DataBase advanced techniques midterm

About Dataset

This dataset contains three csv datas, the first one is actor dataset which features include over 28000 actors from 1874 to 2016 with their english name, actor name, date of birth, gender and the second one is location dataset which contain title, actor1, actor2, actor3,distributor, director, writer, cast, rating,poster, language of the movie, awards won by the movie and much more and third one is composer which includes film name, year, gender and date of birth. I have chosen the dataset from the wikidata open source.

Is the data reliable? On what basis do you make that judgement?

Likely to be reliable since it is a popular open source. The data is reliable; they all present some structure, not requiring pure full text search.

How much detail is there? Is the information helpful?

The all three datasets present some reasonable level of detail over the topic they discuss. However, the Movies datasets present a wealth of data, evaluated and segregated through multiple angles.

Would it be useful to connect to other data sets? Which? How easy would that be?

This really comes down to the problem that needs solving, but because of the segregation of the movie's dataset, it's clear that correlating some of its datasets would be reasonably valuable.

How clear is it what the data means? Where was the documentation? Was it easy to find?

It is clear. There is a pdf file that includes part of the description for some of the files. And inside each file, there are good documents for each dataset, for example in the Location dataset a good explanation about each actor and directors.

What could you use it for? What questions would you like to ask the dataset but can't? What's missing?

Focusing on the movies dataset, it can be used to calculate several things such as a title movie winning over another (considering past results only). Statistics could be potentially improved if we had information about each individual movie's performance within the writers - correlation with other datasets could be valuable.

How easy was it to find open data in your chosen domain? Where did you go? Were there many alternatives?

The website with the movie's dataset was reasonably easy to find, requiring no more than a few seconds on Google Search and I found the dataset in the Wikidata website.

What have you found?

Each one of the Movies Datasets have a different/independent licensing setup.

Where did you find the information?
The Movies Dataset points to Wikidata that makes the licensing option explicit. Other datasets in the same rapidapi website mentioned about being publicly available, but they did not provide any terms on which were the licence terms.

What is not allowed by the terms of the licence?
Wikidata is given with a little more detail in the wikidata Licence Website.

“Wikidata requires a CC0 licence, which is equivalent to public domain; anyone may freely designate any public-domain data to be CC0.”

“Other than public-domain data and CC0 data, no other data licence or designation is compatible with Wikidata's copyright requirements.”

How easy is it for the licensing and rights information to get separated from the data?
Many datasets did not even provide any form of licence in their website. Also, the dataset downloads often did not include the licence and it was expected to be checked on the website. It's extremely simple to get them separated.

Most information presented on the wikidata Web site is considered public domain information.

As mentioned in wikidata for licencing

“Wikidata contributors sometimes avoid uploading data when someone has made any of the following kinds of copyright claims on the data:

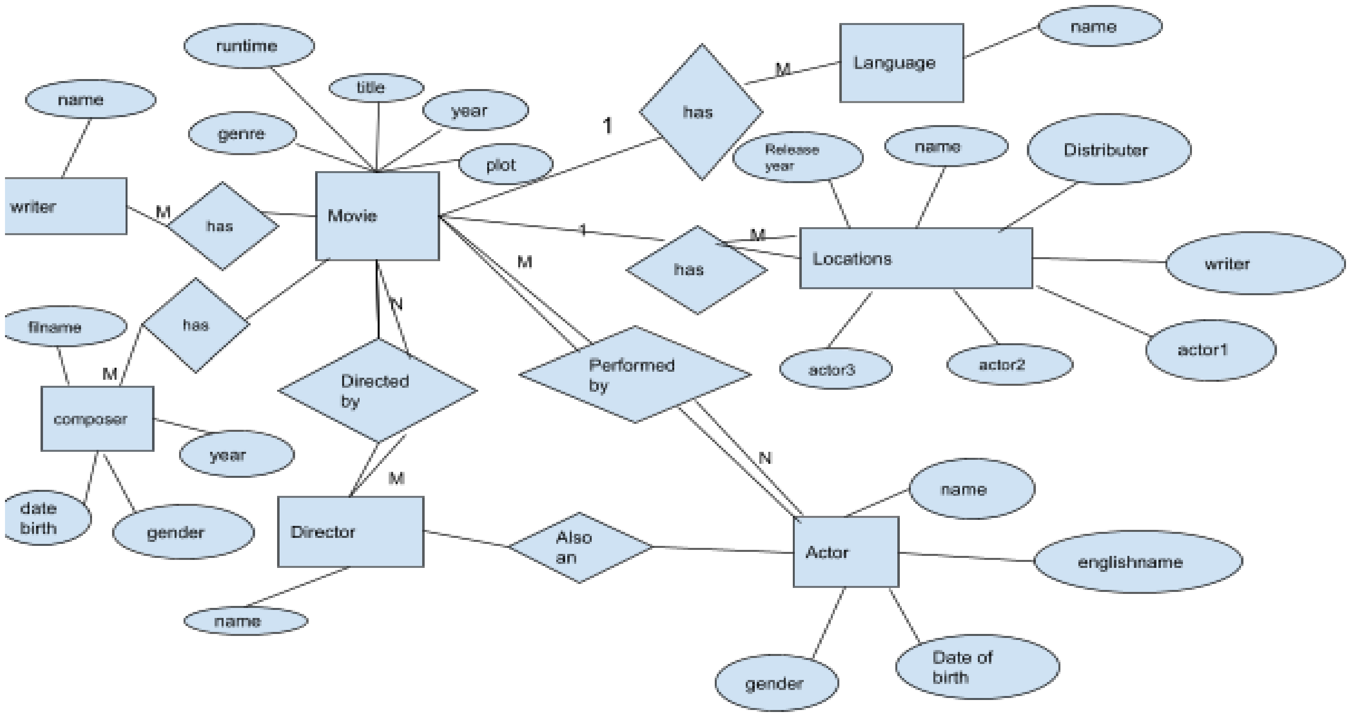
- traditional copyright, even with no copyright notice at all
- any Creative Commons licence other than CC0. This includes Creative Commons Attribution, the very permissive licence accepted on other Wikimedia projects. Some datasets may have Creative Commons Non-Commercial licences as well.
- any specialised licence with restrictions
- any of the many copyright licences applicable to creative works or open source software but which, for some reason, people try to apply to data.”

Inspiration:

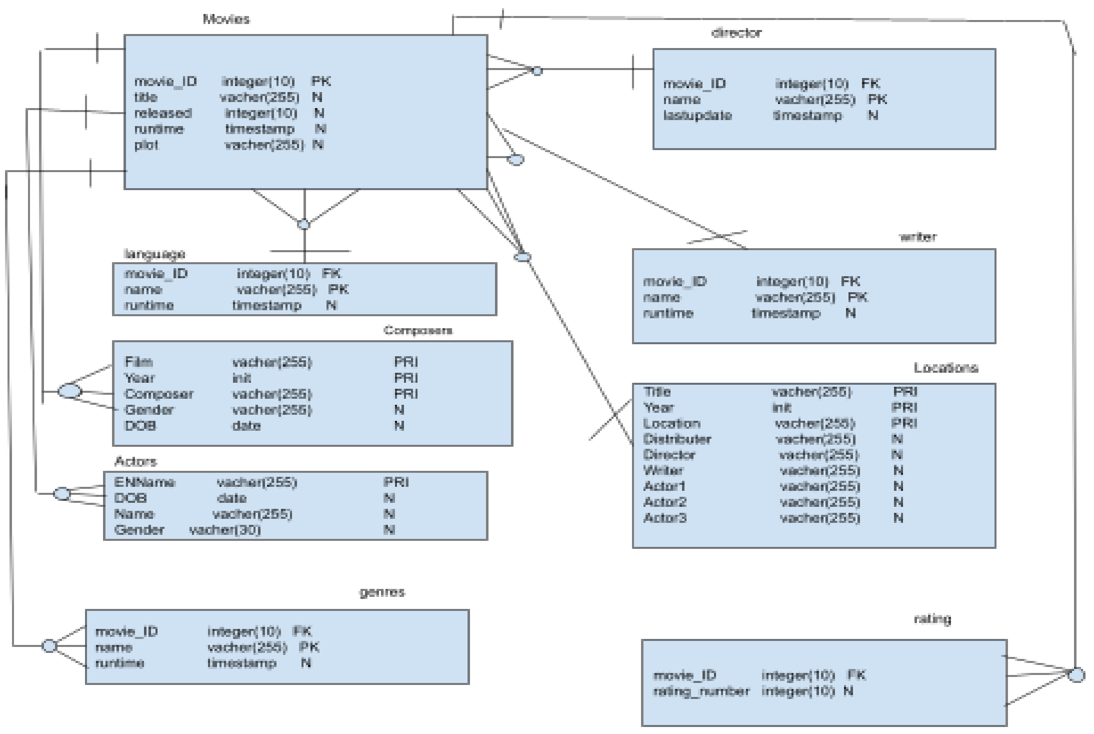
This dataset is assembled as my coursework Database and advanced technique. I wanted to perform an extensive exploratory data on Movie Data to describe the history and the story of Films. For instance, use the Locations data in combination with MovieActors to see various types of Actors with English names.

Is the number of movies produced affected by composers from the previous years?

Draw a complete E/R model of the data:



Add cardinality to your E/R diagram



Movies

movie_ID	integer(10)	PK
title	vacher(255)	N
released	integer(10)	N
runtime	timestamp	N
plot	vacher(255)	N

director

movie_ID	integer(10)	FK
name	vacher(255)	PK
lastupdate	timestamp	N

language

movie_ID	integer(10)	FK
name	vacher(255)	PK
runtime	timestamp	N

writer

movie_ID	integer(10)	FK
name	vacher(255)	PK
runtime	timestamp	N

Composers

Film	vacher(255)	PRI
Year	init	PRI
Composer	vacher(255)	PRI
Gender	vacher(255)	N
DOB	date	N

Locations

Title	vacher(255)	PRI
Year	init	PRI
Location	vacher(255)	PRI
Distributer	vacher(255)	N
Director	vacher(255)	N
Writer	vacher(255)	N
Actor1	vacher(255)	N
Actor2	vacher(255)	N
Actor3	vacher(255)	N

Actors

ENName	vacher(255)	PRI
DOB	date	N
Name	vacher(255)	N
Gender	vacher(30)	N

genres

movie_ID	integer(10)	FK
name	vacher(255)	PK
runtime	timestamp	N

rating

movie_ID	integer(10)	FK
rating_number	integer(10)	N

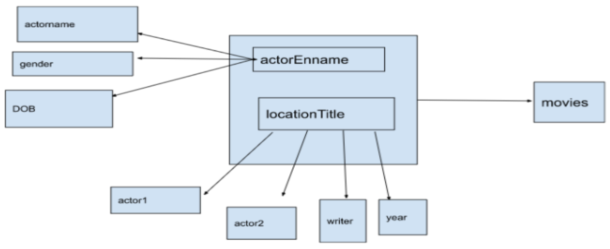
Normal forms:
Provide sets of attributes that guarantee that relations are well structured. These are progressive, with different normal forms. They are defined in terms of functional dependencies, a database is normalised to avoid redundancy of data.

My database is at least in 3NF.

Normal forms

First-norm

movies-info									
actorEnname	name	DOB	gender	location	title	actor1	writer	movies	
Sarah	sarah	1923-09	female	USA	LA	sarah	Jack	earth	



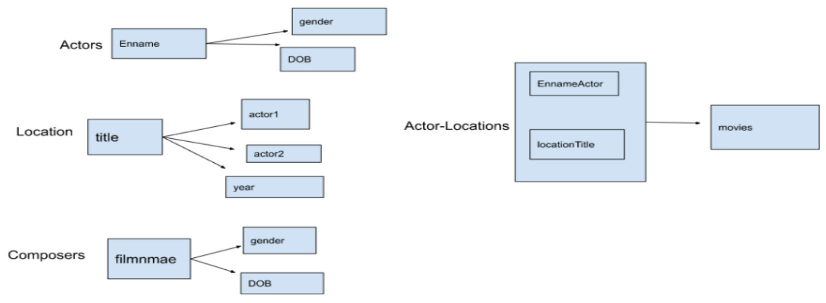
second-norm

Actors			Locations			
Enname(PK)	gender	DOB	Locationtitle(PK)	actor1	actor2	year
Sarah	female	1923-09	USA	Sarah	Sam	1987

Actor-Locations		
Ennmae(PK)	locationTitle(PK)	movies
Sarah	USA	earth

Third-norm

Actors			Composers			
actorEnname	actorname	gender	filmname	year	DOB	

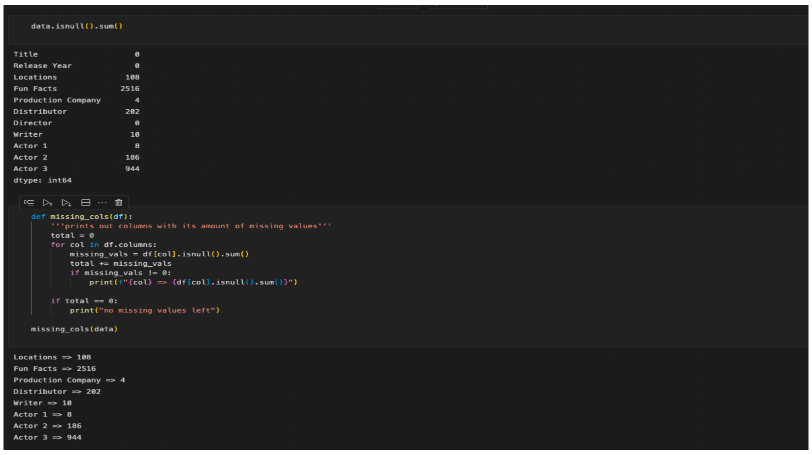


Cleaning data :

I have cleaned the datasets by Python library before loading data.csv to the Movies database.

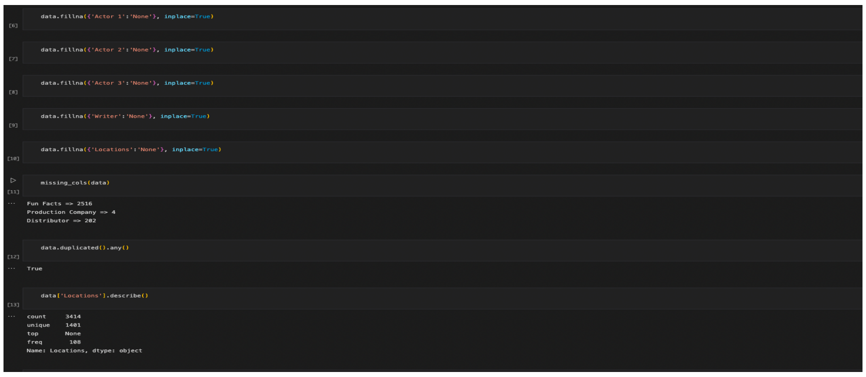
Missing Data

Check for Duplicates



```
data.isnull().sum()

Title                  0
Release Year           0
Locations            108
Fun Facts           2516
Production Company     4
Distributor          202
Director               0
Writer                10
Actor 1                8
Actor 2              186
Actor 3              944
dtype: int64
```

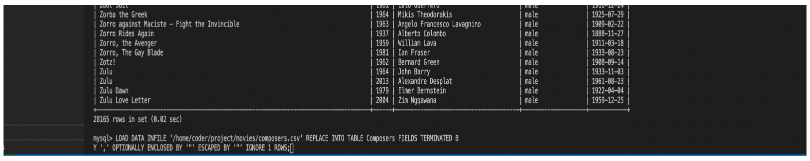
```
def missing_cols(df):
    '''prints out columns with its amount of missing values'''
    total = 0
    for col in df.columns:
        missing_vals = df[col].isnull().sum()
        total += missing_vals
        if missing_vals != 0:
            print(f"{col} => {df[col].isnull().sum()}")

    if total == 0:
        print("no missing values left")

missing_cols(data)

Locations => 108
Fun Facts => 2516
Production Company => 4
Distributor => 202
Writer => 10
Actor 1 => 8
Actor 2 => 186
Actor 3 => 944
```



```
[6]   data.fillna({'Actor 1':'None'}, inplace=True)

[7]   data.fillna({'Actor 2':'None'}, inplace=True)

[8]   data.fillna({'Actor 3':'None'}, inplace=True)

[9]   data.fillna({'Writer':'None'}, inplace=True)

[10]  data.fillna({'Locations':'None'}, inplace=True)

[11]  missing_cols(data)

...   Fun Facts => 2516
      Production Company => 4
      Distributor => 202

[12]  data.duplicated().any()

...   True

[13]  data['Locations'].describe()

...   count     3414
      unique    1401
      top       None
      freq       108
      Name: Locations, dtype: object
```

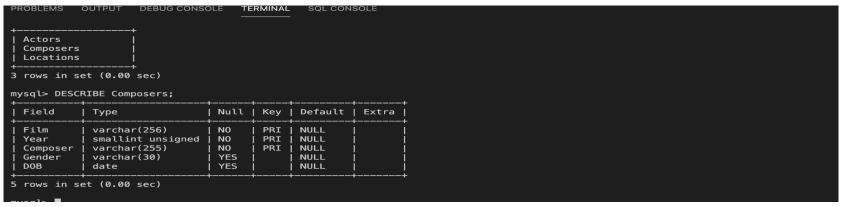
Database Schema(setup sql): List SQL commands

Load composers.csv:



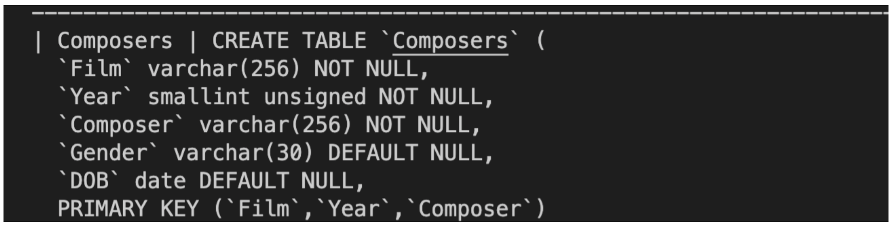
```
| Zorba the Greek                                    | 1964 | Mikis Theodorakis           | male      | 1925-07-29 |
| Zorro against Maciste - Fight the Invincible       | 1963 | Angelo Francesco Lavagnino  | male      | 1909-02-22 |
| Zorro Rides Again                                  | 1937 | Alberto Colombo             | male      | 1888-11-27 |
| Zorro, the Avenger                                 | 1959 | William Lava                | male      | 1911-03-18 |
| Zorro, The Gay Blade                               | 1981 | Ian Fraser                  | male      | 1933-08-23 |
| Zotz!                                              | 1962 | Bernard Green               | male      | 1908-09-14 |
| Zulu                                               | 1964 | John Barry                  | male      | 1933-11-03 |
| Zulu                                               | 2013 | Alexandre Desplat           | male      | 1961-08-23 |
| Zulu Dawn                                          | 1979 | Elmer Bernstein             | male      | 1922-04-04 |
| Zulu Love Letter                                   | 2004 | Zim Ngqawana                | male      | 1959-12-25 |
+----------------------------------------------------+------+-----------------------------+-----------+------------+
28165 rows in set (0.02 sec)

mysql> LOAD DATA INFILE '/home/coder/project/movies/composers.csv' REPLACE INTO TABLE Composers FIELDS TERMINATED B
Y ',' OPTIONALLY ENCLOSED BY '"' ESCAPED BY '"' IGNORE 1 ROWS;
```

Describe Composers:



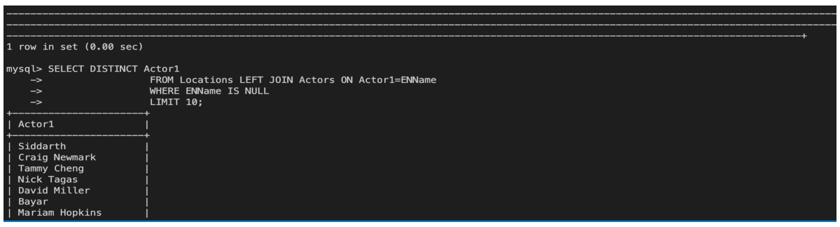
```
+------------------+
| Actors           |
| Composers        |
| Locations        |
+------------------+
3 rows in set (0.00 sec)

mysql> DESCRIBE Composers;
+----------+-------------------+------+-----+---------+-------+
| Field    | Type              | Null | Key | Default | Extra |
+----------+-------------------+------+-----+---------+-------+
| Film     | varchar(256)      | NO   | PRI | NULL    |       |
| Year     | smallint unsigned | NO   | PRI | NULL    |       |
| Composer | varchar(255)      | NO   | PRI | NULL    |       |
| Gender   | varchar(30)       | YES  |     | NULL    |       |
| DOB      | date              | YES  |     | NULL    |       |
+----------+-------------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

CREATE TABLE Composers:

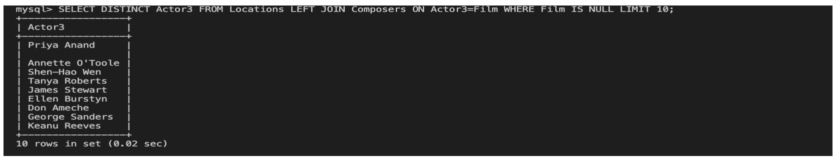


```
 ------------------------------------------------------------------
| Composers | CREATE TABLE `Composers` (
  `Film` varchar(256) NOT NULL,
  `Year` smallint unsigned NOT NULL,
  `Composer` varchar(256) NOT NULL,
  `Gender` varchar(30) DEFAULT NULL,
  `DOB` date DEFAULT NULL,
  PRIMARY KEY (`Film`,`Year`,`Composer`)
```

SELECT Actor1 and 3:

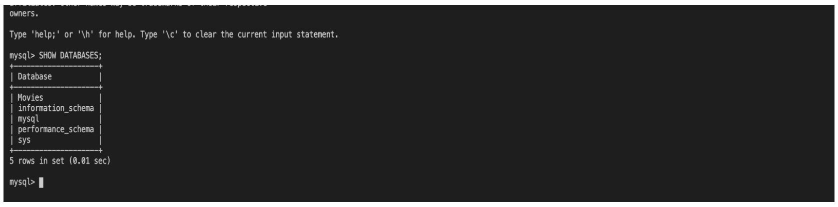


```
1 row in set (0.00 sec)

mysql> SELECT DISTINCT Actor1
    ->                  FROM Locations LEFT JOIN Actors ON Actor1=ENName
    ->                  WHERE ENName IS NULL
    ->                  LIMIT 10;
+-----------------------+
| Actor1                |
+-----------------------+
| Siddarth              |
| Craig Newmark         |
| Tammy Cheng           |
| Nick Tagas            |
| David Miller          |
| Bayar                 |
| Mariam Hopkins        |
```



```
mysql> SELECT DISTINCT Actor3 FROM Locations LEFT JOIN Composers ON Actor3=Film WHERE Film IS NULL LIMIT 10;
+------------------+
| Actor3           |
+------------------+
| Priya Anand      |
|                  |
| Annette O'Toole  |
| Shen-Hao Wen     |
| Tanya Roberts    |
| James Stewart    |
| Ellen Burstyn    |
| Don Ameche       |
| George Sanders   |
| Keanu Reeves     |
+------------------+
10 rows in set (0.02 sec)
```

SHOW DATABASES;



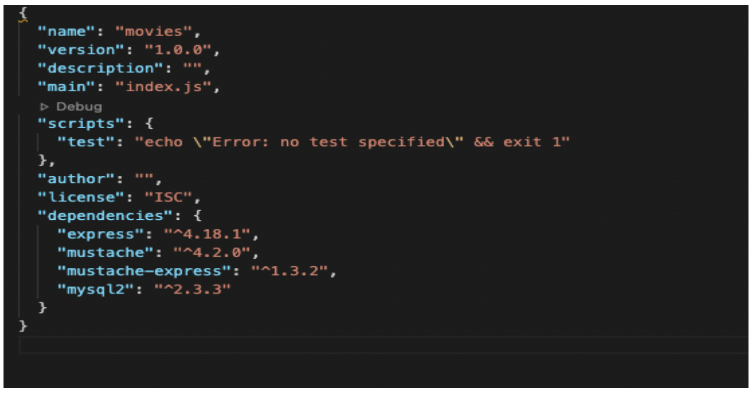
```
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| Movies             |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.01 sec)

mysql> 
```

SELECT *FROM TABLE Composers;

Stage 4. Create a simple web application

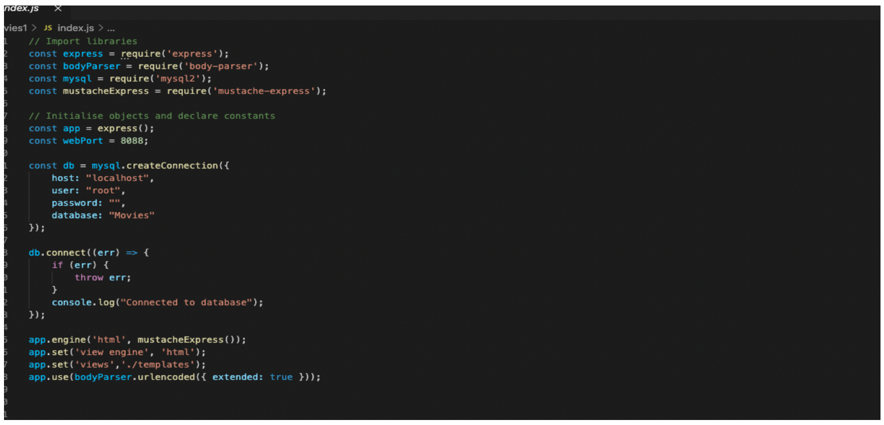
I have created a four simple page web application.
I have Installed the libraries by running npm install there is the list of installation in package.json)

I have used Node.js express and moustache-express which allowed me to talk to database and used to create web pages.



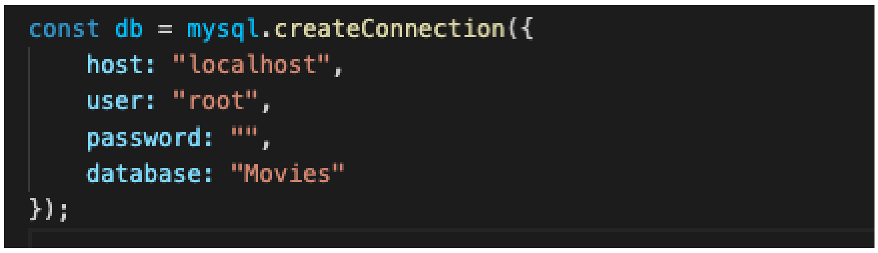
```
{
  "name": "movies",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  ▷ Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.18.1",
    "mustache": "^4.2.0",
    "mustache-express": "^1.3.2",
    "mysql2": "^2.3.3"
  }
}
```

The express server:



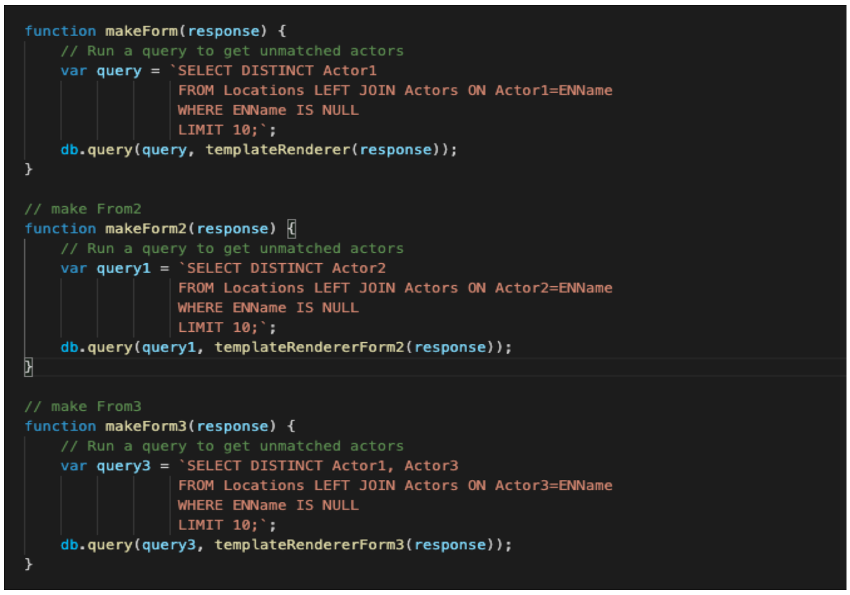
```
ndex.js  ×
vies1 > JS index.js > ...
// Import libraries
const express = require('express');
const bodyParser = require('body-parser');
const mysql = require('mysql2');
const mustacheExpress = require('mustache-express');

// Initialise objects and declare constants
const app = express();
const webPort = 8088;

const db = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "",
    database: "Movies"
});

db.connect((err) => {
    if (err) {
        throw err;
    }
    console.log("Connected to database");
});

app.engine('html', mustacheExpress());
app.set('view engine', 'html');
app.set('views','./templates');
app.use(bodyParser.urlencoded({ extended: true }));
```

This file is used to connect the node express js app to mysql db.



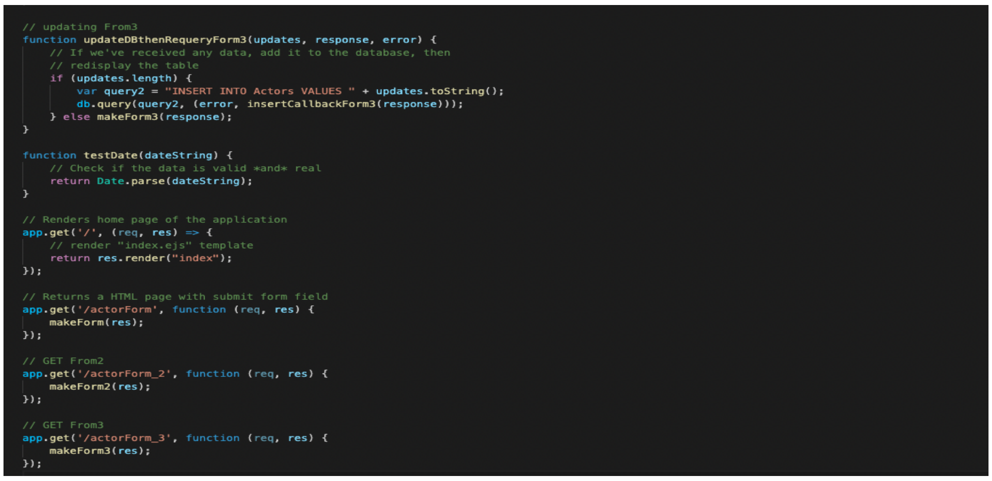
```
const db = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "",
    database: "Movies"
});
```

Database querying:
Once the user press on the submit button then GET request is passed to the server along with the submit query string to the end user while the query string is accessible by using the request.query object and then a submit form is carry out by using the SQL query:



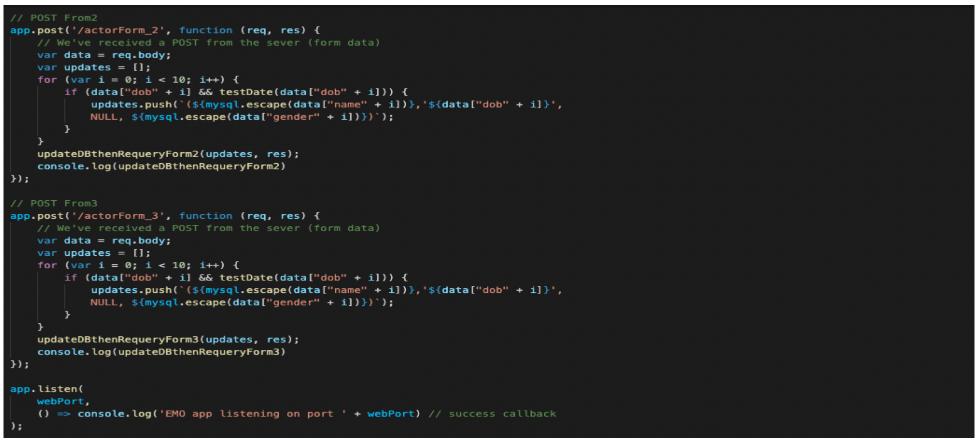
```
function makeForm(response) {
    // Run a query to get unmatched actors
    var query = `SELECT DISTINCT Actor1
                 FROM Locations LEFT JOIN Actors ON Actor1=ENName
                 WHERE ENName IS NULL
                 LIMIT 10;`;
    db.query(query, templateRenderer(response));
}

// make From2
function makeForm2(response) {
    // Run a query to get unmatched actors
    var query1 = `SELECT DISTINCT Actor2
                 FROM Locations LEFT JOIN Actors ON Actor2=ENName
                 WHERE ENName IS NULL
                 LIMIT 10;`;
    db.query(query1, templateRendererForm2(response));
}

// make From3
function makeForm3(response) {
    // Run a query to get unmatched actors
    var query3 = `SELECT DISTINCT Actor1, Actor3
                 FROM Locations LEFT JOIN Actors ON Actor3=ENName
                 WHERE ENName IS NULL
                 LIMIT 10;`;
    db.query(query3, templateRendererForm3(response));
}
```

Updating DB:
Able to access either by sending a GET request to the end user.



```
// updating From3
function updateDBthenRequeryForm3(updates, response, error) {
    // If we've received any data, add it to the database, then
    // redisplay the table
    if (updates.length) {
        var query2 = "INSERT INTO Actors VALUES " + updates.toString();
        db.query(query2, (error, insertCallbackForm3(response)));
    } else makeForm3(response);
}

function testDate(dateString) {
    // Check if the data is valid *and* real
    return Date.parse(dateString);
}

// Renders home page of the application
app.get('/', (req, res) => {
    // render "index.ejs" template
    return res.render("index");
});

// Returns a HTML page with submit form field
app.get('/actorForm', function (req, res) {
    makeForm(res);
});

// GET From2
app.get('/actorForm_2', function (req, res) {
    makeForm2(res);
});

// GET From3
app.get('/actorForm_3', function (req, res) {
    makeForm3(res);
});
```

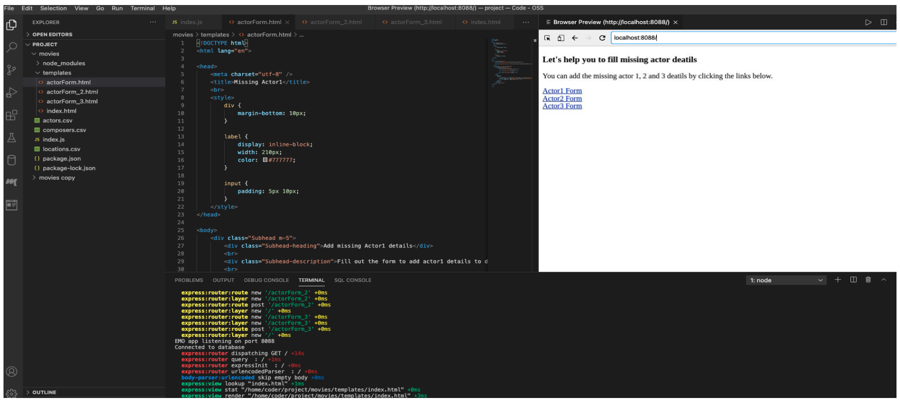
On the server the data is accessible among the request object body parameter and then converted into an array, to pass it to the mysql query function and get request.body as parameter which returns an array which include values for various columns of the Actors table.



```
// POST From2
app.post('/actorForm_2', function (req, res) {
    // We've received a POST from the sever (form data)
    var data = req.body;
    var updates = [];
    for (var i = 0; i < 10; i++) {
        if (data["dob" + i] && testDate(data["dob" + i])) {
            updates.push(`(${mysql.escape(data["name" + i])},'${data["dob" + i]}',
            NULL, ${mysql.escape(data["gender" + i])})`);
        }
    }
    updateDBthenRequeryForm2(updates, res);
    console.log(updateDBthenRequeryForm2)
});

// POST From3
app.post('/actorForm_3', function (req, res) {
    // We've received a POST from the sever (form data)
    var data = req.body;
    var updates = [];
    for (var i = 0; i < 10; i++) {
        if (data["dob" + i] && testDate(data["dob" + i])) {
            updates.push(`(${mysql.escape(data["name" + i])},'${data["dob" + i]}',
            NULL, ${mysql.escape(data["gender" + i])})`);
        }
    }
    updateDBthenRequeryForm3(updates, res);
    console.log(updateDBthenRequeryForm3)
});

app.listen(
    webPort,
    () => console.log('EMO app listening on port ' + webPort) // success callback
);
```

Forms Missing Actors page

Each form displays the first 10 unmatched entries in the database. You need to provide a date of birth in the first box in each row, and gender in the second and then will be added them to the database.



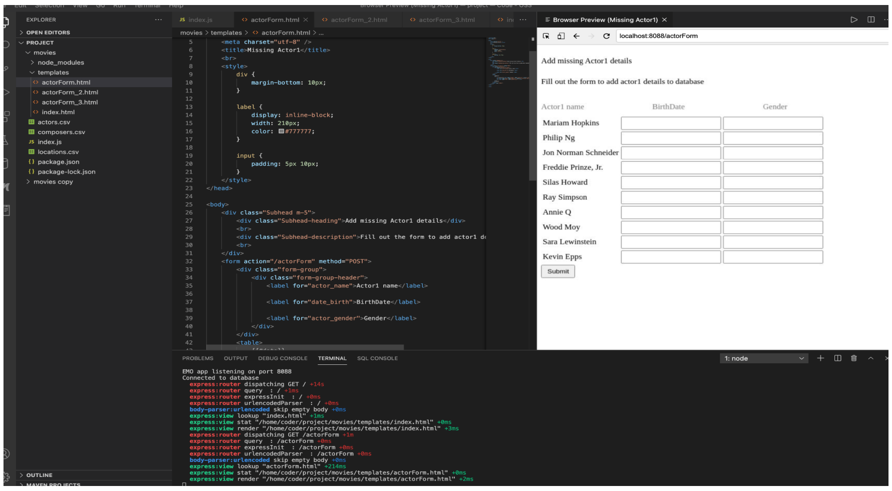
```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8" />
    <title>Missing Actor1</title>
    <br>
    <style>
        div {
            margin-bottom: 10px;
        }

        label {
            display: inline-block;
            width: 210px;
            color: #777777;
        }

        input {
            padding: 5px 10px;
        }
    </style>
</head>

<body>
    <div class="Subhead m-5">
        <div class="Subhead-heading">Add missing Actor1 details</div>
        <br>
        <div class="Subhead-description">Fill out the form to add actor1 details to d
        <br>
```

```
  express:router:route new '/actorForm_2' +0ms
  express:router:layer new '/actorForm_2' +0ms
  express:router:route post '/actorForm_2' +0ms
  express:router:layer new '/' +0ms
  express:router:route new '/actorForm_3' +0ms
  express:router:layer new '/actorForm_3' +0ms
  express:router:route post '/actorForm_3' +0ms
  express:router:layer new '/' +0ms
EMO app listening on port 8088
Connected to database
  express:router dispatching GET / +14s
  express:router query  : / +1ms
  express:router expressInit  : / +0ms
  express:router urlencodedParser  : / +0ms
  body-parser:urlencoded skip empty body +0ms
  express:view lookup "index.html" +1ms
  express:view stat "/home/coder/project/movies/templates/index.html" +0ms
  express:view render "/home/coder/project/movies/templates/index.html" +3ms
```

Let's help you to fill missing actor deatils

You can add the missing actor 1, 2 and 3 deatils by clicking the links below.

Actor1 Form
Actor2 Form
Actor3 Form



```
    <meta charset="utf-8" />
    <title>Missing Actor1</title>
    <br>
    <style>
        div {
            margin-bottom: 10px;
        }

        label {
            display: inline-block;
            width: 210px;
            color: #777777;
        }

        input {
            padding: 5px 10px;
        }
    </style>
</head>

<body>
    <div class="Subhead m-5">
        <div class="Subhead-heading">Add missing Actor1 details</div>
        <br>
        <div class="Subhead-description">Fill out the form to add actor1 de
        <br>
    </div>
    <form action="/actorForm" method="POST">
        <div class="form-group">
            <div class="form-group-header">
                <label for="actor_name">Actor1 name</label>

                <label for="date_birth">BirthDate</label>

                <label for="actor_gender">Gender</label>
            </div>
        </div>
        <table>
```

```
EMO app listening on port 8088
Connected to database
  express:router dispatching GET / +14s
  express:router query  : / +1ms
  express:router expressInit  : / +0ms
  express:router urlencodedParser  : / +0ms
  body-parser:urlencoded skip empty body +0ms
  express:view lookup "index.html" +1ms
  express:view stat "/home/coder/project/movies/templates/index.html" +0ms
  express:view render "/home/coder/project/movies/templates/index.html" +3ms
  express:router dispatching GET /actorForm +1m
  express:router query  : /actorForm +0ms
  express:router expressInit  : /actorForm +0ms
  express:router urlencodedParser  : /actorForm +0ms
  body-parser:urlencoded skip empty body +0ms
  express:view lookup "actorForm.html" +214ms
  express:view stat "/home/coder/project/movies/templates/actorForm.html" +0ms
  express:view render "/home/coder/project/movies/templates/actorForm.html" +2ms
```

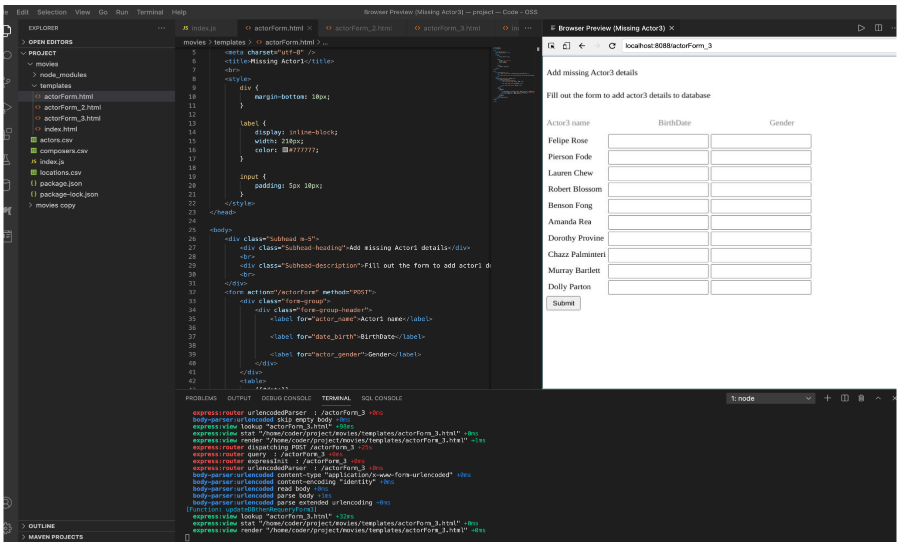
localhost:8088/actorForm

Add missing Actor1 details

Fill out the form to add actor1 details to database

Actor1 name	BirthDate	Gender
Mariam Hopkins		
Philip Ng		
Jon Norman Schneider		
Freddie Prinze, Jr.		
Silas Howard		
Ray Simpson		
Annie Q		
Wood Moy		
Sara Lewinstein		
Kevin Epps		

Submit



Browser Preview (Missing Actor3) — project — Code - OSS

```
    <meta charset="utf-8" />
    <title>Missing Actor1</title>
    <br>
    <style>
        div {
            margin-bottom: 10px;
        }

        label {
            display: inline-block;
            width: 210px;
            color: #777777;
        }

        input {
            padding: 5px 10px;
        }
    </style>
</head>

<body>
    <div class="Subhead m-5">
        <div class="Subhead-heading">Add missing Actor1 details</div>
        <br>
        <div class="Subhead-description">Fill out the form to add actor1 d
        <br>
    </div>
    <form action="/actorForm" method="POST">
        <div class="form-group">
            <div class="form-group-header">
                <label for="actor_name">Actor1 name</label>

                <label for="date_birth">BirthDate</label>

                <label for="actor_gender">Gender</label>
            </div>
        </div>
        <table>
```

```
  express:router urlencodedParser  : /actorForm_3 +0ms
  body-parser:urlencoded skip empty body +0ms
  express:view lookup "actorForm_3.html" +98ms
  express:view stat "/home/coder/project/movies/templates/actorForm_3.html" +0ms
  express:view render "/home/coder/project/movies/templates/actorForm_3.html" +1ms
  express:router dispatching POST /actorForm_3 +25s
  express:router query  : /actorForm_3 +0ms
  express:router expressInit  : /actorForm_3 +0ms
  express:router urlencodedParser  : /actorForm_3 +0ms
  body-parser:urlencoded content-type "application/x-www-form-urlencoded" +0ms
  body-parser:urlencoded content-encoding "identity" +0ms
  body-parser:urlencoded read body +0ms
  body-parser:urlencoded parse body +1ms
  body-parser:urlencoded parse extended urlencoding +0ms
[Function: updateDBthenRequeryForm3]
  express:view lookup "actorForm_3.html" +32ms
  express:view stat "/home/coder/project/movies/templates/actorForm_3.html" +0ms
  express:view render "/home/coder/project/movies/templates/actorForm_3.html" +0ms
```

localhost:8088/actorForm_3

Add missing Actor3 details

Fill out the form to add actor3 details to database

Actor3 name	BirthDate	Gender
Felipe Rose		
Pierson Fode		
Lauren Chew		
Robert Blossom		
Benson Fong		
Amanda Rea		
Dorothy Provine		
Chazz Palminteri		
Murray Barlett		
Dolly Parton		

Submit

Reference:

https://www.wikidata.org/wiki/Wikidata:Licensing

https://rapidapi.com/search/licenses%20on%20publicly