[redacted] DATA ENGINEERING TEST

Language

- Python

Format

- PY or IPYNB file

Note

- Note that you will be evaluated based on script readability and performance
- Assume that you are loading data into a PostgreSQL database for Section I and II
- You may use dummy values for host, port, database, username and password for the database engine
- We should be able to run your code without troubleshooting

Section I – API

You will be tested on API usage in this section.

The mission is to build an extract, transform and load data of Star Wars data. The API used is from https://swapi.dev/. In this mission, you will be required to build three different tables that will be loaded to the database. The script should include the following elements.

- API Data Extraction
- Data Transformation
- Dummy load to database

The tables to be created are as follow.

Missing data should be stored as NULL value.

dim_film		
Field Name	**Field Type**	**Description**
id	Integer	Pkey of table (based on url)
title	String	-
episode_id	Integer	-
opening_crawl	String	-
director	String	-
producer	String	-
character_count	Integer	Distinct count of people in the file
planet_count	Integer	Distinct count of planet in the file
starship_count	Integer	Distinct count of starship in the file
vehicle_count	Integer	Distinct count of vehicle in the file
species_count	Integer	Distinct count of species in the file
release_date	Date	-
created	Datetime	Tz-naïve datetime
edited	Datetime	Tz-naïve datetime

dim_people		
Field Name	**Field Type**	**Description**
id	Integer	Pkey of table (based on url)
name	String	-
height	Integer	-
mass	Integer	-
hair_color	String	-
skin_color	String	-
eye_color	String	-
birth_year	String	-
gender	String	-
homeworld	String	String version of the homeworld Note that url of homeworld is not the value to be stored
species_count	Integer	Distinct count of species in the file
vehicle_count	Integer	Distinct count of vehicle in the file
starship_count	Integer	Distinct count of starship in the file
created	Datetime	Tz-naïve datetime
edited	Datetime	Tz-naïve datetime

film_people_map		
Field Name	**Field Type**	**Description**
film_id	Integer	Pkey of dim_film
people_id	Integer	Pkey of dim_people

Section II – Web Scraping

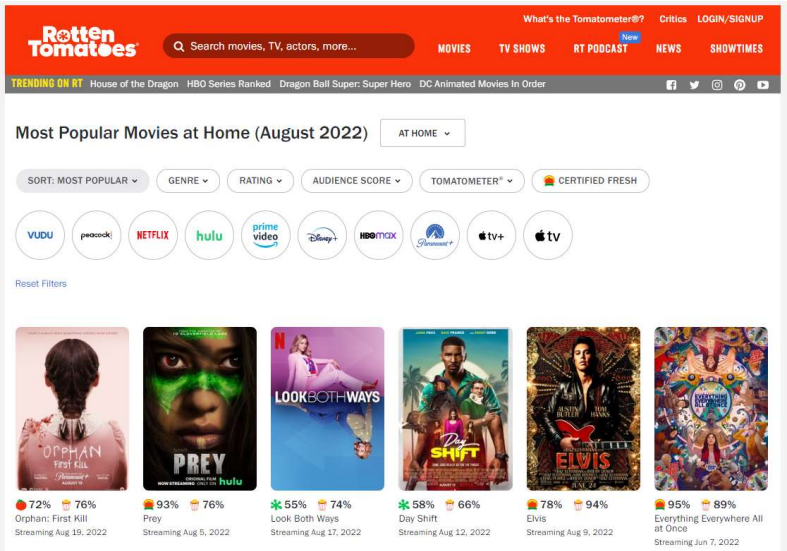
You will be tested on Web Scraping skills in this section.

The mission is to extract, transform and load the data from the first page of the Most Popular Movies at Home in Rotten Tomatoes via web scraping. Build a web scraper using Python packages of your choice to obtain the required data and load it to a Database. The script should include the following elements.

- Web Scraper
- Data Transformation
- Dummy load to database

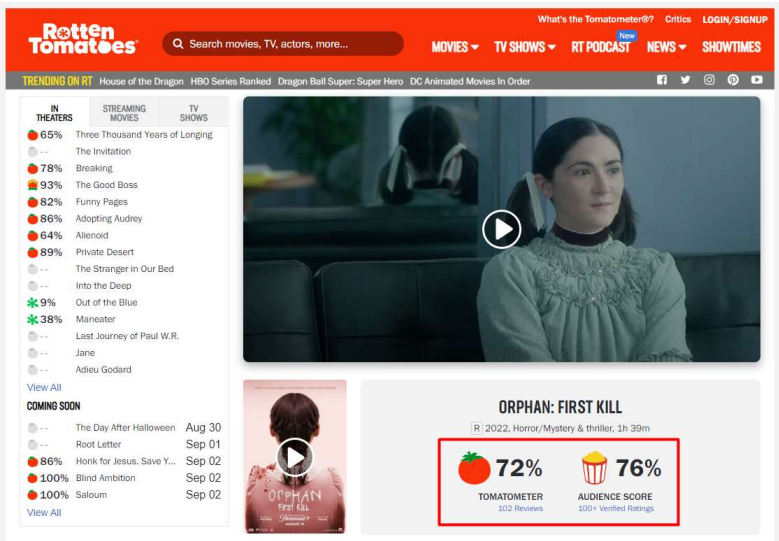
First Page of Most Popular Movies at Home (note that what you see may be different):

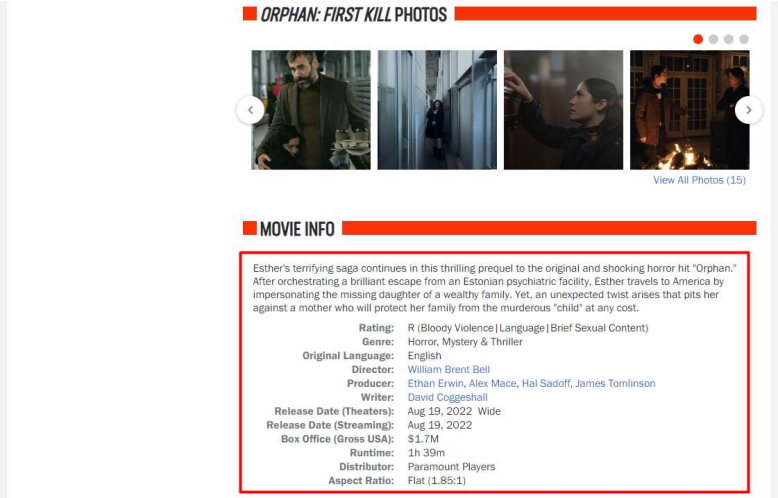
https://www.rottentomatoes.com/browse/movies_at_home/sort:popular?page=1



Movie Detail Page (note that the first movie detail you see may be different):

https://www.rottentomatoes.com/m/orphan_first_kill





The schema of the table to be created and added to the database is as follows.

Take note to remove all consecutive whitespace for string values e.g. tabs, space, newline etc

Missing data should be stored as NULL value.

fact_top_movies		
Field Name	**Field Type**	**Description**
year	Integer	Year of release
title	String	Title of movie Should not contain year information e.g. (2020)
reviews	Integer	# of review
tomatometer	Float	Found at top of Movie Detail Page 98% should be stored as 0.98
audience_score	Float	Found at top of Movie Detail Page 98% should be stored as 0.98
synopsis	String	Found under Movie Info section
rating	String	Found under Movie Info section
genre	String	Found under Movie Info section
original_language	String	Found under Movie Info section
director	String	Found under Movie Info section
producer	String	Found under Movie Info section
writer	String	Found under Movie Info section
release_date_theaters	Datetime	Found under Movie Info section
release_date_streaming	Datetime	Found under Movie Info section
runtime_mins	Integer	Found under Movie Info section Note that runtime should be in minutes
distributor	String	Found under Movie Info section
production_co	String	Found under Movie Info section
aspect_ratio	String	Found under Movie Info section
url	String	Url of Movie Detail Page

Section III – Flask

You will be tested on Flask skills in this section.

The mission is to create RESTful API using Flask package. You will use localhost as the API server with employees.csv file acting as the database. In this mission, the following endpoints are required.

- GET /employee – obtains a json output for all existing employees in the database
- GET /employee/id – obtain a json output of an existing employee from the database
- POST /employee – Add a new employee to the database
- POST /employee/id – Update an existing employee in the database (id should not be updatable)
- DELETE /employee/id – delete an existing employee from the database