

```

##### 1번 전처리 과정 #####

!pip install requests
import requests
from bs4 import BeautifulSoup
import re
# 영화 제목(title)과 url 추출
url1 = 'https://www.imdb.com/search/title/?groups=top_250&sort=user_rating,desc&ref'
url2 = 'https://www.imdb.com/search/title/?groups=top_250&sort=user_rating,desc&sta'
url3 = 'https://www.imdb.com/search/title/?groups=top_250&sort=user_rating,desc&sta'
url4 = 'https://www.imdb.com/search/title/?groups=top_250&sort=user_rating,desc&sta'
url5 = 'https://www.imdb.com/search/title/?groups=top_250&sort=user_rating,desc&sta'
url_list = [url1, url2, url3, url4, url5]

rank_list = []
title_list = []
year_list = []
runtime_list = []
genre_list = []
director_list = []
rating_list = []

for url_i in url_list:
    url = url_i
    req = requests.get(url)
    html = req.text
    soup = BeautifulSoup(html, "html.parser")

    for i in range(0,50):
        rank = re.sub('\.', '', soup.select(".lister-item-index")[i].text)
        title = soup.select(".lister-item-header > a")[i].text
        year = re.sub('\(|\)', '', soup.select(".lister-item-year")[i].text)
        runtime = float(re.sub('min', ' ', soup.select(".runtime")[i].text))
        genre = re.sub('\n', '', soup.select(".genre")[i].text)
        director = soup.select("p > a:nth-of-type(1)")[i].text
        rating = float(soup.select(".inline-block > strong")[i].text)
        rank_list.append(rank)
        title_list.append(title)
        year_list.append(year)
        runtime_list.append(runtime)
        genre_list.append(genre)
        director_list.append(director)
        rating_list.append(rating)
        # print(i)

movie_list = zip(rank_list, title_list, year_list, runtime_list, genre_list, director_list, rating_list)
movie_info_list = []
for movie in movie_list:
    movie_dict = {
        "rank": movie[0],
        "title": movie[1],
        "year": movie[2],
        "runtime": movie[3],
        "genre": movie[4],

```

```

        "director": movie[5],
        "rating": movie[6]
    }

    movie_info_list.append(movie_dict)

# movie_info_list

Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: urllib3!=1.25.0,!1.25.1,<1.26,>=1.21.1 in /usr
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/c
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/

# 1) 가장 빈도가 높은 상위 3개 연도(year)는 언제인가?
import pandas as pd

len(movie_info_list)
movie_data = pd.DataFrame(movie_info_list)
movie_data
movie_data.groupby(["year"]).count()
print(movie_data["year"].value_counts()[:3])
### 1) 답: 1995, 2004, 2009

1995      8
2004      7
2009      6
Name: year, dtype: int64

# 2) 연도별 rating의 평균을 이용해서 평균치가 가장 높은 연도와 평균치를 구하시오.
newdata = movie_data[['year','rating']]
new2 = newdata.groupby(['year']).mean()
# type(new2) => DataFrame
new2.sort_values('rating', ascending=False)[:1] #rating 순으로, descending(내림차순 정렬)
print(new2.sort_values('rating', ascending=False)[:1])
### 2) 답: 1972, 9.2

```

	rank		title	year	runtime	genre
0	1		The Shawshank Redemption	1994	142.0	Drama
1	2		The Godfather	1972	175.0	Crime, Drama

# 3) 가장 빈도가 높은 장르(genre)는 무엇이며 몇 편인가?

# (단, 여러 장르 조합은 하나의 별도의 장르로 구분한다)

```
movie_data.groupby(["genre"]).count()
print(movie_data["genre"].value_counts()[ :1])
#### 3) 답: Drama, 20편
```

```
Drama          20
Name: genre, dtype: int64
```

```
216      217      Aladdin      1992      90.0      Animation Adventure Comedy
```

# 4) 6편 이상을 제작한 감독(director)는 누구인가?

```
print(movie_data["director"].value_counts() >= 6)
```

#### 4) 답: Christopher Nolan, Akira Kurosawa, Stanley Kubrick, Martin Scorsese, Steven Spielberg

```
Christopher Nolan      True
Akira Kurosawa         True
Stanley Kubrick        True
Martin Scorsese        True
Steven Spielberg       True
...
Florian Zeller         False
Jon Watts               False
Asghar Farhadi          False
Thomas Vinterberg      False
Kevin Costner           False
Name: director, Length: 161, dtype: bool
```

# 5) 상영시간(runtime)이 70분 미만인 작품은 총 몇 편인가?

```
print(len(movie_data[movie_data['runtime']<70]))
```

3

##### 2번 전처리 과정 #####

```
!pip install requests
import requests
from bs4 import BeautifulSoup
import re
import pandas as pd
url = "https://movie.naver.com/movie/sdb/rank/rmovie.nhn?sel=pnt&date=20201108"
req = requests.get(url)
html = req.text
soup = BeautifulSoup(html, "html.parser")
```

```
[>] Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: urllib3!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages
```

# 1) 2개의 변수(영화명, 평점, 상세url)를 포함하는 data frame을 만드는 코드를 아래 명령문을 이용하여 작성

```
# for (a,b) in zip(A, B)
titles = soup.select(".tit5 a")
ratings = soup.select(".point")
title_list = []
url_list = []
rating_list = []
for (title, rate) in zip(titles, ratings):
    title_list.append(title.text)
    url_list.append(title["href"])
    rating_list.append(rate.text)

movie_list1 = zip(title_list, url_list, rating_list)
movie_data = pd.DataFrame(movie_list1)
print(movie_data)
```

	0	1	2
0	그린 북	/movie/bi/mi/basic.naver?code=171539	9.59
1	가버나움	/movie/bi/mi/basic.naver?code=174830	9.59
2	디지몬 어드벤처 라스트 에볼루션 : 인연	/movie/bi/mi/basic.naver?code=192613	9.
3	먼 훗날 우리	/movie/bi/mi/basic.naver?code=175092	9.54
4	베일리 어게인	/movie/bi/mi/basic.naver?code=144906	9.53
5	부활: 그 증거	/movie/bi/mi/basic.naver?code=194334	9.53
6	아일라	/movie/bi/mi/basic.naver?code=169240	9.50
7	원더	/movie/bi/mi/basic.naver?code=151196	9.50
8	포드 v 페라리	/movie/bi/mi/basic.naver?code=181710	9.49
9	당갈	/movie/bi/mi/basic.naver?code=157243	9.48
10	주전장	/movie/bi/mi/basic.naver?code=179518	9.48
11	쇼생크 탈출	/movie/bi/mi/basic.naver?code=17421	9.44
12	터미네이터 2:오리지널	/movie/bi/mi/basic.naver?code=10200	9.44
13	덕구	/movie/bi/mi/basic.naver?code=154667	9.42
14	보헤미안 랍소디	/movie/bi/mi/basic.naver?code=156464	9.42
15	라이언 일병 구하기	/movie/bi/mi/basic.naver?code=18988	9.42
16	월-E	/movie/bi/mi/basic.naver?code=69105	9.41
17	나 홀로 집에	/movie/bi/mi/basic.naver?code=10016	9.41
18	클래식	/movie/bi/mi/basic.naver?code=35939	9.41
19	그대, 고맙소 : 김호중 생애 첫 팬미팅	무비 /movie/bi/mi/basic.naver?code=196828	9.
20	헬프	/movie/bi/mi/basic.naver?code=82432	9.40
21	사운드 오브 뮤직	/movie/bi/mi/basic.naver?code=10102	9.40
22	매트릭스	/movie/bi/mi/basic.naver?code=24452	9.40
23	인생은 아름다워	/movie/bi/mi/basic.naver?code=22126	9.40
24	살인의 추억	/movie/bi/mi/basic.naver?code=35901	9.40
25	포레스트 검프	/movie/bi/mi/basic.naver?code=17159	9.40
26	뽕 투 더 퓨처	/movie/bi/mi/basic.naver?code=10002	9.40
27	위대한 쇼맨	/movie/bi/mi/basic.naver?code=106360	9.40
28	소년시절의 너	/movie/bi/mi/basic.naver?code=192066	9.40
29	글래디에이터	/movie/bi/mi/basic.naver?code=29217	9.40
30	센과 치히로의 행방불명	/movie/bi/mi/basic.naver?code=32686	9.39
31	타이타닉	/movie/bi/mi/basic.naver?code=18847	9.39
32	토이 스토리 3	/movie/bi/mi/basic.naver?code=66463	9.39
33	어벤저스: 엔드게임	/movie/bi/mi/basic.naver?code=136900	9.38
34	알라딘	/movie/bi/mi/basic.naver?code=163788	9.38
35	헌터 킬러	/movie/bi/mi/basic.naver?code=92125	9.37
36	아이즈 온 미 : 더 무비	/movie/bi/mi/basic.naver?code=189027	9.37
37	죽은 시인의 사회	/movie/bi/mi/basic.naver?code=10048	9.37
38	캐스트 어웨이	/movie/bi/mi/basic.naver?code=31162	9.37
39	레옹	/movie/bi/mi/basic.naver?code=17170	9.37
40	동주	/movie/bi/mi/basic.naver?code=134899	9.37
41	반지의 제왕: 왕의 귀환	/movie/bi/mi/basic.naver?code=31796	9.37

42	히든 피겨스	/movie/bi/mi/basic.naver?code=147092	9.37
43	아이 캔 스피크	/movie/bi/mi/basic.naver?code=161850	9.37
44	브레이크 더 사일런스: 더 무비	/movie/bi/mi/basic.naver?code=195975	9.36
45	집으로...	/movie/bi/mi/basic.naver?code=34324	9.36
46	안녕 베일리	/movie/bi/mi/basic.naver?code=181700	9.36
47	썬들러 리스트	/movie/bi/mi/basic.naver?code=14450	9.36
48	서유기 2 - 선리기연	/movie/bi/mi/basic.naver?code=18543	9.36
49	클레멘타인	/movie/bi/mi/basic.naver?code=37886	9.36

##### 2-2번 전처리 과정 #####

```

movie_site = "https://movie.naver.com"
score_list = []
type_list = []
country_list = []
run_time_list = []
open_time_list = []
#####
for url_i in url_list:
    html_new = requests.get(movie_site + url_i).text
    soup_new = BeautifulSoup(html_new, "html.parser")
    try:
        score = float(re.sub(r'^0-9', '', soup_new.select(".star_score > span > spa
    except:
        score = None
    type1 = soup_new.select(".info_spec dd > p > span")[0].text
    type1 = re.sub('\t|\r|\n', '', type1)
    country = soup_new.select(".info_spec dd > p > span")[1].text
    country = re.sub('\t|\r|\n', '', country)
    try:
        run_time = soup_new.select(".info_spec dd > p > span")[2].text
        run_time = re.sub('\t|\r|\n', '', run_time)
        run_time = float(re.sub(r'^0-9', '', run_time))
    except:
        run_time = None
    try:
        open_time = soup_new.select(".info_spec dd > p > span")[3].text
        open_time = re.sub('\t|\r|\n|\s', '', open_time)[0:4]
    except:
        open_time = None
    score_list.append(score)
    type_list.append(type1)
    country_list.append(country)
    run_time_list.append(run_time)
    open_time_list.append(open_time)
    # print(len(score_list))
##### 영화 정보 작성 (title, score, type, country, run_time, open_time 묶음)
movie_list = zip(title_list, score_list, type_list, country_list, run_time_list, op
movie_info_list = []

```

```

for movie in movie_list:
    movie_dict = {
        "title": movie[0],
        "score": movie[1],
        "type": movie[2],
        "country": movie[3],

```

```

        "run_time": movie[4],
        "open_time": movie[5]
    }
    movie_info_list.append(movie_dict)
movie_info_list


import pandas as pd
movie_data2 = pd.DataFrame(movie_info_list)

```

```

# 2) 장르가 [다큐멘터리]인 영화의 제목은 무엇인가?
newData5 = movie_data[['title','type']]
newData5[newData5['type'] == '다큐멘터리']
### 2) 답: 부활:그증거, 주전장, 브레이크 더 사일런스:더무비

```

	title	type	
5	부활: 그 증거	다큐멘터리	
10	주전장	다큐멘터리	
44	브레이크 더 사일런스: 더 무비	다큐멘터리	

```

# 3) 영화국적의 빈도가 가장 높은 나라는?
movie_data2["country"].value_counts()[:1]
### 3) 답: 미국

```

```

미국      25
Name: country, dtype: int64

```

```

# 4) 개봉연도 기준으로 관람객평점의 평균이 가장 높은 [1] 개봉연도와 [2] 평균평점을 구하시오.
# 여기서, 관람객평점 결측치가 있는 연도는 데이터에서 제외시킨 뒤 계산 하시오.
# (힌트: data.dropna(axis=0) 이용)
newData3 = movie_data2[['open_time','score']]
new3 = newData3.groupby(['open_time']).mean()
new3.sort_values('score',ascending=False).dropna(axis=0)
### 4) 답: 2003, 972

```

**score** 

```
# 5) 장르 기준으로 상영시간의 평균이 가장 긴 장르를 구하시오.  
newData4 = movie_data2[['type','run_time']]  
new4 = newData4.groupby(['type']).mean()  
new4.sort_values('run_time',ascending=False)  
### 5) 답: 판타지, 모험, 액션, 전쟁
```

run\_time 

type	
판타지, 모험, 액션, 전쟁	263.000000
액션, SF	181.000000
멜로/로맨스, 뮤지컬, 드라마	172.000000
전쟁, 액션, 드라마	170.000000
멜로/로맨스, 드라마	163.000000
드라마, 액션	161.000000
드라마, 전쟁	159.500000
드라마, 모험	143.000000
SF, 액션, 스릴러	137.000000
SF, 액션	136.000000
액션, 드라마	135.333333
멜로/로맨스, 드라마, 범죄, 가족	135.000000
범죄, 액션, 드라마	132.000000
범죄, 미스터리, 스릴러, 코미디, 드라마	132.000000
드라마, 코미디	129.000000
모험, 가족, 판타지, 뮤지컬, 멜로/로맨스	128.000000
애니메이션, 판타지, 모험, 가족	126.000000

✓ 0초

오후 9:36에 완료됨

×