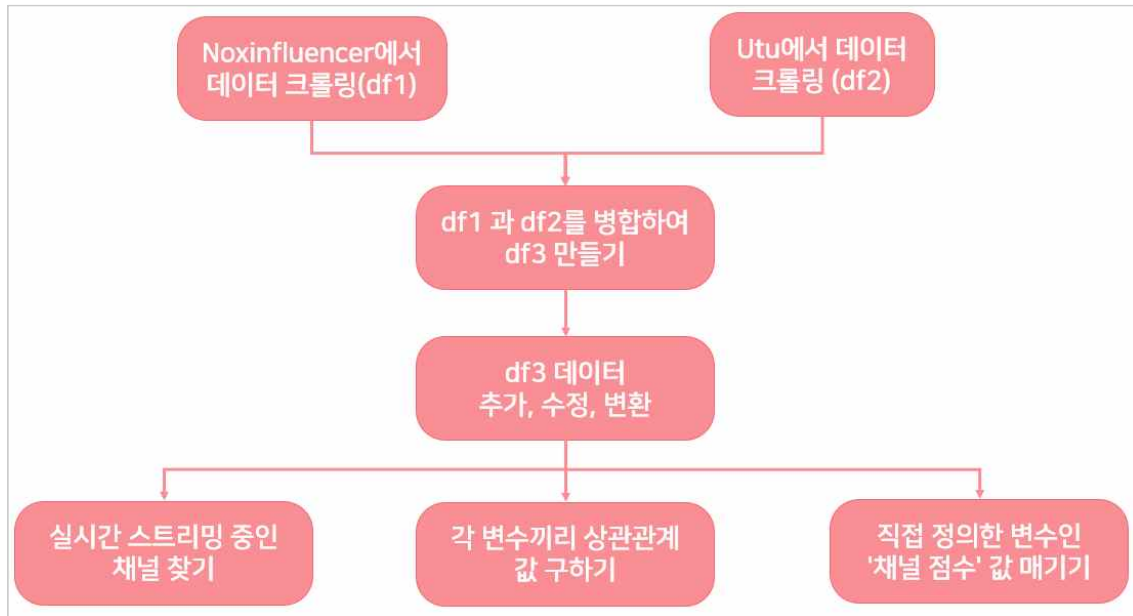


# Youtube Channel Analysis (유튜브 채널 분석)

## 1. 구현한 기능

	세부 기능	결과물
데이터 크롤링 및 생성	'noxinfluencer'에서 한국 top100 유튜브채널 정보 크롤링	df1 (데이터프레임)
	'utu(유투)'에서 한국 top100 유튜브채널 정보 크롤링	df2 (데이터프레임)
	df1과 df2 같은 채널명끼리 항목(column) 합치기	df3 (데이터프레임)
df3 데이터 추가 및 수정	데이터에 '만', '억'이라 붙어있는 값들을 숫자로 변환	df3 (데이터프레임)
	'총 활동일' 열을 새로 만들어 추가	
	string 타입인 숫자값들을 전부 numeric 타입으로 변환	
유튜브채널 분석	실시간 스트리밍 중인 채널 찾기	실시간 스트리밍(Live) 중인 채널명, 영상 제목, 링크
	각 변수끼리 상관관계 값 구하기	corr (데이터프레임), 각 변수별 상관관계가 높은 변수 top3
	직접 정의한 변수인 '채널 점수' 값 매기기	'채널 점수' 열이 추가된 df3

## 2. 프로젝트 수행 과정



### 3. 각 기능에 대한 설명 (코드 설명을 중점으로)

#### 1) 'noxinfluencer'에서 한국 top100 유튜브채널 정보 크롤링

```
driver = webdriver.Chrome('C:/Users/dudtj/Desktop/빅데이터처리및응용/driver/chromedriver')
driver.get("https://kr.noxinfluencer.com/youtube-channel-rank/top-100-kr-all-youtuber-sorted-by-subs-weekly/")
driver.execute_script("window.scrollTo(0, document.body.scrollHeight);") #페이지 제일 밑으로 스크롤
#페이지 제일 밑으로 스크롤을 해야 숨겨진 51~100위 순위가 나오고 숨겨진 모든 소스코드들이 나온다

time.sleep(5) #제대로 모든 소스코드들이 로딩 되도록 5초간 대기

html = driver.page_source
soup = BeautifulSoup(html, 'lxml')
```

<https://kr.noxinfluencer.com/youtube-channel-rank/top-100-kr-all-youtuber-sorted-by-subs-weekly/> (noxinfluencer) 사이트에서 한국 top100 유튜브채널의 정보를 크롤링하기 위해 driver를 연결하여 페이지 소스를 가져오는 코드이다. 이때 주의할 점은 페이지 제일 밑으로 스크롤을 하지 않으면 top50 정보까지만 나오고, 스크롤을 밑으로 내려야만 숨겨진 51~100위 순위와 모든 소스코드들이 로드 되므로 driver를 통해 페이지를 밑으로 스크롤하고 5초 동안 대기하는 코드를 추가했다.

```
channel_name = []
category = []
subscriber = []
avg_views = []
score = []

detail_url = [] # 상세 정보 페이지 주소를 저장할 리스트
url = 'https://kr.noxinfluencer.com'

for i in range(100):
    channel_name.append(soup.find_all("span", {"class": "name kol-name"})[i].string.strip()) #lstrip() : 왼쪽 공백 제거해서 리스트에 추가
    category.append(soup.find_all("a", {"class": "category-text"})[i].string.strip()) #strip() : 앞뒤 공백 모두 제거해서 리스트에 추가

    a = soup.find_all("td", {"class": "text followerNum with-num"})[i]
    subscriber.append(a.find("span", {"class": "num"}).string.strip())

    a = soup.find_all("td", {"class": "text avgView with-num"})[i]
    avg_views.append(a.find("span", {"class": "num"}).string.strip())

    a = soup.find_all("td", {"class": "text nox-score"})[i]
    score.append(a['data-score'])

    detail_url.append(url + soup.find_all("a", {"class": "star-avatar"})[i+1]['href'])
```

다음으로 각 항목에 해당하는 리스트를 만들고 가져올 값에 해당하는 소스태그를 find하여 각 리스트에 값을 추가한다. 총 100개의 유튜브채널 정보를 가져올 것이므로 100번 실행하는 반복문에 코드를 넣어 실행한다. 또한 이후에 각 유튜브 채널의 상세 정보 페이지 주소에서 추가 정보를 크롤링하기 위해 상세 정보 페이지 주소를 저장할 리스트를 생성하고 주소를 저장하였다.

```
df1 = pd.DataFrame({'순위': ['TOP '+str(i+1) for i in range(100)], '채널명': channel_name, '카테고리': category,
                    '구독자': subscriber, '평균 조회수': avg_views, 'Nox평점': score})
```

크롤링한 데이터 값이 들어있는 리스트를 열(column)에 넣어 df1이라는 데이터프레임을 생성하였다.

# 각 채널별 상세 정보 페이지 열어서 추가정보 가져오기

```
for i in range(100):
    driver.get(detail_url[i])

    html = driver.page_source
    soup = BeautifulSoup(html, 'lxml')

    df1.loc[i, '전체 동영상'] = soup.find_all("span", {"class": "strong"})[3].string
    df1.loc[i, '최근 1개월 동영상'] = soup.find_all("span", {"class": "title"})[-16].string
    df1.loc[i, '채널 개설일'] = soup.find_all("span", {"class": "subt itle"})[0].string
    df1.loc[i, '채널주소'] = soup.find_all("a", {"class": "icon=wrapper"})[0]['href']

    time.sleep(5) # 계속 크롤링하면 로봇 확인 페이지가 떠서 이를 방지하기 위해 5초씩 대기

driver.close()
```

# 최종 df1 데이터프레임  
df1

	순위	채널명	카테고리	구독자	평균 조회수	Nox 평점	전체 동영상	최근 1개월 동영상	채널 개설일	채널주소
0	TOP 1	BLACKPINK	음악	3180만	1754.16만	2.05	253	0	2016-06-28	<a href="https://www.youtube.com/channel/UCOmHUn--16B90...">https://www.youtube.com/channel/UCOmHUn--16B90...</a>
1	TOP 2	ibighit	음악	3040만	800.44만	4.76	347	12	2008-06-04	<a href="https://www.youtube.com/channel/UC3lZKseVpdzPS...">https://www.youtube.com/channel/UC3lZKseVpdzPS...</a>
2	TOP 3	BANGTANTV	음악	2380만	204.13만	4.08	1101	26	2012-12-16	<a href="https://www.youtube.com/channel/UCLkAepWjdylmX...">https://www.youtube.com/channel/UCLkAepWjdylmX...</a>
3	TOP 4	SMTOWN	음악	2080만	174.75만	3.39	3939	19	2006-03-18	<a href="https://www.youtube.com/channel/UCEf_Bc-KVd7on...">https://www.youtube.com/channel/UCEf_Bc-KVd7on...</a>
4	TOP 5	1MILLION Dance Studio	엔터테인먼트	1870만	62.03만	3.14	1764	39	2015-02-06	<a href="https://www.youtube.com/channel/UCw8ZhLPdQ0u_Y...">https://www.youtube.com/channel/UCw8ZhLPdQ0u_Y...</a>
5	TOP 6	1theK (원더케이)	엔터테인먼트	1830만	2만	3.27	1.67만	251	2011-01-30	<a href="https://www.youtube.com/channel/UCweOkPb1wVVH0...">https://www.youtube.com/channel/UCweOkPb1wVVH0...</a>
6	TOP 7	jypentertainment	음악	1440만	269.67만	4.48	1038	12	2008-01-24	<a href="https://www.youtube.com/channel/UCaO6TYiIC8U5t...">https://www.youtube.com/channel/UCaO6TYiIC8U5t...</a>

상세 정보 페이지에서만 얻을 수 있는 항목에는 ‘전체 동영상’, ‘최근 1개월 동영상’, ‘채널 개설일’, ‘채널 주소’ 정보가 있다. 따라서 각 채널별 상세 정보 페이지를 열어서 추가항목을 크롤링하였다. 각 페이지를 열기 전 time.sleep(5)를 해주는 이유는 time.sleep을 하지 않으면 빠르게 연속으로 페이지가 바뀌어 사이트가 로봇 확인 페이지로 자동으로 넘어가기에, 이를 방지하고자 5초씩 대기하도록 하였다. 새롭게 크롤링한 데이터를 df1에 새로운 열을 추가하여 넣었다. 이렇게 100 rows × 10 columns인 최종 df1 데이터프레임이 완성되었다.

## 2) 'utu(유투)'에서 한국 top100 유튜브채널 정보 크롤링

```
driver = webdriver.Chrome('C:/Users/dudtj/Desktop/빅데이터처리및응용/driver/chromedriver')
driver.get("https://utu.kr/rank")
driver.find_element_by_xpath("//*[id='srange']").send_keys("구독자 순") # 구독자 순으로 순위 정렬

time.sleep(3) # 페이지가 로딩될 때까지 3초 대기

# 더보기를 눌러야 채널 20개씩 새로 로딩 됨
driver.find_element_by_xpath("//*[id='Rank_main_container']/button").click() # 더보기 클릭
time.sleep(3)
driver.find_element_by_xpath("//*[id='Rank_main_container']/button").click() # 더보기 클릭
time.sleep(3)
driver.find_element_by_xpath("//*[id='Rank_main_container']/button").click() # 더보기 클릭
time.sleep(3)
driver.find_element_by_xpath("//*[id='Rank_main_container']/button").click() # 더보기 클릭
time.sleep(3)
driver.find_element_by_xpath("//*[id='Rank_main_container']/button").click() # 더보기 클릭
time.sleep(3)

html = driver.page_source
soup = BeautifulSoup(html, 'lxml')
```

<https://utu.kr/rank> (유투) 사이트에서 한국 top100 유튜브채널의 정보를 크롤링하기 위해 driver를 연결하여 페이지 소스를 가져오는 코드이다. 구독자 순으로 top100을 정렬하였는데, 이때 주의할 점은 처음에는 top1~top20의 정보밖에 나와있지 않은데 밑에 더보기를 클릭하여야 채널 20개씩 추가로 로딩 된다는 점이다. 총 top100의 데이터를 가져오기위해 더보기를 클릭할때마다 time.sleep(3)을 하여 제대로 로드가 되도록 반복하여 top100까지의 소스를 파싱하였다.

```
channel_name = []
category = []
UTU_score = []
subscriber = []
views = []
like = []
dislike = []
comment = []

for i in range(100):
    channel_name.append(soup.find_all("span", {"class": "chName"})[i].string)
    category.append(soup.find_all("p", {"class": "cateName"})[i].string)
    UTU_score.append(soup.find_all("div", {"class": "list-score text-center"})[i+1].string)
    subscriber.append(soup.find_all("div", {"class": "list-person text-center"})[i+1].string)
    views.append(soup.find_all("div", {"class": "list-view text-center"})[i+1].string.split('*')[0]) # 조회수 옆에 붙은 *는 제거해서 삼입
    like.append(soup.find_all("div", {"class": "list-like text-center View_Desktop"})[i+1].string.split('*')[0])
    dislike.append(soup.find_all("div", {"class": "list-unlike text-center View_Desktop"})[i+1].string)
    comment.append(soup.find_all("div", {"class": "list-reply text-center View_Desktop"})[i+1].string)

driver.close()
```

다음으로 각 항목에 해당하는 리스트를 만들고 가져올 값에 해당하는 소스태그를 find하여 각 리스트에 값을 추가한다. 총 100개의 유튜브채널 정보를 가져올 것이므로 100번 실행하는 반복문에 코드를 넣어 실행한다.

```
df2 = pd.DataFrame({'순위': ['TOP '+str(i+1) for i in range(100)], '채널명': channel_name, '카테고리': category, 'UTU점수': UTU_score,
                    '구독자': subscriber, '조회수': views, '좋아요': like, '싫어요': dislike, '댓글': comment})
```

```
# 최종 df2 데이터프레임
df2
```

	순위	채널명	카테고리	UTU점수	구독자	조회수	좋아요	싫어요	댓글
0	TOP 1	BLACKPINK	영화/엔터 > 음악	75점	3,170 만	79 억	5,088 만	173 만	502 만
1	TOP 2	ibighit	영화/엔터 > 엔터	78점	3,040 만	75 억	1.4 억	270 만	1,930 만
2	TOP 3	GellyBear ToyPudding	키즈/동물 > 놀이	48점	2,550 만	148 억	30 만	13 만	1.7 만
3	TOP 4	BANGTANTV	영화/엔터 > 엔터	81점	2,370 만	32 억	1.5 억	42 만	853 만
4	TOP 5	Boram Tube Vlog [보람튜브 브이로그]	키즈/동물 > 놀이	63점	2,190 만	81 억	423 만	189 만	6.0 만
5	TOP 6	SMTOWN	영화/엔터 > 음악	63점	2,070 만	157 억	2.3 억	486 만	2,203 만
6	TOP 7	1MILLION Dance Studio	영화/엔터 > 음악	75점	1,860 만	45 억	5,827 만	75 만	109 만
7	TOP 8	1theK (원더케이)	영화/엔터 > 음악	62점	1,832 만	150 억	1,020 만	19 만	73 만
8	TOP 9	jypentertainment	영화/엔터 > 음악	79점	1,440 만	78 억	8,785 만	408 만	914 만
9	TOP 10	Boram Tube ToysReview [보람튜브 토이리뷰]	키즈/동물 > 놀이	59점	1,400 만	44 억	623 만	279 만	13 만
10	TOP 11	JFlaMusic	영화/엔터 > 엔터	63점	1,390 만	25 억	1,330 만	35 만	111 만

크롤링한 데이터 값이 들어있는 리스트를 열(column)에 넣어 df2라는 데이터프레임을 생성하였다. 이렇게 100 rows × 9 columns인 최종 df2 데이터프레임이 완성되었다. df2 프레임의 열 항목은 최대한 df1의 열 항목과 겹치지 않은 정보로 가져왔다.

### 3) df1과 df2 채널명이 같은 것끼리 항목(column) 합쳐서 df3 만들기

```
df3 = pd.merge(df1, df2, on='채널명') # df1과 df2 병합 (채널명이 같은 행만 합치고 같지 않은 행은 삭제)
df3.drop(['순위_x', '순위_y', '구독자_y'], axis='columns', inplace=True) # 필요없는 열 삭제
df3.rename(columns={'카테고리_x': '카테고리1', '카테고리_y': '카테고리2', '구독자_x': '구독자'}, inplace=True) # 열 이름 변경
df3 = df3.drop_duplicates('채널명', keep='first') # 중복 채널명 행 삭제 (처음 나오는 행을 유지)
column_titles = ['채널명', '카테고리1', '카테고리2', '구독자', '조회수', '평균 조회수', '전체 동영상', '최근 1개월 동영상', '좋아요', '싫어요', '댓글', 'Nox점점', 'UTU점수', '채널 개설일', '채널주소']
df3 = df3.reindex(columns=column_titles) # 열 순서 재배열
df3 = df3.reset_index(drop=True) # index 순서 0부터 순서대로 초기화

# 최종 df3 데이터프레임
df3
```

	채널명	카테고리1	카테고리2	구독자	조회수	평균 조회수	전체 동영상	최근 1개월 동영상	좋아요	싫어요	댓글	Nox점점	UTU점수	채널 개설일	채널주소
0	BLACKPINK	영화/엔터 > 음악	영화/엔터 > 음악	3180 만	79 억	1754.16 만	253	0	5,088 만	173 만	502 만	2.05	75 점	2016-06-28	<a href="https://www.youtube.com/channel/UCOmHUn--16B90...">https://www.youtube.com/channel/UCOmHUn--16B90...</a>
1	ibighit	영화/엔터 > 엔터	영화/엔터 > 엔터	3040 만	75 억	800.44 만	347	12	1.4 억	270 만	1,930 만	4.76	78 점	2008-06-04	<a href="https://www.youtube.com/channel/UC3lZKseVpdzPS...">https://www.youtube.com/channel/UC3lZKseVpdzPS...</a>

df1과 df2를 병합하는 코드이다. 채널명이 같은 행만 병합하고, 같지 않은 행은 삭제하였다. 필요없는 열(또는 중복되는 열)은 삭제하고, 열 이름을 보기 좋게 변경하였으며, utu사이트 오류로 top100에 똑같은 채널명이 두 개 나와있는 현상 때문에 중복으로 있는 채널명은 처음 나오는 행을 유지하고 뒤에 나오는 행은 삭제했다. 그리고 열 순서를 재배열하고 index를 0부터 순서대로 초기화하여 df1과 df2를 합친 df3 데이터프레임이 완성되었다. (75 rows × 15 columns)



#### 4) df3 데이터에 '만', '억'이라 붙어있는 값들을 숫자로 변환

df3 데이터 값을 보면 '3180만, 79 억, 1754.16만, 5,088 만' 등 문자열로 데이터가 저장돼 있어 데이터 분석이 불가능한 상태이다. '.' 과 ','을 모두 없애고 '만', '억'을 실제 숫자로 변환하여 문자열 데이터들을 전부 실제 숫자 형태로 변환할 필요가 있다는걸 깨달았다.

```
# 데이터에 '만', '억' 이라 붙어있는 값들을 숫자형으로 변환 ex) 2만 -> 200000 5,088 만 -> 50880000 1.2억 -> 120000000
for i in range(len(df3)):
```

```
    # 구독자 열 변환
    df3.loc[i, '구독자'] = df3.loc[i, '구독자'].split('만')[0] + '0000' # '만' 이라 붙어있던걸 0000 숫자로 바꿈
```

```
    # 조회수 열 변환
    if '만' in df3.loc[i, '조회수']:
        df3.loc[i, '조회수'] = df3.loc[i, '조회수'].replace('.', '')
        df3.loc[i, '조회수'] = df3.loc[i, '조회수'].replace('만', '')
        df3.loc[i, '조회수'] = df3.loc[i, '조회수'] + '0000'
    elif '억' in df3.loc[i, '조회수']:
        value = df3.loc[i, '조회수'].split('.')
        if len(value) > 1:
            df3.loc[i, '조회수'] = value[0] + value[1].split(',')[0] + '0000000'
        else:
            df3.loc[i, '조회수'] = value[0].split(',')[0] + '00000000'
```

```
    # 평균 조회수 열 변환
    if '.' in df3.loc[i, '평균 조회수']:
        value = df3.loc[i, '평균 조회수'].split('만')[0]
        value = value.split('.')
        num = len(value[1])
        df3.loc[i, '평균 조회수'] = value[0] + value[1] + '0' * (4 - num)
    else:
        df3.loc[i, '평균 조회수'] = df3.loc[i, '평균 조회수'].split('만')[0] + '0000'
```

```
    # 전체 동영상 열 변환
    if '.' in df3.loc[i, '전체 동영상']:
        value = df3.loc[i, '전체 동영상'].split('만')[0]
        value = value.split('.')
        num = len(value[1])
        df3.loc[i, '전체 동영상'] = value[0] + value[1] + '0' * (4 - num)
```

```
# 좋아요 열 변환
```

```
if '만' in df3.loc[i, '좋아요']:
    if '.' in df3.loc[i, '좋아요']:
        value = df3.loc[i, '좋아요'].split(',')[0]
        value = value.split('.')
        num = len(value[1])
        df3.loc[i, '좋아요'] = value[0] + value[1] + '0' * (4 - num)
    elif ',' in df3.loc[i, '좋아요']:
        value = df3.loc[i, '좋아요'].split(',')[0]
        value = value.split(',')
        df3.loc[i, '좋아요'] = value[0] + value[1] + '0000'
    else:
        df3.loc[i, '좋아요'] = df3.loc[i, '좋아요'].split(',')[0] + '0000'
elif '억' in df3.loc[i, '좋아요']:
    value = df3.loc[i, '좋아요'].split(',')[0]
    value = value.split(',')
    num = len(value[1])
    df3.loc[i, '좋아요'] = value[0] + value[1] + '0' * (8 - num)
elif ',' in df3.loc[i, '좋아요']:
    value = df3.loc[i, '좋아요'].split(',')
    df3.loc[i, '좋아요'] = value[0] + value[1]
```

```

# 싫어요 열 변환
if '만' in df3.loc[i, '싫어요']:
    if '.' in df3.loc[i, '싫어요']:
        value = df3.loc[i, '싫어요'].split(' ')[0]
        value = value.split('.')
        num = len(value[1])
        df3.loc[i, '싫어요'] = value[0] + value[1] + '0' * (4 - num)
    elif ',' in df3.loc[i, '싫어요']:
        value = df3.loc[i, '싫어요'].split(' ')[0]
        value = value.split(',')
        df3.loc[i, '싫어요'] = value[0] + value[1] + '0000'
    else:
        df3.loc[i, '싫어요'] = df3.loc[i, '싫어요'].split(' ')[0] + '0000'
elif '억' in df3.loc[i, '싫어요']:
    value = df3.loc[i, '싫어요'].split(' ')[0]
    value = value.split('.')
    num = len(value[1])
    df3.loc[i, '싫어요'] = value[0] + value[1] + '0' * (8 - num)
elif ',' in df3.loc[i, '싫어요']:
    value = df3.loc[i, '싫어요'].split(',')
    df3.loc[i, '싫어요'] = value[0] + value[1]

# 댓글 열 변환
if '만' in df3.loc[i, '댓글']:
    if '.' in df3.loc[i, '댓글']:
        value = df3.loc[i, '댓글'].split(' ')[0]
        value = value.split('.')
        num = len(value[1])
        df3.loc[i, '댓글'] = value[0] + value[1] + '0' * (4 - num)
    elif ',' in df3.loc[i, '댓글']:
        value = df3.loc[i, '댓글'].split(' ')[0]
        value = value.split(',')
        df3.loc[i, '댓글'] = value[0] + value[1] + '0000'
    else:
        df3.loc[i, '댓글'] = df3.loc[i, '댓글'].split(' ')[0] + '0000'
elif '억' in df3.loc[i, '댓글']:
    value = df3.loc[i, '댓글'].split(' ')[0]
    value = value.split('.')
    num = len(value[1])
    df3.loc[i, '댓글'] = value[0] + value[1] + '0' * (8 - num)
elif ',' in df3.loc[i, '댓글']:
    value = df3.loc[i, '댓글'].split(',')
    df3.loc[i, '댓글'] = value[0] + value[1]

# UTU점수 열 변환
df3.loc[i, 'UTU점수'] = df3.loc[i, 'UTU점수'].split('점')[0]

```

‘2만’, ‘7억’과 같이 단순히 숫자와 단위로만 이루어진 문자열들은 ‘만’ 또는 ‘억’을 기준으로 split하여 숫자 부분 뒤에 ‘0000’ 또는 ‘00000000’을 붙여주었다. 하지만 문제는 ‘1754.16만’, ‘5,088 만’과 같이 ‘.’ 또는 ‘,’과 숫자 단위가 합쳐진 문자열을 변환하는 것이다. 이를 변환



하기 위해 else if문을 사용하여 ‘만’, ‘억’, ‘숫자단위 없이 , 이 있는 경우’ 이 세가지 경우로 나누었고 ‘만’, ‘억’이 있는 if문 안에서는 또 ‘,’이 있는 경우와 ‘.’이 있는 경우로 나누어서 문자열을 변환했다. ‘.’이 있는 경우는 ‘.’을 기준으로 split 한 후 ‘.’ 뒤에 있던 숫자의 개수만큼 0의 개수를 줄여서 뒤에 붙여주었다. ‘,’이 있는 경우는 ‘,’을 replace로 ‘’로 바꾸고 그 뒤에 숫자 단위만큼 0을 붙여주었다. 이와 같은 방법으로 문자열이 섞인 모든 column들의 값을 전부 변환해주었다.

#### 5) '총 활동일' 열을 새로 만들어 추가

```
#채널 개설일과 현재 날짜와의 날짜 차이(day)값을 '총 활동일' 열을 새로 만들어 추가
now = datetime.now() # 현재 날짜

for i in range(len(df3)):
    date = datetime.strptime(df3.loc[i, '채널 개설일'], '%Y-%m-%d')
    df3.loc[i, '총 활동일'] = (now-date).days
```

‘채널 개설일’ 항목 값과 현재 날짜를 뺀 값을 ‘총 활동일’ 값으로 정의하여 새 column을 만들어보고자 생각했다. 현재 날짜 변수를 생성하고 날짜끼리 계산하기 위해 datetime 모듈을 사용하였다. now 변수에 현재 날짜를 저장하였고 ‘채널 개설일’ column 값 역시 date 변수에 datetime 타입으로 저장하여 이 두변수를 뺀 값을 총 활동일 열을 새로 만들어 저장하였다.

#### 6) string 타입인 숫자값들을 전부 numeric 타입으로 변환

```
# string 타입인 숫자값들을 전부 numeric 타입으로 변환
for col in df3.columns:
    if col not in ['채널명', '카테고리1', '카테고리2', '채널 개설일', '채널주소']:
        df3[col] = pd.to_numeric(df3[col])
```

‘채널명’, ‘카테고리1’, ‘카테고리2’, ‘채널 개설일’, ‘채널주소’를 뺀 모든 column 값들은 전부 숫자로 보여도 타입은 string 타입이므로 to\_numeric 메서드를 이용하여 전부 numeric 타입으로 변환해 주었다. 이는 추후 상관관계 값을 구하기 위해 numeric으로 변환해 둘 필요가 있었기에 시행하였다.

# 추가, 수정, 변환한 df3 데이터프레임

df3

	채널명	카 테 고 리 1	카 테 고 리 2	구독자	조회수	평균 조회 수	전체 동영상	최근 1개월 동영상	좋아요	싫어요	댓글	Nox 평점	UTU 점수	채널 개설 일	채널주소
0	BLACKPINK	음악	영화/비디오 > 음악	31800000	7900000000	17541600	253	0	50880000	1730000	5020000	2.05	75	2016-06-28	https://www.youtube.com/c/BLACKPINK
1	ibighit	음악	영화/비디오 > 음악	30400000	7500000000	8004400	347	12	140000000	2700000	19300000	4.76	78	2008-06-04	https://www.youtube.com/c/ibighit

이와 같이 추가, 수정, 변환한 df3 데이터프레임을 출력해보았다.

## 7) 실시간 스트리밍 중인 채널 찾기

```
#실시간 스트리밍 중인 채널의 Live 영상제목과 링크 찾기
driver = webdriver.Chrome('C:/Users/dudt/Desktop/빅데이터처리및응용/driver/chromedriver')
print("실시간 스트리밍(Live) 중인 채널=====")
for i in range(len(df3)):
    driver.get(df3.loc[i, '채널주소']) # 각 채널의 채널주소를 열어서 실시간 스트리밍 중인지 찾기
    html = driver.page_source
    soup = BeautifulSoup(html, 'lxml')
    tag = soup.find_all("span", {'class': 'style-scope ytd-badge-supported-renderer'})
    tag_text = [tag[i].string for i in range(len(tag))]

    if '실시간 스트리밍 중' in tag_text:
        name = df3.loc[i, '채널명']
        title = soup.find("a", {'id': 'video-title'}).string
        url = 'https://www.youtube.com' + soup.find("a", {'id': 'video-title'})['href']
        print("채널명: %s\n영상제목: %s\n링크: %s" % (name, title.strip(), url))
        print('-----')

drive.close()

실시간 스트리밍(Live) 중인 채널=====
채널명: Larva TUBA
영상제목: LIVE LARVA | GRUDGE MATCH - THE FACE OFF | LARVA OFFICIAL
링크: https://www.youtube.com/watch?v=ZY5oov1_jbo
```

df3의 '채널주소' 값을 이용하여 실시간 스트리밍 중인 채널을 찾고자 하였다. driver을 통해 '채널주소' column 값에 접근하여 그 주소로 이동하면 유튜브 채널 페이지가 나오고, class 이름이 'style-scope ytd-badge-supported-renderer'인 태그에서 텍스트가 '실시간 스트리밍 중'이라는 텍스트가 있을 시, 그 채널이 현재 실시간 스트리밍 중이라는 뜻이므로, 채널명과 영상제목, 영상링크를 소스에서 찾아 출력하는 코드를 짜봤다. df3의 row 수 만큼 채널 주소로 페이지가 이동되므로 한번 실행하는데 꽤 오랜 시간이 걸리는 것이 단점이다.

## 8) 각 변수끼리 상관관계 값 구하기

```
#각 변수끼리 상관관계 구하기
corr = df3.corr()
corr
```

	구독자	조회수	평균 조회수	전체 동영상	최근 1개월 동영상	좋아요	싫어요	댓글	Nox평점	UTU점수	총 활동일
구독자	1.000000	0.687863	0.157284	-0.067745	-0.044816	0.721382	0.566491	0.640432	0.051698	0.514563	0.318345
조회수	0.687863	1.000000	0.210945	0.240690	0.267942	0.535824	0.527722	0.519056	-0.078939	0.539810	0.546465
평균 조회수	0.157284	0.210945	1.000000	0.238639	0.285395	-0.009828	0.065998	0.024206	-0.030265	0.295606	0.285877
전체 동영상	-0.067745	0.240690	0.238639	1.000000	0.927921	-0.133863	-0.199137	-0.132913	-0.159672	0.342943	0.257824
최근 1개월 동영상	-0.044816	0.267942	0.285395	0.927921	1.000000	-0.158714	-0.238458	-0.174329	-0.100094	0.394268	0.281706
좋아요	0.721382	0.535824	-0.009828	-0.133863	-0.158714	1.000000	0.668512	0.910387	0.169718	0.356014	0.338074
싫어요	0.566491	0.527722	0.065998	-0.199137	-0.238458	0.668512	1.000000	0.744321	-0.004639	0.298119	0.254265
댓글	0.640432	0.519056	0.024206	-0.132913	-0.174329	0.910387	0.744321	1.000000	0.207655	0.361472	0.348102
Nox평점	0.051698	-0.078939	-0.030265	-0.159672	-0.100094	0.169718	-0.004639	0.207655	1.000000	0.067736	-0.241143
UTU점수	0.514563	0.539810	0.295606	0.342943	0.394268	0.356014	0.298119	0.361472	0.067736	1.000000	0.448447
총 활동일	0.318345	0.546465	0.285877	0.257824	0.281706	0.338074	0.254265	0.348102	-0.241143	0.448447	1.000000

df3의 numeric형에 해당하는 column들을 변수로 두어 각 변수끼리 상관관계 값을 구하는 corr()메서드를 사용하여 상관관계 값을 저장한 corr 데이터프레임을 만들었다. corr() 메서드를 실행하면 자동으로 numeric 형인 column들끼리 상관관계 값을 구해주는 데이터프레임을 반환한다.



그리고 완성된 corr 데이터를 히트맵으로 시각화하여 한눈에 상관관계 정도를 파악할 수 있도록 하였다. 대각선 값은 서로 같은 항목끼리의 상관관계 값을 구한 것이므로 1.00으로 고정이다. 이 대각선 값은 무시하고 그 외 상관관계 값을 서로 비교해보아야 한다.

```
#각 항목별 상관관계가 높은 변수 상위 3개씩 print
for col in corr.columns:
    abs_corr = abs(corr) # 모든 값을 절대값으로 변환 (양수, 음수 상관없이 상관관계도가 높은 항목을 찾기위해)
    abs_sort = abs_corr[col].sort_values() # 오름차순
    corr_series = corr[col]
    print('변수이름:', col)
    for i in range(1,4):
        index = (-1) - i # -1 index는 항상 1.000000 이므로 (자신과 자신의 상관관계이므로) index -2부터 출력
        # 출력은 corr 원본 값을 출력하여 양수, 음수 여부도 출력되게 함
        print('%d. %s %.6f'%(i,abs_sort.index[index], corr_series[abs_sort.index[index]]))
    print('-----')
```






변수이름: 구독자  
1. 좋아요 0.721382  
2. 조회수 0.687863  
3. 댓글 0.640432  
-----

항목들이 너무 많기 때문에 눈으로도 값들을 파악하기 어렵다는 판단이 들어, 각 항목당 상관관계도가 높은 변수 top3를 출력하였다. 양수, 음수 상관없이 오로지 절대값 기준으로 상관관계도가 높은 항목을 찾기 위해 abs 함수를 이용해 corr 데이터 값을 모두 절대값으로 변환시키고 abs\_sort에 따로 저장시켰고, 이를 오름차순 시켜 index -2부터 -3, -4를 출력하였다. -1 index부터 출력하지 않은 이유는 -1 index 값은 항상 1.00 이므로 (자신과 자신의 상관관계이므로) -2 index부터 출력하였고, 상관관계 값은 양수, 음수 여부도 제대로 출력하기 위해 원본 값이 저장된 corr에서 값을 출력하였다.

## 9) 직접 정의한 변수인 '채널 점수' 값 매기기

df3 column의 'Nox평점', 'UTU점수'는 각각 데이터를 크롤링한 사이트에서 자체적으로 기준을 선정하고 계산한 유튜브 채널 점수이다.

NoxScore 이란?

-  구독자 및 채널 성장 추이 (35% 중량)  
구독자 및 조회수가 많을수록 점수가 높습니다.  
최근 30의 구독자 및 조회수 기반으로 산정됩니다.
-  동영상 업로드 빈도 (15% 중량)  
최근 3개월 업로드된 동영상 기준으로 산정됩니다. 일정 시간 간격으로 꾸준히 업로드할수록 점수가 높게 나옵니다.  
1주일 3-5개의 동영상 업로드가 적절합니다. 최근 3개월 동영상 업로드가 없을 경우 평점이 매우 낮게 나옵니다.
-  동영상 품질 분석 (35% 중량)  
최근 3개월 동영상 기준으로 각 동영상의 조회수와 구독자 수의 비율을 산정하여 비율이 높을수록 품질이 좋은 동영상으로 판단합니다.
-  구독자 참여도 분석 (10% 중량)  
댓글수, 좋아요 그리고 시청자의 비율을 종합적으로 산정된 값입니다. 비율이 높을수록 참여율도 높다는 뜻입니다.
-  소셜 네트워크 종류의 다양성 (5% 중량)  
동영상을 Instagram, Twitter, Facebook 등 SNS 플랫폼으로 공유 가능한지 판단하는 기준입니다. 소셜 네트워크 종류가 풍부할수록 점수가 높습니다.

예를 들어 위 사진과 같이 noxinfluencer 사이트에서 자체적으로 'Nox평점'이라는 채널점수 기준을 만들어 계산하였다. 이번 프로젝트에서도 자체적으로 '채널 점수'라는 변수 값을 직접 정의하여 계산해보기로 하였다.

### '채널 점수'의 기준-----

#### 1. 채널 성장도(구독자수, 전체 조회수) - 비중 30%

구독자수와 전체조회수는 그 채널의 성장규모를 가장 잘 보여주는 지표이므로 비중을 제일 크게 두었다. 각각 15%씩 (구독자수 15%, 전체조회수 15%)

#### 2. 활동도, 업데이트 빈도(전체동영상 수/총 활동일, 최근 1개월 동영상 수) - 비중 25%

하루에 올리는 동영상수 = 전체동영상 수/총 활동일, 최근 1개월 동영상 수, 이 두 항목은 활동빈도, 업데이트 빈도 등 얼마나 활발한 채널인지를 나타내주는 지표이다. 각각 12.5%씩

#### 3. 전반적인 동영상 평가(좋아요 수, 싫어요 수) - 비중 25%

좋아요 수가 클수록, 싫어요 수가 적을수록 동영상 품질이 좋다는 뜻이므로 '채널 점수'또한 점수가 높게 나올 것이다. 각각 12.5%씩

#### 4. 동영상 화제도(평균 조회수) - 비중 10%

평균 조회수가 클수록 각 동영상의 화제도가 크다는 의미이므로 '채널 점수'가 높게 나올 것이다

#### 5. 시청자 소통,참여 (댓글 수) - 비중 10%

댓글 수가 클수록 시청자가 활발히 의견을 주고받는 뜻이므로 시청자 소통, 참여 지표로 선정했다.

### 계산 방법 (총 100점 만점)-----

df3 75개의 row 항목(구독자수, 전체 조회수 등등)의 값을 sort했을때 각 해당하는 순위에 따라 점수를 차등 지급한다. 예를 들어 댓글 수를 sort했을때 댓글 수 (비중 10%) 1위인 채널은 10점을 가져가고, 꼴등인 채널은 0점을 가져가고, 중간 순위인 채널은 5점을 가져간다. 순위별 차등 점수는 (만점 점수)/(전체 등수-1)로 계산한다.

위 기준과 같이 '채널 점수' 변수에 대해 정의해 보았다. 물론 이 '채널 점수'의 한계점이 존재하는데 점수를 차등 지급하는 기준이 없으므로 총 75개의 전체 채널 데이터 각 column 값의 순위가 기준이 되어 그 순위에 따라 점수를 지급하는 방법을 썼다. 즉 점수를 계산하는 전체 집단이 기준이 되므로 집단의 수와 분포에 따라 '채널 점수' 결과값이 언제든지 달라진다.

```
# 채널 점수 값을 추가하고 0점으로 초기화
df3.loc[:, '채널 점수'] = 0.0

# 구독자 순위별로 점수 지급 (만점: 15점)
df3 = df3.sort_values(by='구독자', ascending=False)
score = 15.0
gap = score/(len(df3)-1)
gap = round(gap,6)
for i in range(len(df3)):
    df3.iloc[i, -1] += score
    score -= gap

# 전체 조회수 순위별로 점수 지급 (만점: 15점)
df3 = df3.sort_values(by='조회수', ascending=False)
score = 15.0
gap = score/(len(df3)-1)
gap = round(gap,6)
for i in range(len(df3)):
    df3.iloc[i, -1] += score
    score -= gap
```



```

# 하루에 올리는 동영상 수(전체 동영상 수/총 활동일) 열 추가
df3['하루에 올리는 동영상'] = df3['전체 동영상']/df3['총 활동일']

# 하루에 올리는 동영상 순위별로 점수 지급 (만점: 12.5점)
df3 = df3.sort_values(by='하루에 올리는 동영상', ascending=False)
score = 12.5
gap = score/(len(df3)-1)
gap = round(gap,6)
for i in range(len(df3)):
    df3.iloc[i, -2] += score
    score -= gap

# 최근 1개월 동영상 순위별로 점수 지급 (만점: 12.5점)
df3 = df3.sort_values(by='최근 1개월 동영상', ascending=False)
score = 12.5
gap = score/(len(df3)-1)
gap = round(gap,6)
for i in range(len(df3)):
    df3.iloc[i, -2] += score
    score -= gap

```

```

# 좋아요 순위별로 점수 지급 (만점: 12.5점)
df3 = df3.sort_values(by='좋아요', ascending=False)
score = 12.5
gap = score/(len(df3)-1)
gap = round(gap,6)
for i in range(len(df3)):
    df3.iloc[i, -2] += score
    score -= gap

# 싫어요 순위별로 점수 지급 (만점: 12.5점) *싫어요 수가 적을수록 점수를 많이 지급해야 하므로 오름차순으로 정렬할 것
df3 = df3.sort_values(by='싫어요', ascending=True)
score = 12.5
gap = score/(len(df3)-1)
gap = round(gap,6)
for i in range(len(df3)):
    df3.iloc[i, -2] += score
    score -= gap

# 평균 조회수 순위별로 점수 지급 (만점: 10점)
df3 = df3.sort_values(by='평균 조회수', ascending=False)
score = 10.0
gap = score/(len(df3)-1)
gap = round(gap,6)
for i in range(len(df3)):
    df3.iloc[i, -2] += score
    score -= gap

# 댓글 수 순위별로 점수 지급 (만점: 10점)
df3 = df3.sort_values(by='댓글', ascending=False)
score = 10.0
gap = score/(len(df3)-1)
gap = round(gap,6)
for i in range(len(df3)):
    df3.iloc[i, -2] += score
    score -= gap

```

우선 '채널 점수'라는 column을 추가하고 값을 0.0으로 초기화시켜둔다. 그리고 '채널 점수' 계산 기준에 따라 각 항목을 기준으로 내림차순 정렬한 후 score 변수에 만점에 해당하는 점수를 저장해두고, gap 변수에 순위가 내려갈때마다 바로 위 순위 점수에서 까이는 점수를 계산하여 저장해둔다. 그리고 순위가 가장 높은 index 0번부터 점수를 차등 지급하는 반복문을 실행하여 '채널 점수'에 각 순위에 해당하는 점수를 더한다. 이와 같은 과정을 '채널 점수' 기준에 해당하는 항목별로 모두 실행하면 '채널 점수' 계산이 완료된다.

```
# 채널 점수 높은 순서대로 정렬
df3.sort_values(by='채널 점수', ascending=False)
```

평균 조회 수	전체 동영상	최근 1개월 동영상	좋아요	싫어요	댓글	Nox 평점	UTU 점수	채널 개설일	채널주소	총 팔로워	채널 점수	하루에 올리는 동영상
94790000	17800	292	4590000	54000	360000	4.36	78	2006-03-08	<a href="https://www.youtube.com/channel/UCbD8EppRX3ZwJ...">https://www.youtube.com/channel/UCbD8EppRX3ZwJ...</a>	5024.0	77.668911	3.542994
1747500	3939	19	230000000	4860000	22030000	3.39	63	2006-03-18	<a href="https://www.youtube.com/channel/UCEf_Bc-KVd7on...">https://www.youtube.com/channel/UCEf_Bc-KVd7on...</a>	5014.0	73.040531	0.785600

그리고 다음과 같이 계산한 '채널 점수'가 높은 순서대로 내림차순으로 정렬하여 결과가 제대로 나왔는지 확인했다. 총 100점 만점이지만 가장 높은 '채널 점수'는 77.669라는 결과가 나온걸 확인하였다.