

왓차 데이터 분석

(Analyzing Watcha Data)

202144095 이서영

- 목 차 -

1	개 요.....	4
2	데이터 수집.....	4
2.1	수집 방법.....	4
2.2	개발 환경.....	4
2.3	왓차 크롤링(1)(seleni_watcha_id_type.py, watcha_id_type.csv)	5
2.4	왓차 크롤링(2)(seleni_watcha_epnum_rtime.py, watcha_tv_info.csv).....	6
2.5	왓차피디아 크롤링(seleni_wpedia_info.py, watcha_content_info.csv)	8
3	데이터 전처리(data_preprocessing.py).....	10
3.1	TV 프로그램의 에피소드 수와 상영시간 중복 데이터 처리(watcha_tv_info.csv).....	11
3.2	콘텐츠의 상세정보 중복 데이터 처리(watcha_content_info.csv)	12
3.3	콘텐츠 정보 split하고 merge(watcha_content_info.csv, + watcha_id_type.csv)	13
3.4	콘텐츠 정보와 TV 프로그램 정보 merge(split_merge.csv, + watcha_tv_info.csv).....	17
3.5	데이터 형식 및 누락 데이터 처리	17
4	데이터 분석 및 시각화 및 결론(data_analysis.py).....	19
4.1	데이터 정보 확인	19
4.2	종류별 작품 수.....	20
4.3	국가별 작품 수(상위 15개)	21
4.4	장르별 작품 수.....	24
4.5	개봉년도 구간별 작품 수.....	28
4.6	연령 등급별 작품수	30
4.7	종류별 작품 수(평균 평점 4.3 이상).....	33

4.8	연령 등급별 작품 수(평균 평점 4.3 이상)	34
4.9	장르별 작품 수(평균 평점 4.3 이상)	35
4.10	제작 국가별 작품 수(평균 평점 4.3 이상)	37
4.11	개봉년도별 작품 수(평균 평점 4.3 이상)	38
4.12	결론	39

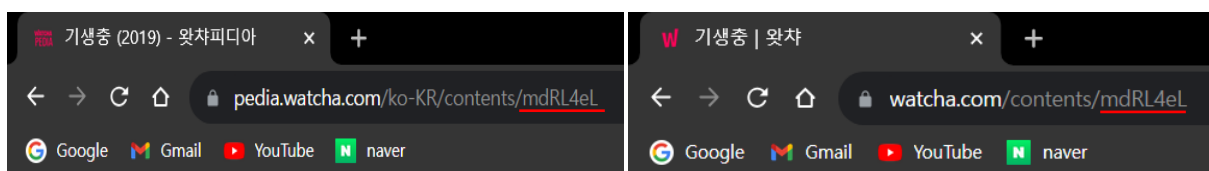
1 개 요

왓차 데이터를 분석을 통해 왓차라는 플랫폼에 어떤 콘텐츠가 있는지 분석하고, 평균 평점 정보를 통해 사람들이 어떤 콘텐츠를 선호하는지 알아본다.

2 데이터 수집

2.1 수집 방법

왓차 웹사이트와 왓차피디아 웹사이트는 동일한 콘텐츠를 나타내는 id 값이 동일하다. 이점을 활용하여 왓차와 왓차피디아에 등록되어있는 콘텐츠의 정보를 셀레니움을 사용하여 수집한다.



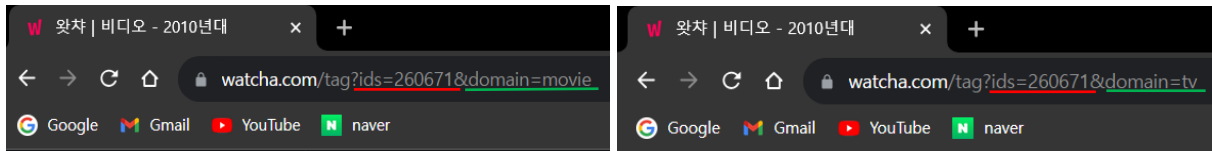
왓차 웹사이트에서는 콘텐츠 id, 종류, TV 프로그램의 에피소드 수와 상영시간 정보를 가져오고, 왓차피디아 웹사이트에서는 모든 콘텐츠의 상세 정보를 가져온다.

2.2 개발 환경

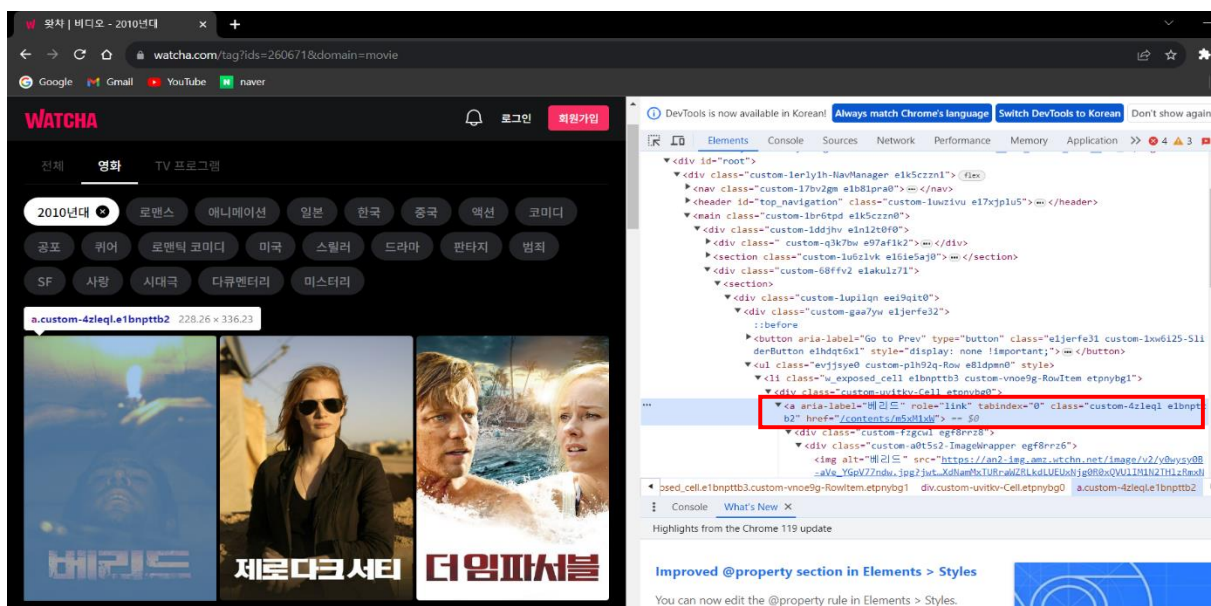
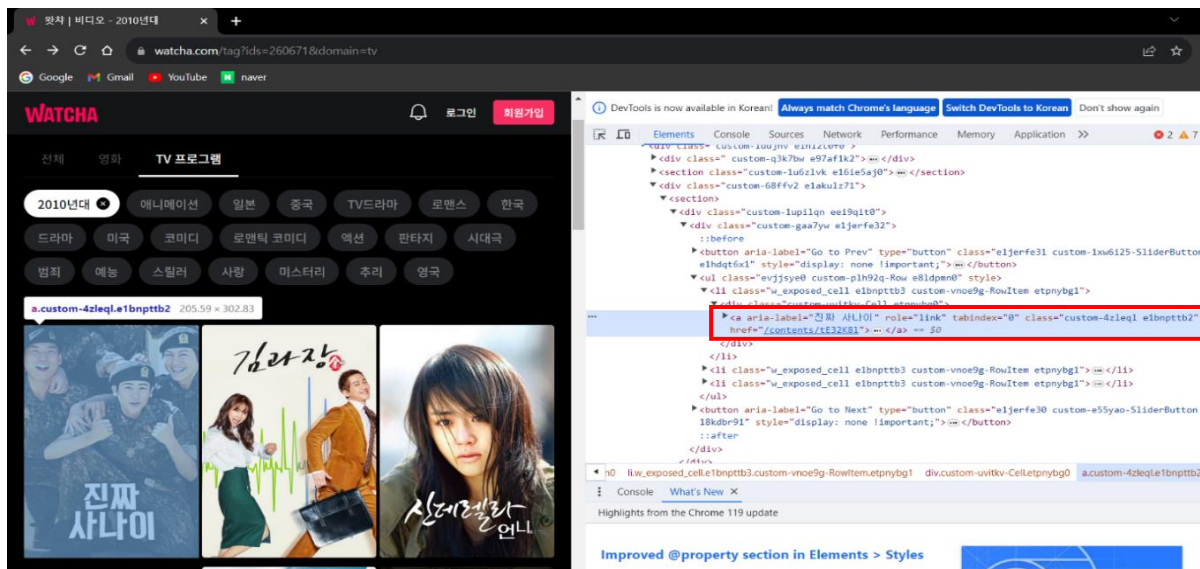
운영체제	Windows 11 Home
개발언어	Python(3.11.5)
개발도구	Spyder(5.4.3), Selenium(4.14.0)
개발환경	ANACONDA(23.7.4)

2.3 왓차 크롤링(1)(seleni_watcha_id_type.py, watcha_id_type.csv)

왓차에서는 콘텐츠 종류(domain)와 해당하는 각 년도별 태그(ids)가 존재하며, 이를 활용하여 모든 콘텐츠의 id값과 종류를 크롤링하고 csv 파일에 저장한다.



스크롤을 맨 밑으로 내린 후 각 년대에 해당하는 종류의 목록에서 각 콘텐츠의 a 태그의 href 속성을 가져오고 split하여 콘텐츠의 id값을 가져온다.

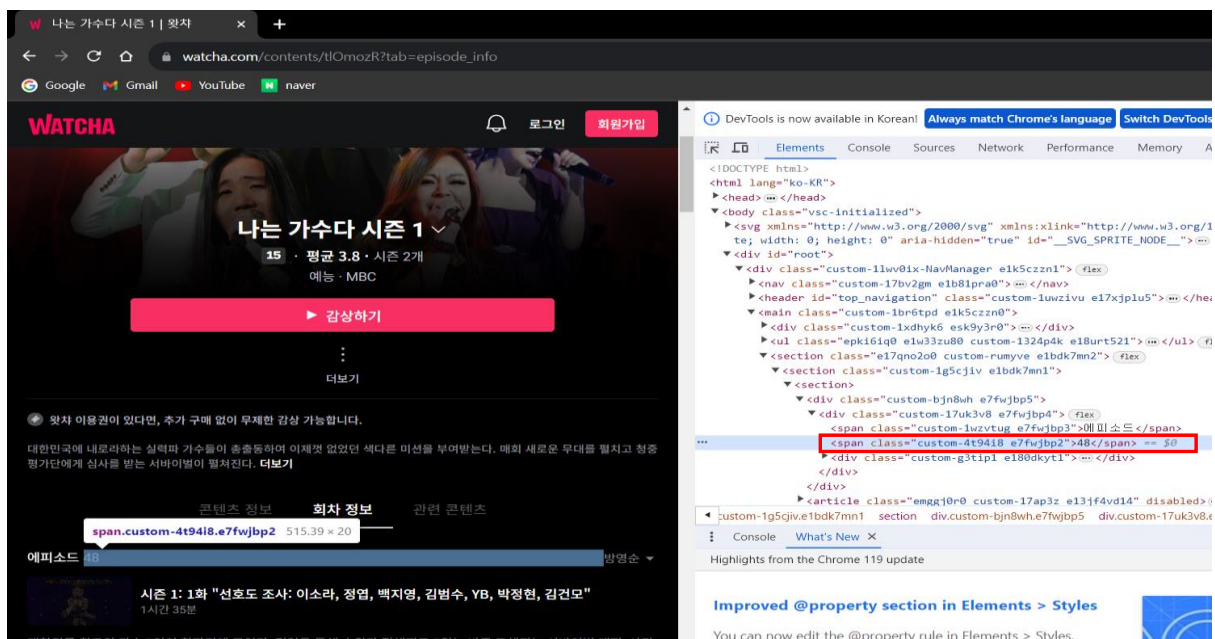


2.4 왓차 크롤링(2)(seleni_watcha_epnum_rtime.py, watcha_tv_info.csv)

TV 프로그램에 해당하는 상영시간과 에피소드 수는 왓차에만 존재한다. 따라서 앞서 수집한 데이터에서 종류가 TV 프로그램인 id만 추출하여 TV프로그램의 상영시간과 에피소드 수를 크롤링하고 csv 파일에 저장한다. 이때, 상영시간은 첫 화를 기준으로 한다.

셀레니움을 사용하여 크롤링하기 때문에 상당 시간이 소요되어 안정성을 높이기 위해 100개씩 데이터를 크롤링하고 동일한 csv 파일에 저장하여 데이터를 쌓는다. 또한 100개를 크롤링한 후 WebDirver 인스턴스를 종료 후 다시 생성하며, 전체를 한 번 크롤링한 후 요소를 찾지 못한 id를 추출하여 최대 10번 다시 크롤링한다. 이때, 다시 크롤링한 데이터도 동일한 csv에 저장하고 후에 전처리한다.

그리고 많은 수의 데이터를 크롤링하므로 봇으로 간주되어 차단당하지 않기 위해 랜덤 시간동안 대기하는 코드를 추가한다.



나는 가수다 시즌 1 | 왓차

watcha.com/contents/1lOm0zR?tab=episode_info

Google Gmail YouTube naver

WATCHA

로그인 회원가입

나는 가수다 시즌 1

15 · 평균 3.8 · 시즌 2개

예능 · MBC

▶ 감상하기

더보기

왓차 이용권이 있다면, 추가 구매 없이 무제한 감상 가능합니다.

대한민국에 내로라하는 실력과 가수들이 총출동하여 이제껏 없었던 색다른 미션을 부여받는다. 매회 새로운 무대를 펼치고 청중평가단에게 심사를 받는 서바이벌이 펼쳐진다. **더보기**

콘텐츠 정보 **회사 정보** 관련 콘텐츠

에피소드 48

span.custom-1kx6jlp.e13jf4vd3 56.31 × 18

김범수, YB, 박정현, 김건모

대한민국 최고의 가수 7명이 한자리에 모인다. 경연을 통해 순위가 정해지고 7위는 바로 교체되는 서바이벌 대전. 사전 선호도 조사 1위를 차지하기 위한 가수들의 자존심을 담은 무대가 펼쳐진다.

DevTools is now available in Korean! Always match Chrome's language Switch DevTools to Korean Don't show again

Elements Console Sources Network Performance Memory Application

<div class="custom-bjn8uh e7fw3bp5"></div>

<article class="emggj0r0 custom-17ap3z e13jf4vd14" disabled></article>

<div class="custom-1n4s4j4 e13jf4vd11"></div>

<div disabled class="custom-ns1fi2 e13jf4vd10"></div>

<div class="custom-82a6rk e13jf4vd6">

<div class="custom-kpnix e13jf4vd5">

<p class="custom-ulsr2d e13jf4vd4">시즌 1: 1화 "선호도 조사: 이소라, 정엽, 박지영, 김범수, YB, 박정현, 김건모"</p>

1시간 39분 </div>

<p class="custom-kyc8f1 e13jf4vd2"></p>

</div>

<p class="custom-its1ej e13jf4vd1"></p>

</article>

<article class="emggj0r0 custom-17ap3z e13jf4vd14" disabled></article>

<div class="custom-1n729k6 e1bjxyx3"></div>

<div class="custom-1n729k6 e1bjxyx3"></div>

<div class="custom-1n729k6 e1bjxyx3"></div>

<div class="custom-1n729k6 e1bjxyx3"></div>

stom-1n4s4j4.e13jf4vd11 div.custom-82a6rk.e13jf4vd6 div.custom-kpnix.e13jf4vd5 span.custom-1kx6jlp.e13jf4vd3

Console What's New X

Highlights from the Chrome 119 update

Improved @property section in Elements > Styles

2.5 왓차피디아 크롤링(seleni_wpedia_info.py, watcha_content_info.csv)

왓차피디아에서는 왓차에서 크롤링한 모든 콘텐츠의 id를 가지고 콘텐츠의 상세정보와 평균 평점, 줄거리, 제작진 정보를 크롤링하고 csv 파일에 저장한다.

전체적인 구조는 앞서 왓차에서 TV 프로그램의 에피소드 수와 상영시간과 같다. 100개씩 크롤링 하며, 전체를 한 번 크롤링한 후 누락된 요소가 있는 id만 추출하고 최대 10번 다시 크롤링한다.

콘텐츠의 상세정보에는 제목, 개봉년도, 장르, 국가, 상영시간(영화), 연령 등급, 방송사(TV 프로그램) 정보가 포함되어 있다. 이후 전처리에서 split한다.

The screenshot shows the Watcha Pedia website for the movie '기생충' (Parasite). The page displays the movie title, a rating of 4.3, and a synopsis. The DevTools console on the right shows the HTML structure of the page, with a red box highlighting the following code snippet:

```
<div class="css-oext2p elyew28612">기생충</div>
<div class="css-12psq2j elyew28611">기생충</div>
<div class="css-12psq2j elyew28611">2019 · 드라마 · 한국</div>
<div class="css-12psq2j elyew28611">2시간 11분 · 15세</div>
</div>
<section class="css-d1nd9e esh6962">
</section>
</div>
<div class="css-2tqido elyew2869">
</div>
</div>
```

The screenshot shows the Watcha Pedia website for the movie '기생충' (Parasite). The page displays the movie title, a rating of 4.3, and a synopsis. The DevTools console on the right shows the HTML structure of the page, with a red box highlighting the following code snippet:

```
<section class="css-291cts elyew2864">
</section>
</div>
<div class="css-2tqido elyew2869">
<section class="css-3btvmx elyew2868">
<div class="css-7ckx12 elyew2867">
<div class="css-g8juaf elyew2865">
<section class="css-1ha5mfc elyew2864">
</section>
</div>
</section>
</div>
<div class="css-1jmb7hb elyew2862">
</div>
</div>
```


기생충 (2019) - 왓치피디아

WATCHA PEDIA 영화 TV 책 웹툰

기생충

2019 · 드라마 · 한국

2시간 11분 · 15세

평가하기

4.3

평균 평점 (121.7만명)

“배 끼치고 싶진 않았어요”

DevTools is now available in Korean! Always match Chrome's language Switch DevTools to Korean Don't show again

Elements Console Sources Network Performance Memory Application

Improved @property section in Elements > Styles

You can now edit the @property rule in Elements > Styles.

Updated list of devices

기생충 (2019) - 왓치피디아

WATCHA PEDIA 영화 TV 책 웹툰

매월 1일, 100원으로 누리는 최고의 가치!

100원에 평생 소장

리브러터 100원

코멘트 20000+

더보기

DevTools is now available in Korean! Always match Chrome's language Switch DevTools to Korean Don't show again

Elements Console Sources Network Performance Memory Application

Improved @property section in Elements > Styles

You can now edit the @property rule in Elements > Styles.

Updated list of devices

3 데이터 전처리(data_preprocessing.py)

크롤링한 데이터를 전처리한다. 여러 번의 크롤링 결과를 각 하나의 csv 파일에 저장하였으므로 동일한 id에 결측치 개수가 다른 행이 여러개 존재한다. 이를 처리하고, 데이터를 split하거나 동일한 형식으로 수정하며, 앞서 수집한 3개의 csv 파일을 id를 기준으로 merge한다.

Import 코드는 다음과 같다.

```
import pandas as pd
import numpy as np
import re # 구분자를 2 개 이상으로 자르기 위한 import, 문자열에서 원하는 패턴을 검색하거나
대체하는데 사용
```

3.1 TV 프로그램의 에피소드 수와 상영시간 중복 데이터 처리(watcha_tv_info.csv)

왓차에서 수집해온 TV 프로그램의 에피소드 수와 상영시간은 결측치가 없는 레코드를 모두 수집하였다. 따라서 결측치가 있는 행을 drop하고 동일한 csv 파일에 덮어쓴다.

```
# watcha_tv_info.csv 전처리(tv 프로그램의 에피소드 수와 상영시간)
# columns=['id', '에피소드 수', '상영시간']
# 결측치가 없는 레코드를 모두 수집했으므로 결측치가 있는 행 drop

# 1. csv 파일 읽어오기
tv_info_df = pd.read_csv('C:/Crawling_Watcha/csv/watcha_tv_info.csv', encoding='cp949')

# 2. ? 값을 NaN(결측치)으로 치환
tv_info_df.replace('?', pd.NA, inplace = True)

# 3. 결측치가 있는 행 제거
tv_info_df.dropna(inplace = True)

# 4. watcha_tv_info.csv 에 덮어쓰기
tv_info_df.to_csv('C:/Crawling_Watcha/csv/watcha_tv_info.csv', encoding='cp949', mode='w',
index=False)

# 5. 개수 비교
id_type_df = pd.read_csv('C:/Crawling_Watcha/csv/watcha_id_type.csv', encoding='cp949')
mask = id_type_df['종류'] == 'tv'
tv_id_type_df = id_type_df.loc[mask,:]

print('나와야 하는 tv 프로그램 수 : ' + str(tv_id_type_df['id'].count()))
print('전처리한 tv 프로그램 수 : ' + str(tv_info_df['id'].count()))
```

3.2 콘텐츠의 상세정보 중복 데이터 처리(watcha_content_info.csv)

왓차피디아에서 수집해온 정보에는 결측치가 있는 레코드가 존재하였다. 따라서 한 레코드당 결측치가 제일 적은 레코드만 남기고 drop 한다.

```
# watcha_content_info.csv 전처리
# columns=['id', '컨텐츠 정보','평균 평점','줄거리','제작진']

# 1. 결측치가 있기 때문에 한 레코드당 결측치가 제일 적은 레코드만 남기고 drop

# 1-1. csv 파일 읽어오기
content_info_df = pd.read_csv('C:/Crawling_Watcha/csv/watcha_content_info.csv',
encoding='utf-8-sig')

# 1-2. ? 값을 NaN(결측치)으로 치환
content_info_df.replace('?', pd.NA, inplace = True)

# 1-3 각 id 그룹 내에서 나머지 칼럼들의 누락된 값이 가장 작은 레코드만 남도록 처리
def keep_row_with_least_nulls(group):

    # ID를 제외한 나머지 칼럼들에 대해 누락된 값의 개수를 계산
    null_counts = group.iloc[:, 1:].isnull().sum(axis=1)

    # 누락된 값의 개수가 가장 작은 인덱스를 반환
    min_null_count_idx = null_counts.idxmin()

    # 누락된 값이 가장 작은 레코드를 반환
    return group.loc[min_null_count_idx]

# id를 기준으로 그룹화하여 각 그룹에 대해 함수를 적용하고 결과를 저장
content_info_df =
content_info_df.groupby('id').apply(keep_row_with_least_nulls).reset_index(drop=True)

# 1-4 watcha_tv_info.csv에 덮어쓰기
content_info_df.to_csv('C:/Crawling_Watcha/csv/watcha_content_info.csv',encoding='utf-8-
sig', mode='w', index=False)

# 1-5 개수 비교
id_type_df = pd.read_csv('C:/Crawling_Watcha/csv/watcha_id_type.csv', encoding='cp949')
print('나와야 하는 콘텐츠 수 : ' + str(id_type_df['id'].count()))
print('전처리한 콘텐츠 수 : ' + str(content_info_df['id'].count()))
```

3.3 콘텐츠 정보 split하고 merge(watcha_content_info.csv, + watcha_id_type.csv)

왓차피디아에서 크롤링해온 콘텐츠 정보를 split해서 재저장한다. 이때, 영화와 TV 프로그램의 정보가 다르므로 구분해서 처리하기 위해 watcha_id_type.csv 파일과 merge하여 처리한다.

또한 split하여 해당하는 각 정보 리스트에 저장할 때, 특정 정보가 웹사이트에서 처음부터 누락되어 있을 수도 있다. 이를 위한 예외처리를 길이와 연령정보를 사용하여 처리한다.

split한 데이터를 이용하여 만든 데이터프레임과 watcha_id_type에서 가져온 데이터프레임을 merge하여 새로운 csv 파일을 만든다.(split_merge.csv)

처리 후 칼럼은 id, 종류, 평균 평점, 줄거리, 제작진, 제목, 개봉년도, 장르, 국가, 상영시간, 연령등급, 방송국 이렇게 12개이다.

```
# 2. 콘텐츠 정보 split 해서 재저장
# 영화와 TV 프로그램의 정보가 다르므로 구분해서 처리하기 위해 merge

# 2-1 두 csv 파일 merge
merge_df = pd.merge(id_type_df, content_info_df, on='id', how='outer')

# 2-2 필요한 정보만 리스트로 추출
id_data = merge_df['id'].values.tolist()
type_data = merge_df['종류'].values.tolist()
info_data = merge_df['콘텐츠 정보'].values.tolist()

# 2-3 split 한 데이터를 저장할 리스트
titleList = [] # 제목
release_yerList = [] # 개봉년도
genreList = [] # 장르
contryList = [] # 국가
running_timeList = [] # 상영시간
age_gradelist = [] # 연령 등급
tv_stationList = [] # 방송국

# 연령정보 고유값
age_grade = ['전체', '7 세', '12 세', '15 세', '청소년']

# 콘텐츠 정보 split
#for i in range(10):
for i in range(len(id_data)):
```

```

split_info = re.split(r' · |\n', info_data[i])

# 연령 정보가 split 리스트에 있는지 확인
age_found = False
for item in age_grade:
    if item in split_info:
        age_found = True
        break

titleList.append(split_info[0])
release_yerList.append(split_info[2])

if type_data[i] == "tv":

    running_timeList.append(None)

    if len(split_info) >= 7: # 모든 정보가 있는 경우

        if age_found == True: # 방송국 정보만 없는 경우
            tv_stationList.append(split_info[3])
            genreList.append(split_info[4])
            contryList.append(split_info[5])
            age_gradeList.append(split_info[6])

        else:
            tv_stationList.append(split_info[3])
            genreList.append(split_info[4])
            contryList.append(split_info[5])
            age_gradeList.append(None)

    elif len(split_info) == 6:

        if age_found == True: # 방송국 정보만 없는 경우
            tv_stationList.append(None)
            genreList.append(split_info[3])
            contryList.append(split_info[4])
            age_gradeList.append(split_info[5])

        else: # 연령 등급이 없는 경우
            tv_stationList.append(split_info[3])

```

```

        genreList.append(split_info[4])
        contryList.append(split_info[5])
        age_gradeList.append(None)

    else: #len(split_info) == 5:

        tv_stationList.append(None) # 방송국 정보가 없고

        if age_found == True: # 국가정보가 없는 경우
            genreList.append(split_info[3])
            contryList.append(None)
            age_gradeList.append(split_info[4])

        else: # 연령 등급이 없는 경우
            genreList.append(split_info[3])
            contryList.append(split_info[4])
            age_gradeList.append(None)

    else: # 영화인 경우

        tv_stationList.append(None)

        if len(split_info) >= 7: # 모든 정보가 있는 경우

            genreList.append(split_info[3])
            contryList.append(split_info[4])
            running_timeList.append(split_info[5])

            if age_found == True: # 연령 등급이 있는 경우(7 or 10)
                age_gradeList.append(split_info[6])

            else: # 연령 등급이 없는 경우(9)
                age_gradeList.append(None)

        elif len(split_info) == 6:

            if age_found == True: # 상영시간만 없는 경우
                genreList.append(split_info[3])
                contryList.append(split_info[4])

```

```

        running_timelist.append(None)
        age_gradelist.append(split_info[5])

    else: # 연령 등급이 없는 경우
        genrelist.append(split_info[3])
        contrylist.append(split_info[4])
        running_timelist.append([split_info[5]])
        age_gradelist.append(None)

    else: #len(split_info) == 5, 상영시간과 연령정보가 없는 경우

        genrelist.append(split_info[3])
        contrylist.append(split_info[4])
        running_timelist.append(None)
        age_gradelist.append(None)

# merged_df 에 merge
# 데이터프레임 생성
split_df = pd.DataFrame({
    'id': id_data,
    '제목': titleList,
    '개봉년도': release_yerList,
    '장르': genrelist,
    '국가': contrylist,
    '상영시간': running_timelist,
    '연령등급': age_gradelist,
    '방송국': tv_stationList,
})

merge2_df = pd.merge(merge_df, split_df, on='id', how='outer')
merge2_df.drop(labels='컨텐츠 정보',axis=1, inplace=True)

print(merge2_df.count())
print(merge2_df.head(5))

merge2_df.to_csv('C:/Crawling_Watcha/csv/split_merge.csv',encoding='utf-8-sig', mode='w',
index=False)

```


3.4 콘텐츠 정보와 TV 프로그램 정보 merge(split_merge.csv, + watcha_tv_info.csv)

앞서 만든 데이터에 종류가 TV 프로그램인 레코드에 상영시간과 에피소드 수를 합친다.

```
# 모든 데이터 합치기
tv_info_df = pd.read_csv('C:/Crawling_Watcha/csv/watcha_tv_info.csv', encoding='cp949')
content_info_df = pd.read_csv('C:/Crawling_Watcha/csv/split_merge.csv', encoding='utf-8-sig')

final_df = pd.merge(content_info_df, tv_info_df, on='id', how='outer')

# 조건에 따라 상영시간_y와 상영시간 칼럼 값 합치기
final_df['상영시간'] = final_df.apply(lambda x: x['상영시간_y'] if x['종류'] == 'tv' else
x['상영시간_x'], axis=1)
final_df.drop(['상영시간_x', '상영시간_y'], axis=1, inplace=True)
```

3.5 데이터 형식 및 누락 데이터 처리

데이터 형식을 통일하도록 처리하고, 결측치를 처리하여 최종 csv 파일을 생성한다.(watcha.csv)

```
# 밀린 데이터 drop
drop_id = ['m5GX0v2', 'm0LEGLd', 'mOVvmLg', 'tR4JKKy', 'tR72L5x']
final_df = final_df.drop(final_df[final_df['id'].isin(drop_id)].index)

# '연령 등급' 열의 누락 데이터를 바로 앞에 있는 값으로 치환
final_df['연령등급'].fillna(method='ffill', inplace=True)

# '평균 평점' 열의 누락 데이터를 바로 앞에 있는 값으로 치환
final_df['평균 평점'].fillna(method='ffill', inplace=True)

# 결측치를 'unknown'으로 변경
final_df.fillna('unknown', inplace=True)

# 국가와 장르의 첫번째 값만 남기기
final_df['국가'] = final_df['국가'].apply(lambda x: x.split(",")[0])
final_df['장르'] = final_df['장르'].apply(lambda x: x.split("/")[0])

# 상영시간 칼럼에서 '[', ']', '"' 문자 제거하는 함수 정의
def remove_characters(text):
    text = text.replace('[', '').replace(']', '').replace('"', '')
    return text
```

```
# 상영시간 칼럼에 함수 적용하여 문자 제거 후 업데이트
final_df['상영시간'] = final_df['상영시간'].apply(remove_characters)

# '종류' 열의 값을 변경
final_df['종류'] = final_df['종류'].apply(lambda x: 'TV 프로그램' if x == 'tv' else '영화' if
x == 'movie' else x)

final_df.to_csv('C:/Crawling_Watcha/csv/watcha.csv',encoding='utf-8-sig', mode='w',
index=False)
```

4 데이터 분석 및 시각화 및 결론(data_analysis.py)

4.1 데이터 정보 확인

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

plt.rcParams['font.family'] = 'Malgun Gothic' # 한국어 텍스트에 폰트 지정

# csv 파일 읽어오기
watcha = pd.read_csv('C:/Crawling_Watcha/csv/watcha.csv', encoding='utf-8-sig')

# 데이터 정보 확인
watcha.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9659 entries, 0 to 9658
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype  
---  --
 0   id          9659 non-null   object 
 1   종류        9659 non-null   object 
 2   평균 평점    9659 non-null   float64
 3   줄거리      9659 non-null   object 
 4   제작진      9659 non-null   object 
 5   제목        9659 non-null   object 
 6   개봉년도    9659 non-null   int64  
 7   장르        9659 non-null   object 
 8   국가        9659 non-null   object 
 9   연령등급    9659 non-null   object 
10   방송국      9659 non-null   object 
11   에피소드 수 9659 non-null   object 
12   상영시간    9659 non-null   object 
dtypes: float64(1), int64(1), object(11)
memory usage: 981.1+ KB
```

4.2 종류별 작품 수

1. 종류별 작품 수

```
ax = sns.countplot(data=watcha, x='종류')
```

각 막대 위에 수치 표시 (정수형태로)

```
for p in ax.patches:
```

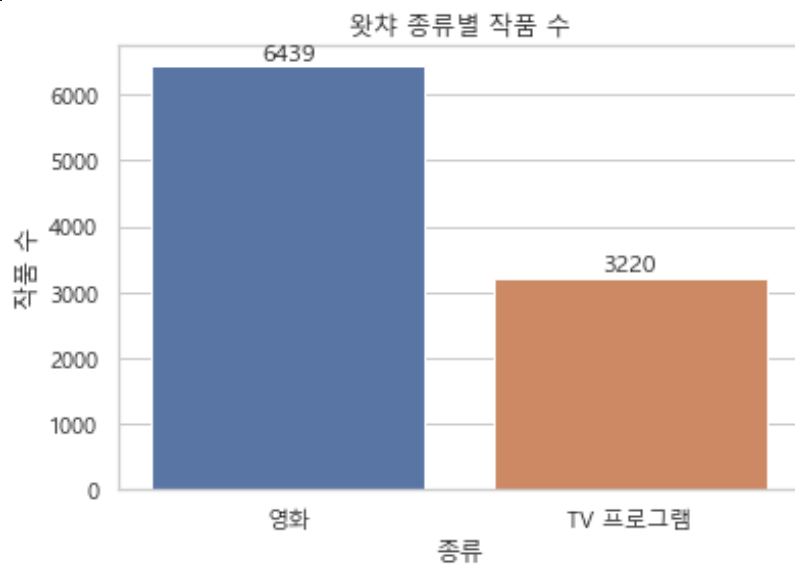
```
    height = p.get_height() # 막대의 높이(데이터 개수)
```

```
    ax.text(p.get_x() + p.get_width() / 2., height, f'{int(height)}', ha='center',  
           va='bottom')
```

```
plt.ylabel('작품 수') # y 축 이름 설정
```

```
plt.title('왓차 종류별 작품 수') # 제목 추가
```

```
plt.show() # 시각화된 plot 보여줌
```



4.3 국가별 작품 수(상위 15개)

- 전체

```
# 2. 제작 국가별 작품 수(상위 15 개)

# 2-1 전체

# 전체 종류 중 제작 국가별 작품 수 상위 15 개 확인
print(watcha['국가'].value_counts().head(15))

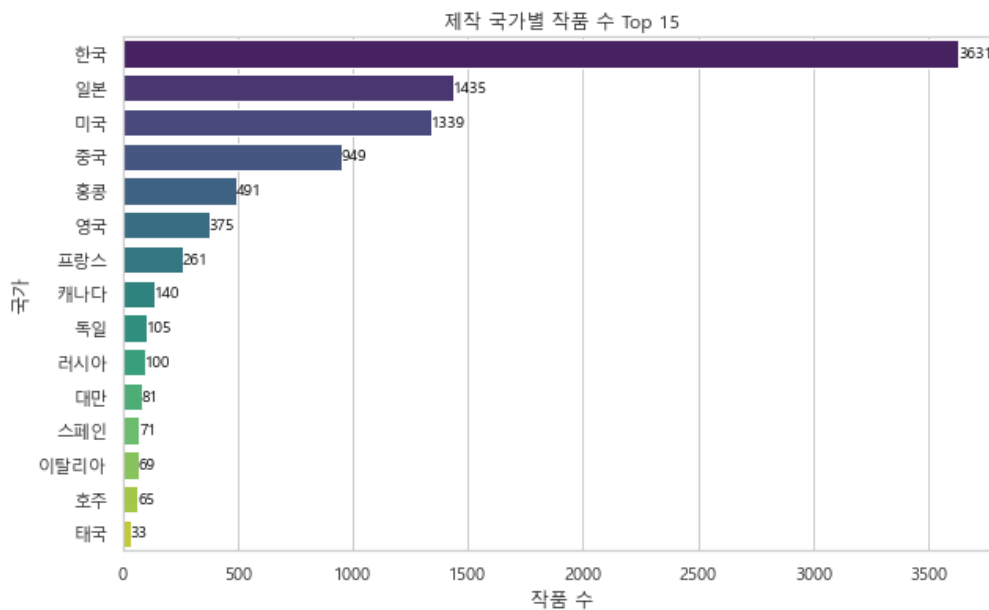
# 국가별 작품 수 계산 후 정렬
country_counts = watcha['국가'].value_counts().head(15).sort_values(ascending=False)

# 가로 막대 그래프로 시각화
plt.figure(figsize=(10, 6)) # 그래프 크기 설정
sns.barplot(x=country_counts.values, y=country_counts.index, palette='viridis')

# 각 막대 안에 수치 표시
for index, value in enumerate(country_counts):
    plt.text(value, index, str(value), va='center', ha='left', fontsize=10, color='black')

plt.xlabel('작품 수')
plt.ylabel('국가')
plt.title('제작 국가별 작품 수 Top 15')

plt.show()
```



- 영화

```
# 2-2 영화

# 종류가 영화인 부분만 가져옴
movie = watcha[watcha['종류'] == '영화']

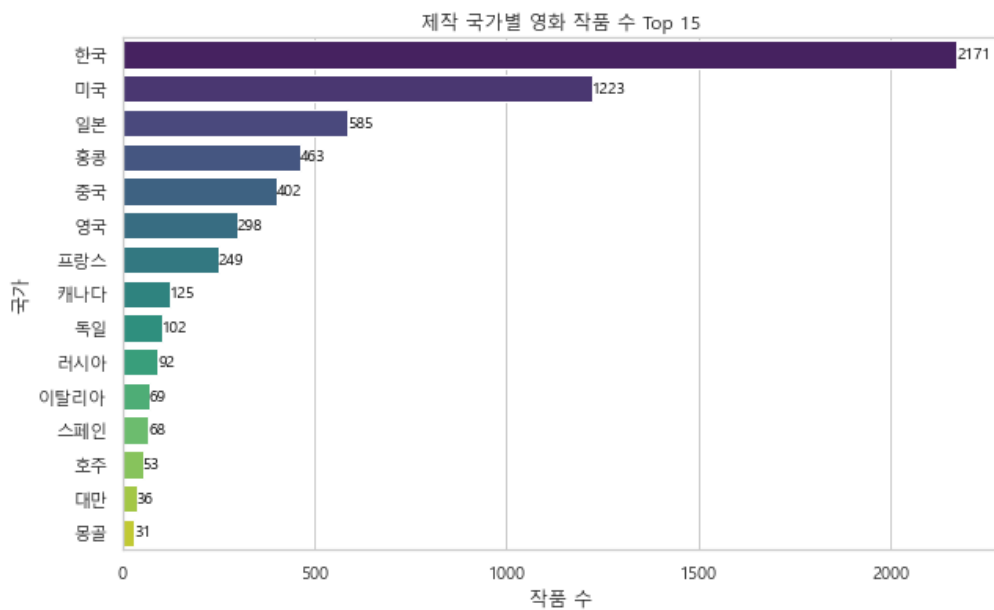
# 영화 중 제작 국가별 작품 수 상위 15 개 확인
print(movie['국가'].value_counts().head(15))

# 국가별 작품 수 계산 후 정렬
country_counts = movie['국가'].value_counts().head(15).sort_values(ascending=False)

# 가로 막대 그래프로 시각화
plt.figure(figsize=(10, 6)) # 그래프 크기 설정
sns.barplot(x=country_counts.values, y=country_counts.index, palette='viridis')

# 각 막대 안에 수치 표시
for index, value in enumerate(country_counts):
    plt.text(value, index, str(value), va='center', ha='left', fontsize=10, color='black')

plt.xlabel('작품 수') # x 축 레이블 설정
plt.ylabel('국가') # y 축 레이블 설정
plt.title('제작 국가별 영화 작품 수 Top 15') # 그래프 제목 설정
plt.show() # 그래프 표시
```



- TV 프로그램

```
# 2-3 TV 프로그램

# 종류가 tv 프로그램인 부분만 가져옴
tv = watcha[watcha['종류'] == 'TV 프로그램']

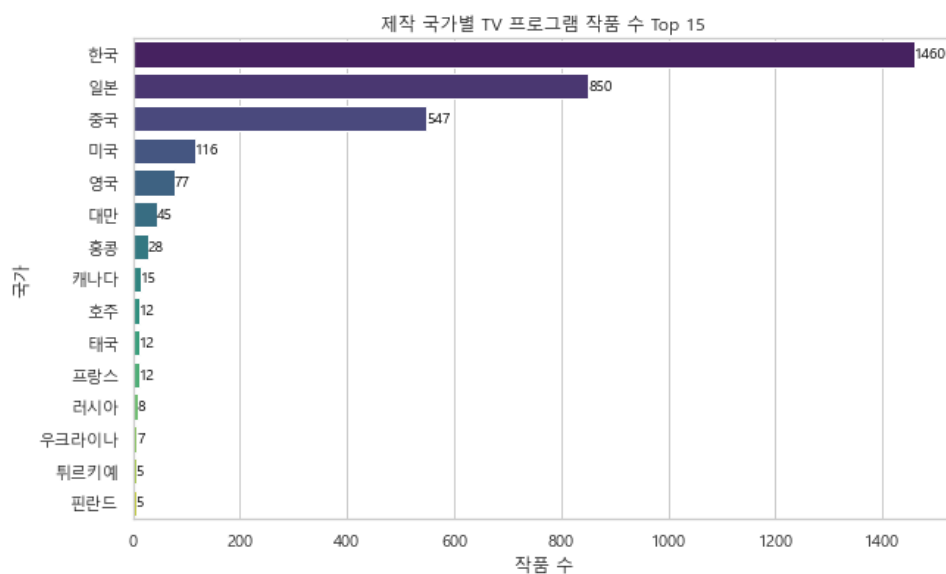
# tv 프로그램 중 제작 국가별 작품 수 상위 15 개 확인
print(tv['국가'].value_counts().head(15))

# 국가별 작품 수 계산 후 정렬
country_counts = tv['국가'].value_counts().head(15).sort_values(ascending=False)

# 가로 막대 그래프로 시각화
plt.figure(figsize=(10, 6)) # 그래프 크기 설정
sns.barplot(x=country_counts.values, y=country_counts.index, palette='viridis')

# 각 막대 안에 수치 표시
for index, value in enumerate(country_counts):
    plt.text(value, index, str(value), va='center', ha='left', fontsize=10, color='black')

plt.xlabel('작품 수') # x 축 레이블 설정
plt.ylabel('국가') # y 축 레이블 설정
plt.title('제작 국가별 TV 프로그램 작품 수 Top 15') # 그래프 제목 설정
plt.show() # 그래프 표시
```



4.4 장르별 작품 수

- 영화

```
# 3. 장르별 작품 수

# 3-1 영화

# 영화만 필터링
movie = watcha[watcha['종류'] == '영화']

# 영화의 장르별 작품 수 계산
genre_counts = movie['장르'].value_counts()

# 작품 수의 비율 계산
genre_percentages = genre_counts / genre_counts.sum() * 100

# 1% 미만인 장르들을 '기타'로 통합
threshold = 1 # 임계치 설정
other_genres = genre_percentages[genre_percentages < threshold].index
movie.loc[movie['장르'].isin(other_genres), '장르'] = '기타'

# 다시 장르별 작품 수 계산
new_genre_counts = movie['장르'].value_counts()

# 새로운 작품 수의 비율 계산
new_genre_percentages = new_genre_counts / new_genre_counts.sum() * 100

# 가장 많은 비율을 차지하는 조각 강조를 위해 explode 설정
max_index = new_genre_percentages.idxmax()
explode = [0.1 if label == max_index else 0 for label in new_genre_percentages.index]

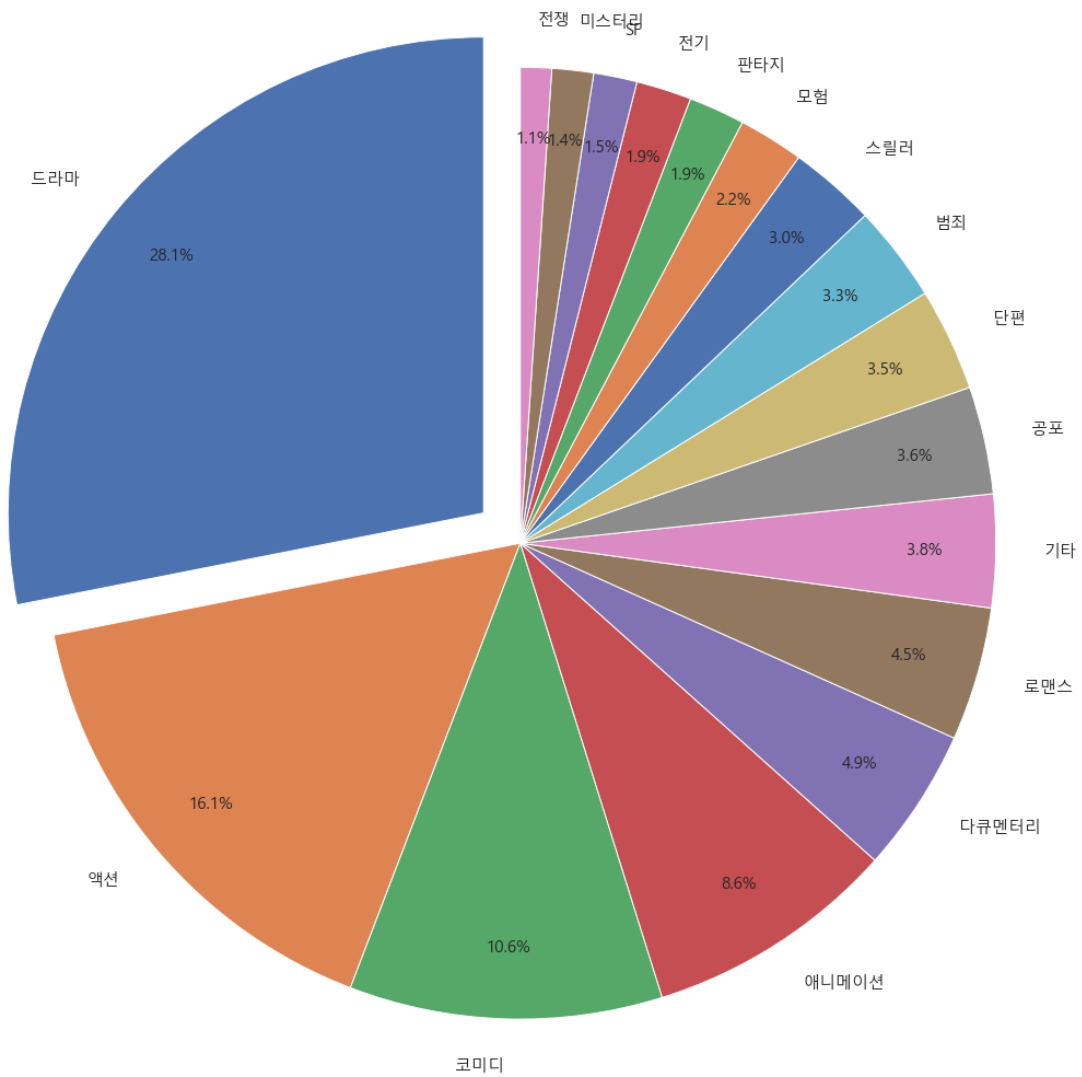
# 원 그래프 그리기 (크기 조절 및 텍스트 설정)
plt.figure(figsize=(20, 20)) # 그래프 크기 설정
patches, texts, autotexts = plt.pie(new_genre_percentages,
labels=new_genre_percentages.index,
autopct='%1.1f%%', startangle=90, pctdistance=0.85,
labeldistance=1.1, explode=explode)
plt.title('장르별 영화 작품 수 비율', fontsize=20) # 그래프 제목 설정
```



```
# 텍스트 사이즈 설정
for text in texts + autotexts:
    text.set_fontsize(15)

plt.show() # 그래프 표시
```

장르별 영화 작품 수 비율



- TV 프로그램

```
# 3-2 TV 프로그램

# TV 프로그램만 필터링
tv = watcha[watcha['종류'] == 'TV 프로그램']

# TV 프로그램의 장르별 작품 수 계산
genre_counts = tv['장르'].value_counts()

# 작품 수의 비율 계산
genre_percentages = genre_counts / genre_counts.sum() * 100

# 1% 미만인 장르들을 '기타'로 통합
threshold = 1 # 임계치 설정
other_genres = genre_percentages[genre_percentages < threshold].index
tv.loc[tv['장르'].isin(other_genres), '장르'] = '기타'

# 다시 장르별 작품 수 계산
new_genre_counts = tv['장르'].value_counts()

# 새로운 작품 수의 비율 계산
new_genre_percentages = new_genre_counts / new_genre_counts.sum() * 100

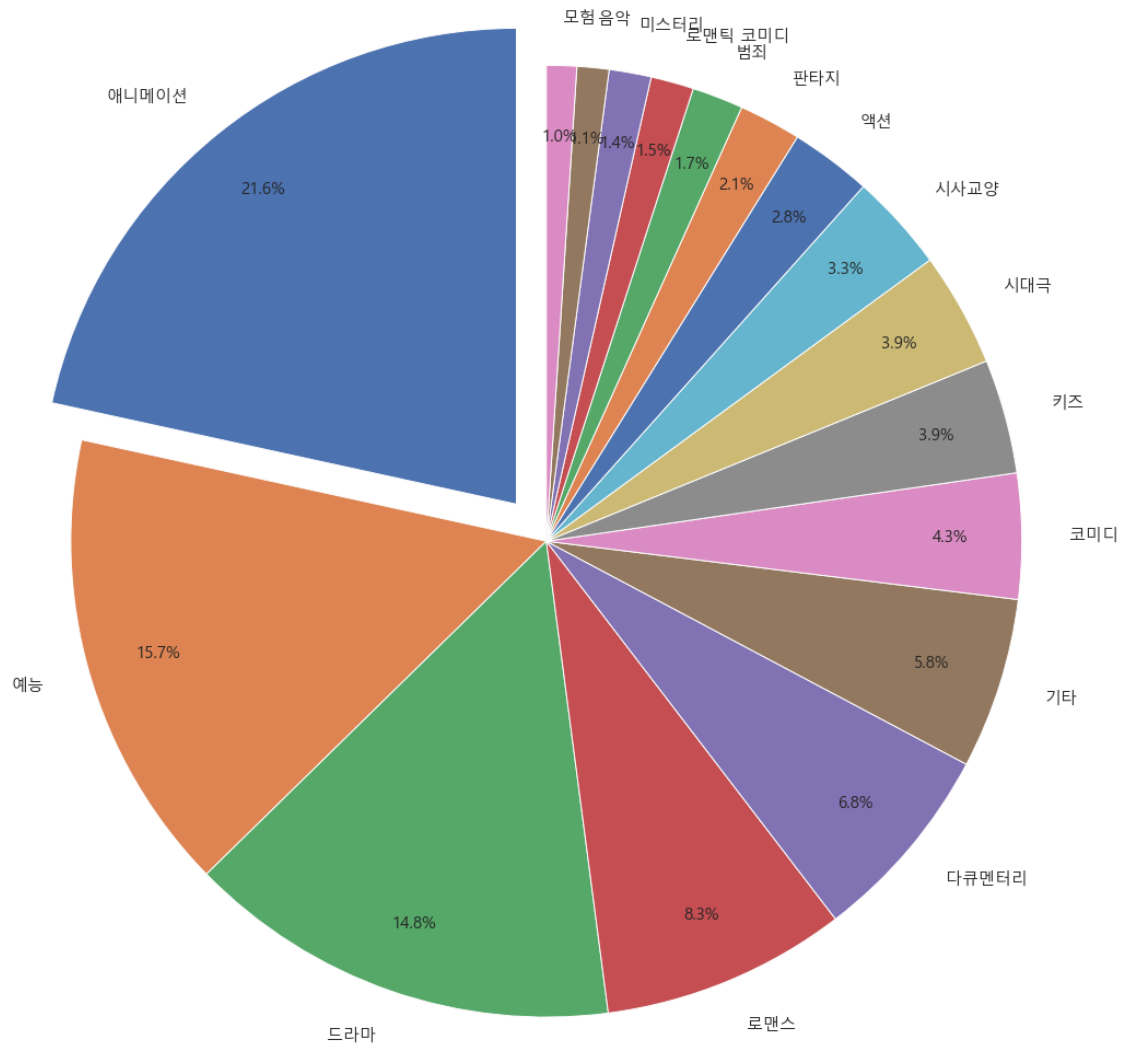
# 가장 많은 비율을 차지하는 조각 강조를 위해 explode 설정
max_index = new_genre_percentages.idxmax()
explode = [0.1 if label == max_index else 0 for label in new_genre_percentages.index]

# 원 그래프 그리기 (크기 조절 및 텍스트 설정)
plt.figure(figsize=(20, 20)) # 그래프 크기 설정
patches, texts, autotexts = plt.pie(new_genre_percentages,
labels=new_genre_percentages.index,
autopct='%1.1f%%', startangle=90, pctdistance=0.85,
labeldistance=1.1, explode=explode)
plt.title('장르별 TV 프로그램 작품 수 비율', fontsize=20) # 그래프 제목 설정

# 텍스트 사이즈 설정
for text in texts + autotexts:
    text.set_fontsize(15)

plt.show() # 그래프 표시
```

장르별 TV 프로그램 작품 수 비율



4.5 개봉년도 구간별 작품 수

```
# 4. 개봉년도 구간별 작품 수

# 개봉년도별 작품 수 확인
print(watcha['개봉년도'].unique)

# 1920년대부터 2020년대까지 10년 단위로 구간 설정
bins = [1920, 1930, 1940, 1950, 1960, 1970, 1980, 1990, 2000, 2010, 2020, 2030]

# 개봉년도를 각 구간으로 나누기
watcha['연도_구간'] = pd.cut(watcha['개봉년도'], bins=bins, labels=['1920년대', '1930년대',
'1940년대', '1950년대', '1960년대', '1970년대', '1980년대', '1990년대', '2000년대',
'2010년대', '2020년대'])

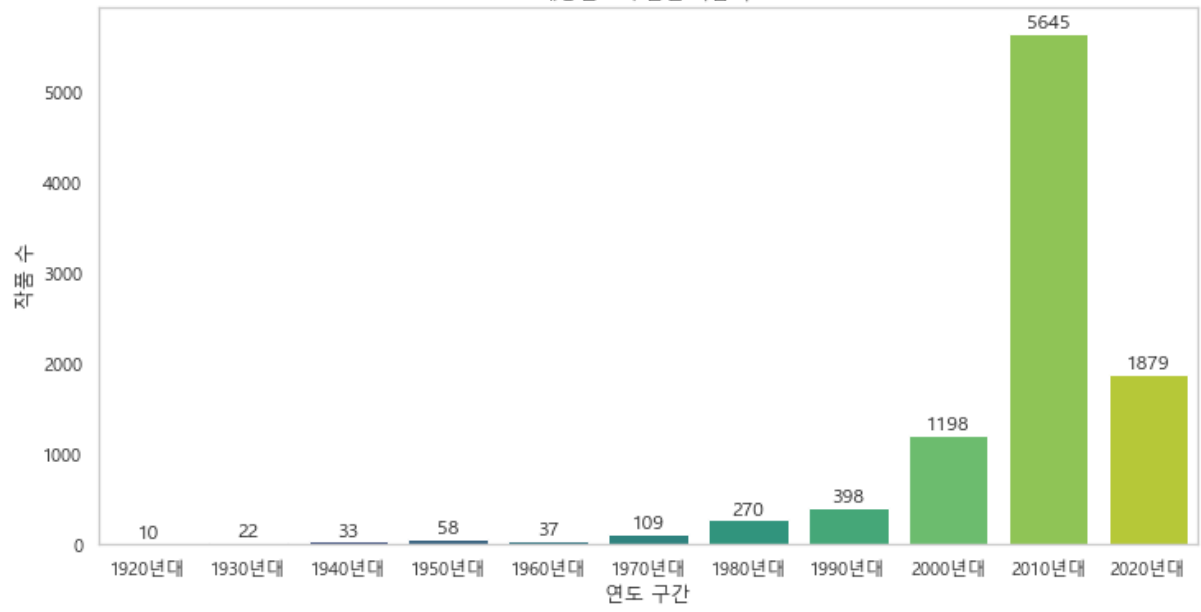
# 각 구간별 작품 수 계산
yearly_counts_grouped = watcha['연도_구간'].value_counts().sort_index()

# 바 그래프 그리기
plt.figure(figsize=(12, 6)) # 그래프 크기 설정
ax = sns.barplot(x=yearly_counts_grouped.index, y=yearly_counts_grouped.values,
palette='viridis') # 바 그래프 생성

# 각 막대 위에 수치 표시 (정수형태로)
for p in ax.patches:
    height = p.get_height() # 막대의 높이(데이터 개수)
    ax.text(p.get_x() + p.get_width() / 2., height, f'{int(height)}', ha='center',
va='bottom')

plt.xlabel('연도 구간') # x축 레이블 설정
plt.ylabel('작품 수') # y축 레이블 설정
plt.title('개봉년도 구간별 작품 수') # 그래프 제목 설정
plt.grid(axis='y') # y축 기준으로만 격자 표시
plt.show() # 그래프 표시
```

개봉년도 구간별 작품 수



4.6 연령 등급별 작품수

- 전체

```
# 5. 연령 등급별 작품 수

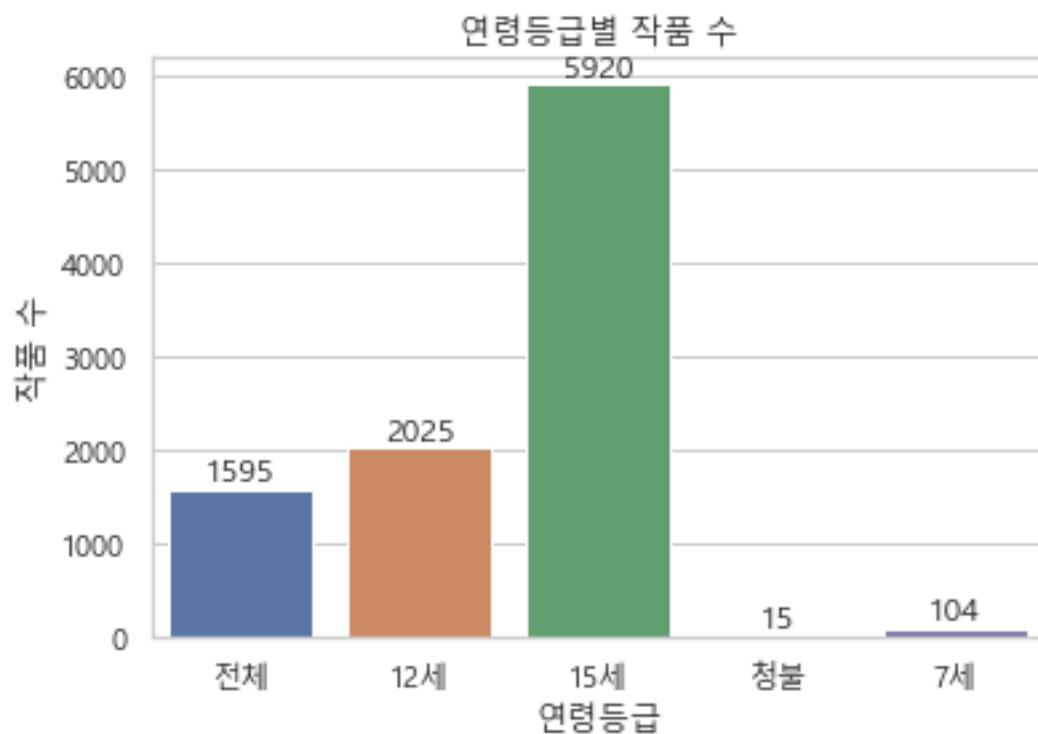
# 5-1 전체

ax = sns.countplot(data=watcha, x='연령등급')

# 각 막대 위에 수치 표시 (정수형태로)
for p in ax.patches:
    height = p.get_height() # 막대의 높이(데이터 개수)
    ax.text(p.get_x() + p.get_width() / 2., height, f'{int(height)}', ha='center',
            va='bottom')

plt.ylabel('작품 수') # y 축 이름 설정
plt.title('연령등급별 작품 수') # 제목 추가

plt.show() # 시각화된 plot 보여줌
```



- 영화

```
# 5-2 영화

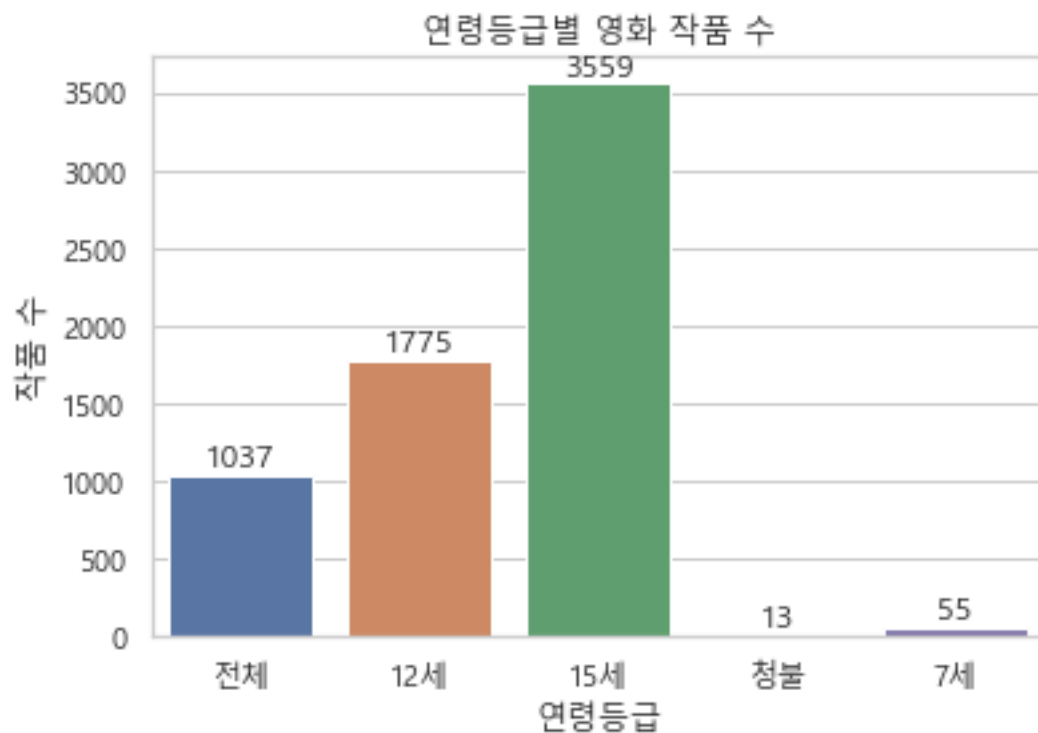
# 영화만 필터링
movie = watcha[watcha['종류'] == '영화']

ax = sns.countplot(data=movie, x='연령등급')

# 각 막대 위에 수치 표시 (정수형태로)
for p in ax.patches:
    height = p.get_height() # 막대의 높이(데이터 개수)
    ax.text(p.get_x() + p.get_width() / 2., height, f'{int(height)}', ha='center',
            va='bottom')

plt.ylabel('작품 수') # y 축 이름 설정
plt.title('연령등급별 영화 작품 수') # 제목 추가

plt.show() # 시각화된 plot 보여줌
```



- TV 프로그램

```
# 5-3 tv 프로그램

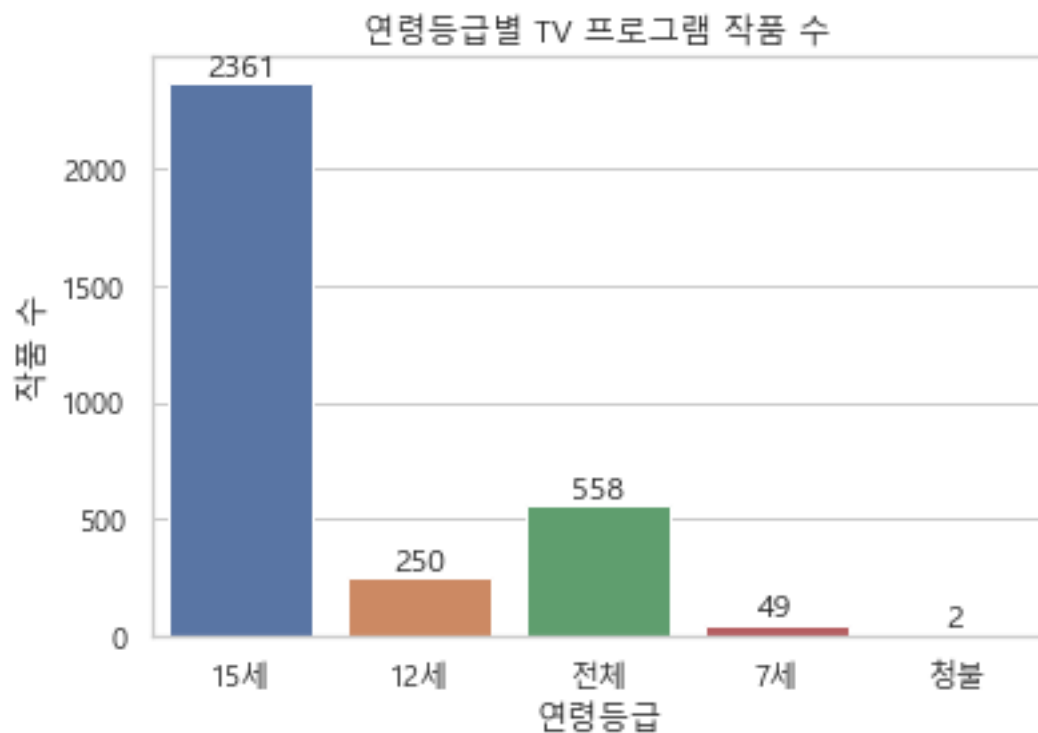
# 영화만 필터링
tv = watcha[watcha['종류'] == 'TV 프로그램']

ax = sns.countplot(data=tv, x='연령등급')

# 각 막대 위에 수치 표시 (정수형태로)
for p in ax.patches:
    height = p.get_height() # 막대의 높이(데이터 개수)
    ax.text(p.get_x() + p.get_width() / 2., height, f'{int(height)}', ha='center',
            va='bottom')

plt.ylabel('작품 수') # y 축 이름 설정
plt.title('연령등급별 TV 프로그램 작품 수') # 제목 추가

plt.show() # 시각화된 plot 보여줌
```



4.7 종류별 작품 수(평균 평점 4.3 이상)

```
# 7. 평균 평점 4.3 이상의 작품 분석

# '평균 평점' 열의 데이터를 숫자(float) 타입으로 변환
watcha['평균 평점'] = watcha['평균 평점'].astype(float)

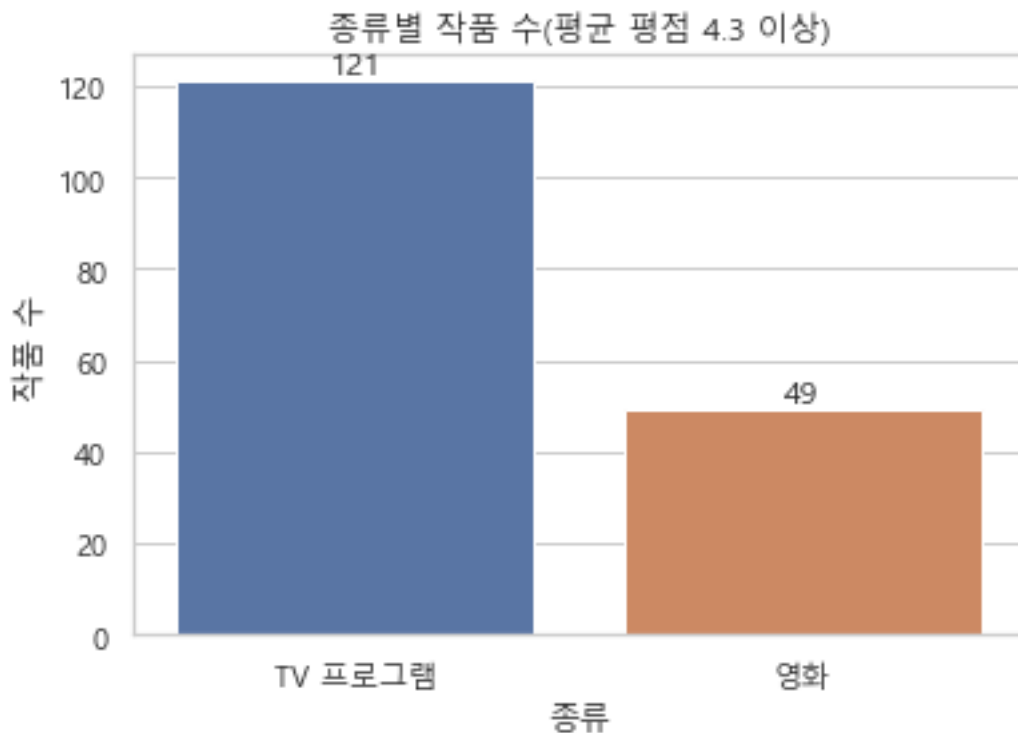
# 평균 평점이 4.3 이상인 행 추출
content_4_3 = watcha[watcha['평균 평점'] >= 4.3]

# 7-1 종류별
ax = sns.countplot(data=content_4_3, x='종류')

# 각 막대 위에 수치 표시 (정수형태로)
for p in ax.patches:
    height = p.get_height() # 막대의 높이(데이터 개수)
    ax.text(p.get_x() + p.get_width() / 2., height, f'{int(height)}', ha='center',
va='bottom')

plt.ylabel('작품 수') # y 축 이름 설정
plt.title('종류별 작품 수(평균 평점 4.3 이상)') # 제목 추가

plt.show() # 시각화된 plot 보여줌
```



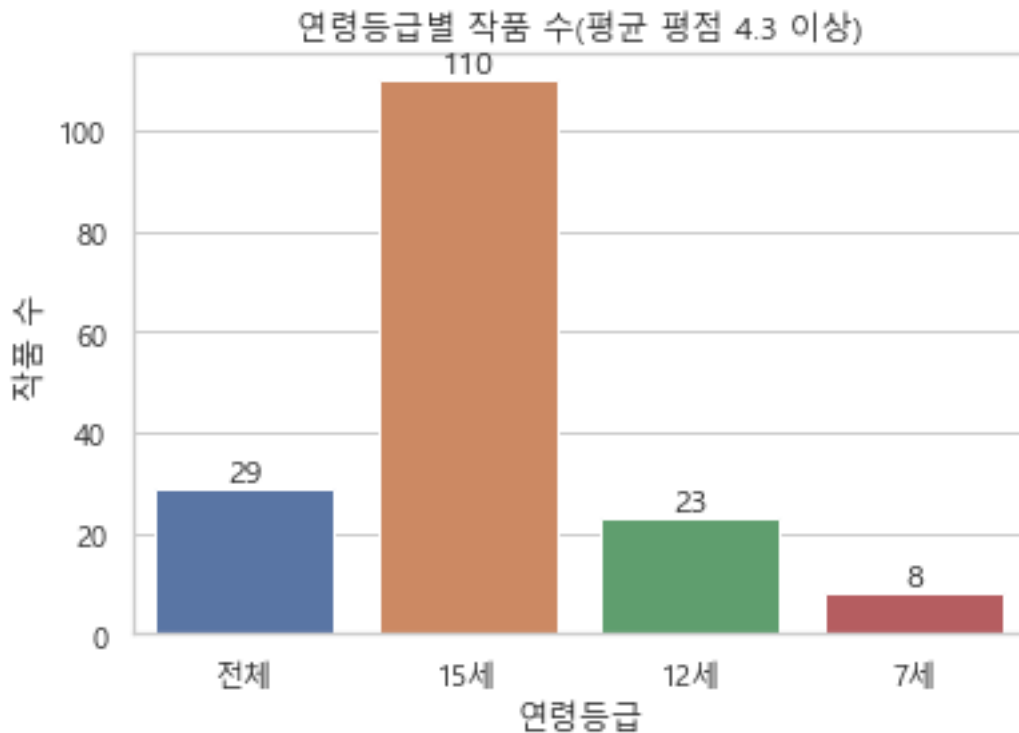
4.8 연령 등급별 작품 수(평균 평점 4.3 이상)

```
# 7-2 연령등급별
ax = sns.countplot(data=content_4_3, x='연령등급')

# 각 막대 위에 수치 표시 (정수형태로)
for p in ax.patches:
    height = p.get_height() # 막대의 높이(데이터 개수)
    ax.text(p.get_x() + p.get_width() / 2., height, f'{int(height)}', ha='center',
            va='bottom')

plt.ylabel('작품 수') # y 축 이름 설정
plt.title('연령등급별 작품 수(평균 평점 4.3 이상)') # 제목 추가

plt.show() # 시각화된 plot 보여줌
```



4.9 장르별 작품 수(평균 평점 4.3 이상)

```
# 7-3 장르별

genre_counts = content_4_3['장르'].value_counts()
print(genre_counts)

# 작품 수의 비율 계산
genre_percentages = genre_counts / genre_counts.sum() * 100

# 2% 미만인 장르들을 '기타'로 통합
threshold = 1 # 임계치 설정
other_genres = genre_percentages[genre_percentages < threshold].index
content_4_3.loc[content_4_3['장르'].isin(other_genres), '장르'] = '기타'

# 다시 장르별 작품 수 계산
new_genre_counts = content_4_3['장르'].value_counts()

# 새로운 작품 수의 비율 계산
new_genre_percentages = new_genre_counts / new_genre_counts.sum() * 100

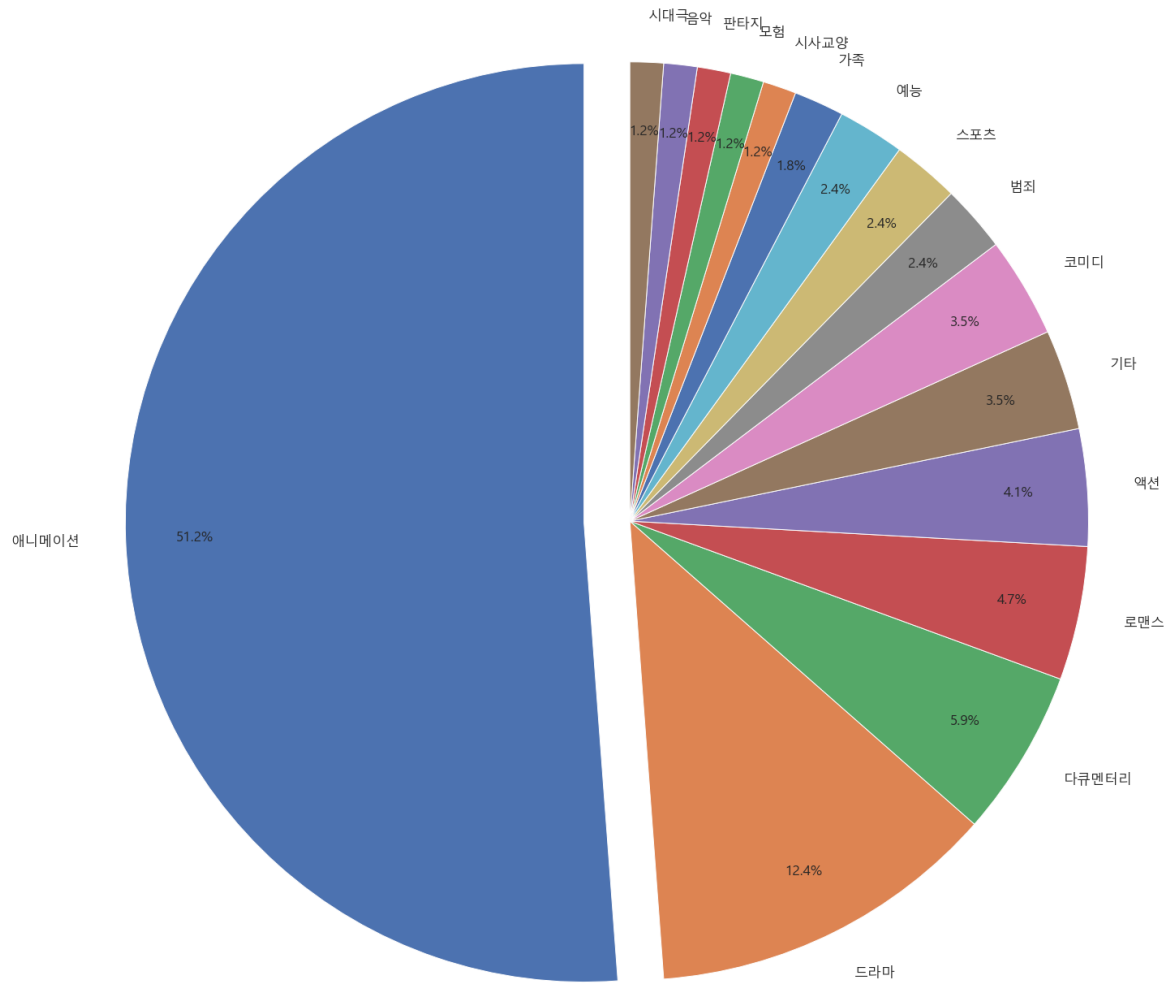
# 가장 많은 비율을 차지하는 조각 강조를 위해 explode 설정
max_index = new_genre_percentages.idxmax()
explode = [0.1 if label == max_index else 0 for label in new_genre_percentages.index]

# 원 그래프 그리기 (크기 조절 및 텍스트 설정)
plt.figure(figsize=(26, 26)) # 그래프 크기 설정
patches, texts, autotexts = plt.pie(new_genre_percentages,
labels=new_genre_percentages.index,
autopct='%1.1f%%', startangle=90, pctdistance=0.85,
labeldistance=1.1, explode=explode)
plt.title('장르별 작품 수 비율(평균 평점 4.3 이상)', fontsize=20) # 그래프 제목 설정

# 텍스트 사이즈 설정
for text in texts + autotexts:
    text.set_fontsize(17)

plt.show() # 그래프 표시
```

장르별 작품 수 비율(평균 평점 4.3 이상)



4.10 제작 국가별 작품 수(평균 평점 4.3 이상)

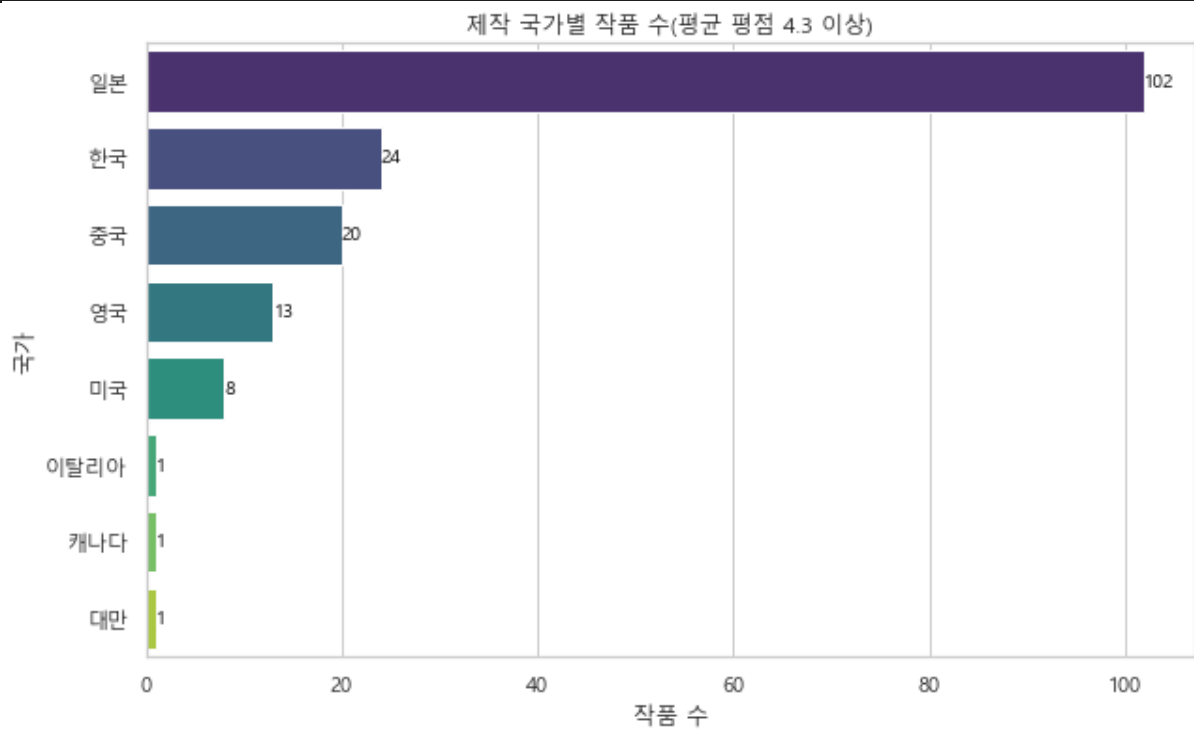
```
# 7-4 제작 국가별

# 제작 국가별 작품 수 계산 후 정렬
country_counts = content_4_3['국가'].value_counts().sort_values(ascending=False)

# 가로 막대 그래프로 시각화
plt.figure(figsize=(10, 6)) # 그래프 크기 설정
sns.barplot(x=country_counts.values, y=country_counts.index, palette='viridis')

# 각 막대 안에 수치 표시
for index, value in enumerate(country_counts):
    plt.text(value, index, str(value), va='center', ha='left', fontsize=10, color='black')

plt.xlabel('작품 수') # x 축 레이블 설정
plt.ylabel('국가') # y 축 레이블 설정
plt.title('제작 국가별 작품 수(평균 평점 4.3 이상)') # 그래프 제목 설정
plt.show() # 그래프 표시
```



4.11 개봉년도별 작품 수(평균 평점 4.3 이상)

```
# 7-5 개봉년도별

# 개봉년도별 작품 수 확인
print(content_4_3['개봉년도'].unique)

# 1920년대부터 2020년대까지 10년 단위로 구간 설정
bins = [1920, 1930, 1940, 1950, 1960, 1970, 1980, 1990, 2000, 2010, 2020, 2030]

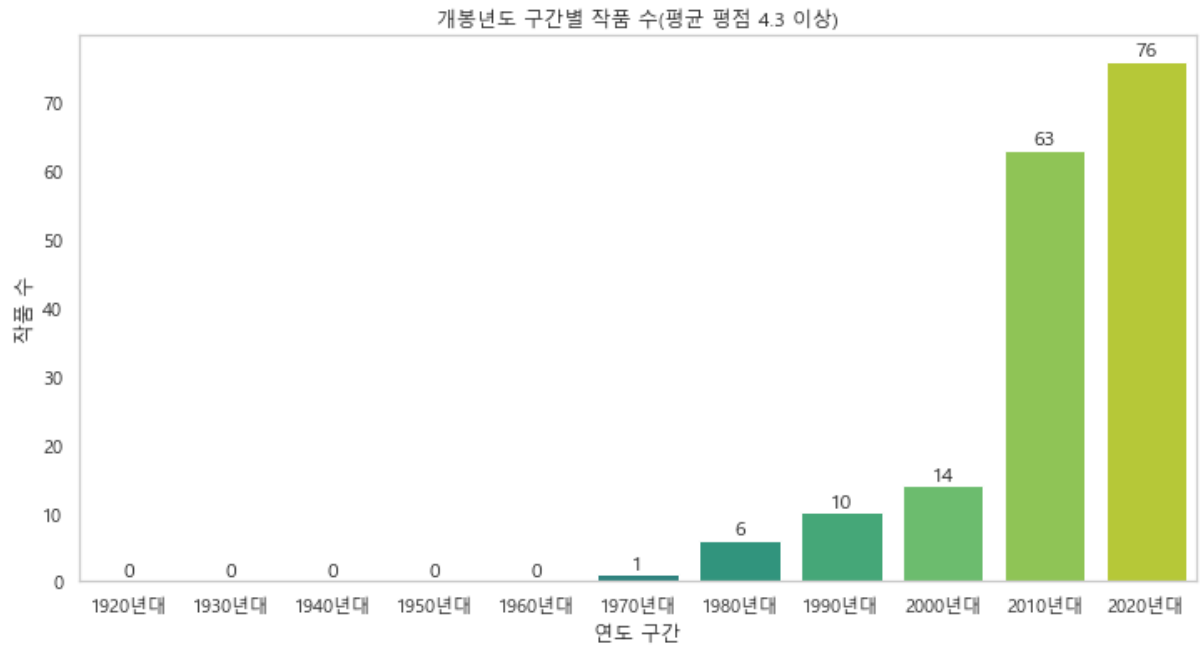
# 개봉년도를 각 구간으로 나누기
content_4_3['연도_구간'] = pd.cut(content_4_3['개봉년도'], bins=bins, labels=['1920년대',
'1930년대', '1940년대', '1950년대', '1960년대', '1970년대', '1980년대', '1990년대',
'2000년대', '2010년대', '2020년대'])

# 각 구간별 작품 수 계산
yearly_counts_grouped = content_4_3['연도_구간'].value_counts().sort_index()

# 바 그래프 그리기
plt.figure(figsize=(12, 6)) # 그래프 크기 설정
ax = sns.barplot(x=yearly_counts_grouped.index, y=yearly_counts_grouped.values,
palette='viridis') # 바 그래프 생성

# 각 막대 위에 수치 표시 (정수형태로)
for p in ax.patches:
    height = p.get_height() # 막대의 높이(데이터 개수)
    ax.text(p.get_x() + p.get_width() / 2., height, f'{int(height)}', ha='center',
va='bottom')

plt.xlabel('연도 구간') # x축 레이블 설정
plt.ylabel('작품 수') # y축 레이블 설정
plt.title('개봉년도 구간별 작품 수(평균 평점 4.3 이상)') # 그래프 제목 설정
plt.grid(axis='y') # y축 기준으로만 격자 표시
plt.show() # 그래프 표시
```



4.12 결론

1) 종류별 작품 수

- 영화가 6,439개, TV 프로그램이 3,220개로 영화가 약 2배 더 많이 등록되어 있다.

2) 국가별 작품 수(상위 15개)

- 한국 작품이 총 3,631개(영화 2,171개, TV 프로그램 1,460개)로 가장 많이 등록되어 있다.

3) 장르별 작품 수

- 영화 장르에서는 드라마가 28.1%로 가장 많다.
- TV 프로그램 장르에서는 애니메이션이 21.6%로 가장 많다.

4) 개봉년도 구간별 작품 수

- 2010년대에 개봉된 작품이 5,645개로 가장 많다.

5) 연령 등급별 작품수

- 영화, TV프로그램 모두 15세 이상 등급이 제일 많았다.

6) 종류별 작품 수(평균 평점 4.3 이상)

- 170개의 작품 중 121개가 TV 프로그램이고 49개가 영화로, TV 프로그램이 영화의 약 2.5배 더 많다.

7) 연령 등급별 작품 수(평균 평점 4.3 이상)

- 15세 이상이 110개로 가장 많았다.

8) 장르별 작품 수(평균 평점 4.3 이상)

- 애니메이션이 51.2%로 가장 많았다.

9) 제작 국가별 작품 수(평균 평점 4.3 이상)

- 102개로 일본에서 제작된 작품이 가장 많았다.
- 한국 작품은 24개로 두 번째로 많았다.

10) 개봉년도별 작품 수(평균 평점 4.3 이상)

- 2020년대에 개봉한 작품이 76개로 가장 많았다.