클라우드기반 데이터 보안 전문가 양성 과정

모듈 프로젝트 3

2021년 3월 19일

이름: 공경선

■ 주제

- 영화 데이터 수집 및 분석

■ 기능

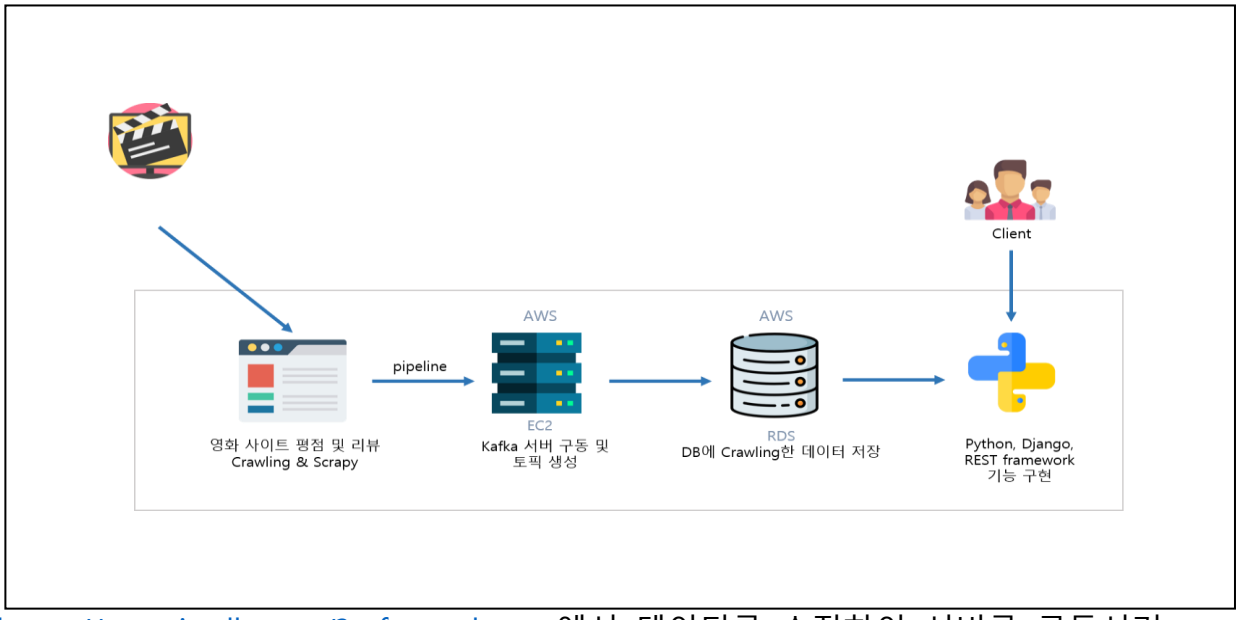
1) 연대별(1920~2020) 평균별점 보여주기

2) 연대별(1920~2020) 평균등수 보여주기

3) 지정 개수의 스포일러가 포함된 영화정보 보여주기

4) n순위까지의 영화 정보 보여주기

▪ 설계



- https://www.imdb.com/?ref_=nv_home에서 데이터를 수집하여 서버를 구동시켜 데이터베이스에 저장 후 데이터를 분석하고 기능을 구현한다.

■ 구현

0.환경

EC2		54.159.115.189
RDS	Hostname	project-db.cfalcgsq6onn.us-east-1.rds.amazonaws.com
	Username	admin
	Password	123123123

1. 과정

1) Scrapy & Crawling

- 필요한 데이터만 추출하여 저장한다.

(1) items.py

```
import scrapy
class ModuleProjectItem(scrapy.Item):

    Ranking = scrapy.Field()
    Title = scrapy.Field()
    R_year = scrapy.Field()
    Rating = scrapy.Field()
    Spoilers = scrapy.Field()
```

- 수집할 데이터구조를 정의하는 items.py 파일 작성

(2) pipelines.py

```
from itemadapter import ItemAdapter
from kafka import KafkaProducer
from json import dumps
import time
class ModuleProjectPipeline:
    def __init__(self):
        self.producer = KafkaProducer(acks=0,compression_type='gzip',bootstrap_servers=[
'54.159.115.189:9092'], value_serializer=lambda x: dumps(x).encode('utf-8'))

    def process_item(self, item, spider):
        data = {"schema":{"type":"struct","fields":[{"type":"int32","optional":"false","
field":"ID"},{"type":"string","optional":"true","field":"Ranking"},{"type":"string","opt
ional":"true","field":"Title"},{"type":"string","optional":"true","field":"R_year"},{"ty
pe":"string","optional":"true","field":"Rating"},{"type":"string","optional":"true","fie
ld":"Spoiler"}],"optional":"false","name":"MOVIE"},"payload":{"ID":10,"Ranking":item.get
('Ranking'),"Title":item.get('Title'),"R_year":item.get('R_year'),"Rating":item.get('Rat
ing'),"Spoiler":item.get('Spoilers')}}
```

```
        self.producer.send("topic_MOVIE", value=data)
        time.sleep(1)
        self.producer.flush()
```

- 수집된 데이터 처리 방식(DB에 저장)을 정의하는 piplines.py 파일 작성

(3) settings.py

```
BOT_NAME = 'module_project'

SPIDER_MODULES = ['module_project.spiders']
NEWSPIDER_MODULE = 'module_project.spiders'

ROBOTSTXT_OBEY = False
CONCURRENT_REQUESTS = 1

ITEM_PIPELINES = {
    'module_project.pipelines.ModuleProjectPipeline': 300,
}

FEED_FORMAT = "csv"
FEED_URI = "my_news.csv"
FEED_EXPORT_ENCODING = 'utf-8'
```

- 프로젝트 모듈간 연결 및 기본 설정을 정의하는 settings.py 파일 작성

(4) mybots.py

```
import scrapy
from scrapy.http import Request
from module_project.items import ModuleProjectItem
import requests
from bs4 import BeautifulSoup

class MybotsSpider(scrapy.Spider):
    name = 'mybots'

    url = 'http://www.imdb.com/chart/top'
    response = requests.get(url)
    if response.status_code == 200:
        html = response.text
        soup = BeautifulSoup(html, 'html.parser')

        items = []
        title_id = []
        for i in range(1,101):
            Title = soup.select_one('#main > div > span > div > div > div.lister > table
 > tbody > tr:nth-child(' + str(i) + ') > td.titleColumn > a').get_text()
            R_year = soup.select_one('#main > div > span > div > div > div.lister > tabl
e > tbody > tr:nth-child(' + str(i) + ') > td.titleColumn > span').get_text()
            Rating = soup.select_one('#main > div > span > div > div > div.lister > tabl
e > tbody > tr:nth-
child(' + str(i) + ') > td.ratingColumn.imdbRating > strong').get_text()
            title_raw = soup.select_one('#main > div > span > div > div > div.lister > t
able > tbody > tr:nth-child(' + str(i) + ') > td.titleColumn > a')
```

```
            item = {}
            item['Ranking'] = str(i)
            item['Title'] = str(Title)
            item['R_year'] = str(R_year).replace("(","").replace(")","")
            item['Rating'] = str(Rating)
            items.append(item)
            title_id.append(str(title_raw).split("/")[2])
    else :
        print(response.status_code)

    allowed_domains = ['imdb.com/title/']
    start_urls = []
    for id in title_id:
        start_urls.append('http://imdb.com/title/' + id + '/reviews?ref_=tt_urv')

    idx = 0
    def parse(self, response):
        Title = response.xpath('//*[@id="main"]/section/div[1]/div/div/h3/a/text()').ext
ract()
        Spoiler = response.xpath('//*[@id="main"]/section/div[2]/div[2]/div/div/div[1]/s
pan/text()').extract()

        items2 = []
        item = ModuleProjectItem()
        item['Title'] = Title[0]
        item['Ranking'] = self.items[self.idx]['Ranking']
        item['R_year'] = self.items[self.idx]['R_year']
        item['Rating'] = self.items[self.idx]['Rating']
        item['Spoilers'] = len(Spoiler)
        items2.append(item)
        self.idx = self.idx+1

        return items2
```

- 데이터 수집을 위한 수행 코드를 정의하는 mybots.py 파일 작성

2) EC2에서 server 구동

- aws의 EC2와 RDS의 데이터베이스를 연결하여 데이터를 저장한다.

① Zookeeper server 구동

```
[ec2-user@ip-172-31-21-85 kafka_2.13-2.7.0]$ ./bin/zookeeper-server-start.sh ./config/zookeeper.properties
OpenJDK 64-Bit Server VM warning: If the number of processors is expected to increase from one, then you should configure t
```

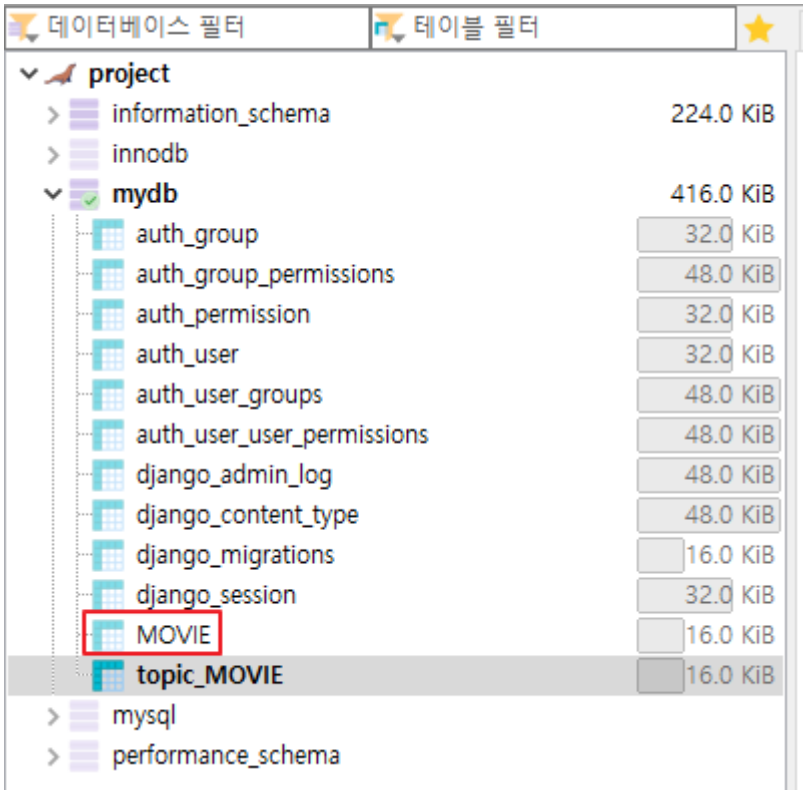
② Kafka server 구동

```
[ec2-user@ip-172-31-21-85 kafka_2.13-2.7.0]$ ./bin/kafka-server-start.sh ./config/server.properties
```

③ Connect 구동

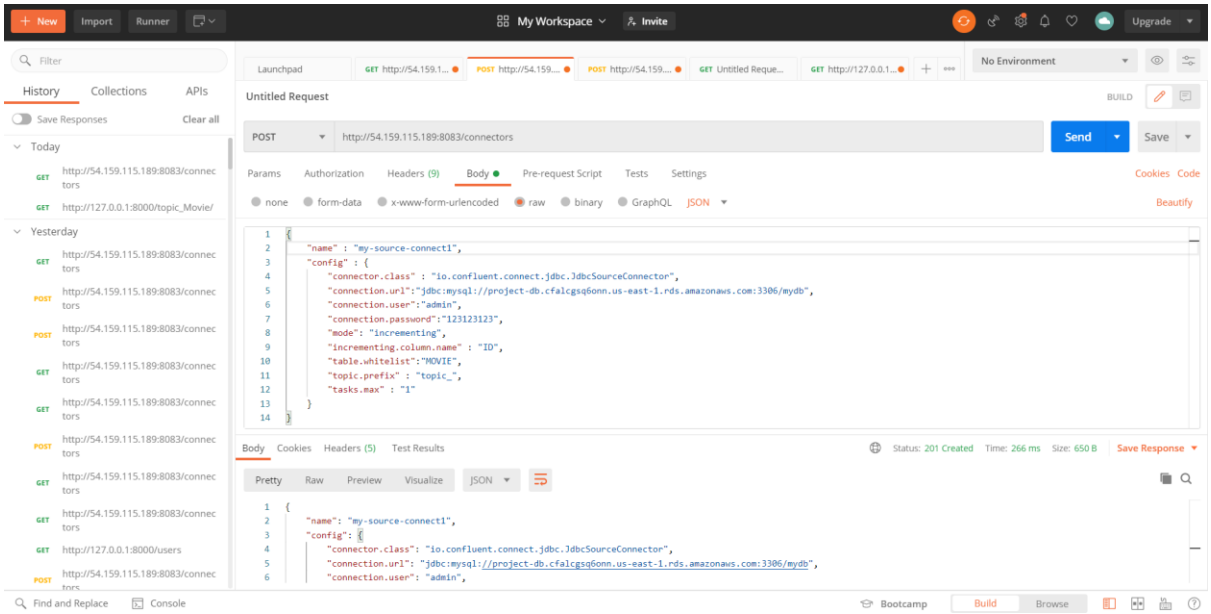
```
[ec2-user@ip-172-31-21-85 confluent-6.1.0]$ ./bin/connect-distributed ./etc/kafka/connect-distributed.properties
```

3) HeidiSQL로 MOVIE table 생성

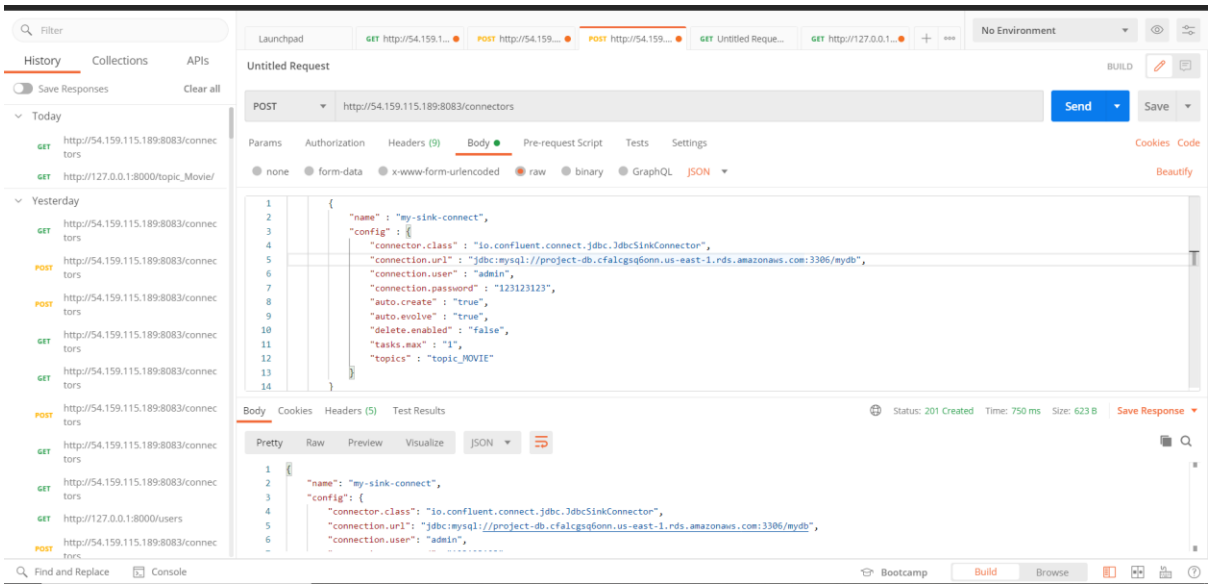


4) Postman

① source 전송

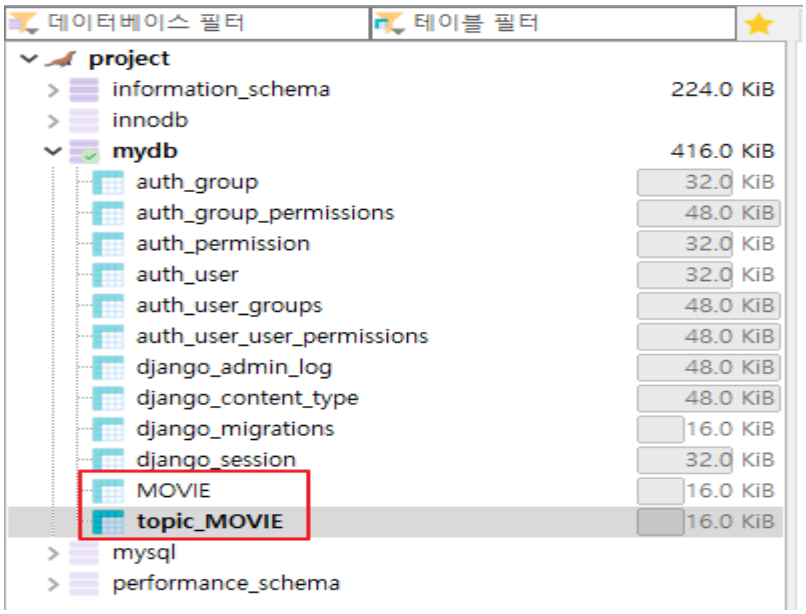


② **sink 전송**

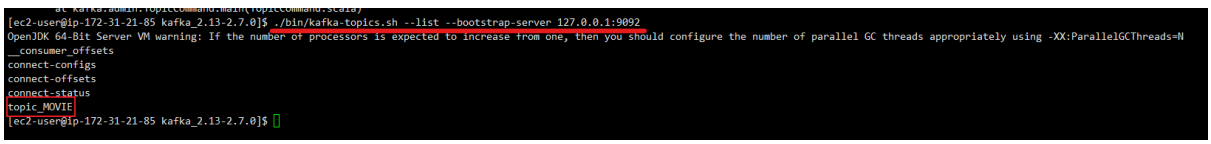


- postman으로 EC2로 source와 sink를 전송하여 MOVIE 테이블을 복사한 topic_MOVIE 테이블과 topic_MOVIE 토픽을 생성한다.

5) topic_MOVIE 테이블 생성



6) topic 생성



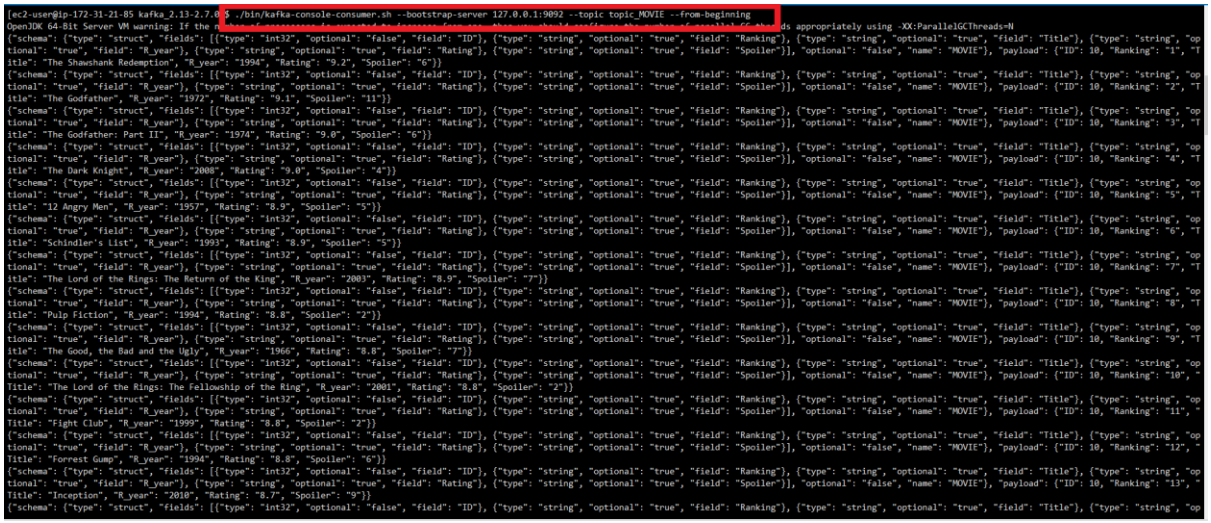
7) scrapy crawl mybots 실행 후 RDS DB에 데이터가 저장됨

Ranking	Title	R_year	Rating	Spoiler
1	The Shawshank Redemption	1994	9.2	6
2	The Godfather	1972	9.1	11
3	The Godfather: Part II	1974	9.0	6
4	The Dark Knight	2008	9.0	4
5	12 Angry Men	1957	8.9	5
6	Schindler's List	1993	8.9	5
7	The Lord of the Rings: The Return of the King	2003	8.9	7
8	Pulp Fiction	1994	8.8	2
9	The Good, the Bad and the Ugly	1966	8.8	7
10	The Lord of the Rings: The Fellowship of the Ring	2001	8.8	2
11	Fight Club	1999	8.8	2
12	Forrest Gump	1994	8.8	6
13	Inception	2010	8.7	9
14	The Lord of the Rings: The Two Towers	2002	8.7	7
15	Star Wars: Episode V - The Empire Strikes Back	1980	8.7	6
16	The Matrix	1999	8.6	3
17	Goodfellas	1990	8.6	5
18	One Flew Over the Cuckoo's Nest	1975	8.6	7
19	Seven Samurai	1954	8.6	5
20	Se7en	1995	8.6	12
21	Life Is Beautiful	1997	8.6	5
22	City of God	2002	8.6	6
23	The Silence of the Lambs	1991	8.6	5
24	It's a Wonderful Life	1946	8.6	6
25	Star Wars: Episode IV - A New Hope	1977	8.6	7
26	Saving Private Ryan	1998	8.6	3
27	The Green Mile	1999	8.5	2
28	Spirited Away	2001	8.5	1
29	Interstellar	2014	8.5	4
30	Parasite	2019	8.5	7
31	Léon: The Professional	1994	8.5	3
32	Hara-Kiri	1962	8.5	4
33	The Usual Suspects	1995	8.5	14
34	The Lion King	1994	8.5	7
35	The Pianist	2002	8.5	5
36	Terminator 2: Judgment Day	1991	8.5	4
37	Back to the Future	1985	8.5	2
38	American History X	1998	8.5	8
39	Modern Times	1936	8.5	5
40	Gladiator	2000	8.5	6

Ranking	Title	R_year	Rating	Spoiler
41	Psycho	1960	8.5	6
42	The Departed	2006	8.5	2
43	City Lights	1931	8.5	5
44	The Intouchables	2011	8.5	3
45	Whiplash	2014	8.5	10
46	Grave of the Fireflies	1988	8.5	8
47	The Prestige	2006	8.5	7
48	Once Upon a Time in the West	1968	8.4	3
49	Casablanca	1942	8.4	3
50	Cinema Paradiso	1988	8.4	7
51	Rear Window	1954	8.4	4
52	Alien	1979	8.4	7
53	Apocalypse Now	1979	8.4	6
54	Memento	2000	8.4	12
55	The Great Dictator	1940	8.4	2
56	Indiana Jones and the Raiders of the Lost Ark	1981	8.4	1
57	Django Unchained	2012	8.4	2
58	The Lives of Others	2006	8.4	6
59	Hamilton	2020	8.4	0
60	Paths of Glory	1957	8.4	12
61	Joker	2019	8.4	4
62	WALL·E	2008	8.4	11
63	The Shining	1980	8.4	8
64	Avengers: Infinity War	2018	8.4	4
65	Sunset Blvd.	1950	8.4	9
66	Witness for the Prosecution	1957	8.4	7
67	Oldboy	2003	8.3	10
68	Spider-Man: Into the Spider-Verse	2018	8.3	3
69	Princess Mononoke	1997	8.3	4
70	Dr. Strangelove or: How I Learned to Stop Worryin...	1964	8.3	7
71	The Dark Knight Rises	2012	8.3	8
72	Once Upon a Time in America	1984	8.3	5
73	Your Name.	2016	8.3	0
74	Aliens	1986	8.3	6
75	Coco	2017	8.3	1
76	Avengers: Endgame	2019	8.3	9
77	Capernaum	2018	8.3	3
78	American Beauty	1999	8.3	6
79	Braveheart	1995	8.3	5
80	High and Low	1963	8.3	5

81	Das Boot	1981	8.3	6
82	Toy Story	1995	8.3	5
83	3 Idiots	2009	8.3	3
84	Amadeus	1984	8.3	5
85	Inglourious Basterds	2009	8.3	12
86	Good Will Hunting	1997	8.3	6
87	Star Wars: Episode VI - Return of the Jedi	1983	8.3	12
88	Like Stars on Earth	2007	8.3	4
89	Reservoir Dogs	1992	8.3	6
90	2001: A Space Odyssey	1968	8.3	9
91	Requiem for a Dream	2000	8.3	9
92	The Hunt	2012	8.3	10
93	Vertigo	1958	8.3	7
94	M	1931	8.3	3
95	Eternal Sunshine of the Spotless Mind	2004	8.3	12
96	Citizen Kane	1941	8.3	8
97	Dangal	2016	8.3	0
98	Singin' in the Rain	1952	8.2	2
99	Bicycle Thieves	1948	8.2	9
100	The Kid	1921	8.2	3

8) consumer에 데이터가 들어오는 것을 확인



- EC2의 topic_MOIVE에 데이터가 들어 오는 것을 확인할 수 있다.

9) Django API 설계 및 구현

- 프로젝트 및 앱 개발에 필요한 디렉토리와 파일을 생성한다.

① models.py

```
from django.db import models

class Movie(models.Model):
    id = models.AutoField(db_column='ID', primary_key=True)
    ranking = models.CharField(db_column='Ranking', max_length=100, blank=True, null=Tru
e)
    title = models.CharField(db_column='Title', max_length=200, blank=True, null=True)
    r_year = models.CharField(db_column='R_year', max_length=20, blank=True, null=True)
    rating = models.CharField(db_column='Rating', max_length=10, blank=True, null=True)
    spoiler = models.CharField(db_column='Spoiler', max_length=1000, blank=True, null=Tr
ue)
    class Meta:
        managed = False
        db_table = 'MOVIE'
class TopicMovie(models.Model):
    id = models.IntegerField(primary_key='True')
    ranking = models.TextField(db_column='Ranking')  # Field name made lowercase.
    title = models.TextField(db_column='Title')  # Field name made lowercase.
    r_year = models.TextField(db_column='R_year')  # Field name made lowercase.
    rating = models.TextField(db_column='Rating')  # Field name made lowercase.
    spoiler = models.TextField(db_column='Spoiler')  # Field name made lowercase.

    class Meta:
        managed = False
```

```
        db_table = 'topic_MOVIE'
```

- 테이블 관련 내용을 개발하는 models.py 파일 작성

② serializers.py

```
from restapi.api.models import TopicMovie
from rest_framework import serializers

class TopicMovieSerializer(serializers.ModelSerializer):
    class Meta:
        model = TopicMovie
        fields = ['id', 'ranking','title','r_year','rating','spoiler']
```

③ views.py

```
from rest_framework import viewsets
from rest_framework import permissions
from restapi.api.models import TopicMovie
from restapi.api.serializers import TopicMovieSerializer
from rest_framework.decorators import action
from rest_framework.response import Response
import pymysql
import re
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pandas import Series, DataFrame
import os

# Create your views here.
class TopicMovieViewSet(viewsets.ReadOnlyModelViewSet):
    queryset = TopicMovie.objects.all()
    serializer_class = TopicMovieSerializer

    permissions_classes=[permissions.IsAuthenticated]

    @action(detail=False, methods=['GET'])
    def rating_year(self, request):
        q = request.GET['year']
        df = mean_rating()

        # 연도 별 평점 평균
        temp = df.groupby('year_Range',as_index=False)['Rating']
        data = temp.mean()

        temp2 = df.groupby('year_Range')['Rating']
        data2 = temp2.mean()
        plt.plot(data2,color='#e35f62')
        plt.xlabel('year_Range')
        plt.ylabel('Rating')
        plt.savefig('year_mean.png')
        data['png_path'] = str(os.path.abspath("year_mean.png"))

        data = data[data['year_Range'] == q]
        return Response(data)

    @action(detail=False, methods=['GET'])
    def ranking_year(self, request):
        q = request.GET['year']
```

```
        df = mean_rating()

        # 연도 별 평점 평균 등수
        temp = df.groupby('year_Range',as_index=False)['Ranking']
        data = temp.mean()

        temp2 = df.groupby('year_Range')['Ranking']
        data2 = temp2.mean()
        plt.plot(data2)
        plt.xlabel('year_Range')
        plt.ylabel('Ranking')
        plt.savefig('year_rank.png')
        data['png_path'] = str(os.path.abspath("year_rank.png"))

        data = data[data['year_Range'] == q]
        return Response(data)

    @action(detail=False, methods=['GET'])
    def top_rating(self, request):
        q = request.GET['num']
        df = db_connect()

        # 평점 순위
        df = df.astype({'Ranking' : int,'R_year' : int, 'Rating' : np.float})
        data = df.drop(['ID', 'Spoiler'], axis=1)

        data = data.head(int(q)).T
        return Response(data)

    @action(detail=False, methods=['GET'])
    def spoiler(self, request):
        q = request.GET['num']
        df = db_connect()

        # 영화 리뷰의 스포일러 개수(n개 이하)
        df = df.astype({'Ranking' : int,'R_year' : int, 'Rating' : np.float, 'Spoiler' : int})
        data = df.drop(['ID'],axis=1)

        plt.plot(df['Ranking'],df['Spoiler'])
        plt.xlabel('Ranking')
        plt.ylabel('Spoiler CNT')
        plt.savefig('spoiler_count.png')

        data = data[data['Spoiler'] <= int(q)]
        data['png_path'] = None
        data.iloc[[0], [5]] = str(os.path.abspath("year_rank.png"))
        data2 = data.T

        return Response(data2)

# DB 연결
def db_connect():
    connect = pymysql.connect(host='project-db.cfalcgsq6onn.us-east-
1.rds.amazonaws.com', user='admin', password='123123123', db='mydb',charset='utf8mb4')
    cur = connect.cursor()
    query = "select * from topic_MOVIE"
    cur.execute(query)
    connect.commit()

    global df
    df = pd.read_sql(query, connect)
    return df
```

```
# 연도 별로 묶기
def mean_rating():
    df = db_connect()
    df = df.astype({'Ranking' : int,'R_year' : int, 'Rating' : np.float, 'Spoiler' : int})
    df1 = df.drop(['ID'],axis=1)
    conditionlist = [
        (df1['R_year'] >= 2020) ,
        (df1['R_year'] >= 2010) & (df1['R_year'] <2020),
        (df1['R_year'] >= 2000) & (df1['R_year'] <2010),
        (df1['R_year'] >= 1990) & (df1['R_year'] <2000),
        (df1['R_year'] >= 1980) & (df1['R_year'] <1990),
        (df1['R_year'] >= 1970) & (df1['R_year'] <1980),
        (df1['R_year'] >= 1960) & (df1['R_year'] <1970),
        (df1['R_year'] >= 1950) & (df1['R_year'] <1960),
        (df1['R_year'] >= 1940) & (df1['R_year'] <1950),
        (df1['R_year'] >= 1930) & (df1['R_year'] <1940),
        (df1['R_year'] <1930)]
    choicelist = ['2020', '2010', '2000', '1990', '1980', '1970', '1960', '1950', '1940', '1930', '1
920']
    df1['year_Range'] = np.select(conditionlist, choicelist, default='Not Specified')

    return df1
```

- 애플리케이션 로직을 개발하는 views.py 파일 작성

④ settings.py

```
from pathlib import Path

BASE_DIR = Path(__file__).resolve().parent.parent
SECRET_KEY = 'ei3il)#*_=9%kfl(m8w60l2(meh!ikpe$v-i^40xh(6!r791q&'
DEBUG = True
ALLOWED_HOSTS = []
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'restapi.api',
    'rest_framework']

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',]
ROOT_URLCONF = 'restapi.urls'
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
```

```
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',],},},]

WSGI_APPLICATION = 'restapi.wsgi.application'
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'mydb',
        'USER' : 'admin',
        'PASSWORD' : '123123123',
        'HOST' : 'project-db.cfalcgsq6onn.us-east-1.rds.amazonaws.com',
        'PORT' : 3306}}

AUTH_PASSWORD_VALIDATORS = [
    {'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },]

LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'Asia/Seoul'
USE_I18N = True
USE_L10N = True
USE_TZ = True
STATIC_URL = '/static/'
```

- 데이터베이스 설정, 템플릿 항목 설정, 정적 파일 항목 설정, 애플리케이션 등록, 타임존 지정 등을 작성하는 settings.py 파일 작성

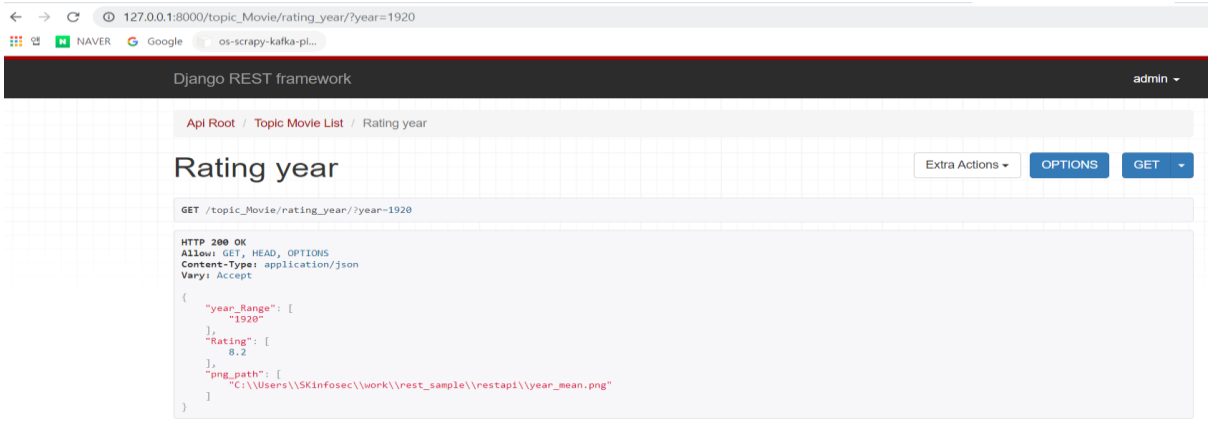
⑤ **urls.py**

```
from django.contrib import admin
from django.urls import path,include
from rest_framework import routers
from restapi.api import views

router = routers.DefaultRouter()
router.register(r'topic_Movie',views.TopicMovieViewSet)

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include(router.urls)),
    path('apt-auth', include('rest_framework.urls',
    namespace = 'rest_framework'))]
```

- URL 및 뷰 매칭 관계를 정의하는 urls.py 파일 작성

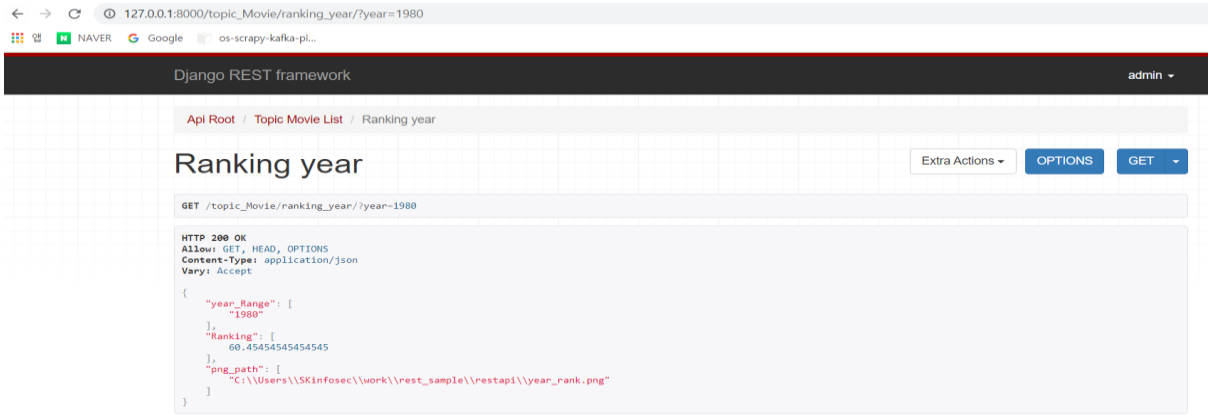
■ 기능 구현

1) 연대별(1920~2020) 평균별점 보여주기



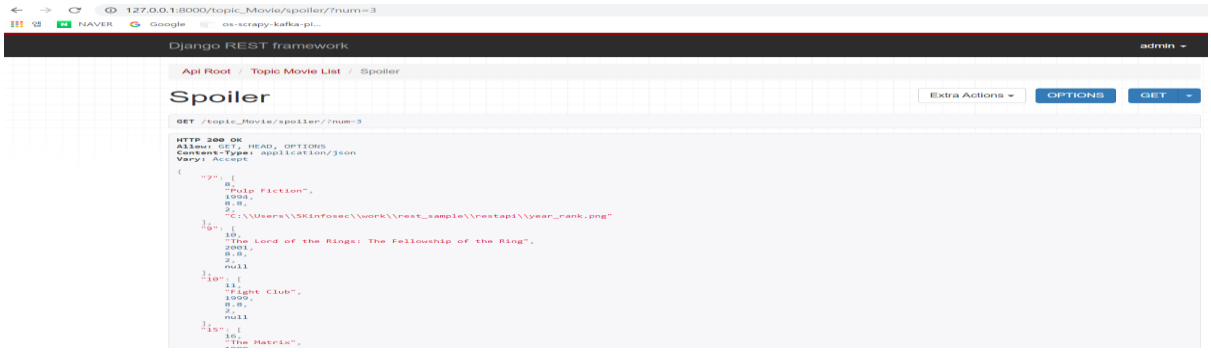
예시) 1920년대 평균평점 및 데이터분석 이미지 경로 출력

2) 연대별(1920~2020) 평균등수 보여주기



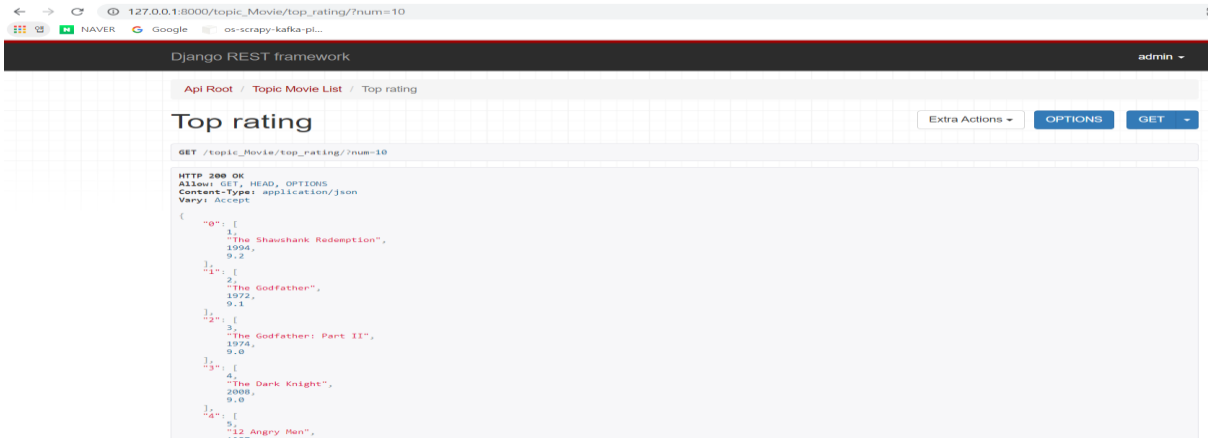
예시) 1980년대 평균등수 및 데이터분석 이미지 경로 출력

3) 지정 개수 이하의 스포일러가 포함된 영화정보 보여주기



예시) 3을 입력하면 스포일러수가 3개이하인 영화정보 출력

4) n순위까지의 영화 정보 보여주기



예시) TOP10 영화정보 출력

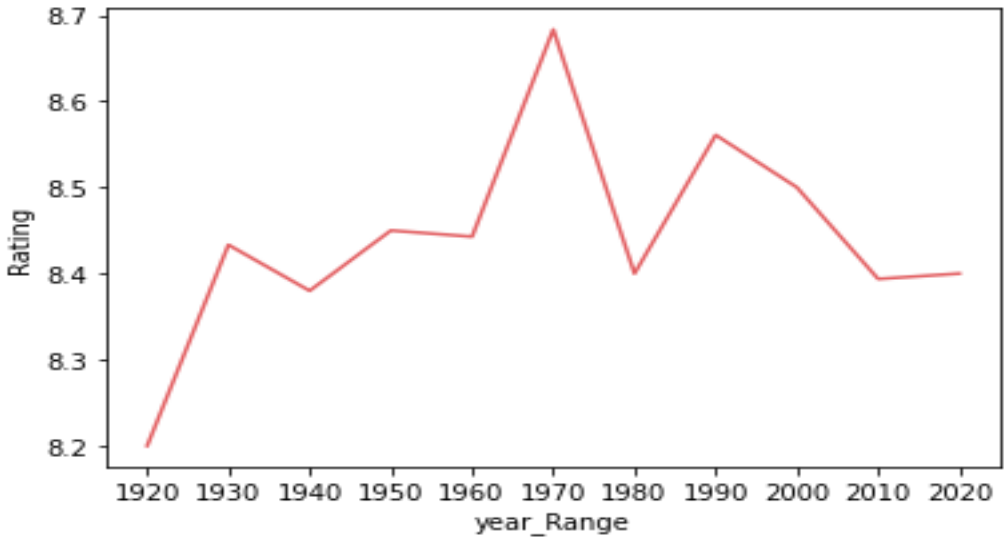
Cf) 데이터 분석 시각화

1) 연대별 평균 등수

- 최고 평균 등수 : 1970년대

- 최저 평균 등수 : 1920년대

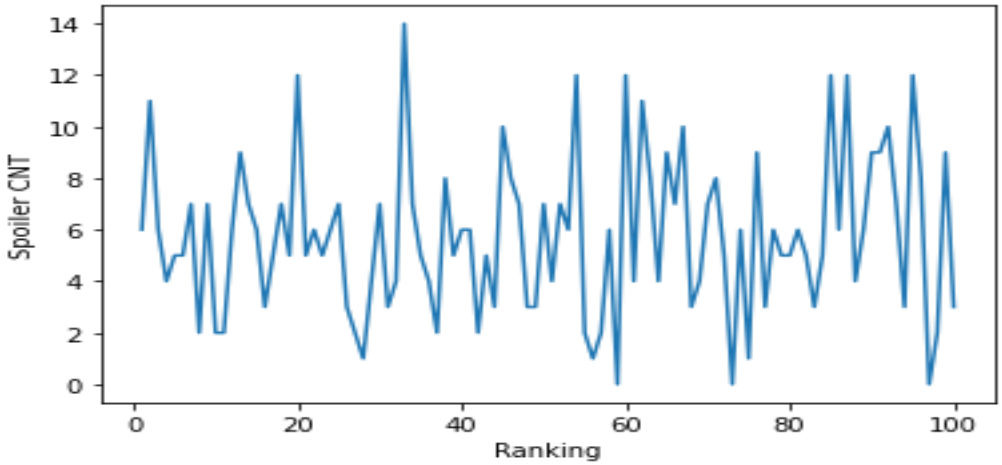
2) 연도별 평균 평점

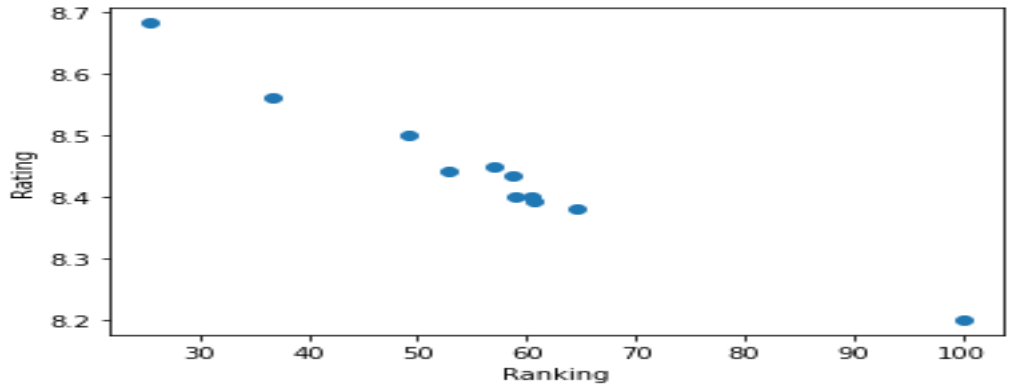


- 최고 평균 평점 : 1970년대

- 최저 평균 평점 : 1920년대

3) 등수별 스포일러리뷰 수





4) 등수와 평점 상관관계

	Ranking	Rating	Spoiler
Ranking	1.000000	-0.980689	-0.496102
Rating	-0.980689	1.000000	0.549042
Spoiler	-0.496102	0.549042	1.000000

■ 프로젝트 후기

7일간의 짧은 기간동안 프로젝트를 하면서 부족한 부분을 많이 발견할 수 있었습니다. 하지

만 1차, 2차 프로젝트와 달리 팀으로 이루어져 팀원들과 서로의 부족한 부분을 채워나가며 프로젝트를 기간내에 끝 마칠 수 있었습니다. 또한, 팀으로 프로젝트를 진행하면서 팀원들과의 소통, 의견 조율 등 많은 것을 배울 수 있는 경험이였습니다. 이번 프로젝트 경험을 통해 다음 프로젝트에서 더욱 성장한 모습으로 임할 수 있을 거라 생각합니다.