

UNIVERSITY OF MALAYA

EXAMINATION FOR THE DEGREE OF MASTER OF DATA SCIENCE

ACADEMIC SESSION 2019/2020 : SEMESTER II

WQD7005 : Data Mining

June 2020

INSTRUCTIONS TO CANDIDATES :

Answer **ALL** questions (50 marks).

Name : Nurullainy Binti Mat Rashid

ID : 17036591

Link to codes and data:

<https://github.com/Nurullainy/Data-Mining-Final-Exam>

Link to video :

https://drive.google.com/drive/folders/1hMqLYw5ubSpMRsPvyJwjDG8MuV3GO_Qt?usp=sharing

(This question paper consists of 5 questions on 3 printed pages)

Mini-assignment (50 marks)

Instructions: Work individually, submission via Spectrum.

1. You are required to make a user-agent that will crawl the WWW (your familiar domain) to produce dataset of a particular website.
 - the web site can be as simple as a list of webpages and what other pages they link to
 - the output does not need to be in XHTML (or HTML) form
a multi-stage approach (e.g. produce the xhtml or html in csv format)

Topic: Web Crawling Data of TV Shows and Movies at Online Internet Movie Database (IMDb)

IMDb is an online database of information related to films, television programs, home videos, video games, and streaming content online – including cast, year released, ratings, production crew, plot summaries, trivia, fan and critical reviews.

I want to analyze data of TV Show and movie from IMDb. The data can be extracted from this website : <https://www.imdb.com/search/title/?year=2017>

1.1 Importing Python Libraries

First of all, I will do the following step:

- 1) Import requests module and BeautifulSoup from bs4
- 2) Assign the address of the web page to a variable named 'url'.
- 3) Request the server the content of the web page by using get(), and store the server's response in the variable 'response'.
- 4) Print HTTP status code and a small part of response's content by accessing its text attribute (response is now a Response object).

```
In [1]: import requests
        from bs4 import BeautifulSoup as soup
        import re
```

```
In [2]: url = "https://www.imdb.com/search/title/?year=2017"

        response = requests.get(url, headers = {"Accept-Language": "en-US, en;q=0.5"})

        print('HTTP status:', response.status_code) # return response status from the server
        print(response.text[:300]) # Print a small part of response's content by accessing its .text attribute

        HTTP status: 200
```

```
<!DOCTYPE html>
<html
  xmlns:og="http://ogp.me/ns#"
  xmlns:fb="http://www.facebook.com/2008/fbml">
  <head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="apple-itunes-app" content="app-id=342792525, app-argument=imd
```

1.2 Using BeautifulSoup Module To Parse The HTML Content

Parsing HTML document and extract the 50 div containers

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

```
In [3]: page_soup = soup(response.text, "html.parser") # .text or .content
        type(page_soup)
        # print(page_soup.prettify())
```

```
Out[3]: bs4.BeautifulSoup
```

Class attribute has two values; 1) lister-item and 2) mode-advanced.

This combination is unique to these div containers. Use the find_all() method to extract all the div containers that have a class attribute of *lister-item mode-advanced* and assign it to variable *movie_container*. It will return a ResultSet object which is a list containing all the 50 divs

```
In [4]: # To extract all the div containers that have a class attribute of lister-item mode-advanced
movies_container = page_soup.find_all('div', class_='lister-item mode-advanced')
# Number of movies in current web page
len(movies_container)
```

Out[4]: 50

1.3 Extracting The Data For A Single Movie

Now I'm selecting one movie container (let say the first container) to extract 9 attributes that I am interested with for next data mining purposes:

```
In [7]: # Access the first container which contains information about a single movie
# From a single movie, using this information to extract more data (date release, ratings, etc)

first_movie = movies_container[0] # 1st movie in the list
print(first_movie.text.strip()) # Print a small part of response's content by accessing its .text attribute
```

1.
Dark
(2017–2020)

TV-MA
|
60 min
|

Crime, Drama, Mystery

8.8

Rate this

1
2
3
4
5
6
7
8
9
10

8.8/10
X

A family saga with a supernatural twist, set in a German town, where the disappearance of two young children exposes the relationships among four families.

Stars:
Karoline Eichhorn,
Louis Hofmann,
Jördis Triebel,
Stephan Kampwirth

Votes:
192,616

The 9 following attributes are as follows:

- 1) The name of the TV show or movie
- 2) The year of release
- 3) Runtime of each TV show or movie
- 4) Genre of TV show or movie
- 5) Revenues from the movie released
- 6) The IMDB rating
- 7) The number of votes from user
- 8) Stars of the TV show or movie (name of director and main cast)
- 9) Hyperlink to the TV show or movie

1.4 Extracting Information For All The Tv Shows And Movies In A Single Page

- 1) Declare list of variables to have something to store the extracted data in.
- 2) Loop through each container in a web page (the variable which contains all the 50 movie containers).
- 3) Extract the data points of interest only if the container is True

In [209]: *# List to store the scraped data in*

```
names = []
years = []
runtimes = []
genres = []
revenues = []
imdb_ratings = []
votes = []
stars = []
hyperlinks = []
```

In [210]: **for** container **in** movies_container:
 if container **is not** None:

```
    name = container.h3.a.text
    names.append(name)

    year = container.h3.find('span', class_='list-item-year text-muted unbold').text
    years.append(year)

    if container.find('span', class_='runtime'):
        runtime = container.find('span', class_='runtime').text
    else:
        runtime = ''
    runtimes.append(runtime)

    genre = container.p.find('span', class_='genre').text
    genre = genre.replace('\n', '')
    genre = genre.rstrip()
    genres.append(genre)

    if container.findAll('span', attrs = {'name': 'nv'})[1:]:
        revenue = container.findAll('span', attrs = {'name': 'nv'})[1:]
        revenue = str(revenue)
        revenue = revenue.replace(' ', '').strip()
        revenue = revenue.replace('<spandata-value=""', '').strip()
        revenue = revenue.replace('name="nv"', '').strip()
        revenue = revenue.replace('</span>', '').strip()

    else:
        revenue = ''
    revenues.append(revenue)
```

```

imdb_rating = container.strong.text
imdb_rating = float(imdb_rating)
imdb_ratings.append(imdb_rating)

vote = container.find('span', attrs = {'name':'nv'})['data-value']
vote = int(vote)
votes.append(vote)

star = container.find('p', class_='').text
star = str(star)
star = star.replace('\n', '').strip()
stars.append(star)

link = container.h3.find('a')['href']
link = "https://www.imdb.com/"+link
hyperlinks.append(link)

else:
    container = ''

```

Let's check the data collected using Pandas library. Then print the 10 movie_container in first web page

```

In [211]: import pandas as pd

output_df = pd.DataFrame({'movie_name':names,
                          'year_released':years,
                          'runtime': runtimes,
                          'genre': genres,
                          'revenues': revenues,
                          'imdb_rating':imdb_ratings,
                          'vote':votes,
                          'Star':stars,
                          'link': hyperlinks,
                          })

print(output_df.info())

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 9 columns):
movie_name      50 non-null object
year_released   50 non-null object
runtime         50 non-null object
genre           50 non-null object
revenues        50 non-null object
imdb_rating     50 non-null float64
vote           50 non-null int64
Artist          50 non-null object
link            50 non-null object
dtypes: float64(1), int64(1), object(7)
memory usage: 3.6+ KB
None

```

```

In [212]: # The last 10 movies in page 1

output_df.tail(10)

```

Out[212]:

	movie_name	year_released	runtime	genre	revenues	imdb_rating	vote	Artist	link
40	Taboo	(2017-)	59 min	Drama, Mystery, Thriller		8.4	110582	Stars:Tom Hardy, David Hayman, Jonathan Pryce...	https://www.imdb.com/title/tt3647998/
41	Wonder Woman	(2017)	141 min	Action, Adventure, Fantasy	412,563,408>\$412.56M	7.4	530070	Director:Patty Jenkins Stars:Gal Gadot, C...	https://www.imdb.com/title/tt0451279/
42	Baby Driver	(2017)	113 min	Action, Crime, Drama	107,825,862>\$107.83M	7.6	414324	Director:Edgar Wright Stars:Ansel Elgort...	https://www.imdb.com/title/tt3890160/
43	Pirates of the Caribbean: Dead Men Tell No Tales	(2017)	129 min	Action, Adventure, Fantasy	172,558,876>\$172.56M	6.6	251090	Directors:Joachim Rønning, Espen Sandberg ...	https://www.imdb.com/title/tt1790809/
44	Logan	(2017)	137 min	Action, Drama, Sci- Fi	226,277,068>\$226.28M	8.1	619693	Director:James Mangold Stars:Hugh Jackman...	https://www.imdb.com/title/tt3115342/
45	GLOW	(2017-)	35 min	Comedy, Drama, Sport		8.0	37573	Stars:Alison Brie, Marc Maron, Betty Gilpin, B...	https://www.imdb.com/title/tt5770786/
46	Jumanji: Welcome to the Jungle	(2017)	119 min	Action, Adventure, Comedy	404,515,480>\$404.52M	6.9	303322	Director:Jake Kasdan Stars:Dwayne Johnson...	https://www.imdb.com/title/tt283362/
47	Young Sheldon	(2017-)	30 min	Comedy		7.4	30359	Stars:Iain Armitage, Zoe Perry, Lance Barber, ...	https://www.imdb.com/title/tt6226232/
48	King Arthur: Legend of the Sword	(2017)	126 min	Action, Adventure, Drama	39,175,066>\$39.18M	6.7	184869	Director:Guy Ritchie Stars:Charlie Hunnam...	https://www.imdb.com/title/tt1972591/
49	Imposters	(2017-)	41 min	Comedy, Crime, Drama		7.8	9287	Stars:Inbar Lavi, Rob Heaps, Parker Young, Mar...	https://www.imdb.com/title/tt5212822/

1.5 Extracting data for all TV shows and movies from multiple pages from year 2000 – 2020

- 1) Create a list called *pages*, and populate it with the strings corresponding to the first 4 pages.
- 2) Create a list called *years_url* and populate it with the strings corresponding to the years 2000 - 2020

```
In [214]: pages = [str(i) for i in range(1,5)]
          years_url = [str(i) for i in range(2000,2021)]
```

1.6 Controlling the crawl-rate

Controlling the rate of crawling is important for the website that I will be scraping. If I not controlling the rate of crawling, much less likely to get my IP address banned.

Need to avoid activity disruption of the scraped website by allowing the server to respond to other users' requests too.

Control the loop's rate by using the *sleep()* function from Python's time module. *sleep()* will pause the execution of the loop for a specified amount of seconds. To mimic human behavior, I'll vary the amount of waiting time between requests by using the *randint()* function from the Python's random module. *randint()* randomly generates integers within a specified interval.

```
In [215]: from time import sleep
          from random import randint
          sleep(randint(1,4))
```

Since the web scraping is more than 10 pages, it would be nice to find a way to monitor the scraping process as it's still going. The greater the number of pages, the more helpful the monitoring becomes.

For my script, I'll make use of this feature, and monitor the following parameters:

- 1) The *frequency (speed) of requests*, to ensure our program is not overloading the server
- 2) The *number of requests*, so I can halt the loop in case the number of expected requests is exceeded
- 3) The *status code of our requests*, to make sure the server is sending back the proper responses

To get a frequency value, I divide the number of requests by the time elapsed since the first request.

- 1) Set a starting time using the *time()* function from the time module, and assign the value to *start_time*.

- 2) Assign 0 to the variable requests which to be use to count the number of requests.
- 3) Start a loop, and then with each iteration:
 - Simulate a request
 - Increment the number of requests by 1
 - Pause the loop for a time interval between 8 and 15 seconds
 - Calculate the elapsed time since the first request, and assign the value to elapsed_time
 - Print the number of requests and the frequency

If I set my request to 100 requests, my output return will look a lengthy and a bit untidy as the output accumulates. To avoid that, I'll clear the output after each iteration, and replace it with information about the most recent request. I use the `clear_output()` function from the IPython's `core.display` module. Then set the wait parameter of `clear_output()` to `True` to wait with replacing the current output until some new output appears.

```
In [241]: from time import time
from IPython.core.display import clear_output

start_time = time()
requests = 0

for _ in range(3):

    # A request would go here
    requests += 1
    sleep(randint(1,3))
    current_time = time()
    elapsed_time = current_time - start_time
    print('Request: {}; Frequency: {} requests/s'.format(requests, requests/elapsed_time))

    clear_output(wait = True) # set wait = True, to wait with replacing the current output until some new output ap

Request: 3; Frequency: 0.7473912696391304 requests/s
```

```
In [242]: # Redefining the lists variables so they become empty again.

names = []
years = []
runtimes = []
genres = []
revenues = []
imdb_ratings = []
votes = []
stars = []
hyperlinks = []
```



```

In [219]: from requests import get
          from warnings import warn

          headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.106 Safari/537.36'}

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

          # For every year in the interval 2000-2020
          for year_url in years_url:

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

                  # Make a get request
                  url = "http://www.imdb.com/search/title?release_date=" + year_url + "&sort=num_votes,desc&page=1" + page, headers = headers)
                  response = get(url + year_url + '&sort=num_votes,desc&page=1' + page, headers = headers)

                  # Pause the loop interval between 8 and 15 seconds
                  sleep(randint(8,15))

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

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

              # Break the loop if the number of requests is greater than expected
              if requests > 100:
                  warn('Number of requests exceeds than expected.')
                  break

          # Parse the content of the request with BeautifulSoup
          page_html = soup(response.text, 'html.parser')

          # Select all the 50 movie containers from a single page
          movies_container = page_html.find_all('div', class_ = 'lister-item mode-advanced')

          for container in movies_container:
              if container is not None:

                  name = container.h3.a.text
                  names.append(name)

                  year = container.h3.find('span', class_='lister-item-year text-muted unbold').text
                  years.append(year)

                  if container.find('span', class_='runtime'):
                      runtime = container.find('span', class_='runtime').text
                  else:
                      runtime = ' '
                  runtimes.append(runtime)

                  genre = container.p.find('span', class_='genre').text
                  genre = genre.replace('\n', '')
                  genre = genre.rstrip()
                  genres.append(genre)

```

```

if container.findAll('span', attrs = {'name':'nv'})[1:]:
    revenue = container.findAll('span', attrs = {'name':'nv'})[1:]
    revenue = str(revenue)
    revenue = revenue.replace(' ', '').strip()
    revenue = revenue.replace('<spandata-value=', '').strip()
    revenue = revenue.replace('name="nv"', '').strip()
    revenue = revenue.replace('</span>]', '').strip()

else:
    revenue = ''
    revenues.append(revenue)

imdb_rating = container.strong.text
imdb_rating = float(imdb_rating)
imdb_ratings.append(imdb_rating)

vote = container.find('span', attrs = {'name':'nv'})['data-value']
vote = int(vote)
votes.append(vote)

star = container.find('p', class_='').text
star = str(star)
star = star.replace('\n', '').strip()
stars.append(star)

link = container.h3.find('a')['href']
link = "https://www.imdb.com/"+link
hyperlinks.append(link)

else:
    container = ''

```

Request:84; Frequency: 0.06464969451330399 requests/s

I set the loop limit to 100 but the requests stop at request number 84. This indicates that I have collected all TV shows and movies data from year 2000 – 2020.

Let's check the data collected using Pandas library. Then print the 10 movie_container in first web page

```

In [236]: output_df = pd.DataFrame({'movie_name':names,
                                   'year_released':years,
                                   'runtime_in_min': runtimes,
                                   'genre' : genres,
                                   'revenues' : revenues,
                                   'imdb_rating':imdb_ratings,
                                   'number_of_votes':votes,
                                   'Artist':stars,
                                   'link' : hyperlinks,
                                   })

print(output_df.info())

```

In [237]: output_df.head()

Out [237]:

	movie_name	year_released	runtime_in_min	genre	revenues	imdb_rating	number_of_votes	Artist
0	Gladiator	(2000)	155 min	Action, Adventure, Drama	187,705,427>\$187.71M	8.5	1295296	Director:Ridley Scott Stars:Russell Crowe...
1	Memento	(2000)	113 min	Mystery, Thriller	25,544,867>\$25.54M	8.4	1088494	Director:Christopher Nolan Stars:Guy Pear...
2	Snatch	(2000)	104 min	Comedy, Crime	30,328,156>\$30.33M	8.3	760513	Director:Guy Ritchie Stars:Jason Statham,...
3	Requiem for a Dream	(2000)	102 min	Drama	3,635,482>\$3.64M	8.3	742056	Director:Darren Aronofsky Stars:Ellen Bur...
4	X-Men	(2000)	104 min	Action, Adventure, Sci-Fi	157,299,717>\$157.30M	7.4	558637	Director:Bryan Singer Stars:Patrick Stewa...

Total number of TV shows and movies collected are 4200 titles.

In [51]: `output_df.shape`

Out[51]: (4200, 9)

Export dataframe to csv file

Export dataframe to csv file

In [239]: `import csv`

`output_df.to_csv('movies_imdb_with_revenue.csv', mode = 'w', index=True, header=True)`

Snapshot of exported csv file (movie_imdb_with_revenue) dataset

movies_imdb_with_revenue									
	movie_name	year_released	runtime_in_min	genre	revenues	imdb_rating	number_of_votes	Star	link
1	Glaadiator	-2000	155 min	Action, Adventure, Drama	187,705,427-\$187.71M	8.5	1295546	Director: Ridley Scott Stars: Russell Crowe, Joaquin Phoenix, Connie Nielsen, Oliver Reed	https://www.imdb.com/title/tt0111161/
2	Memento	-2000	113 min	Mystery, Thriller	25,544,867-\$25.54M	8.4	1088700	Director: Christopher Nolan Stars: Guy Pearce, Carrie-Anne Moss, Joe Pantoliano, Mark Boone Junior	https://www.imdb.com/title/tt0203154/
3	Snatch	-2000	104 min	Comedy, Crime	30,328,156-\$30.33M	8.3	700646	Director: Guy Ritchie Stars: Jason Statham, Brad Pitt, Benicio Del Toro, Dennis Farina	https://www.imdb.com/title/tt0266993/
4	Requiem for a Dream	-2000	102 min	Drama	3,635,482-\$3.64M	8.3	742193	Director: Darren Aronofsky Stars: Ellen Burstyn, Jared Leto, Jennifer Connelly, Marlon Wayans	https://www.imdb.com/title/tt0272619/
5	X-Men	-2000	104 min	Action, Adventure, Sci-Fi	157,299,717-\$157.30M	7.4	558716	Director: Bryan Singer Stars: Patrick Stewart, Hugh Jackman, Ian McKellen, Famke Janssen	https://www.imdb.com/title/tt0120668/
6	Cast Away	-2000	143 min	Adventure, Drama, Romance	233,632,142-\$233.63M	7.8	505434	Director: Robert Zemeckis Stars: Tom Hanks, Helen Hunt, Paul Sanchez, Lari White	https://www.imdb.com/title/tt0285383/
7	American Psycho	-2000	101 min	Comedy, Crime, Drama	15,070,285-\$15.07M	7.6	467017	Director: Mary Harron Stars: Christian Bale, Justin Theroux, Josh Lucas, Bill Sage	https://www.imdb.com/title/tt0261472/
8	Unbreakable	-2000	106 min	Drama, Mystery, Sci-Fi	95,011,339-\$95.01M	7.3	377418	Director: M. Night Shyamalan Stars: Bruce Willis, Samuel L. Jackson, Robin Wright, Spencer Treat Clark	https://www.imdb.com/title/tt0261472/
9	Mission: Impossible II	-2000	123 min	Action, Adventure, Thriller	215,400,889-\$215.41M	6.1	305909	Director: John Woo Stars: Tom Cruise, Doug Gray Scott, Thandie Newton, Ving Rhames	https://www.imdb.com/title/tt0208151/
10	Meet the Parents	-2000	108 min	Comedy, Romance	166,244,045-\$166.24M	7	304312	Director: Jay Roach Stars: Ben Stiller, Robert De Niro, Teri Polo, Blythe Danner	https://www.imdb.com/title/tt0208151/
11	O Brother, Where Art Thou?	-2000	107 min	Adventure, Comedy, Crime	45,512,580-\$45.51M	7.7	276674	Director: Joel Coen, Ethan Coen Stars: George Clooney, John Turturro, Tim Blake Nelson, John Goodman	https://www.imdb.com/title/tt0208151/
12	Gone in Sixty Seconds	-2000	118 min	Action, Crime, Thriller	101,648,571-\$101.65M	6.5	255336	Director: Dominic Sena Stars: Nicolas Cage, Angelina Jolie, Giovanni Ribisi, T.J. Cross	https://www.imdb.com/title/tt0208151/
13	The Patriot	-2000	165 min	Action, Drama, History	113,330,342-\$113.33M	7.2	248749	Director: Roland Emmerich Stars: Mel Gibson, Heath Ledger, Josely Richardson, Jason Isaacs	https://www.imdb.com/title/tt0208151/
14	Wo hu cang long	-2000	120 min	Action, Adventure, Fantasy	128,078,872-\$128.08M	7.8	248276	Director: Ang Lee Stars: Yun-Fat Chow, Michelle Yeoh, Ziyi Zhang, Chen Chang	https://www.imdb.com/title/tt0208151/
15	Almost Famous	-2000	122 min	Adventure, Comedy, Drama	32,534,850-\$32.53M	7.9	245339	Director: Cameron Crowe Stars: Billy Crudup, Patrick Fugit, Kate Hudson, Frances McDormand	https://www.imdb.com/title/tt0208151/
16	Scary Movie	-2000	88 min	Comedy	157,019,771-\$157.02M	6.2	226495	Director: Keenen Ivory Wayans Stars: Anna Faris, Jon Abrahams, Marlon Wayans, Carmen Electra	https://www.imdb.com/title/tt0208151/
17	Final Destination	-2000	98 min	Horror, Thriller	53,331,147-\$53.33M	6.7	223140	Director: James Wong Stars: Devon Sawa, Ali Larter, Kerr Smith, Kristen Cloke	https://www.imdb.com/title/tt0208151/
18	Pitch Black	-2000	109 min	Action, Horror, Sci-Fi	39,240,659-\$39.24M	7.1	221231	Director: David Twohy Stars: Radha Mitchell, Cole Hauser, Vin Diesel, Keith David	https://www.imdb.com/title/tt0208151/
19	The Beach	(t) (2000)	119 min	Adventure, Drama, Romance	39,785,027-\$39.79M	6.7	217887	Director: Danny Boyle Stars: Leonardo DiCaprio, Daniel York, Patcharawan Patarakijanon, Virginie Ledoyen	https://www.imdb.com/title/tt0208151/
20	Me, Myself & Irene	-2000	116 min	Comedy	90,570,999-\$90.57M	6.6	217782	Director: Bobby Farrelly, Peter Farrelly Stars: Jim Carrey, Renée Zellweger, Anthony Anderson, Mongo Brownlee	https://www.imdb.com/title/tt0208151/
21	Amores perros	-2000	154 min	Drama, Thriller	5,383,834-\$5.38M	8.1	217564	Director: Alejandro G. Iñárritu Stars: Emilio Echevarría, Gael García Bernal, Goya Toledo, Alvaro Guerrero	https://www.imdb.com/title/tt0208151/
22	How the Grinch Stole Christmas	-2000	104 min	Comedy, Family, Fantasy	260,044,825-\$260.04M	6.2	192078	Director: Ron Howard Stars: Jim Carrey, Taylor Momsen, Kelley, Jeffrey Tambor	https://www.imdb.com/title/tt0208151/
23	Remember the Titans	-2000	113 min	Biography, Drama, Sport	115,654,751-\$115.65M	7.8	192675	Director: Boaz Yakin Stars: Denzel Washington, Will Patton, Wood Harris, Ryan Hurst	https://www.imdb.com/title/tt0208151/
24	Traffic	-2000	147 min	Crime, Drama, Thriller	124,115,725-\$124.12M	7.6	192088	Director: Steven Soderbergh Stars: Michael Douglas, Benicio Del Toro, Catherine Zeta-Jones, Jacob Vargas	https://www.imdb.com/title/tt0208151/
25	What Women Want	-2000	127 min	Comedy, Fantasy, Romance	182,811,707-\$182.81M	6.4	187538	Director: Nancy Meyers Stars: Mel Gibson, Helen Hunt, Maria Tormel, Alan Alda	https://www.imdb.com/title/tt0208151/
26	Miss Congeniality	-2000	109 min	Action, Comedy, Crime	106,807,667-\$106.81M	6.2	181185	Director: Donald Petrie Stars: Sandra Bullock, Michael Caine, Benjamin Bratt, Candice Bergen	https://www.imdb.com/title/tt0208151/
27	Chocolat	-2000	121 min	Drama, Romance	71,509,363-\$71.51M	7.2	174415	Director: Lasse Hallström Stars: Juliette Binoche, Judi Dench, Alfred Molina, Carrie-Anne Moss	https://www.imdb.com/title/tt0208151/
28	The Emperor's New Groove	-2000	78 min	Animation, Adventure, Comedy	89,302,687-\$89.30M	7.3	173318	Director: Mark Dindal Stars: David Spade, John Goodman, Eartha Kitt, Patrick Warburton	https://www.imdb.com/title/tt0208151/
29	Chicken Run	-2000	84 min	Animation, Adventure, Comedy	106,834,564-\$106.83M	7	172691	Director: Peter Lord, Nick Park Stars: Mel Gibson, Julia Sawalha, Phil Daniels, Lynn Ferguson	https://www.imdb.com/title/tt0208151/
30	Charlie's Angels	-2000	98 min	Action, Adventure, Comedy	125,305,545-\$125.31M	5.5	172421	Director: McG Stars: Cameron Diaz, Drew Barrymore, Lucy Liu, Bill Murray	https://www.imdb.com/title/tt0208151/
31	Erin Brockovich	-2000	131 min	Biography, Drama	125,595,205-\$125.60M	7.3	169081	Director: Steven Soderbergh Stars: Julia Roberts, Albert Finney, David Brisbin, Dawn Dawidow	https://www.imdb.com/title/tt0208151/
32	Batman Returns	-2000	114 min	Adventure, Drama, Sci-Fi	27,277,055-\$27.28M	7.5	165300	Director: Tim Burton Stars: Michael Keaton, Danny Keefe, Michael Gough, Christopher Penn	https://www.imdb.com/title/tt0208151/
33	High Fidelity	-2000	93 min	Comedy, Drama, Music	27,277,055-\$27.28M	7.5	164459	Director: John Dahl Stars: John Cusack, Iben Hjejle, Todd Louiso, Jack Black	https://www.imdb.com/title/tt0208151/
34	Road Trip	-2000	93 min	Comedy	68,540,777-\$68.54M	6.4	154443	Director: Todd Phillips Stars: Breckin Meyer, Sean William Scott, Amy Smart, Paulo Costanzo	https://www.imdb.com/title/tt0208151/
35	The Perfect Storm	-2000	130 min	Action, Adventure, Drama	182,818,434-\$182.82M	6.4	152772	Director: Wolfgang Petersen Stars: George Clooney, Mark Wahlberg, John C. Reilly, Diane Lane	https://www.imdb.com/title/tt0208151/
36	Dude, Where's My Car?	-2000	83 min	Comedy, Mystery, Sci-Fi	46,729,800-\$46.73M	5.5	132361	Director: Danny Leiner Stars: Ashton Kutcher, Seann William Scott, Jennifer Garner, Maria Sokoloff	https://www.imdb.com/title/tt0208151/

(10 marks)

2. Draw snowflake schema diagram for the above dataset. Justify your attributes to be selected in the respective dimensions.

(10 marks)

3. You are required to write code to create a decision tree (DT) model using the above dataset (Question 1). In order to achieve the task, you are going to cover the following steps:

- Importing required libraries
- Loading Data
- Feature Selection
- Splitting Data
- Building Decision Tree Model
- Evaluating Model
- Visualizing Decision Trees

(10 marks)

4. You are required to write code to find frequent itemsets using the above dataset (Question 1). In order to achieve the task, you are going to cover the following steps:

- Importing required libraries
- Creating a list from dataset (Question 1)
- Convert list to dataframe with boolean values
- Find frequently occurring itemsets using Apriori Algorithm
- Find frequently occurring itemsets using F-P Growth
- Mine the Association Rules

(10 marks)

5. It will be appeared in week 14.

(10 marks)

Submissions:

The student is expected to submit answers to each question individually, and submit the document in PDF format. The student can include online materials, screenshots, videos and/or codes (ipynb format) to support your answer