### INTRODUCTION

Welcome to Hollywood! You’ve been hired as the newest data engineer for a data science group that specializes in entertainment. The team is building a set of models that will assess historical box office trends to provide recommendations on the type of movies that will be successful. The studio is interested in expanding their portfolio of films that garner critical or popular acclaim.

Your role as a data engineer is to build a data pipeline. You’ll need to fetch data from several different sources, perform cleaning steps, and store the data in a structured format. This dataset will be used by other members on the team for modeling efforts and analyses.

### DATA SOURCES

1. ‘movies.db’ is a sqlite database with one table of ~4,700 rows and 33 dimensions.
2. ‘Movie\_Data\_Dictionary.xlsx’ is a data dictionary file which includes descriptions for the 33 dimensions. As this is a sample data set, incomplete records may exist.
3. ‘OMDb API’ is a RESTful web service to obtain movie information.

Web address: <http://www.omdbapi.com/>

API key: 273f5165

Example request: <http://www.omdbapi.com/?i=tt3896198&apikey=273f5165>

*IMPORTANT:* because querying the API may take up to a couple of hours, we recommend starting Step 2 under “PROCESS” as soon as possible upon receiving these instructions.

### PROCESS

1. Using any language (Python is preferred), write code to connect to the provided database. done
2. Use the provided OMDb API key to query the Open Movie Database and obtain any and all additional metrics to enrich the dataset. done
3. Perform any necessary data cleansing, as well as some simple exploratory analysis -- whatever steps you would normally take in order to get a sense of what is in a new data set. done
4. Make note of any problematic data and missing values in each of the sources. done
5. Break out the ‘genre’ column (containing a list of genres) into multiple binary “dummy” columns. done
6. Store the cleaned and enriched dataset as a separate table in movies.db. done
7. Using any language or methodology, devise a scheme to automatically run your code on a weekly schedule. done
8. Create a simple web applcation using any web framework (python preffered) which will let a user choose one of (movie\_facebook\_likes, director\_facebook\_likes, actor\_1\_facebook\_likes, num\_critic\_for\_reviews, imdb\_score) and then updates the names of the top 20 movies in a Google sheet. Not completed
9. Host your application to any PaaS, like OpenShift, Heroku, Google AppEngine etc

### DELIVERABLES

Create a zip file of the following, and email them to the team (addresses provided below)

1. An export of the database files:

**Location:**

case\_study\_hollywood/data

**Details:**

contains the same database where 2 new tables are added as below:

1. **movies\_cleaned**
2. **audit\_movies\_cleaned**
3. All code / script(s), with comments: given

**Python job code:**

**case\_study\_hollywood-> main**

**Sql Script(s):**

**case\_study\_hollywood-> sql scripts**

**Shell Script(s):**

**case\_study\_hollywood->shell\_scripts**

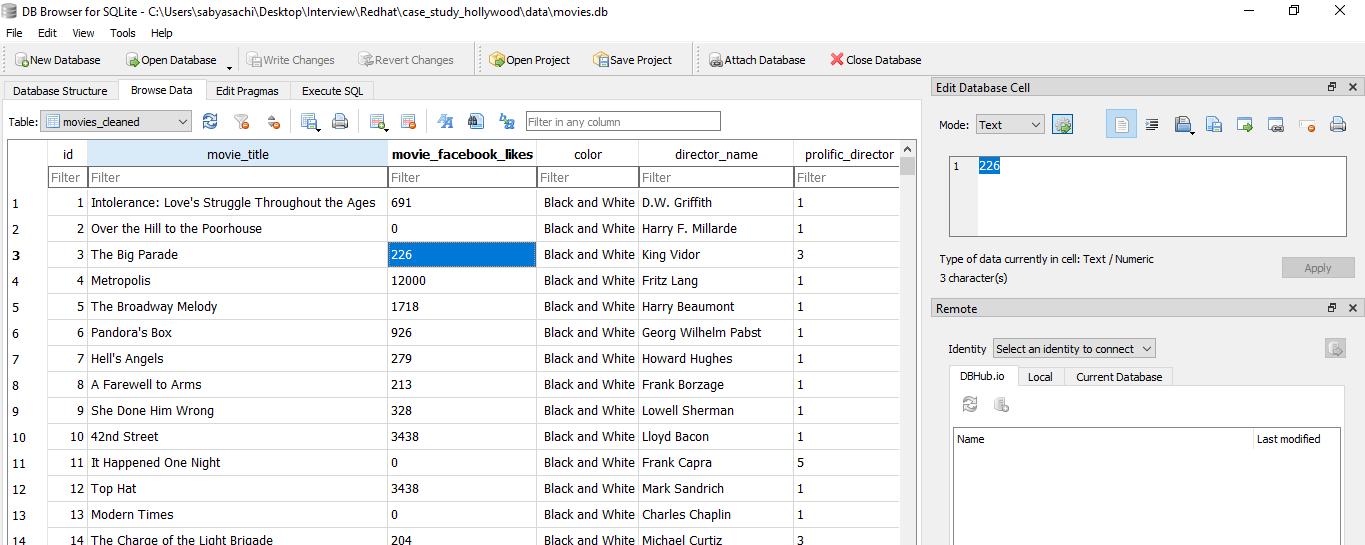
**cron tab can be used scheduling the shell script, on required interval**

1. A text file containing any SQL queries you used (if not present in the code):

Sql queries for transaction are in the code itself.

Also, they are present as standalone in the below path:

**case\_study\_hollywood-> sql scripts(can be used from tool as below):**



1. Link to the hosted web application. – **not Completed.**
2. Also, attached the jupyter notebook, used to perform the basic EDA in the below path, just in case its required for a check:

case\_study\_hollywood -> bin

**Areas of Optimisation of the code:**

* + - 1. MultiProcessing can be used for two parts: a. API data pull part and b. insert into DB part, if resources are available
      2. Further Indexing can be done on the tables, if DB is MySQL or Oracle or SQL Server, for better query performance.
      3. Data can be put in flat files after ingestion, to perform EDA and reporting-> from there- one set will go to DB and other will go for reports. Again, it requires sufficient space in the server, if data is large.
      4. Unittesting is not implemented in this code, which could be done.