**MOVIE IMDB Database**

**Extract Transform Load Project**

**Team 5:**

**Amy Guan**

**Beth Emborsky**

**Shanker Nair**

August 1, 2020

**Table of Contents**

[1. Introduction 4](#_Toc47224672)

[2. Data Source 4](#_Toc47224673)

[2.1 Movies CSV File 4](#_Toc47224674)

[2.2 Names CSV File 5](#_Toc47224675)

[3. Methodology 5](#_Toc47224676)

[3.1 Extraction 6](#_Toc47224677)

[3.2 Transformation 6](#_Toc47224678)

[3.2.1. ERD Diagram 6](#_Toc47224679)

[3.2.2. Data Selection 7](#_Toc47224680)

[3.2.3. Data Aggregation 7](#_Toc47224681)

[4. Load 9](#_Toc47224682)

**List of Figures**

[**Figure 1: Kaggle IMDB Movies Data** 4](#_Toc47225108)

[Figure 2: IMDb movies.csv Snapshot 4](#_Toc47225109)

[Figure 3: IMDb names.csv Snapshot 5](#_Toc47225110)

[Figure 4: Flow Chart of ETL Methodology 5](#_Toc47225111)

[Figure 5: ERD Diagram 6](#_Toc47225112)

[Figure 6: ERD with 6 Supporting Tables Highlighted by Red Ellipses 8](#_Toc47225113)

[Figure 7: ERD with 4 Junction Tables Highlighted by Green Ellipses 9](#_Toc47225114)

**List of Tables**

[**Table 1: List of Parameters Selection** 7](#_Toc47224719)

[**Table 2: List of Primary Keys Created using Python** 8](#_Toc47224720)

# Introduction

The goal of this ETL project is to extract, transform and load at least 2 datasets. The large movie IMDB datasets were best converted into a relational database using SQL. This report is organized as follows: Section 2 contains the data source; Methodologies including data extraction and transforming are detailed in Section 3; Section 4 summarizes the loading of the final relational database.

# Data Source

Two extensive IMDB dataset from Kaggle were selected for this project:

* One is a movie-oriented CSV file *IMDb movies.csv* with 81k+ records;
* The other is a cast-member-oriented file *IMDb names.csv* with 175k+ records.



**Figure 1:** [**Kaggle IMDB Movies Data**](https://www.kaggle.com/stefanoleone992/imdb-extensive-dataset)

The two files are detailed in Section 2.1 and 2.2 below.

# Movies CSV File

The *IMDb movies.csv* file includes 81,273 movies with attributes such as title, year, genre, duration, country, language, director, production\_company, description, avg\_vote, votes, genre, etc.

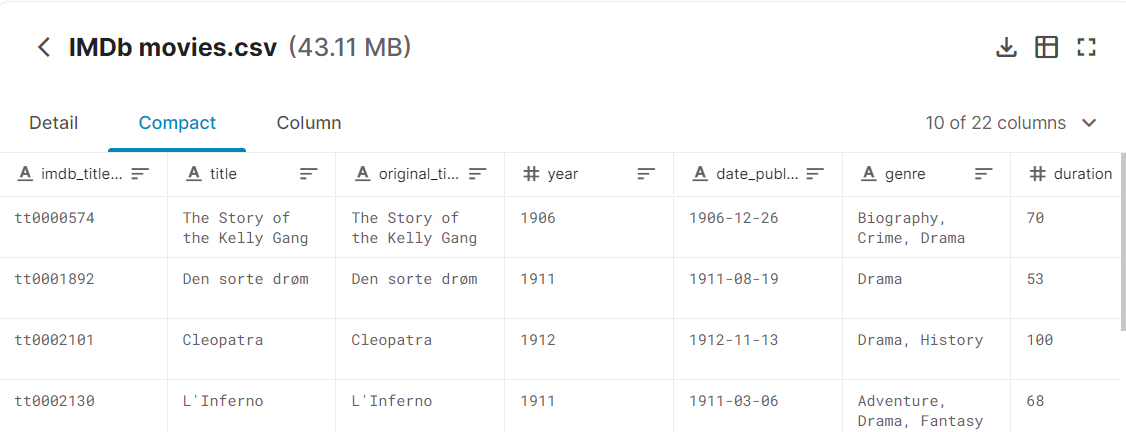


Figure 2: IMDb movies.csv Snapshot

# Names CSV File

The *IMDb names.csv* file dataset includes 175,719 members with personal attributes such as name, height, birth\_year, spouses, children, etc.

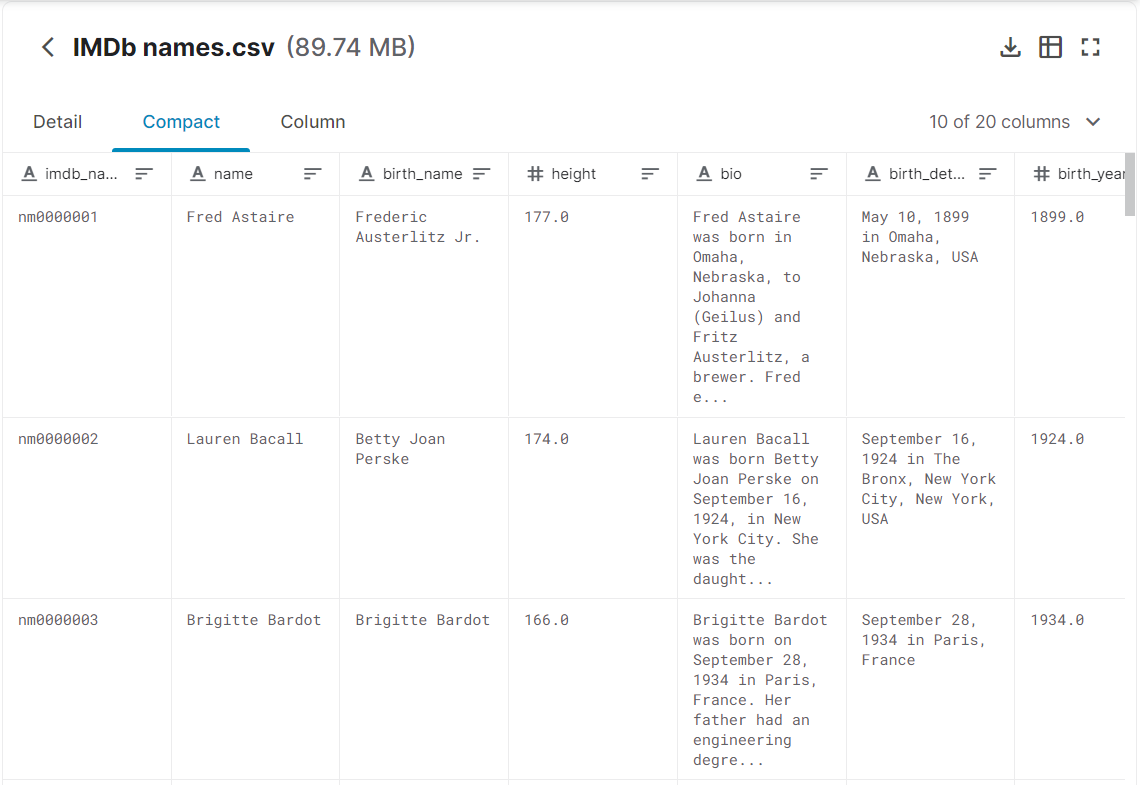


Figure 3: IMDb names.csv Snapshot

# Methodology

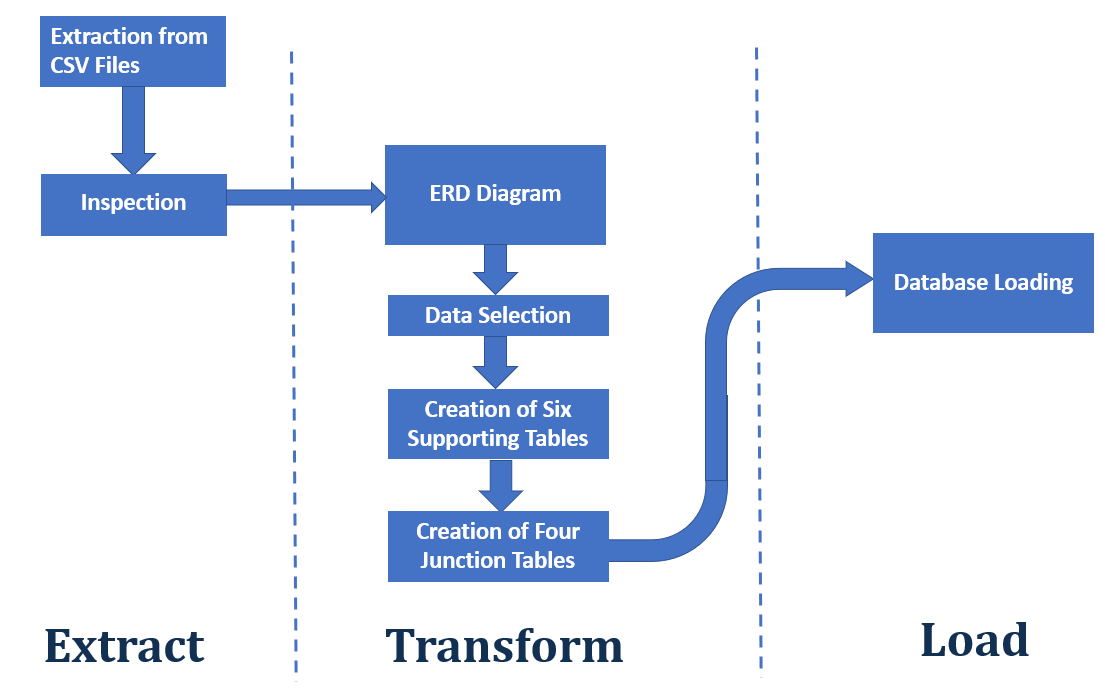


Figure 4: Flow Chart of ETL Methodology

# Extraction

After the CSV files were downloaded from Kaggle, Python was used to convert the CSVs into Pandas DataFrames. Initial inspection was conducted to find out the number of null and valid elements for all the columns. Please refer to the ***EXTRACT*** Section of the main script ‘Main\_ETL\_project2.ipynb’.

# Transformation

# ERD Diagram

The following ERD diagram shows the collection of tables, their relationship, the information included in each table, as well as the primary keys and foreign keys. It provides the overview of the framework of the final database.

All the transformation steps are detailed in Sections 3.2.2 and 3.2.3.

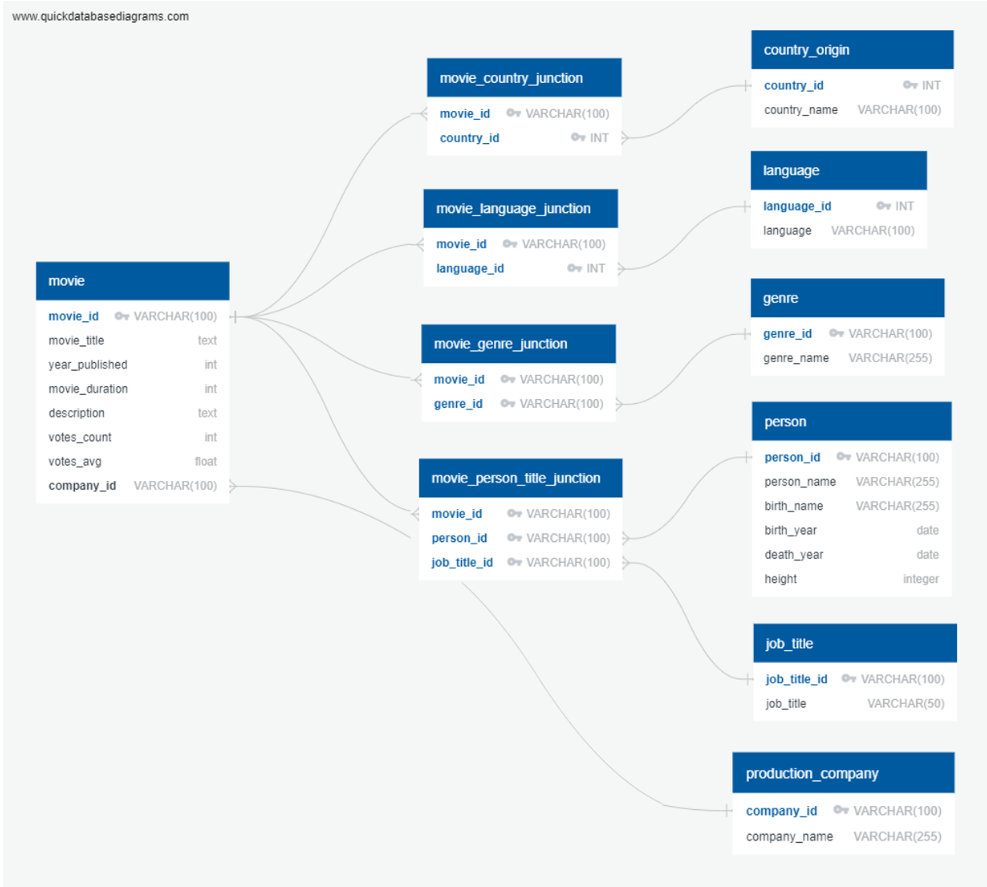


Figure 5: ERD Diagram

# Data Selection

After the CSV files were downloaded, the information of interest was selected using Python. Please refer to the main script ‘Main\_ETL\_project2.ipynb’ for details.

**Table 1** below lists the parameters extracted, as well as their corresponding source CSV files.

**Table 1: List of Parameters Selection**

|  |  |
| --- | --- |
| **Source File** | **Parameters Selected** |
| IMDB movies.csv (81K+ record) | movie\_id |
| movie\_title |
| year\_published |
| movie\_duration |
| description |
| votes\_count |
| votes\_avg |
| **company** |
| **country** |
| **language** |
| **genre** |
| IMDB names.csv (175k+ record) | movie\_id |
| **job\_title** |
| **person\_name** |
| birth\_name |
| birth\_year |
| death\_year |
| height |

# Data Aggregation

After data inspection, it was discovered that special attention had to be paid to those parameters with many-to-many relationships, including country, language, genre, person name and job title.

The data separation process included two main steps - creation of primary keys for five of six supporting tables, and creation of four junction tables, as detailed below.

Each team member was responsible for creating two of the six supporting tables and their corresponding junction tables.

***Creation of Primary Keys for Five Supporting Tables***

In order to facilitate the relational dataset establishment of the five supporting tables using SQL, as highlighted by a red ellipse in Figure 6, the following unique primary keys were created:

**Table 2: List of Primary Keys Created using Python**

|  |  |
| --- | --- |
| **Tables** | **Created Primary Keys (absent from the CSV files)** |
| production\_company | company\_id |
| job\_title | job\_title\_id |
| genre | genre\_id |
| language | language\_id |
| country\_origin | country\_id |

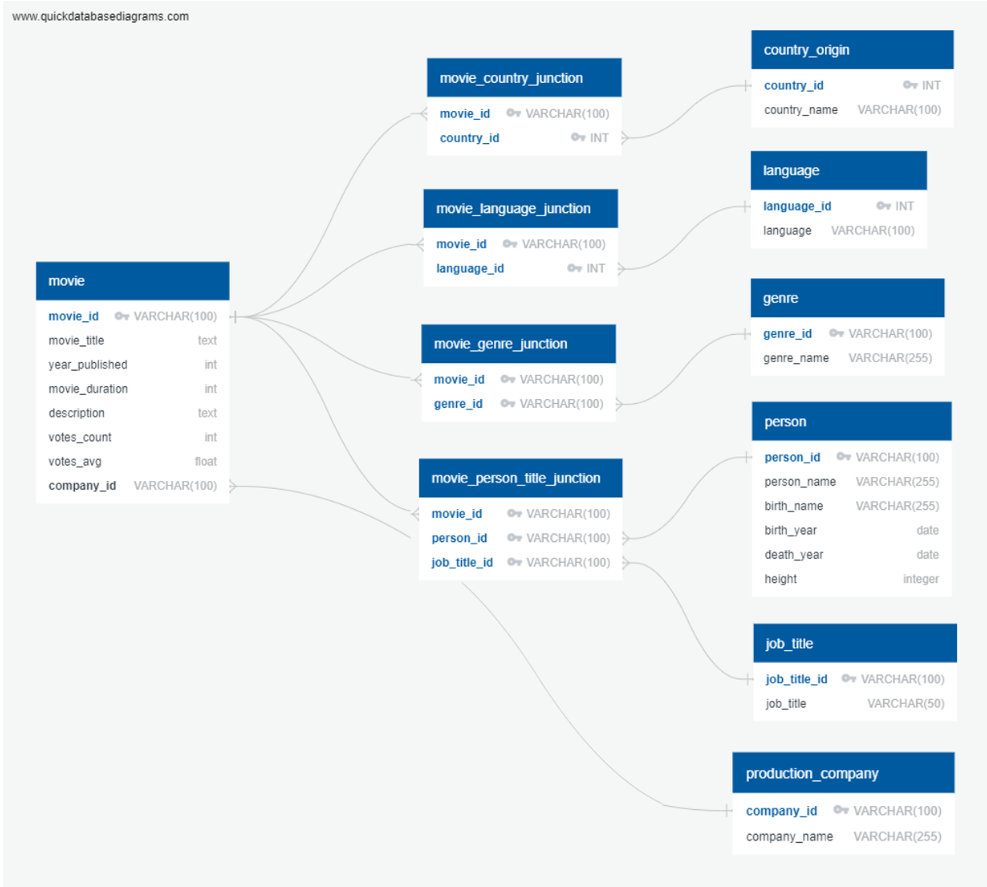


Figure 6: ERD with 6 Supporting Tables Highlighted by Red Ellipses

***Creation of Four Relational Junction Tables***

The four junction tables were essential to connect the main movie table and the supporting tables. Python was used to create the junction tables to ensure the bridging between the main movie table and the five supporting tables with the information of the country, language, genre, cast member and job title.

*The production\_company table had a many-to-one relationship and therefore did not need a junction table*

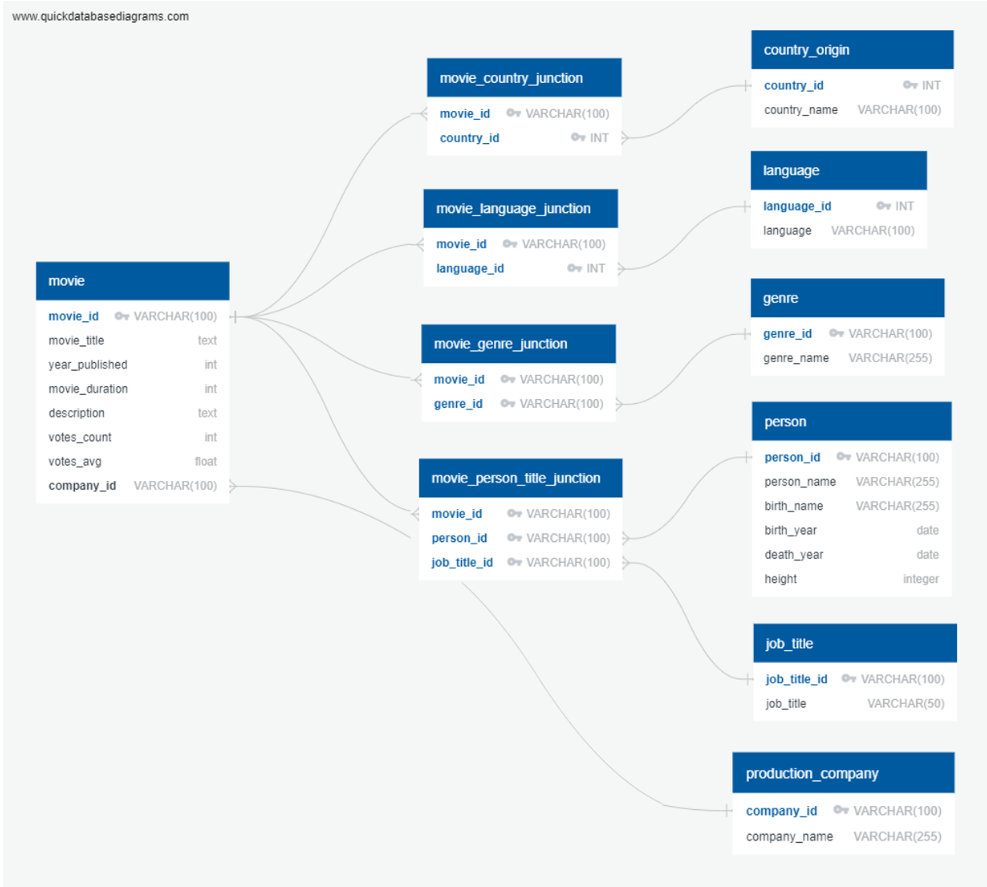


Figure 7: ERD with 4 Junction Tables Highlighted by Green Ellipses

# Load

Once the eleven Pandas DataFrames were established with necessary information, SQL was used to load all tables into a SQL database ‘movies\_db’.