We, all three project partners are big fans of movies, and we often had discussions about the Oscars winners. So for this project we decided to create a small database about the movies which won an Oscar since the year 1960. Along with the name of the movie, we tabulated, the director of the movie, production studio and the IMDB rating as well.

**Extract:**

Data for this ETL project was extracted from two sources.

* + <https://www.filmsite.org/bestpics4.html> - HTML
  + <http://www.omdbapi.com/> - JSON

**Transform:**

Data scraped from [www.filmsite.org](http://www.filmsite.org) website and obtained in the table format.

(See Schema below)

{“Film Year”: “int”  
, “Best Picture Academy Award Winners”: “string”  
, “Director (\* Did not also win Best Director)”: “string”  
, “Title Screen and Studio or Production Company”: “string”  
, “NOT Nominated for Best Picture”: “string”  
, “Should Have Won”: “string”  
}

From all the tables that could have been scrape, we selected the one that has useful information for the project: Year that movie was released, movie that won the Oscar for Best picture, movies’ director name and so on. The selected table then was converted into a DataFrame. The next step was to eliminate the unwanted columns and rows from the DataFrame, and rename the columns using more appropriate/easier names. Once the “Movies” table was finalized, all the names of the movies were captured in a list, which we used to perform a “for loop” to make JSON calls to the www.omdbapi.com website (our second source of data).

[{'Title': “string”,  
 'Year': int,  
 'Rated': “string”,  
 'Released': “string”,  
 'Runtime': 'string',  
 'Genre': 'string',  
 'Director': 'string',  
 'Writer': 'string',  
 'Actors': 'string',  
 'Plot': 'string',  
 'Language': 'string',  
 'Country': 'string',  
 'Awards': 'string',  
 'Poster': 'string',  
 'Ratings': [{'Source': 'string', 'Value': “float”},  
  {'Source': “string”, 'Value': 'string'},  
  {'Source': 'string', 'Value': 'string'}],

 'Metascore': 'int',  
 'imdbRating': 'float',  
 'imdbVotes': 'int',  
 'imdbID': 'string',  
 'Type': 'string',  
 'DVD': 'string',  
 'BoxOffice': 'string',  
 'Production': 'string',  
 'Website': 'string',  
 'Response': “boolean”},

From the JSON call, we wanted the movie name and IMDB rating, which we appended in new separate lists (movies and ratings) to create our second table.

**Load:**

Once, the appropriate data was fetched, cleaned and properly tabulated as DataFrames, the data was then transferred to the Postgres SQL. To transfer data to Postgres SQL from jupyter notebook, we created two empty tables in Postgres (movies and ratings). Meanwhile, we created an engine in jupyter notebook to communicate our information to SQL. The next step was to run the queries in jupyter to transfer our DataFrames to the appropriate tables in the SQL database. At last, we run an SQL query in jupyter to merge both tables.