DELHI TECHNOLOGICAL UNIVERSITY

DATABASE MANAGEMENT SYSTEM

LAB PROJECT

INTERNET MOVIE DATABASE

SUBMITTED TO:

Mr. MANOJ SETHI

SUBMITTED BY:

VAIBHAV VASHISHT

2K19/CO/420

ACKNOWLEDGEMENT

I would like to express a deep sense of thanks and gratitude to our teacher Mr. Manoj Sethi for giving me the golden opportunity to do this project on the topic “IMDb Database” and guiding me immensely through the course of the project. His constructive advice and constant motivation have been responsible for the successful completion of the project.

I would also like to thank Mr. Yogesh Singh, our Vice-Chancellor, for his co-ordination in extending every possible support for completion of this project.

Last but not the least, I would like to thank all those who had helped me directly or indirectly towards the completion of the project.

CERTIFICATE

This is to certify that Vaibhav Vashisht (2K19/CO/420) has successfully completed the Database Management Lab Project entitled “IMDb database”. The project report is a result of his efforts and endeavors. The project is found worthy of acceptance as final project for Database Management Systems Lab. The project has been prepared under my knowledge consistently.

Mr. Manoj Sethi

(DBMS TEACHER)

CONTENTS

INTRODUCTION

Every year there are hundreds and thousands of movies released in many different genres. Thus, there is a vast amount of movie data available on the internet. IMDb is the most popular website that maintains movie dataset. In this project I will build a MySQL database using the Internet Movie Database (IMDb) dataset. The dataset consists of 7 compressed tab-separated-value (*.tsv) files that can be downloaded from the IMDb website.

Dataset Used

Each dataset is contained in a gzipped tab-separated-values (TSV) formatted file in the UTF-8-character set. The first line in each file contains headers that describe what is in each column. A "\N" is used to denote that a particular field is missing or has a NULL value for that title or name. It should be noted that the data available for download from the IMDb website is not the full dataset, but it will suffice for our purposes. The available IMDb data files are as follows:

name.basics.tsv.gz

Contains the following information for names:

- nconst (string) - alphanumeric unique identifier of the name/person.
- primaryName (string)– name by which the person is most often credited.
- birthYear – in YYYY format.
- deathYear – in YYYY format if applicable, else "\N".
- primaryProfession (array of strings) – the top-3 professions of the person.
- knownForTitles (array of tconsts) – titles the person is known for.

title.basics.tsv.gz

Contains the following information for titles:

- tconst (string) - alphanumeric unique identifier of the title.
- titleType (string) – the type/format of the title (e.g. movie, short, tvseries, tvepisode, video, etc).

• primaryTitle (string) – the more popular title / the title used by the filmmakers on promotional materials at the point of release.
• originalTitle (string) - original title, in the original language.
• isAdult (boolean) - 0: non-adult title; 1: adult title.
• startYear (YYYY) – represents the release year of a title. In the case of TV Series, it is the series start year.
• endYear (YYYY) – TV Series end year. "\N" for all other title types.
• runtimeMinutes – primary runtime of the title, in minutes.
• genres (string array) – includes up to three genres associated with the title.

title.akas.tsv.gz
Contains the following information for titles:
• titleId (string) - a tconst which is an alphanumeric unique identifier of the title.
• ordering (integer) – a number to uniquely identify rows for a given titleId.
• title (string) – the localised title.
• region (string) - the region for this version of the title.
• language (string) - the language of the title.
• types (array) - Enumerated set of attributes for this alternative title. One or more of the following: "alternative", "dvd",
"festival", "tv", "video", "working", "original", "imdbDisplay". New values may be added in the future without warning.
• attributes (array) - Additional terms to describe this alternative title, not enumerated.
• isOriginalTitle (boolean) – 0: not original title; 1: original title.

title.crew.tsv.gz
Contains the director and writer information for all the titles in IMDb. Fields include:
• tconst (string) - alphanumeric unique identifier of the title.
• directors (array of nconsts) - director(s) of the given title.
• writers (array of nconsts) – writer(s) of the given title.

title.episode.tsv.gz
Contains the tv episode information. Fields include:
• tconst (string) - alphanumeric identifier of episode.
• parentTconst (string) - alphanumeric identifier of the parent TV Series.
• seasonNumber (integer) – season number the episode belongs to.

• episodeNumber (integer) – episode number of the tconst in the TV series.

title.principals.tsv.gz

Contains the principal cast/crew for titles

• tconst (string) - alphanumeric unique identifier of the title.
• ordering (integer) – a number to uniquely identify rows for a given titleId.
• nconst (string) - alphanumeric unique identifier of the name/person.
• category (string) - the category of job that person was in.
• job (string) - the specific job title if applicable, else "\N".
• characters (string) - the name of the character played if applicable, else "\N"

title.ratings.tsv.gz

Contains the IMDb rating and votes information for titles

• tconst (string) - alphanumeric unique identifier of the title.
• averageRating – weighted average of all the individual user ratings.
• numVotes - number of votes the title has received.

PROPOSED WORK

In this project work we aim to create an IMDb database which stores the data from the IMDb website. In this project we first understand the data in the IMDb dataset and then design a relational database for the data. This includes creating an ER diagram and schema for the database, create the database in MySQL, load the data into the database and then add primary and foreign key constraints. The objectives of the project are:

- Understand the data in the IMDb dataset.
- Design a relational database and store the IMDb data in it.
    - Model the database using an Entity-Relationship (ER) diagram.
    - Perform normalisation and restructure the IMDb data.
    - Create the schema for the database
    - Create MySQL database.
    - Load data into the database.
    - Add primary and foreign key constraints.
- Ask questions of the IMDb data, so as to practice simple and more advanced SQL queries.

IMPLEMENTATION

ER DIAGRAM

ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.

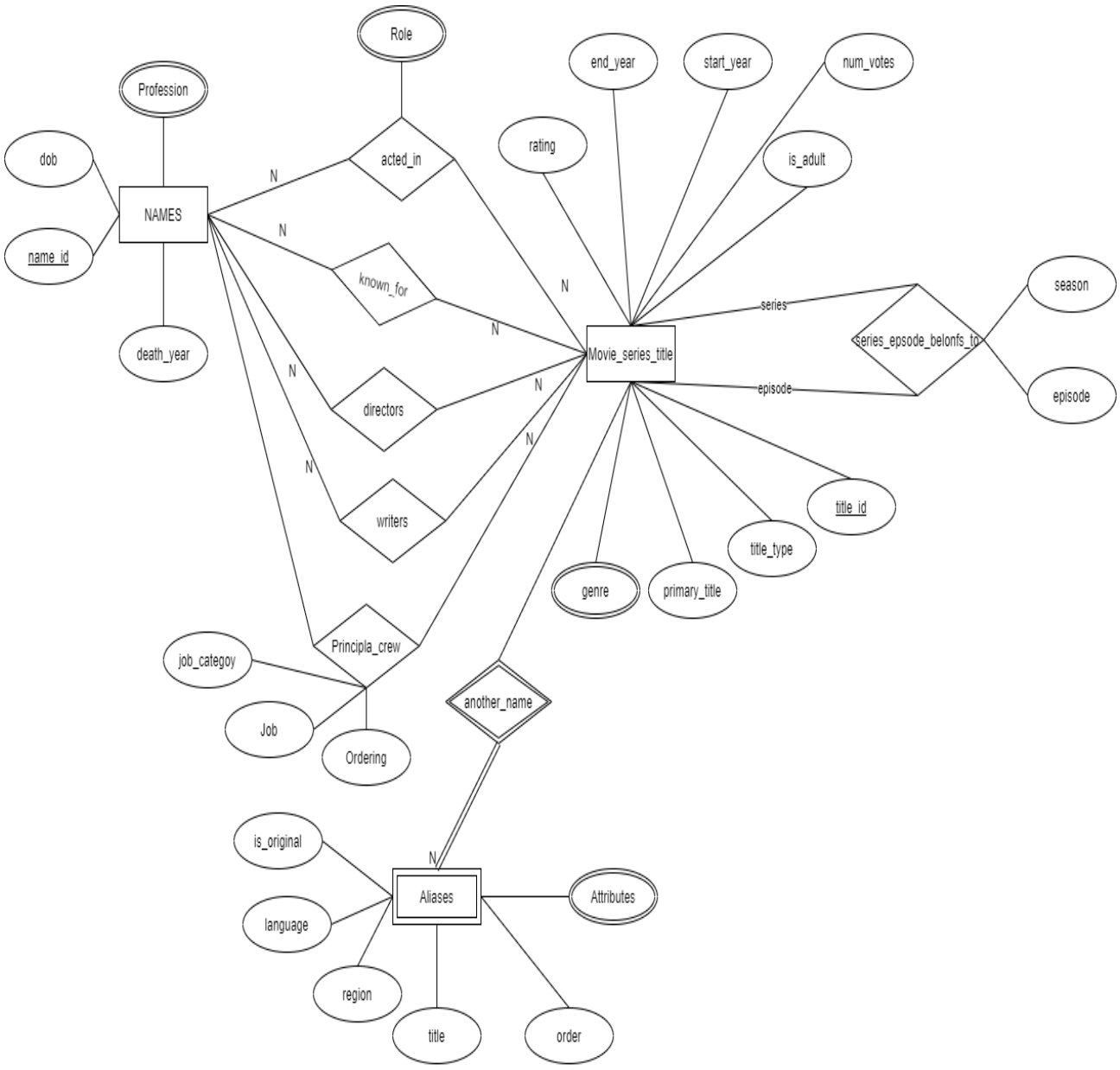
ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

The ER diagram of my database consists of three entities namely:

- Names
- Movie_series_title
- Aliases

Aliases is a weak entity and has a total participation constraint in its many to one identifying relationship another_name with the movie_series_title entity. Names and Movie_series_titles entities have several many to many relationships between them namely acted_in, known_for, directors, writers and principal crew.

Movie_series_title has a recursive many to one relationship series_epsiode_belongs_to with itself.


Role
Profession
dob
NAMES
name_id
death_year
acted_in
known_for
directors
writers
N
N
N
N
N
N
N
rating
end_year
start_year
num_votes
is_adult
Movie_series_title
series
episode
series_epsode_belonfs_to
season
episode
title_id
title_type
primary_title
genre
Principla_crew
job_categoy
Job
Ordering
another_name
N
is_original
language
region
Aliases
Attributes
title
order

NORMALISATION

Normalization is the process of organizing the data in the database. Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies. Normalization divides the larger table into the smaller table and links them using relationship. The normal form is used to reduce redundancy from the database table.

The ER diagram is not in 1 NF form due to the presence of multivalued attributes such as genre in movie_series_title, profession in names entity, attributes in aliases entity and role in acted_in relationship. These were decomposed to form separate relations as is shown in the schema.

After bringing it to 1NF form, the conditions for 2NF and 3NF are checked.

Following the second normal form, every non-prime attribute should be fully functionally dependent on prime key attribute. That is, if $X \rightarrow A$ holds, then there should not be any proper subset Y of X, for which $Y \rightarrow A$ also holds true. This means there should be no partial dependency. This condition is satisfied by all the relations and hence they are in second normal form.

For a relation to be in Third Normal Form, it must be in Second Normal form and no non-prime attribute must be transitively dependent on prime key attribute and for any non-trivial functional dependency, $X \rightarrow A$, then either −

- X is a super key or,
- A is prime attribute.

This condition is also satisfied by all the relations, that means, they are in the third normal form.

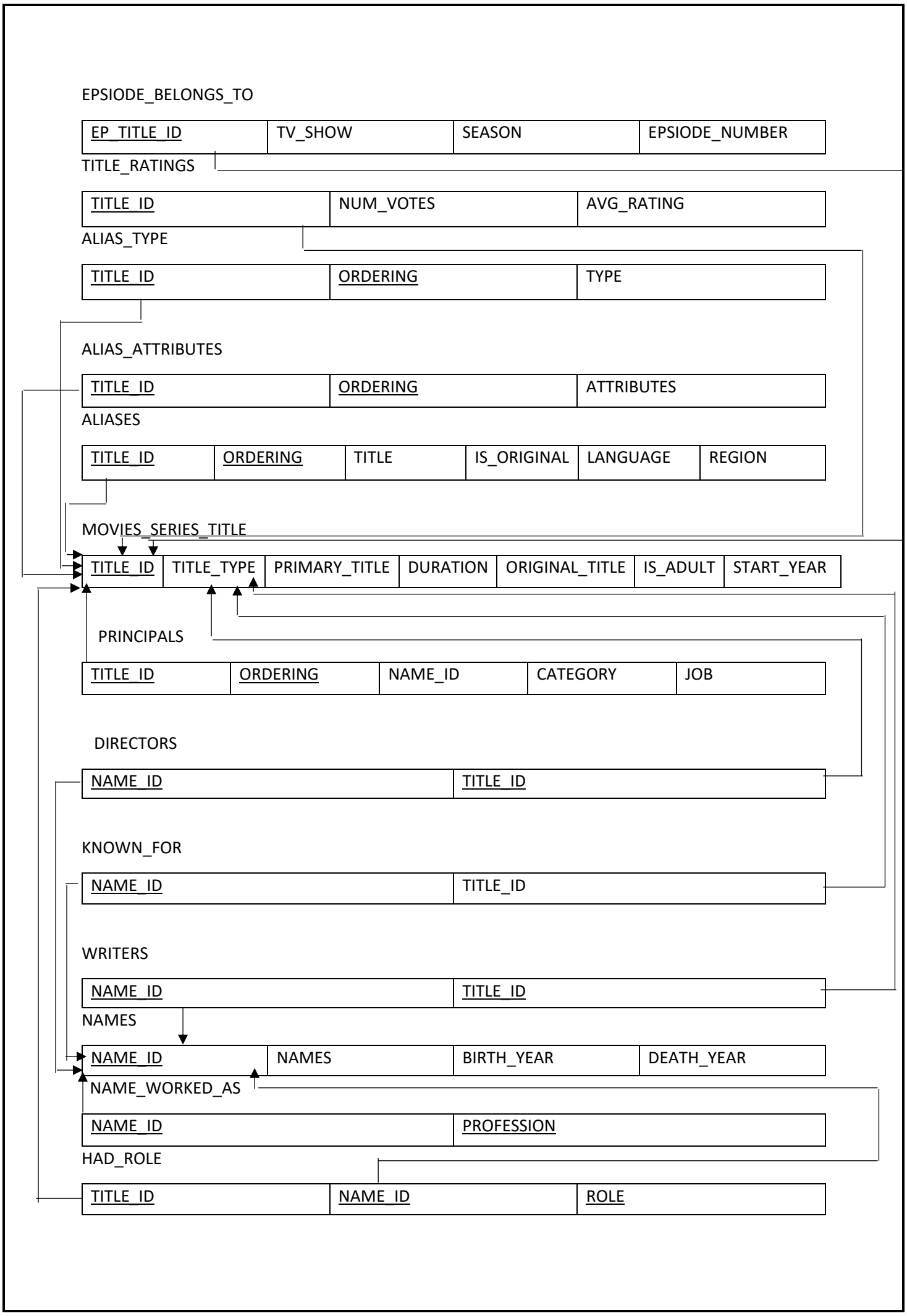
SCHEMA

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams

The logical schema of my database consists of 13 tables or relations:

- Movies_series_Titles
- Aliases
- Alias_attributes
- Alias_types
- Title_ratings
- Title_genre
- Names_
- Directors
- Writers
- Principals
- Episode_belongs_to
- Had_role
- Known_for


EPSIODE_BELONGS_TO
EP_TITLE_ID
TV_SHOW
SEASON
EPSIODE_NUMBER
TITLE_RATINGS
TITLE_ID
NUM_VOTES
AVG_RATING
ALIAS_TYPE
TITLE_ID
ORDERING
TYPE
ALIAS_ATTRIBUTES
TITLE_ID
ORDERING
ATTRIBUTES
ALIASES
TITLE_ID
ORDERING
TITLE
IS_ORIGINAL
LANGUAGE
REGION
MOVIES_SERIES_TITLE
TITLE_ID
TITLE_TYPE
PRIMARY_TITLE
DURATION
ORIGINAL_TITLE
IS_ADULT
START_YEAR
PRINCIPALS
TITLE_ID
ORDERING
NAME_ID
CATEGORY
JOB
DIRECTORS
NAME_ID
TITLE_ID
KNOWN_FOR
NAME_ID
TITLE_ID
WRITERS
NAME_ID
TITLE_ID
NAMES
NAME_ID
NAMES
BIRTH_YEAR
DEATH_YEAR
NAME_WORKED_AS
NAME_ID
PROFESSION
HAD_ROLE
TITLE_ID
NAME_ID
ROLE

PRE-PROCESSING DATA

The scheme created for the databases consists of 13 relations or tables. But the data collected from the IMDb website is in the form of 7 tsv files. These 7 files were divided into 13 files to be imported to the database. Also, the format of the files was converted from .tsv to .csv to facilitate the easy import of data. Python scripts were written for this data pre-processing.

```
def make_Aliases(title_akas):

    Aliases = pd.DataFrame()
    Aliases[['title_id','ordering','title','region','language',
    'is_original_title']] = title_akas[['titleId','ordering','title','region','language',
    'isOriginalTitle']]
    Aliases.to_csv('Aliases.csv',index=False)

def make_Alias_types(title_akas):

    Alias_types = pd.DataFrame()
    Alias_types[['title_id','ordering','type']] = title_akas[['titleId','ordering','types']]
    Alias_types = Alias_types.dropna()
    Alias_types.to_csv('Alias_types.csv',index=False)


def make_Alias_attributes(title_akas):

    Alias_attributes = pd.DataFrame()
    Alias_attributes[['title_id','ordering','attribute']] =  title_akas[['titleId','ordering','attributes']]
    Alias_attributes = Alias_attributes.dropna()
    Alias_attributes.to_csv('Alias_attributes.csv',index=False)


def make_Directors_and_Writers(title_crew):

    Directors = pd.DataFrame()
    Directors[['title_id','name_id']] = title_crew[['tconst','directors']]
    Writers[['title_id','name_id']] = title_crew[['tconst','writers']]

    Directors = Directors.dropna()
    Writers = Writers.dropna()

    Directors = Directors.assign(name_id=Directors.name_id.str.split(',')).explode('name_id').reset_index(drop=True)
    Writers = Writers.assign(name_id=Writers.name_id.str.split(',')).explode('name_id').reset_index(drop=True)

    Directors.to_csv('Directors.csv',index=False)
    Writers.to_csv('Writers.csv',index=False)
```

```
def make_Episode_belongs_to(title_episode):
    Episode_belongs_to = title_episode.rename(columns={
    'tconst':'title_id',
    'parentTconst':'parent_tv_show_title_id',
    'seasonNumber':'season_number',
    'episodeNumber':'episode_number'
    })

    Episode_belongs_to.to_csv('Episode_belongs_to.csv',index=False)


def make_Names_(name_basics):
    Names_[['name_id','name_','birth_year','death_year']] = name_basics[['nconst','primaryName','birthYear','deathYear']]
    Names_.to_csv('Names_.csv',index=False)

def make_Name_worked_as(name_basics):
    Name_worked_as[['name_id','profession']] = name_basics[['nconst','primaryProfession']]
    Name_worked_as = Name_worked_as.dropna()
    Name_worked_as = Name_worked_as.assign(profession=Name_worked_as.profession.str.split(',')).explode('profession').reset_i
    Name_worked_as.to_csv('Name_worked_as.csv',index=False)


def make_Known_for(name_basics):


    Known_for[['name_id','title_id']] = name_basics[['nconst','knownForTitles']]
    Known_for = Known_for.dropna()
    Known_for = Known_for.assign(title_id=Known_for.title_id.str.split(',')).explode('title_id').reset_index(drop=True)
    Known_for.to_csv('Known_for.csv',index=False)


def make_Principals(title_principals):
    Principals[['title_id','ordering','name_id','job_category','job']] = title_principals[['tconst','ordering','nconst','cate
    Principals.to_csv('Principals.csv',index=False)

def make_Had_role(title_principals):
    Had_role[['title_id','name_id','role_']] = title_principals[['tconst','nconst','characters']]
    Had_role = Had_role.dropna()
```

```
def make_Had_role(title_principals):
    Had_role[['title_id','name_id','role_']] = title_principals[['tconst','nconst','characters']]
    Had_role = Had_role.dropna()

    Had_role['role_'] = Had_role['role_'].str.replace('[\"\[\]]','',regex=True)
    Had_role['role_'] = Had_role['role_'].str.replace('\\','|')
    Had_role = Had_role.assign(role_=Had_role.role_.str.split(',')).explode('role_').reset_index(drop=True)
    Had_role['role_'] = Had_role['role_'].str.title()
    Had_role['role_'] = Had_role['role_'].str.replace('^ | $','',regex=True)
    Had_role.drop_duplicates(keep=False,inplace=True)
    Had_role.to_csv('Had_role.csv',index=False)


def make_Titles(title_basics):
    Titles[['title_id','title_type','primary_title',
    'original_title', 'is_adult', 'start_year', 'end_year', 'runtime_minutes']] = title_basics[['tconst','titleType','primary
    'originalTitle', 'isAdult', 'startYear', 'endYear', 'runtimeMinutes']]
    Titles.to_csv('Titles.csv',index=False)


def make_Title_genres(title_basics):

    Title_genres[['title_id','genre']] = title_basics[['tconst','genres']]

    Title_genres = Title_genres.dropna()

    Title_genres = Title_genres.assign(genre=Title_genres.genre.str.split(',')).explode('genre').reset_index(drop=True)

    Title_genres.to_csv('Title_genres.csv',index=False)
```

CREATING THE DATABASE

After creating the ER diagram and logical schema, the database was created in MySQL using SQL. Structured Query Language (SQL) as we all know is the database language by the use of which we can perform certain operations on the existing database and also, we can use this language to create a database .SQL can be used as a Data Definition Language which includes using the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database.

We create the database using the CREATE DATABASE command

```
CREATE DATABASE imdb;
use imdb;
```

The tables were created using the create table SQL command

```
CREATE TABLE Titles (
  title_id  VARCHAR(100),
  title_type VARCHAR(50),
  primary_title TEXT,
  original_title TEXT,
  is_adult  BOOLEAN,
  start_year INTEGER,
  end_year  INTEGER,
  runtime_minutes INTEGER,
  Primary Key(title_id)
);

CREATE TABLE Title_ratings (
  title_id   VARCHAR(100) ,
  average_rating FLOAT,
  num_votes INTEGER,
  Primary Key(title_id)
);

CREATE TABLE Aliases (
  title_id   VARCHAR(100) ,
  ordering  INTEGER,
  title   TEXT,
  region CHAR(4),
  language  CHAR(4),
  is_original_title BOOLEAN,
  Primary Key(title_id,ordering)
);

CREATE TABLE alias_types (
  title_id   VARCHAR(100) ,
  ordering  INTEGER,
  type  VARCHAR(255),
  Primary Key(title_id,ordering)
);

CREATE TABLE alias_attributes (
  title_id  VARCHAR(100),
  ordering  INTEGER,
  attribute VARCHAR(255),
  Primary Key(title_id,ordering)
);

CREATE TABLE episode_belongs_to (
  episode_title_id  VARCHAR(100),
  parent_tv_show_title_id   VARCHAR(100),
  season_number  INTEGER,
  episode_number INTEGER,
  Primary Key(episode_title_id)
);

CREATE TABLE title_genres (
  title_id   VARCHAR(100),
  genre VARCHAR(150),
  Primary Key(title_id)
);
```

```sql
CREATE TABLE names_ (
  name_id VARCHAR(100) ,
  name_   VARCHAR(100) NOT NULL,
  birth_year   INT,
  death_year  INT,
  Primary Key(name_id)
);
```

```sql
CREATE TABLE name_worked_as (
  name_id  VARCHAR(100),
  profession VARCHAR(200),
  Primary Key(name_id,profession)
);
```

```sql
CREATE TABLE had_role (
  title_id VARCHAR(100),
  name_id   VARCHAR(100),
  role_  TEXT,
  Primary key(title_id,name_id,role_)
);
```

```sql
CREATE TABLE known_for (
  name_id   VARCHAR(100),
  title_id  VARCHAR(100),
  Primary Key(name_id,title_id)
);
```

```sql
CREATE TABLE directors (
  title_id VARCHAR(100),
  name_id  VARCHAR(100),
  Primary Key(title_id,name_id)
);
```

```sql
CREATE TABLE writers (
  title_id VARCHAR(100),
  name_id  VARCHAR(100),
  Primary Key(title_id,name_id)
);
```

```sql
CREATE TABLE principals (
  title_id   VARCHAR(100),
  ordering    INT,
  name_id  VARCHAR(100),
  job_category  VARCHAR(255),
  job  TEXT,
  Primary Key(title_id,ordering)
);
```

IMPORTING THE DATA

After creating the database schema in MySQL, the next step is to import the data into the created tables. The data has already been pre-processed and ready to be imported. The files to be imported are of very large size, with some of them having records as much as 2 crores. So, to optimize the import of data, pymysql package of python was used to establish a connection between the python jupyter notebook and the MySQL database. Then a python script was written to import the data of the files into the database in the form of batches, wherein each file was first divided into batches or chunks of a million records each and then exported to the database.

```
import pymysql
mydb = pymysql.connect(
    host='localhost',
    user='root',
    passwd='221928122',
    database='imdb')
```

```
cursor = mydb.cursor()
sql = "select database();"
cursor.execute(sql)
record = cursor.fetchone()
for df in batches(d,10000):
    my_data = []
    for i,row in df.iterrows():
        my_data.append(tuple(row))
    sql = "INSERT INTO imdb.had_role VALUES (%s,%s,%s)"
    cursor.executemany(sql,my_data)
    print("Records inserted")
    mydb.commit()
```

ADDING CONSTRAINTS

All the foreign key constraints were added using the ALTER TABLE SQL command.

```
ALTER TABLE directors
ADD CONSTRAINT FOREIGN KEY (title_id) REFERENCES movies_series_titles(title_id);

ALTER TABLE writers
ADD CONSTRAINT FOREIGN KEY (title_id) REFERENCES movies_series_titles(title_id);


ALTER TABLE episode_belongs_to
ADD CONSTRAINT FOREIGN KEY (episode_title_id) REFERENCES movies_series_titles(title_id);

ALTER TABLE name_worked_as
ADD CONSTRAINT FOREIGN KEY (name_id) REFERENCES Names_(name_id);


ALTER TABLE known_for
ADD CONSTRAINT FOREIGN KEY (name_id) REFERENCES names_(name_id);

ALTER TABLE title_genres
ADD CONSTRAINT FOREIGN KEY (title_id) REFERENCES movies_series_titles(title_id);


ALTER TABLE title_ratings
ADD CONSTRAINT FOREIGN KEY (title_id) REFERENCES movies_series_titles(title_id);


ALTER TABLE aliases
ADD CONSTRAINT FOREIGN KEY (title_id) REFERENCES movies_series_titles(title_id);

ALTER TABLE alias_attributes
ADD CONSTRAINT FOREIGN KEY (title_id) REFERENCES movies_series_titles(title_id);
```

```sql
ALTER TABLE alias_types
ADD CONSTRAINT FOREIGN KEY (title_id) REFERENCES movies_series_titles(title_id);


ALTER TABLE episode_belongs_to
ADD CONSTRAINT FOREIGN KEY (parent_tv_show_title_id) REFERENCES movies_series_titles(title_id);

ALTER TABLE known_for
ADD CONSTRAINT FOREIGN KEY (title_id) REFERENCES movies_series_titles(title_id);

ALTER TABLE principals
ADD CONSTRAINT FOREIGN KEY (name_id) REFERENCES names_(name_id);

ALTER TABLE principals
ADD CONSTRAINT FOREIGN KEY (title_id) REFERENCES movies_series_titles(title_id);

ALTER TABLE had_role
ADD CONSTRAINT FOREIGN KEY (title_id) REFERENCES movies_series_titles(title_id);


ALTER TABLE had_role
ADD CONSTRAINT FOREIGN KEY (name_id) REFERENCES names_(name_id);

ALTER TABLE directors
ADD CONSTRAINT FOREIGN KEY (name_id) REFERENCES names_(name_id);

ALTER TABLE writers
ADD CONSTRAINT FOREIGN KEY (name_id) REFERENCES names_(name_id);
```

RESULT

SQL QUERIES

Displaying the average rating for all the episodes of a series (for example, displaying the average rating of all episode of Brooklyn nine nine)

Query

```
SELECT p.primary_title,t.average_rating,t.num_votes FROM title_ratings t,movies_series_titles p
 WHERE t.title_id IN
 (SELECT episode_title_id FROM episode_belongs_to WHERE parent_tv_show_title_id
 IN( SELECT title_id FROM titles WHERE primary_title ='Brooklyn Nine-Nine'))
 AND p.title_id = t.title_id ORDER BY average_rating;
```

primary_title	average_rating	num_votes
Debbie	7.4	1302
The Slump	7.6	2987
Karen Peralta	7.6	1844
The Tagger	7.6	3193
The Funeral	7.7	2032
Hostage Situation	7.7	1901
The Ebony Falcon	7.7	2411
Fancy Brudgom	7.7	2297
Sal's Pizza	7.7	2572
The Apartment	7.7	2309
M.E. Time	7.7	2889
Pilot	7.8	3897
Captain Peralta	7.8	2036
Lockdown	7.8	2161
AC/DC	7.8	1996
Chocolate Milk	7.8	2279
Defense Rests	7.8	2047
Into the Woods	7.8	1945
The Mattress	7.8	1945
House Mouses	7.8	1797
Cheddar	7.8	1859
Adrian Pimento	7.8	1841
Terry Kitties	7.8	1773
Old School	7.8	2622
Full Boyle	7.9	2386
Boyle's Hunch	7.9	1984
The Swedes	7.9	2000
USPIS	7.9	2157
Stakeout	7.9	2102
Sabotage	7.9	2021
Trying	7.9	1173
The Jimmy Jab G...	8	1218
Admiral Peralta	8	1098
The Vulture	8	2793
Det. Dave Majors	8	2019
The Wednesday ...	8	2051

Payback	8	2090
9 Days	8	2020
The 9-8	8	1866
Pontiac Bandit	8.1	2576
48 Hours	8.1	2746
The Cruise	8.1	1943
Maximum Security	8.1	1794
Jake and Sophia	8.2	2263
Ava	8.2	1979
Paranoia	8.2	1812
The Mole	8.2	2251
The Road Trip	8.2	2219
Boyle-Linetti We...	8.2	2114
Operation: Broke...	8.2	2540
Christmas	8.2	2510
Unsolvable	8.2	2326
Undercover	8.2	2413
The Takeback	8.3	1227
Thanksgiving	8.3	2634
The Pontiac Ban...	8.3	2191
Beach House	8.3	2212
Windbreaker City	8.3	2102
New Captain	8.3	2261
The Jimmy Jab G...	8.3	2369
Captain Kim	8.4	1338
Valloweaster	8.4	1235
Halloween	8.5	3118
The Oolong Slayer	8.5	2128
Tactical Village	8.5	2486
Pimemento	8.6	1411
The Party	8.6	2666
The Chopper	8.6	2119
Bureau	8.7	1913
The Bet	8.8	2921
Halloween II	8.8	2664
Yippie Kayak	8.8	2229

Displaying the number of movies of each genre released in 2021

Query

```
SELECT genre,COUNT(title_id) FROM title_genres WHERE
title_id IN (SELECT title_id FROM movies_series_titles
WHERE start_year = 2020 AND title_type = 'movie') GROUP BY genre;
```

genre	COUNT(title_id)
Drama	3239
Documentary	3662
War	88
Comedy	1553
Romance	544
Biography	229
History	165
Adventure	329
Family	310
Action	728
Sport	178
Mystery	275
Horror	871
Fantasy	253
Thriller	862
Crime	382
Musical	95
Animation	262
Music	227
Sci-Fi	263
Western	39
News	12
Adult	147
Reality-TV	76
Short	1

Displaying the average movie ratings of a director

QUERIES:

```
SELECT AVG(t.average_rating),n.name_ FROM title_ratings t, names_ n, directors d
where t.title_id = d.title_id AND n.name_id = d.name_id AND n.name_ = 'Christopher Nolan';
```

AVG(t.average_rating)	name_
7.96153842485868	Christopher Nolan

```
SELECT AVG(t.average_rating),n.name_ FROM title_ratings t, names_ n, directors d
where t.title_id = d.title_id AND n.name_id = d.name_id AND n.name_ = 'Sanjay Leela Bhansali';
```

AVG(t.average_rating)	name_
7.1099999904632565	Sanjay Leela Bhansali

The top 10 actors who have made the most movies

Query

```
SELECT n.name_,COUNT(DISTINCT h.title_id) FROM
had_role h,names_ n,name_worked_as w WHERE h.name_id = n.name_id AND n.name_id = w.name_id
AND (w.profession = 'actor' OR w.profession = 'actress') GROUP BY h.name_id;
```

name_	COUNT(DISTINCT h.title_id)
Johnny Gilbert	7826
Alex Trebek	7372
Bob Barker	7038
Pat Sajak	6540
Vanna White	6401
Carol Vorderman	5673
Janice Pennington	5531
Johnny Olson	5087
Jay Leno	4899
Richard Whiteley	4577
Susie Dent	4518
Charlie O'Donnell	4513
David Letterman	4504
Rod Roddy	3859
Dick Clark	3809
Aaron Elliott	3710
Luz Stella Luengas	3568
Luis Eduardo Mo...	3559
Pallavi Ramisetty	3521
Dian Parkinson	3498
Holly Hallstrom	3490
Charlie Rose	3300
Kevin Eubanks	3118
Elina Akritidou	3108
Dimitri Aronis	3093
Epistimi Aretaki	3090
Ray Meagher	3070
Ed McMahon	3015
Ana Blanco	2971
Paul Shaffer	2958
Tibor Gazdag	2944
John Aniston	2936
Calvin Grubb	2908
Alan Fletcher	2896
Stefan Dennis	2873
Kristian Alfonso	2774
Omar Fajardo	2770

Jon Stewart	2767
Caroline de Bruijn	2764
Matías Prats	2762
Edd Hall	2758
Eric Braeden	2734
Jef Desmedt	2725
Stephen Colbert	2721
Ellen DeGeneres	2712
Peter Bergman	2704
Gene Rayburn	2700
Dilip Joshi	2676
Disha Vakani	2653
Rachel Riley	2637
Susan Flannery	2635
Bartho Braat	2581
Eva Pedraza	2540
Victoria Wyndham	2530
Constance Ford	2520

Constance Ford	2520
Carson Daly	2515
Howard Stern	2512
Matthew Ashford	2504
Anne Igartiburu	2470
Kit Hoover	2468
Christian Gálvez	2468
Alexandra Lenca...	2465
Pepa Bueno	2451
Andy Richter	2449
Craig Ferguson	2422
Drew Carey	2416
Jeroen Pauw	2408
Johnny Carson	2401
María Teresa Ca...	2396
Whoopi Goldberg	2365
Ana Velázquez	2364
Shane Farley	2334
Rachael Ray	2331

The actors who have played spiderman in a movie.

Query

```
CREATE VIEW COMICS(name_id,name_,number_) AS SELECT n.name_id,n.name_,COUNT(*)
AS number_ FROM names_ n,had_role h,movies_series_titles t WHERE h.role_ LIKE
'Peter Parker' AND t.title_type LIKE 'movie' AND t.title_id = h.title_id AND n.name_id = h.name_id GROUP BY n.name_id;
SELECT * FROM COMICS;
```

name_id	name_	number_
nm0917076	Peter Weck	1
nm0001497	Tobey Maguire	3
nm0200991	Ron Darby	1
nm0106755	Patrick Breen	1
nm1940449	Andrew Garfield	2
nm9060711	Ronan Doss	1
nm10900601	Seth Beckson	1
nm10900332	Levi Balla	1
nm4043618	Tom Holland	2
nm11762833	Michael Stirling	1
nm7201417	Mark Ricci	1
nm5562562	Joey Lever	1
nm7155492	Kosta Stylianou	1
nm2896043	John Nania	1

Not all these actors have played spider man, some of them must include movies where their character's name is peter parker but it is not a spider man movie. So

Wrote a query to display movie name.

Query

```
SELECT COMICS.name_, T.title_id, T.primary_title, T.start_year
FROM COMICS, movies_series_titles AS T, Had_role AS H
WHERE COMICS.name_id = H.name_id
AND H.role_ LIKE 'Peter Parker'
AND T.title_id = H.title_id
AND T.title_type LIKE 'movie'
ORDER BY T.start_year DESC;
```

name_	title_id	primary_title	start_year
Tom Holland	tt10872600	Untitled Spider-Man Sequel	2021
Michael Stirling	tt12767902	Spider-Man 4 Fan Film	2021
Mark Ricci	tt12906558	Spider-Man: Destiny	2021
Seth Beckson	tt10778364	The Spider-Man (Fan Film)	2020
Levi Balla	tt10782740	The Spider-Man 2	2020
Ronan Doss	tt10443172	Spider-Man: Identity	2019
Tom Holland	tt2250912	Spider-Man: Homecoming	2017
Kosta Stylianou	tt4321248	The Avenging Spider-Man	2015
Andrew Garfield	tt1872181	The Amazing Spider-Man 2	2014
Joey Lever	tt2803854	Spider Man: Lost Cause	2014
Andrew Garfield	tt0948470	The Amazing Spider-Man	2012
John Nania	tt4664384	Shamelessly She-Hulk	2009
Tobey Maguire	tt0413300	Spider-Man 3	2007
Tobey Maguire	tt0316654	Spider-Man 2	2004
Tobey Maguire	tt0145487	Spider-Man	2002
Patrick Breen	tt0198438	East of A	2000
Ron Darby	tt0180401	Lord Farthingay's Holiday	1972
Peter Weck	tt0049534	Mädchen mit schwachem …	1956

Th last three movies are not spider man movies. The actors just play a character whose name is peter parker.

Different professions in the database

Query

```
SELECT P.job_category, COUNT(*)
FROM principals AS P
GROUP BY P.job_category
ORDER BY P.job_category ASC;
```

job_category	COUNT(*)
actor	6341477
actress	4598335
archive_footage	183323
archive_sound	1962
cinematographer	997115
composer	1057558
director	2890126
editor	982628
producer	1656003
production_designer	216257
self	5177697
writer	3321532

The top 100 movies as determined by the average rating

Query

```
SELECT T.title_id, T.primary_title, R.average_rating
FROM movies_series_titles AS T, title_ratings AS R
WHERE T.title_id = R.title_id
AND T.title_type = 'movie'
AND R.num_votes > 100000
ORDER BY R.average_rating DESC
LIMIT 100;
```

title_id	primary_title	average_rating
tt0111161	The Shawshank Redemption	9.3
tt0068646	The Godfather	9.2
tt0050083	12 Angry Men	9
tt0071562	The Godfather: Part II	9
tt0468569	The Dark Knight	9
tt0108052	Schindler's List	8.9
tt0110912	Pulp Fiction	8.9
tt0167260	The Lord of the Rings: The Return of the King	8.9
tt0060196	The Good, the Bad and the Ugly	8.8
tt0109830	Forrest Gump	8.8
tt0120737	The Lord of the Rings: The Fellowship of the ...	8.8
tt0137523	Fight Club	8.8
tt1375666	Inception	8.8
tt0073486	One Flew Over the Cuckoo's Nest	8.7
tt0080684	Star Wars: Episode V - The Empire Strikes Back	8.7
tt0099685	Goodfellas	8.7
tt0133093	The Matrix	8.7
tt0167261	The Lord of the Rings: The Two Towers	8.7
tt0047478	Seven Samurai	8.6
tt0076759	Star Wars: Episode IV - A New Hope	8.6
tt0102926	The Silence of the Lambs	8.6
tt0114369	Se7en	8.6
tt0118799	Life Is Beautiful	8.6
tt0120689	The Green Mile	8.6
tt0120815	Saving Private Ryan	8.6
tt0245429	Spirited Away	8.6
tt0317248	City of God	8.6
tt0816692	Interstellar	8.6
tt0054215	Psycho	8.5
tt0064116	Once Upon a Time in the West	8.5
tt0088763	Back to the Future	8.5
tt0095327	Grave of the Fireflies	8.5
tt0095765	Cinema Paradiso	8.5
tt0103064	Terminator 2: Judgment Day	8.5
tt0110357	The Lion King	8.5
tt0110413	Léon: The Professional	8.5
tt0114814	The Usual Suspects	8.5
tt0120586	American History X	8.5

tt0114814	The Usual Suspects	8.5
tt0120586	American History X	8.5
tt0172495	Gladiator	8.5
tt0253474	The Pianist	8.5
tt0407887	The Departed	8.5
tt0482571	The Prestige	8.5
tt1675434	The Intouchables	8.5
tt2582802	Whiplash	8.5
tt0047396	Rear Window	8.4
tt0050825	Paths of Glory	8.4
tt0051201	Witness for the Prosecution	8.4
tt0057012	Dr. Strangelove or: How I Learned to Stop ...	8.4
tt0078748	Alien	8.4
tt0078788	Apocalypse Now	8.4
tt0081505	The Shining	8.4
tt0082971	Indiana Jones and the Raiders of the Lost Ark	8.4
tt0087843	Once Upon a Time in America	8.4
tt0119698	Princess Mononoke	8.4
tt0209144	Memento	8.4
tt0364569	Oldboy	8.4
tt0209144	Memento	8.4
tt0364569	Oldboy	8.4
tt0405094	The Lives of Others	8.4
tt0910970	WALL·E	8.4
tt0986264	Like Stars on Earth	8.4
tt1187043	3 Idiots	8.4
tt1345836	The Dark Knight Rises	8.4
tt1853728	Django Unchained	8.4
tt2380307	Coco	8.4
tt4154756	Avengers: Infinity War	8.4
tt4154796	Avengers: Endgame	8.4
tt4633694	Spider-Man: Into the Spider-Verse	8.4
tt5074352	Dangal	8.4
tt0045152	Singin' in the Rain	8.3
tt0052357	Vertigo	8.3
tt0053125	North by Northwest	8.3
tt0053604	The Apartment	8.3
tt0056172	Lawrence of Arabia	8.3
tt0059578	For a Few Dollars More	8.3
tt0062622	2001: A Space Odyssey	8.3

tt0066921	A Clockwork Orange	8.3
tt0070735	The Sting	8.3
tt0075314	Taxi Driver	8.3
tt0082096	Das Boot	8.3
tt0086190	Star Wars: Episode VI - Return of the Jedi	8.3
tt0086250	Scarface	8.3
tt0086879	Amadeus	8.3
tt0090605	Aliens	8.3
tt0093058	Full Metal Jacket	8.3
tt0105236	Reservoir Dogs	8.3
tt0112573	Braveheart	8.3
tt0114709	Toy Story	8.3
tt0119217	Good Will Hunting	8.3
tt0169547	American Beauty	8.3
tt0180093	Requiem for a Dream	8.3
tt0208092	Snatch	8.3
tt0211915	Amélie	8.3
tt0338013	Eternal Sunshine of the Spotless Mind	8.3
tt0361748	Inglourious Basterds	8.3
tt1255953	Incendies	8.3
tt1832382	A Separation	8.3
tt2106476	The Hunt	8.3
tt0046912	Dial M for Murder	8.2
tt0050976	The Seventh Seal	8.2
tt0053291	Some Like It Hot	8.2
tt0055630	Yojimbo	8.2

Displaying the average rating of movies of each genre

Query

```
SELECT AVG(t.average_rating),g.genre FROM title_ratings t,title_genres g,movies_series_titles p
 WHERE t.title_id = g.title_id AND p.title_id = t.title_id
AND p.title_type = 'movie' GROUP BY g.genre ORDER BY average_rating;
```

AVG(t.average_rating)	genre
7.2154723118493145	News
7.213340248830989	Documentary
6.968296208124089	Biography
6.822962643851717	Music
6.820373393302165	History
6.584868716037056	Sport
6.461913878666727	War
6.444062497466803	Film-Noir
6.400000095367432	Game-Show
6.35	Short
6.226198756854	Drama
6.17620421088598	Animation
6.171188489353912	Family
6.109836878513736	Musical
6.088461527457604	Reality-TV
6.0666667222976685	Talk-Show
6.054924973574153	Romance
5.971227732113996	Crime
5.906088305115067	Adventure
5.875330864554575	Comedy
5.863900743616838	Fantasy
5.850919730587548	Western
5.845793865955012	Mystery
5.771905000791699	Adult
5.672948136772738	Action
5.555385330731712	Thriller
5.2569289084294395	Sci-Fi
4.998721695573663	Horror

List of all the actor names who had a role in the movie Tangled

QUERY

```
SELECT N.name_, H.role_
FROM movies_series_titles AS T, had_role AS H, names_ AS N
WHERE T.primary_title LIKE 'Tangled'
AND T.title_type LIKE 'movie'
AND T.title_id = H.title_id
AND H.name_id = N.name_id;
```

name_	role_
Al Vicente	Carlos Menedez
Rich Komenich	Frank Welby
Bruce Orendorf	Denny Palmer
Yolanda Androzzo	Diamond James
Rachael Leigh Cook	Jenny Kelley
Jonathan Rhys Meyers	Alan Hammond
Shawn Hatosy	David Klein
Estella Warren	Elise Stevens
Ron Perlman	Stabbington Brother
Mandy Moore	Rapunzel
Donna Murphy	Mother Gothel
Zachary Levi	Flynn Rider

List the count of all movies and series of each genre in which Chris Evans had a role

QUERY

```
SELECT G.genre,COUNT( DISTINCT G.title_id) FROM names_ N, title_genres G, had_role H
WHERE H.name_id = N.name_id AND H.title_id = G.title_id AND N.name_ = 'Chris Evans' GROUP BY G.genre;
```

genre	COUNT(DISTINCT G.title_id)
Action	23
Adult	9
Adventure	19
Animation	5
Biography	5
Comedy	87
Crime	8
Documentary	19
Drama	31
Family	9
Fantasy	7
Game-Show	14
Music	63
Mystery	1
News	118
Reality-TV	5
Romance	11
Sci-Fi	10
Short	19
Sport	4
Talk-Show	107
Thriller	4

CONCLUSION:

In this project I built a MySQL database using the Internet Movie Database (IMDb) dataset. The dataset consists of 7 compressed tab-separated-value (*.tsv) files that can be downloaded from the IMDb website. During the database management course, I learned about the basics of database design. I designed a database that can be used by movie enthusiasts as well as data analysts to efficiently work with the massive amount of movie data available on the IMDb website. This project gave me the opportunity to try my new skills in practice. While doing this project I also gained deeper understanding on database design and how it can be implemented in real life situations.

REFERENCES:

- https://datasets.imdbws.com/ for the dataset
- https://docs.python.org/3.9/ for data pre-processing and importing