**Introduction:**

* The purpose of this project's database is to store data on movies so that we can extract basic insights from data within a web application setting. This data will be used to show trends across movies such as the amount of shared genres between movies, how many movies a director has made, and the number of movies that were released within a certain time span, etc.

**Functional Requirements:**

* For this project, the database should be designed in such a way that it makes it easy to retrieve specific information from the source data and display it in an easier to read format.
* Data Entry:
  + The data for this project comes from Kaggle and is from the [IMDB Top 250 Movies Dataset (kaggle.com)](https://www.kaggle.com/datasets/rajugc/imdb-top-250-movies-dataset). The data is in a csv file and will be imported into the SQLite database.
  + All data came from kaggle
  + Example of Data Entry in SQLite:
    - Command FileLocation TableName
    - .import C:\Sqlite-Proj\Import-CSV\importFile.csv students
* Data Storage:
  + Most of the data found in our data set will be stored as TEXT, with a few other areas of data being stored as INTEGER and REAL
  + The TEXT data will be assigned through the VARCHAR() function, fro elements such as title this can be stored at about 125 characters while other elements such as names or description should be stored around 64 and 255 characters
  + When it comes to the REAL data, this will be data that are decimales such as ratings or mony elements. This can be assigned by using the DECIMAL() function, such as (2, 1) for ratings and (10, 2) for box office results.
  + One prerequisite for our data will be to parse the run time of these movies into a singular time format (minutes) as they are given to us in hours and minutes, once they are parsed as minutes they can be stored as INTEGERS.
* Retrieval:
  + This project will be using APIs programing in python in order to retrieve the data within this database and display it in a web application
  + Having separate API calls for specific search or filter conditions can help in displaying the desired data.

**Non-functional Requirements:**

* Response Time:
  + Having our repeating data such as movie genres stored in separate tables, while being connected to our main table through foreign keys, can help improve database performance
    - This can be established with the age rating data and possibly the genre and cast data
* Security:
  + Using API calls will establish a surface level security, by not having any query call directly in the Web App UI the user will not be able to tamper with the responses
  + Setting up permissions so that only a certain user or group can safeguard us against any unwanted users; This can function within a python script allowing an easy security level for our project
    - The details can be found here <https://dev.to/stephenc222/basic-security-practices-for-sqlite-safeguarding-your-data-23lh>
* Scalability:
  + One of the limitations of SQLite is its scalability, it can store up to 100GB of data.
  + However, our project plans to use a limited amount of data in order to display our concept.

**Data Dictionary:**

* Name VARCHAR(125)
* Year INTEGER
* Rating DECIMAL(2, 1)
* Rank INTEGER
* Genre TEXT
* Age\_Rating VARCHAR(32)
* Run\_time INTEGER
* tagline VARCHAR(255)
* Release\_Date DATE
* Budget DECIMAL(10, 2)
* Box\_office DECIMAL(10, 2)
* Casts TEXT
* Directors TEXT
* Writers TEXT
* Possible Foreign Key Elements:
  + Move\_id INTEGER
  + Age\_Rating\_id INTEGER
  + Director\_id INTEGER
  + Genre\_id INTEGER
  + Cast\_id INTEGER