THIS IS HOW TO SET UP MYSQL DATABASE WITH LOADING ALL THE .TSV FILES

\* DB COMMANDS

in case you mess up the Database. DROP DATABASE movie\_review & create it again after. CREATE DATABASE movie\_review.

DROP DATABASE movie\_review;

CREATE DATABASE movie\_review;

SHOW databases;

DROP TABLE IF EXISTS TMDB\_Title;

TRUNCATE TABLE TMDB\_Title;

SELECT \* FROM TMDB\_Title LIMIT 1;

Alright, now create 10 tables SQL QUERIES

Final SQL SCHEMA, JUST COPY FROM USE movie\_review to end of table 10

USE movie\_review;

-- Table 1: IMDB\_Title

CREATE TABLE IMDB\_Title (

title\_id VARCHAR(20) PRIMARY KEY, -- title\_id should be the primary key

title VARCHAR(255) NOT NULL,

year INT,

runtime INT,

genres VARCHAR(255)

);

-- Table 2: Rating

-- Modified Rating Table

CREATE TABLE Rating (

rating\_id INT AUTO\_INCREMENT PRIMARY KEY, -- Added a new primary key column

title\_id VARCHAR(20), -- Foreign key referencing IMDB\_Title

rating DECIMAL(3, 1),

num\_votes INT,

FOREIGN KEY (title\_id) REFERENCES IMDB\_Title(title\_id) ON DELETE CASCADE

);

-- Table 3: Person

CREATE TABLE Person (

person\_id VARCHAR(20) PRIMARY KEY, -- Primary Key: person\_id

name VARCHAR(255) NOT NULL,

birth\_year INT,

death\_year INT,

known\_for\_titles VARCHAR(255),

professions VARCHAR(255)

);

-- Table 4: Collaborated\_On (Many-to-Many relationship between IMDB\_Title and Person)

CREATE TABLE Collaborated\_On (

title\_id VARCHAR(20), -- Should be VARCHAR to match IMDB\_Title

person\_id VARCHAR(20),

professions VARCHAR(255),

PRIMARY KEY (title\_id, person\_id), -- Composite Primary Key

FOREIGN KEY (title\_id) REFERENCES IMDB\_Title(title\_id) ON DELETE CASCADE,

FOREIGN KEY (person\_id) REFERENCES Person(person\_id) ON DELETE CASCADE

);

-- Table 5: TMDB\_Title (Table to store data from TMDB)

CREATE TABLE TMDB\_Title (

tmdb\_id INT PRIMARY KEY, -- Primary Key: tmdb\_id

title VARCHAR(255) NOT NULL, -- Movie title

vote\_average FLOAT, -- Movie rating (example: 8.3)

vote\_count INT, -- Number of votes

status VARCHAR(50), -- Status of the movie (e.g., "Released")

release\_date DATE, -- Release date (YYYY-MM-DD)

revenue BIGINT, -- Revenue (BIGINT for larger values)

runtime INT, -- Runtime in minutes

adult BOOLEAN, -- Adult status (True/False)

backdrop\_path VARCHAR(255), -- Path to the backdrop image

budget BIGINT, -- Budget (BIGINT for larger values)

homepage VARCHAR(255), -- Homepage URL

imdb\_id VARCHAR(20), -- IMDb ID (foreign key to IMDB\_Title)

original\_language VARCHAR(50), -- Original language of the movie

original\_title VARCHAR(255), -- Original title of the movie

overview TEXT, -- Overview/description of the movie

popularity FLOAT, -- Popularity score

poster\_path VARCHAR(255), -- Path to the poster image

tagline VARCHAR(255), -- Movie tagline

genres VARCHAR(255), -- Genres (comma-separated)

production\_companies TEXT, -- Production companies (comma-separated)

production\_countries VARCHAR(255), -- Production countries (comma-separated)

spoken\_languages VARCHAR(255), -- Spoken languages (comma-separated)

keywords MEDIUMTEXT, -- Keywords (comma-separated)

FOREIGN KEY (imdb\_id) REFERENCES IMDB\_Title(title\_id) ON DELETE CASCADE

);

-- Table 6: TV\_Series (One-to-Many relationship with TV\_Episode)

CREATE TABLE TV\_Series (

title\_id VARCHAR(20) PRIMARY KEY, -- Primary Key & Foreign Key referencing IMDB\_Title

title VARCHAR(255) NOT NULL,

start\_year INT,

end\_year INT,

genres VARCHAR(255),

FOREIGN KEY (title\_id) REFERENCES IMDB\_Title(title\_id) ON DELETE CASCADE

);

-- Table 7: TV\_Episode (Many-to-One relationship with TV\_Series)

CREATE TABLE TV\_Episode (

title\_id VARCHAR(20) PRIMARY KEY, -- title\_id as the primary key of the episode

tv\_series\_id VARCHAR(20), -- Foreign Key referencing TV\_Series (title\_id is VARCHAR)

season\_number INT,

episode\_number INT,

FOREIGN KEY (title\_id) REFERENCES IMDB\_Title(title\_id) ON DELETE CASCADE,

FOREIGN KEY (tv\_series\_id) REFERENCES TV\_Series(title\_id) ON DELETE CASCADE

);

-- Table 8: Short (Subclass of IMDB\_Title)

CREATE TABLE Short (

title\_id VARCHAR(20) PRIMARY KEY, -- Foreign Key referencing IMDB\_Title

FOREIGN KEY (title\_id) REFERENCES IMDB\_Title(title\_id) ON DELETE CASCADE

);

-- Table 9: Feature\_Film (Subclass of IMDB\_Title)

CREATE TABLE Feature\_Film (

title\_id VARCHAR(20) PRIMARY KEY, -- Foreign Key referencing IMDB\_Title

FOREIGN KEY (title\_id) REFERENCES IMDB\_Title(title\_id) ON DELETE CASCADE

);

-- Table 10: Is\_Same\_Title (One-to-One relationship between IMDB\_Title and TMDB\_Title)

CREATE TABLE Is\_Same\_Title (

tmdb\_id INT, -- Foreign Key referencing TMDB\_Title

imdb\_id VARCHAR(20), -- Foreign Key referencing IMDB\_Title

PRIMARY KEY (tmdb\_id, imdb\_id),

FOREIGN KEY (tmdb\_id) REFERENCES TMDB\_Title(tmdb\_id) ON DELETE CASCADE,

FOREIGN KEY (imdb\_id) REFERENCES IMDB\_Title(title\_id) ON DELETE CASCADE

);

-- Table 11: Users

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50) NOT NULL UNIQUE,

email VARCHAR(100) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

Ok, Loading over 1 million data sucks. I use LOAD DATA statement.

After manually inserting all this data and getting errors. I found a way to insert the whole data files into mySQL.

BUT U can’t load files because mySQL has security.

Long story short, I decided not to load all data. I SPLIT the data file. Here’s how:

IMPORTANT SETUP

FIND THE PRIV DIRECTORY. In MySQL Query this in mySQL:

SHOW VARIABLES LIKE 'secure\_file\_priv';

A screenshot of a computer

Description automatically generated

You see the directory or FILE PATH. This is the path to let u upload files into mySQL.

Copy the highlighted path ROW.

PASTE IT INTO your file explorer. THE ADDRESS BAR OF MY FILES OR FILE EXPLORER. LOOK LIKE:

**'secure\_file\_priv'**, 'C:**\\**ProgramData**\\**MySQL**\\**MySQL Server 8.0**\\**Uploads**\\'**

BUT delete the name and double back lines so it looks something like this:

C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\

It should look like this in your file explorer (YOURS WILL BE EMPTY)

A screenshot of a computer

Description automatically generated

As you see, I divided the original files. This is because I kept getting errors or mySQL would not load over 1 million data files.

1. Open the original .tsv files into a word editior. I used VSCODE.
2. COPY the lines up to line whatever.( I used 10000) \*notes all other files have to match the same lines, so title.crew, title.rating, title.episode wil have to match this.
3. PASTE the lines into notepad and save it as part 1 into the PRIV directory:

\* C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\ THE FILE MUST BE IN YOUR PRIV PATH

JUST COPY PASTE. Don’t mess up the tab or lines.

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Rename the copy paste file as the same but part 1.

Alright, now you have the datebase, the 10 tables, and the date files in your PRIV directory. Now it’s time to INSERT or LOAD the data files into the tables.

**Example Insertion Sequence:**

**IMDB\_Title**:

* Load data from title.basics.tsv.

**Rating**:

* Load data from title.ratings.tsv.

**Person**:

* Load data from name.basics.tsv.

**Collaborated\_On**:

* Load data from title.principals.tsv.

**TMDB\_Title**:

* Load data from TMDB\_movie\_dataset\_v11.csv.

**Is\_Same\_Title**:

* Insert corresponding imdb\_id and tmdb\_id relationships (from TMDB\_movie\_dataset\_v11.csv).

**TV\_Series**:

* Load TV series data from title.basics.tsv (filter by titleType).

**TV\_Episode**:

* Load episode data from title.episode.tsv.

**Short/Feature\_Film**:

* Insert data for shorts or feature films from title.basics.tsv.

SQL QUERIES

**1.IMDB\_Title (Load from**title.basics.tsv**)**

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\title.basics.part1.tsv'

INTO TABLE IMDB\_Title

FIELDS TERMINATED BY '\t'

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(title\_id, @dummy, @title, @dummy, @dummy, @year, @dummy, @runtime, @genres)

SET

title = LEFT(@title, 255), -- Truncate title to 255 characters if it's too long

year = NULLIF(@year, '\N'), -- Set year to NULL if it's '\N' in the file

runtime = NULLIF(@runtime, '\N'), -- Set runtime to NULL if it's '\N' in the file

genres = @genres; -- Assuming genres are directly stored

**2. Rating (Load from**title.ratings.tsv**) Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`movie\_review`.`rating`, CONSTRAINT `rating\_ibfk\_1` FOREIGN KEY (`title\_id`) REFERENCES `imdb\_title` (`title\_id`) ON DELETE CASCADE)**

SET foreign\_key\_checks = 0;

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\title.ratings.part1.tsv'

INTO TABLE Rating

FIELDS TERMINATED BY '\t'

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(title\_id, rating, num\_votes);

SET foreign\_key\_checks = 1;

**I bypass the foreign key constraint, which could result in orphaned records in your Rating table that do not correspond to any title\_id in IMDB\_Title because some data is missing like title\_id, is duplicated, or won’t update at all ratings.tsv to basics.tsv. \**

**3. Person (Load from**name.basics.tsv**)**

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\name.basics.part1.tsv'

INTO TABLE Person

FIELDS TERMINATED BY '\t'

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(person\_id, name, birth\_year, death\_year, known\_for\_titles, professions);

**4. Collaborated\_On (Load from**title.principals.tsv**)**

SET foreign\_key\_checks = 0;

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\title.principals.part1.tsv'

REPLACE INTO TABLE Collaborated\_On

FIELDS TERMINATED BY '\t'

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(@tconst, @ordering, @nconst, @category, @job, @characters)

SET

title\_id = @tconst,

person\_id = @nconst,

professions = @category;

SET foreign\_key\_checks = 1;

**5. TMDB\_Title (Load from**TMDB\_movie\_dataset\_v11.csv**)**

SET foreign\_key\_checks = 0;

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\TMDB\_movie\_dataset\_v11.csv'

REPLACE INTO TABLE TMDB\_Title

FIELDS TERMINATED BY ',' -- CSV fields are comma-separated

ENCLOSED BY '"' -- Quoted fields are enclosed by double quotes

LINES TERMINATED BY '\r\n' -- Ensure line breaks are properly handled (Windows line endings)

IGNORE 1 ROWS -- Skip the header row

(

tmdb\_id, title, vote\_average, vote\_count, status, release\_date, revenue, runtime, adult, backdrop\_path,

budget, homepage, imdb\_id, original\_language, original\_title, overview, popularity, poster\_path,

tagline, genres, production\_companies, production\_countries, spoken\_languages, keywords

);

SET foreign\_key\_checks = 1;

**6. FINAL SERIES**

SET foreign\_key\_checks = 0;

ALTER TABLE TV\_Series

ADD COLUMN title\_type VARCHAR(50);

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\name.basics.part1.tsv'

INTO TABLE TV\_Series

FIELDS TERMINATED BY '\t' -- Tab-separated values

ENCLOSED BY '"' -- Enclosed by quotes (if any)

LINES TERMINATED BY '\n' -- Newline character for each row

IGNORE 1 ROWS -- Ignore the header row

(title\_id, title, start\_year, end\_year, genres, @extra1, @extra2) -- Map columns and discard extras

SET title\_type = 'tvSeries'; -- Set title\_type to 'tvSeries'

ALTER TABLE TV\_Episode MODIFY episode\_number INT NULL;

SET foreign\_key\_checks = 1;

* 1. **FINAL EPISODE**

SET foreign\_key\_checks = 0;

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\title.episode.part1.tsv'

INTO TABLE tv\_episode

FIELDS TERMINATED BY '\t'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(title\_id, tv\_series\_id, @season\_number, @episode\_number)

SET

season\_number = IFNULL(NULLIF(TRIM(@season\_number), ''), 0),

episode\_number = IFNULL(NULLIF(TRIM(@episode\_number), ''), 0);

SET foreign\_key\_checks = 1;

* 1. **SHORT FINAL INSERTING IN SHORT TABLE (3 Queries)**

SQL Q:

SELECT title\_id, title, genres

FROM IMDB\_Title

WHERE genres LIKE '%Short%';

SQL Q:

INSERT INTO Short (title\_id)

SELECT title\_id

FROM IMDB\_Title

WHERE genres LIKE '%Short%';

SQL Q:

SELECT \* FROM Short;

* 1. **FINAL FEATURE FILMS QUERIIES (2 Queries)**

SQL Q:

SELECT title\_id, title, genres

FROM IMDB\_Title

WHERE genres NOT LIKE '%Short%';

SQL Q:

INSERT INTO Feature\_Film (title\_id)

SELECT title\_id

FROM IMDB\_Title

WHERE genres NOT LIKE '%Short%';

THE DATABASE movie\_review is now set up. Now connect it to backend and frontend. – John

Since my team did not communicate with presenting the demo,

I did it myself.

localhost:8888

This is my local host links: will not work with yall. SHOWN in DEMO.

[Movie Review Website](http://localhost/movie_review_solo/)

Localhost/movie\_review\_solo