Group 9 ETL-Project

Group 9 ETL Project proposal

Explore available data for streaming services across similar platforms and create a database that could be used for further analysis of various aspects of streaming services, movies, etc

Group 9 ETL Project members

1. Ankita Puri
2. Carmen Jiaman Li
3. Kevin Swartz

Project overview

Explored relationship between two streaming service datasets from different sources. Focussing on movie titles across various platforms/service providers - as opposed to including tv shows as well. Service providers were narrowed down to three main providers for which available data was most consistent.

The final dataset allows for further analysis of a number of avenues of interest. For example, information about which actors appear most regularly, which actors attract the highest BoxOffice revenue most regularly, which Directors attract the highest BoxOffice revenue most regularly, average movie ratings by actor appearances, average movie ratings by Director, etc.

Further analysis is beyond the scope of this project

Data sources (EXTRACT)

1. Kaggle

We have found a recent dataset about the information of movies available on different streaming platforms. The entire data is stored in a CSV file by 17 columns including title, year, genre, age group, IMDB rating and platforms to watch. The data objects include text, character, and integer.

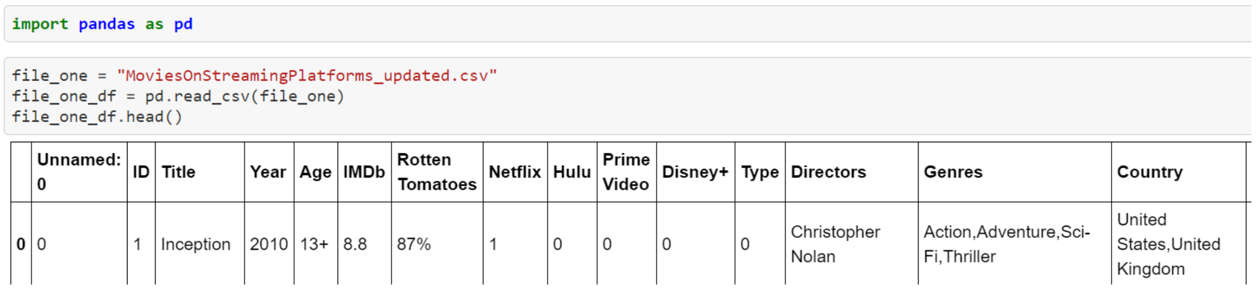
2. OMDB API

Using OMDB API, we have obtained three-thousands of additional movie information such as movie directors, writers, plots and sales by Boxoffice. The data was called by the same title from the first database where a title list was generated and stored in json format. All information was then appended to the empty list which then allowed us to perform data cleaning and analysis.

Clean-up and analysis (TRANSFORM)

• After downloading several similar datasets, we narrowed these down to two relatively distinct sets that had some common data but largely offered different information on the same movies and streaming platforms. The aim was to provide a set from which a wide range of analysis could be performed.

Our first process to extract the data was by reading the Kaggle CSV file on Pandas as a dataframe.

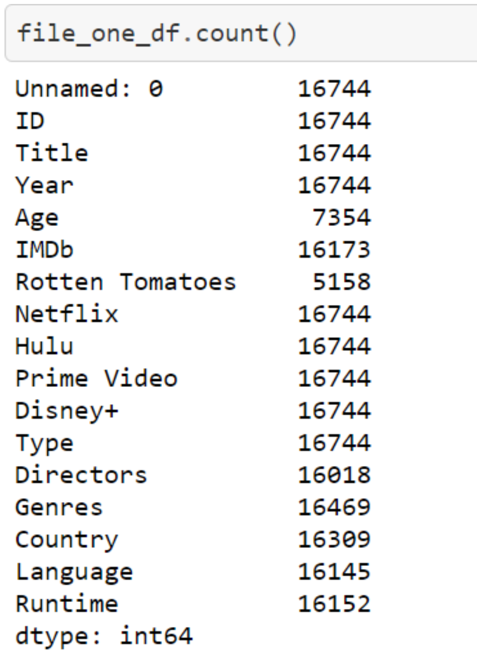


```
import pandas as pd
```

```
file_one = "MoviesOnStreamingPlatforms_updated.csv"
file_one_df = pd.read_csv(file_one)
file_one_df.head()
```

	Unnamed: 0	ID	Title	Year	Age	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Type	Directors	Genres	Country
0	0	1	Inception	2010	13+	8.8	87%	1	0	0	0	0	Christopher Nolan	Action,Adventure,Sci-Fi,Thriller	United States,United Kingdom

To check which data to be cleaned, we have used the "count" function as an indicator that which columns contain the least unique value. It helped us to reduce the number of rows that only contained valid information. Then we have dropped all rows that contained Null values.

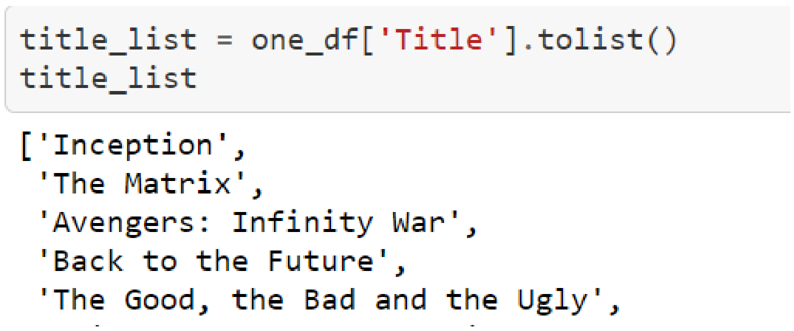


```
file_one_df.count()
```

```
Unnamed: 0         16744
ID                 16744
Title              16744
Year               16744
Age                 7354
IMDb               16173
Rotten Tomatoes     5158
Netflix            16744
Hulu               16744
Prime Video        16744
Disney+            16744
Type               16744
Directors          16018
Genres             16469
Country            16309
Language           16145
Runtime            16152
dtype: int64
```

```
one_df=file_one_df.dropna(how='
```

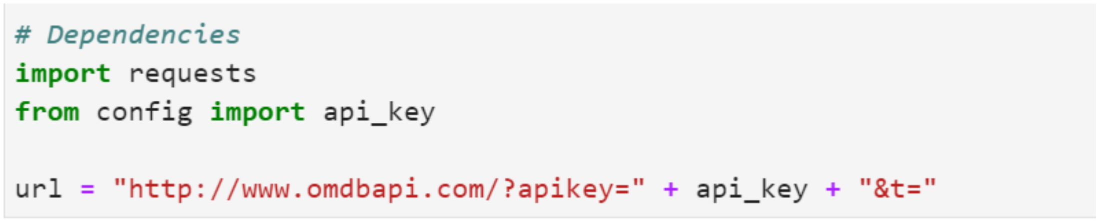
In order to successfully call the right information from the second API data source, we have generated a title list based on the Kaggle Pandas dataframe.



```
title_list = one_df['Title'].tolist()
title_list
```

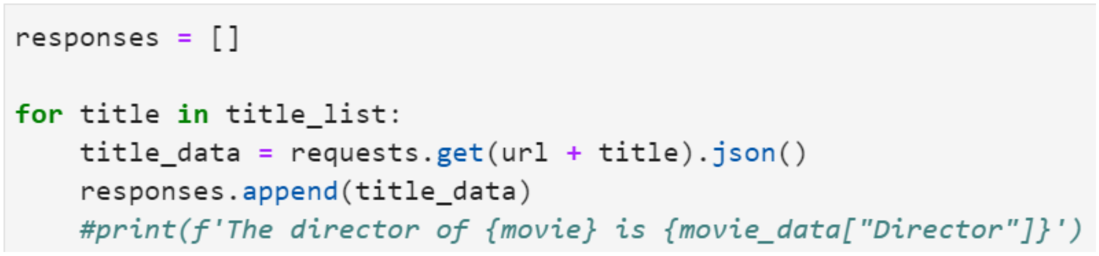
```
['Inception',
 'The Matrix',
 'Avengers: Infinity War',
 'Back to the Future',
 'The Good, the Bad and the Ugly',
```

The title list was then used as the search item to call information from the OMDB API website through a "for loop" function. We were able to store the jason formatted data into the empty response list and transformed into a Pandas data frame. The responses returned invalid rows and columns so we performed data-cleaning process to drop NaN values.



```python
# Dependencies
import requests
from config import api_key

url = "http://www.omdbapi.com/?apikey=" + api_key + "&t="
```

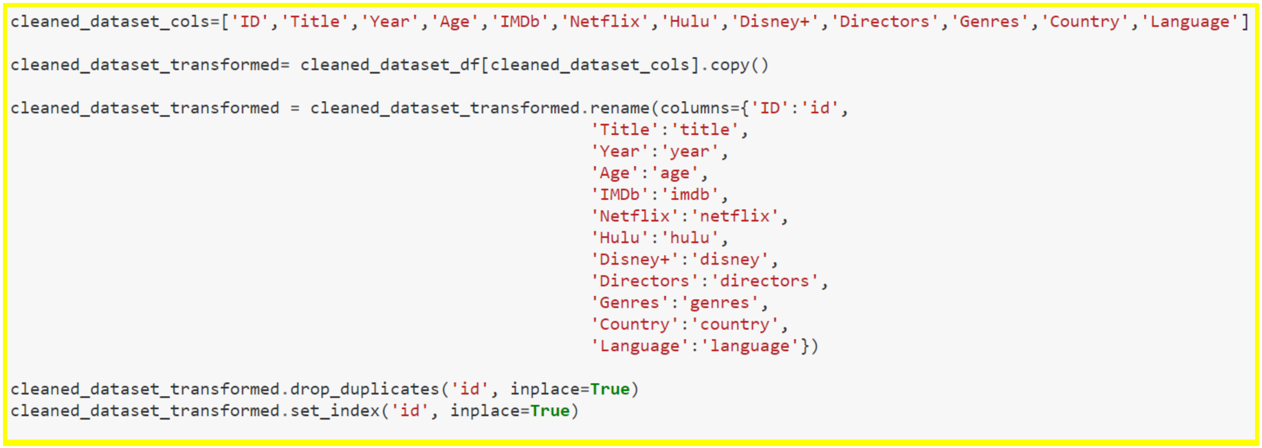


```python
responses = []

for title in title_list:
    title_data = requests.get(url + title).json()
    responses.append(title_data)
    #print(f'The director of {movie} is {movie_data["Director"]}')
```

```
omdb_df = pd.DataFrame(responses)
```

Once the API calling process was successfully done and cleaned, we were able to compare and discuss what information would be useful for this project. It enabled us to select columns and conducted the data transform process by making a copy of the original dataset. All transformed Pandas datasets were renamed columns to match the database column names created at the Postgresql in order to set connection successfully. Then we set up the connection from Pandas to the Postgresql and checked the connect was working by printing out the correct table names.

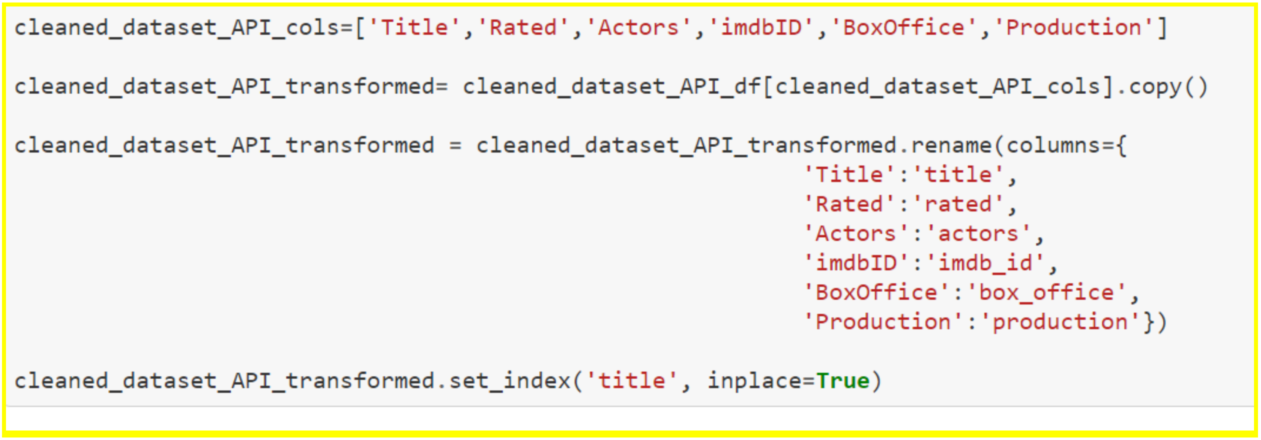


```python
cleaned_dataset_cols=['ID','Title','Year','Age','IMDb','Netflix','Hulu','Disney+','Directors','Genres','Country','Language']

cleaned_dataset_transformed= cleaned_dataset_df[cleaned_dataset_cols].copy()

cleaned_dataset_transformed = cleaned_dataset_transformed.rename(columns={'ID':'id',
                                                          'Title':'title',
                                                          'Year':'year',
                                                          'Age':'age',
                                                          'IMDb':'imdb',
                                                          'Netflix':'netflix',
                                                          'Hulu':'hulu',
                                                          'Disney+':'disney',
                                                          'Directors':'directors',
                                                          'Genres':'genres',
                                                          'Country':'country',
                                                          'Language':'language'})

cleaned_dataset_transformed.drop_duplicates('id', inplace=True)
cleaned_dataset_transformed.set_index('id', inplace=True)
```



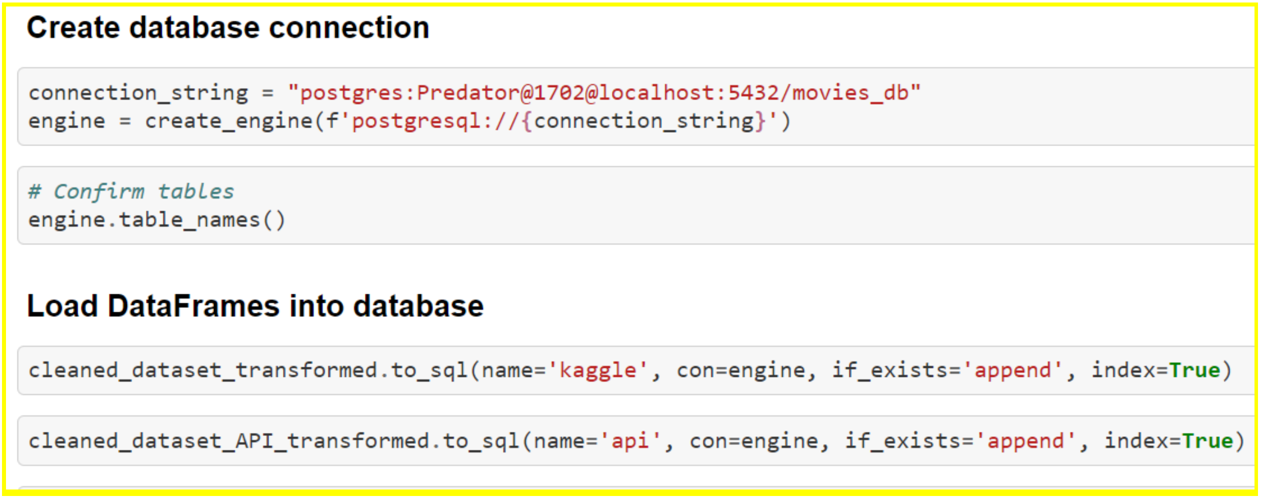
```
cleaned_dataset_API_cols=['Title','Rated','Actors','imdbID','BoxOffice','Production']

cleaned_dataset_API_transformed= cleaned_dataset_API_df[cleaned_dataset_API_cols].copy()

cleaned_dataset_API_transformed = cleaned_dataset_API_transformed.rename(columns={
                                                         'Title':'title',
                                                         'Rated':'rated',
                                                         'Actors':'actors',
                                                         'imdbID':'imdb_id',
                                                         'BoxOffice':'box_office',
                                                         'Production':'production'})

cleaned_dataset_API_transformed.set_index('title', inplace=True)
```

Our final step was loading the data from Pandas to SQL through the connect we set up and checking the tables on Postgresql database can be seen.



Create database connection

```
connection_string = "postgres:Predator@1702@localhost:5432/movies_db"
engine = create_engine(f'postgresql://{connection_string}')
```

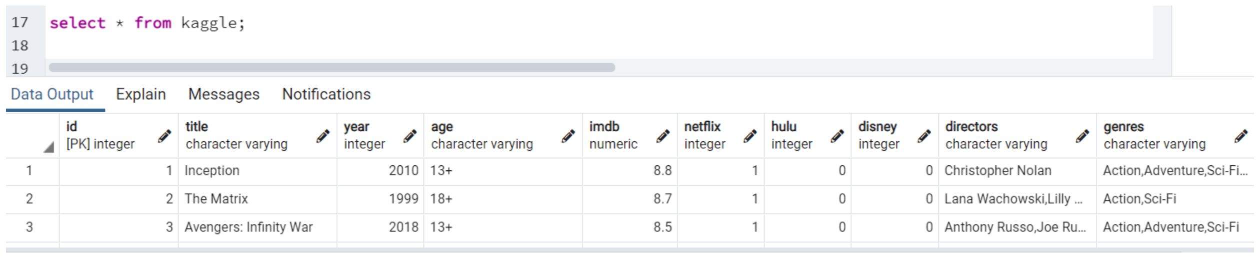
```
# Confirm tables
engine.table_names()
```

Load DataFrames into database

```
cleaned_dataset_transformed.to_sql(name='kaggle', con=engine, if_exists='append', index=True)
```

```
cleaned_dataset_API_transformed.to_sql(name='api', con=engine, if_exists='append', index=True)
```

```
create table kaggle(
  ID serial PRIMARY KEY,
  Title VARCHAR ,
  Year integer ,
  Age VARCHAR ,
  IMDb decimal,
  Netflix integer,
  Hulu integer ,
  Disney integer,
  Directors VARCHAR,
  Genres VARCHAR,
  Country VARCHAR,
  Language VARCHAR
);
```



```
select * from kaggle;
```

Data Output | Explain | Messages | Notifications

	id [PK] integer	title character varying	year integer	age character varying	imdb numeric	netflix integer	hulu integer	disney integer	directors character varying	genres character varying
1	1	Inception	2010	13+	8.8	1	0	0	Christopher Nolan	Action,Adventure,Sci-Fi...
2	2	The Matrix	1999	18+	8.7	1	0	0	Lana Wachowski,Lilly ...	Action,Sci-Fi
3	3	Avengers: Infinity War	2018	13+	8.5	1	0	0	Anthony Russo,Joe Ru...	Action,Adventure,Sci-Fi

We then integrated the tables by inner join on title.

```
SELECT kaggle.id, kaggle.title, kaggle.year, kaggle.age,kaggle.netflix, kaggle.hulu,
FROM kaggle
JOIN api
ON kaggle.title = api.title;
```

• The data was integrated using merges.

• Discussed and identified potential issues with our chosen datasets

• Cleaned the two remaining datasets to remove columns and rows which had missing or inadequate data and compiled two new separate and clean dataframes for later use in combining datasets

• All work on the chosen datasets/dataframes was done in Jupyter Notebook

Data storage (LOAD)

• We chose Relational Database (Postgres) as it provides extensive data capacity and is trusted for its data integrity.

• We chose this database as it will allow us to easily apply analytical functions and derived data

• Our tables will be kaggle and api. We created the tables in postgres and after establishing the connection ,data has been loaded to the corresponding tables.

• The potential limitations we can run into is the dataset having limited data than expected ~ This dataset has data mainly focussed on Netflix over the other streaming services.

• We will explore different sources of data.

Repository files

Resources

Contains chosen csv download after eliminating unwanted datasets, then added cleaned csv files:

- **MoviesOnStreamingPlatforms_updated.csv.** Downloaded from Kaggle
- **cleaned_dataset.csv**. Cleaned version of **MoviesOnStreamingPlatforms_updated.csv**
- **cleaned_dataset_API.csv.** Cleaned version of dataset derived from API extracted from OMDB API

Script

- **API for project.ipynb.** Rough pre-work to extract only movie data from API data and to explore data available from MoviesOnStreamingPlatforms.csv
- **Grp_9_ETL_main.ipynb**. Final workings for presentation
- **config.py**. Contains api key for retrieving data from OMDB API
- Tables.sql.