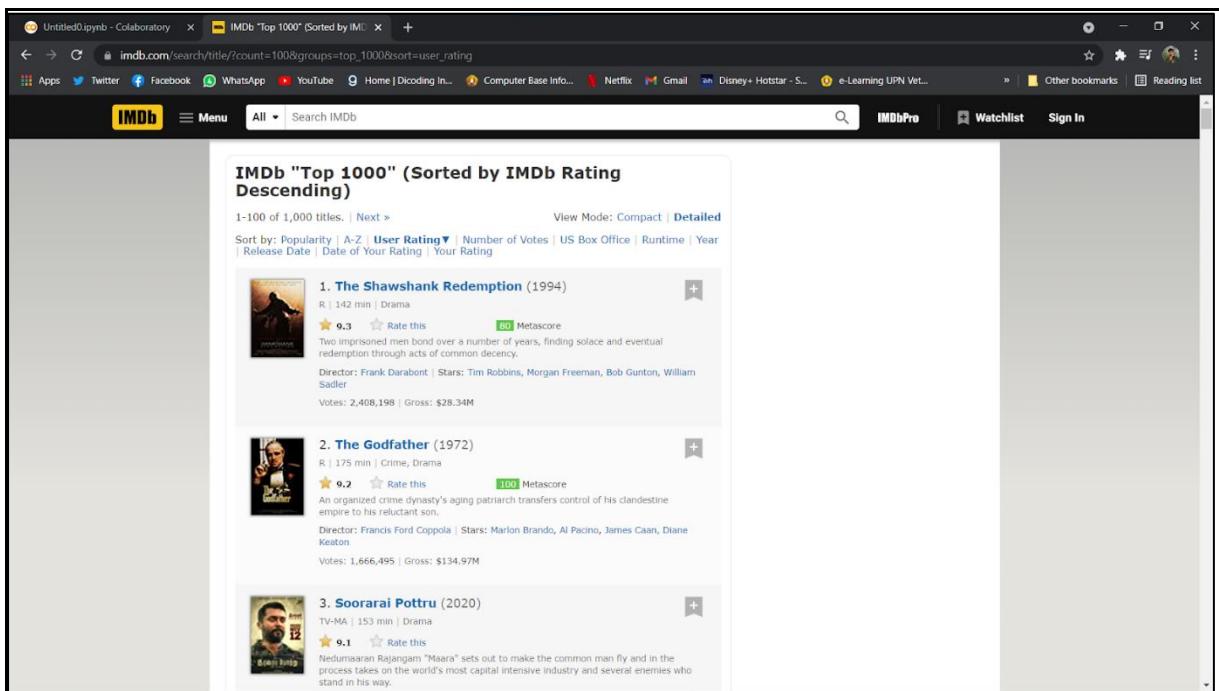


Gambar 2. 2 Tampilan awal halaman *notebook google colab*

Pada halaman tersebut, kita dapat melihat tampilan yang berbeda dari IDE *Visual Studio Code* yang digunakan sebelumnya. *Google Colab* merupakan IDE berjenis *notebook*. Dengan jenis *notebook* tersebut, kita dapat memisahkan kode-kode yang akan kita ketik menjadi beberapa bagian. Bagian-bagian tersebut nantinya akan kita sebut sebagai *cell*. Kita dapat menjalankan *cell-cell* tersebut secara terpisah dan perlu diingat bahwa *cell* tersebut harus dijalankan secara berurutan dari *cell* yang terletak paling atas hingga *cell* paling bawah agar kode yang sudah kita ketik tidak mengalami *error* ketika dijalankan.

Selanjutnya mari kita buka situs IMDB yang akan kita *scraping*. Silahkan kunjungi *link* berikut:

[https://www.imdb.com/search/title/?count=100&groups=top\\_1000&sort=user\\_rating](https://www.imdb.com/search/title/?count=100&groups=top_1000&sort=user_rating). Situs tersebut merupakan situs yang berisi daftar 1000 film terbaik yang diurutkan berdasarkan user rating tertinggi. Setelah membuka situs tersebut, halaman yang akan tampil adalah sebagai berikut:



Gambar 2. 2 Tampilan awal halaman notebook google colab

Setelah situs terbuka, hal yang akan kita lakukan selanjutnya adalah melakukan *inspect* pada halaman tersebut. Setelah kita *inspect*, mari kita tentukan data apa saja yang akan kita *scraping*. Pada kesempatan kali ini, kita akan melakukan *scraping* untuk mendapatkan data-data berupa:

- Judul film
- Tahun film tersebut ditayangkan
- Durasi film tersebut
- Rating
- Metascore
- Votes, dan
- Pendapatan kotor film tersebut (*gross*)

Setelah kita menentukan data apa saja yang akan kita ambil, mari buka kembali halaman *google collab*. Pada *cell* pertama halaman *google colab*, mari kita *import* beberapa *library* yang akan digunakan seperti kode dibawah ini:

```
import pandas as pd
import requests
from bs4 import BeautifulSoup
import numpy as np
```

Selanjutnya kita akan membuat beberapa variabel. Variabel tersebut nantinya akan digunakan untuk menyimpan beberapa hal, seperti:

- url, untuk menyimpan *link* dari halaman yang akan kita *scraping*
- req, untuk menampung response
- soup, untuk menampung objek dari Beautiful Soup.

Untuk membuat variabel tersebut, silahkan tambah *cell* baru di bawah *cell* *import library* yang sudah kita ketik. Setelah itu silahkan ketik kode seperti berikut:

```

url =
"https://www.imdb.com/search/title/?count=100&groups=top_1000&sort=user_rating"
req = requests.get(url)
soup = BeautifulSoup(response.content, 'html.parser') #get the html from the page

```

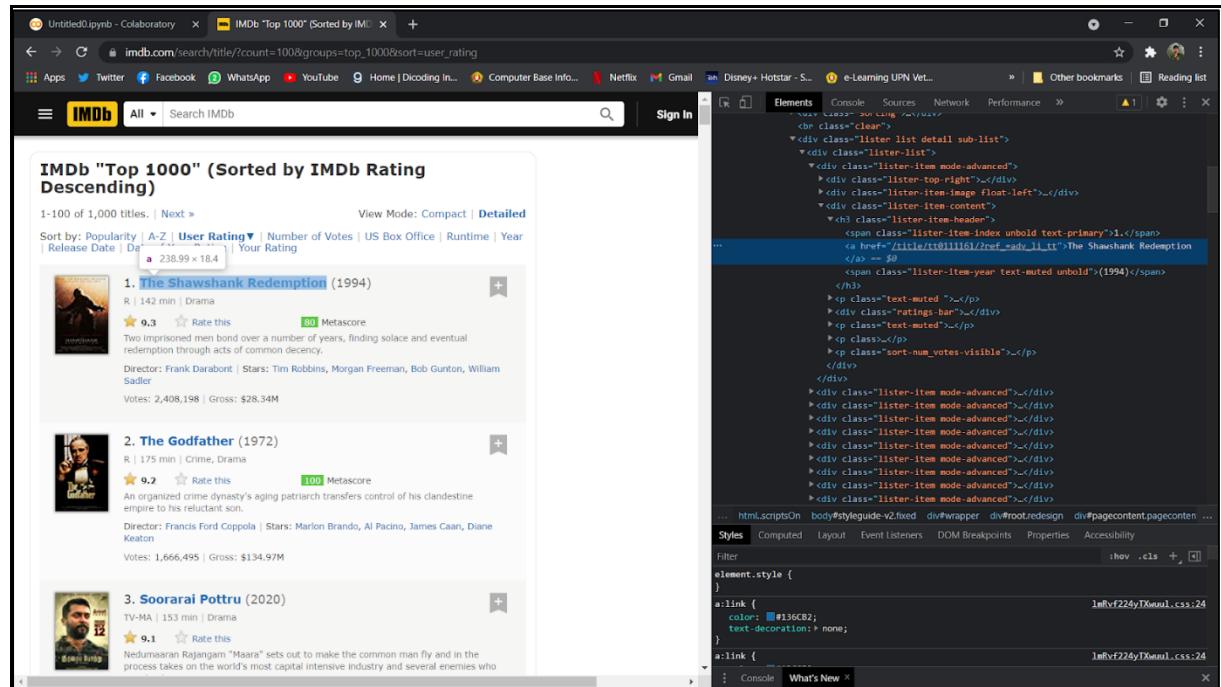
Selanjutnya kita akan membuat variabel baru yang akan dideklarasikan sebagai sebuah list kosong. Variabel baru tersebut nantinya akan digunakan untuk menampung masing-masing dari data yang kita butuhkan. Untuk mendeklarasikan variabel baru sebagai *list*, kita dapat mengetik kode dibawah ini pada *cell* baru:

```

#declare empty list to store the data
movie_title = []
year_of_release = []
movie_runtime = []
movie_rating = []
movie_metascore = []
movie_vote = []
movie_gross_list = []

```

Selanjutnya, mari kita kembali ke halaman IMDB untuk melihat data-data yang kita perlukan terletak pada bagian dari HTML yang mana. Setelah membuka halaman IMDB, kita dapat melakukan *inspect* pada data yang akan digunakan. Sebagai contoh, disini kita akan melakukan *inspect* pada judul film. Hal yang perlu kita lakukan adalah dengan klik kanan pada judul film tersebut, lalu pilih *inspect*. Setelah melakukan hal tersebut, maka halaman yang akan ditampilkan adalah sebagai berikut:

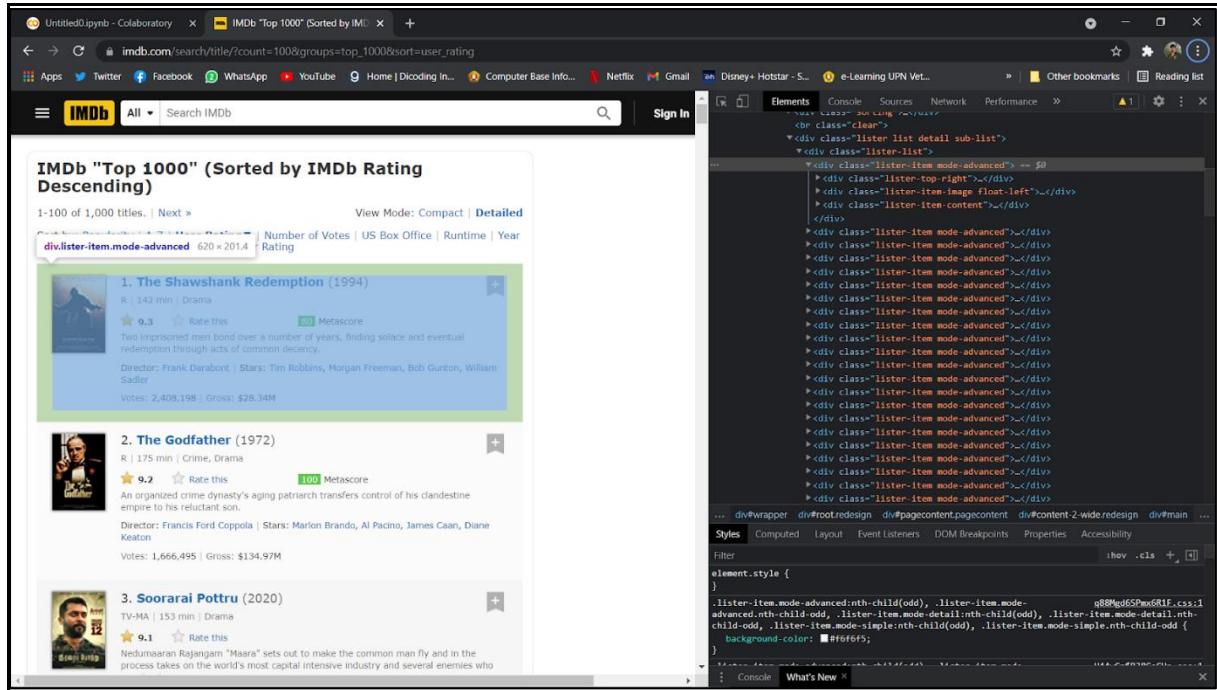


Gambar 2. 3 Tampilan halaman ketika *inspect*

Pada gambar diatas, kita dapat melihat judul film terletak pada tag [tag](#) yang berada di dalam tag 

### . Lalu kita juga bisa melihat tag yang berisi judul film tersebut terletak di dalam tag yang memiliki class 'lister-item-content'. Selanjutnya, mari coba kita tutup tag tersebut dan arahkan kursor ke arah tag yang memiliki class 'lister-item mode-advanced'. Kita dapat lihat pada

halaman situs tersebut, bagian yang di-*highlight* adalah keseluruhan data yang akan kita *scraping* seperti pada gambar berikut:



Gambar 2.4 Tampilan halaman ketika mengarahkan kursor ke tag <div>

Lalu, mari coba kita arahkan kursor kita ke tag <div> dengan *class* serupa lainnya. Kita dapat melihat data-data yang dibutuhkan sebenarnya terbungkus pada tag <div> yang memiliki *class* 'lister-item mode-advanced'. Oleh karena itu, selanjutnya kita akan mengambil keseluruhan tag <div> yang memiliki *class* 'lister-item mode-advanced' tersebut kedalam sebuah variabel agar proses *scraping* menjadi lebih mudah.

Untuk melakukan hal tersebut, mari kita kembali kembali ke *Google Colab* dan ketikkan kode berikut pada cell baru:

```
#retrieve all the necessary data
movie_data = soup.findAll('div', attrs= {'class': 'lister-item mode-advanced'})
```

Untuk pengecekan, silahkan ketik `movie_data` lalu kita jalankan *cell* tersebut dan hasil yang akan didapatkan adalah sebagai berikut:

```

[1] import pandas as pd
import requests
from bs4 import BeautifulSoup
import numpy as np

[3] url = "https://www.imdb.com/search/title/?count=100&groups=top_1000&sort=user_rating"
req = requests.get(url)
soup = BeautifulSoup(req.content, 'html.parser') #get the html from the page

[4] #declare empty list to store the data
movie_title = []
year_of_release = []
movie_runtime = []
movie_rating = []
movie_metascore = []
movie_vote = []
movie_gross_list = []

[5] #retrieve all the necessary data
movie_data = soup.findAll('div', attrs= {'class': 'lister-item mode-advanced'})
movie_data

```

**Gambar 2. 5** Tampilan halaman setelah kita jalankan cell movie\_data

Pada gambar 2. 4 diatas, terlihat bahwa seluruh tag `<div>` yang memiliki `class` ‘`lister-item mode-advanced`’ tersimpan ke dalam sebuah variabel `movie_data`. Langkah selanjutnya adalah kita akan mengambil data-data yang kita perlukan dan menyimpannya ke dalam variabel list kosong yang sudah kita buat. Untuk melakukan hal tersebut, silahkan ketikan kode berikut pada cell baru:

```

#store data to the list
for store in movie_data:
    title = store.h3.a.text
    movie_title.append(title)

    year = store.h3.find('span', attrs= {'class': 'lister-item-year text-muted
unbold'}).text
    year_of_release.append(year)

    runtime = store.p.find('span', attrs= {'class': 'runtime'}).text.replace(
'min','')
    movie_runtime.append(runtime)

    rating = store.find('div', attrs= {'class': 'inline-block ratings-imdb-
rating'}).text.replace('\n', '')
    movie_rating.append(rating)

    if store.find('span', attrs= {'class': 'metascore'}):
        metascore = store.find('span', attrs= {'class': 'metascore'}).text.replace(
',','')
    else:
        metascore = ''
    movie_metascore.append(metascore)

```

```

value = store.findAll('span', attrs= {'name': 'nv'})

```

```

vote = value[0].text
movie_vote.append(vote)

if len(value) > 1:
    gross = value[1].text
else:
    gross = ''
movie_gross_list.append(gross)

```

Setelah kita mendapatkan semua data yang diperlukan dan telah disimpan ke dalam masing-masing variabel, kita akan menggabungkan semua *list* yang berisi data-data tersebut ke dalam sebuah *dataframe* agar nantinya data tersebut menjadi lebih rapi. Untuk melakukan hal tersebut, silahkan ketikkan kode berikut pada *cell* baru:

```

#create movie dataframe
movie_df = pd.DataFrame({'Movie Title': movie_title,
                         'Year of Release': year_of_release,
                         'Runtime': movie_runtime,
                         'Rating': movie_rating,
                         'Metascore': movie_metascore,
                         'Votes': movie_vote,
                         'Gross': movie_gross_list
                         })

```

Lalu mari kita cek bentuk dan isi dari *dataframe* tersebut dengan membuat *cell* baru, ketikkan `movie_df` dan jalankan *cell* tersebut. Hasil yang didapatkan dapat dilihat pada gambar berikut:

	Movie Title	Year of Release	Runtime	Rating	Metascore	Votes	Gross
0	The Shawshank Redemption	(1994)	142	9.3	80	2,408,198	\$28.34M
1	The Godfather	(1972)	175	9.2	100	1,666,506	\$134.97M
2	Soorarai Pottru	(2020)	153	9.1		83,253	
3	The Dark Knight	(2008)	152	9.0	84	2,367,391	\$534.86M
4	The Godfather: Part II	(1974)	202	9.0	90	1,157,870	\$57.30M
...	...	...	...	...	...	...	...
95	My Father and My Son	(2005)	112	8.3		81,274	
96	Inglourious Basterds	(2009)	153	8.3	69	1,304,181	\$120.54M
97	Eternal Sunshine of the Spotless Mind	(2004)	108	8.3	89	931,245	\$34.40M
98	Amélie	(2001)	122	8.3	69	715,474	\$33.23M
99	Snatch	(2000)	102	8.3	55	797,461	\$30.33M

100 rows × 7 columns

**Gambar 2. 6** Tampilan hasil *dataframe* yang sudah dibuat