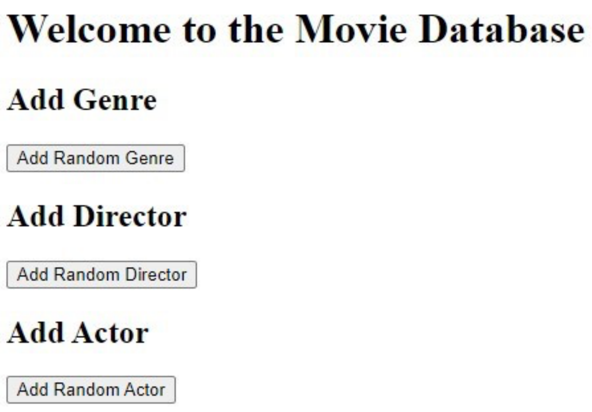
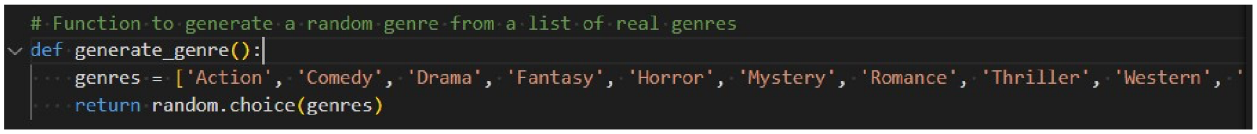
Using the same Movies database that you created in Part 1 of the individual project to do the following:

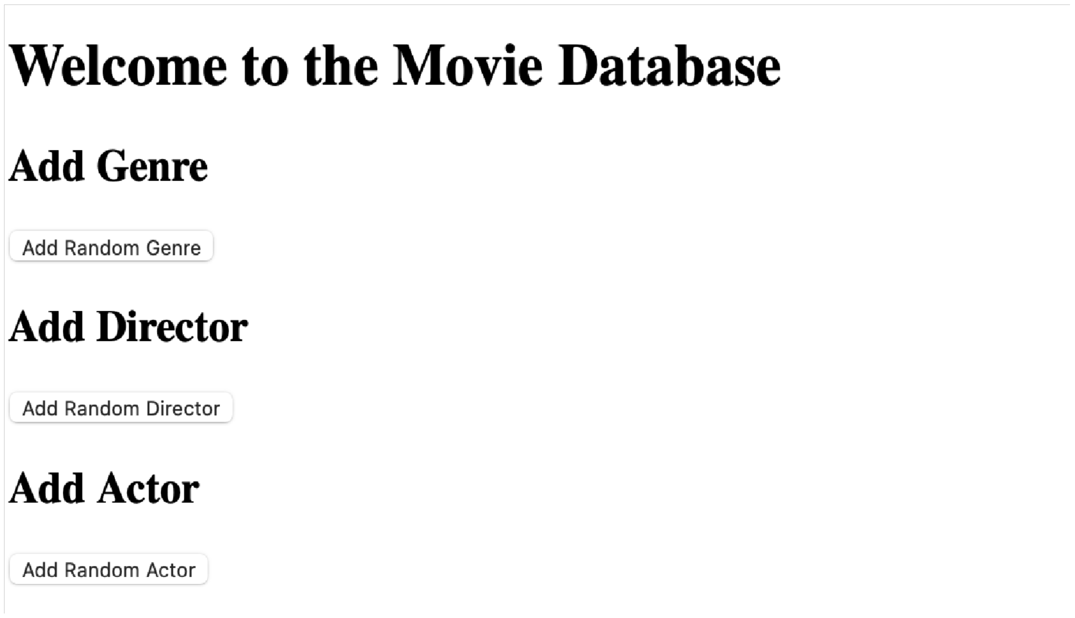
1. Add new funcCons to generate genres, directors, and actors. Below is one example to generate random records. In addiCon to that, I provided a Python funcCon to generate random genres. Remember that you need to "*import random*" to use the random package. You can do something similar with directors and actors.





```
# Function to generate a random genre from a list of real genres
def generate_genre():
    genres = ['Action', 'Comedy', 'Drama', 'Fantasy', 'Horror', 'Mystery', 'Romance', 'Thriller', 'Western', '
    return random.choice(genres)
```

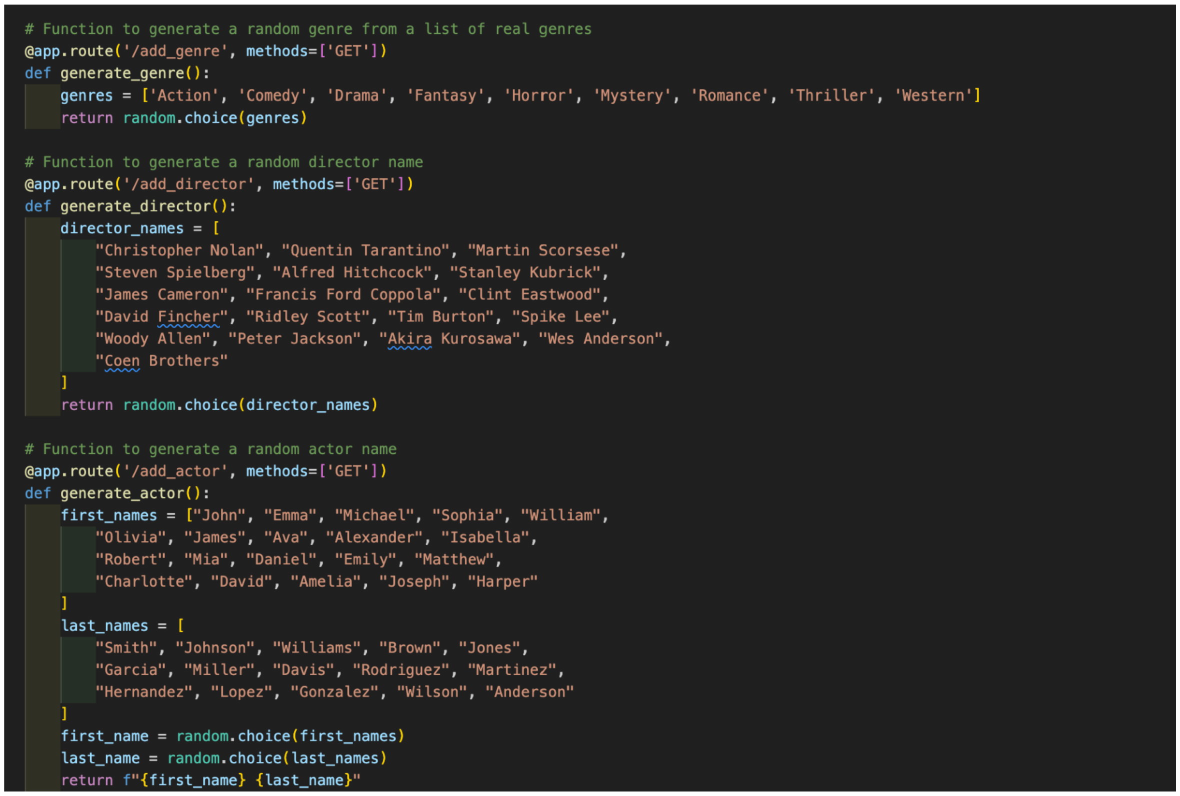
Ans:


Welcome to the Movie Database

Add Genre

Add Director

Add Actor

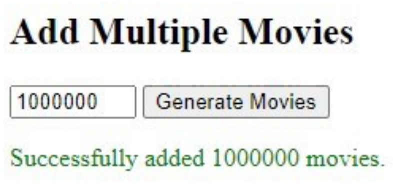

```
# Function to generate a random genre from a list of real genres
@app.route('/add_genre', methods=['GET'])
def generate_genre():
    genres = ['Action', 'Comedy', 'Drama', 'Fantasy', 'Horror', 'Mystery', 'Romance', 'Thriller', 'Western']
    return random.choice(genres)

# Function to generate a random director name
@app.route('/add_director', methods=['GET'])
def generate_director():
    director_names = [
        "Christopher Nolan", "Quentin Tarantino", "Martin Scorsese",
        "Steven Spielberg", "Alfred Hitchcock", "Stanley Kubrick",
        "James Cameron", "Francis Ford Coppola", "Clint Eastwood",
        "David Fincher", "Ridley Scott", "Tim Burton", "Spike Lee",
        "Woody Allen", "Peter Jackson", "Akira Kurosawa", "Wes Anderson",
        "Coen Brothers"
    ]
    return random.choice(director_names)

# Function to generate a random actor name
@app.route('/add_actor', methods=['GET'])
def generate_actor():
    first_names = ["John", "Emma", "Michael", "Sophia", "William",
        "Olivia", "James", "Ava", "Alexander", "Isabella",
        "Robert", "Mia", "Daniel", "Emily", "Matthew",
        "Charlotte", "David", "Amelia", "Joseph", "Harper"
    ]
    last_names = [
        "Smith", "Johnson", "Williams", "Brown", "Jones",
        "Garcia", "Miller", "Davis", "Rodriguez", "Martinez",
        "Hernandez", "Lopez", "Gonzalez", "Wilson", "Anderson"
    ]
    first_name = random.choice(first_names)
    last_name = random.choice(last_names)
    return f"{first_name} {last_name}"
```

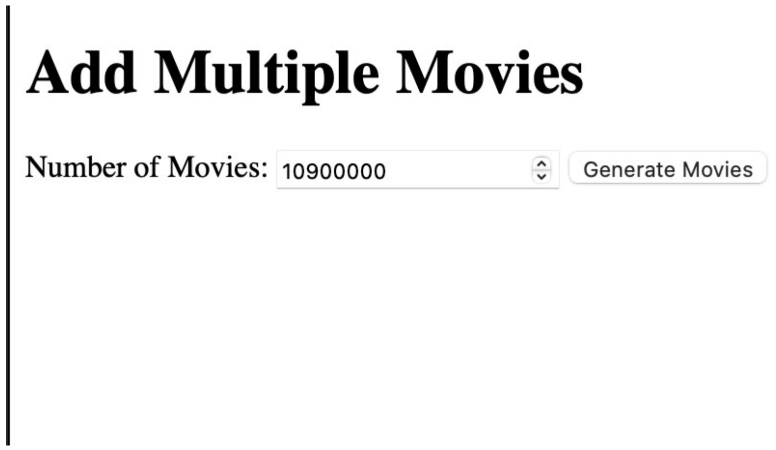
2. Add a new funcCon to generate movies. Depending on your database design, you need to design something like the below image. The idea is to generate millions of movies! I provided some code snippets in Python (helper.py file) to give you an idea. Start with 10K, 50K, 100K,

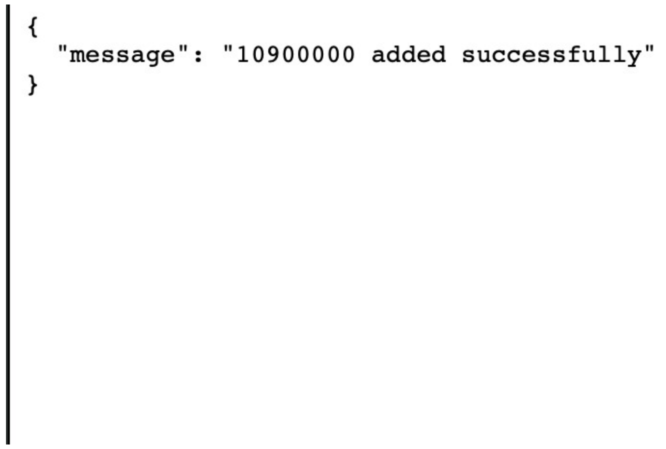
500K, 1M, 5M, 10M. You may use the browser developer tools to see the status of the request, especially for large values.



Successfully added 1000000 movies.

Ans:

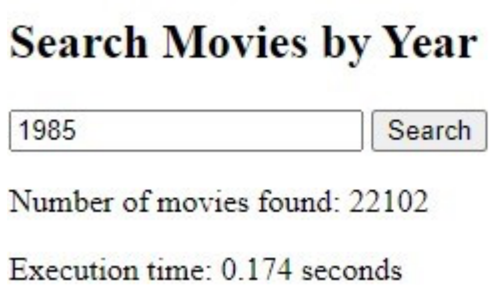




```
{
  "message": "10900000 added successfully"
}
```

3. The reason we need to generate millions of random movies is to search the movies database before and aWer adding an index. You are going to search for the following queries and to create the needed funcCons/design to support that.

a. Search Movies by year *(the number of movies and the execu5on 5me may be different than what you will get, and that is OK)*

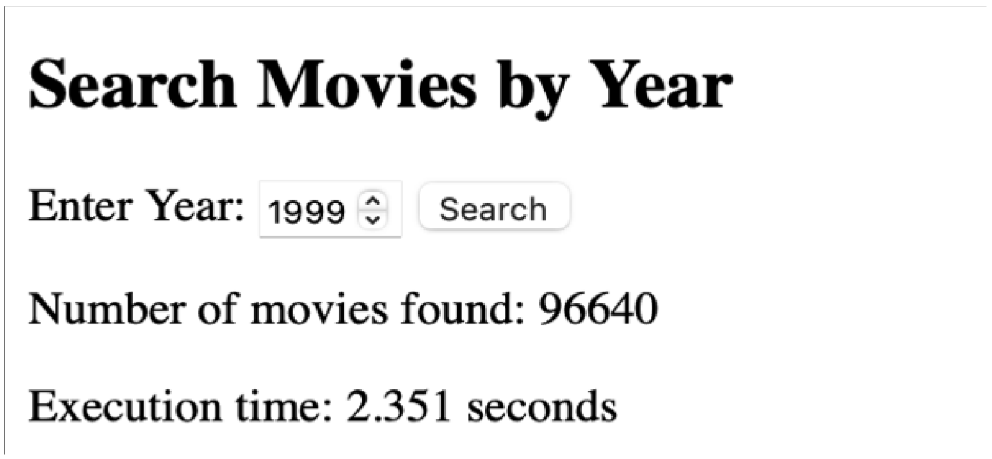

Search Movies by Year

1985 Search

Number of movies found: 22102

Execution time: 0.174 seconds

Ans:

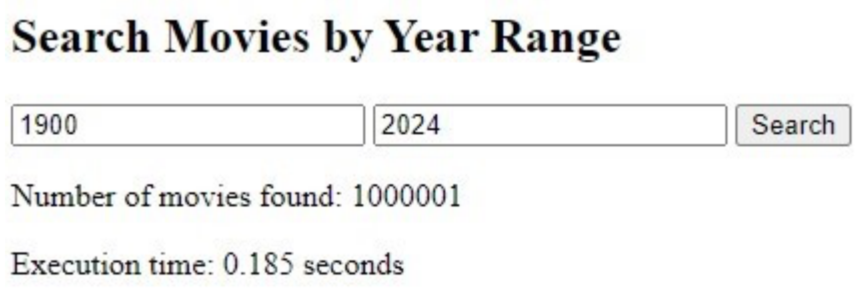

Search Movies by Year

Enter Year: 1999 Search

Number of movies found: 96640

Execution time: 2.351 seconds

b. Search Movies by year Range *(the number of movies and the execu5on 5me may be different than what you will get, and that is OK)*

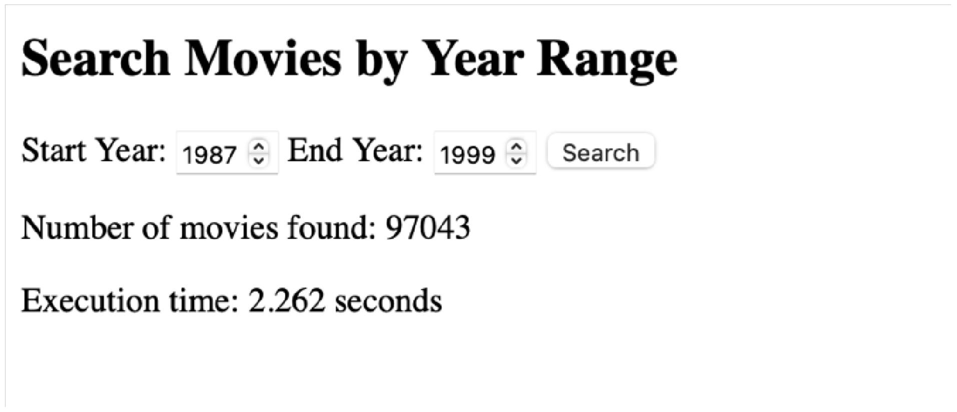

Search Movies by Year Range

1900 2024

Number of movies found: 1000001

Execution time: 0.185 seconds

Ans:


Search Movies by Year Range

Start Year: 1987 End Year: 1999

Number of movies found: 97043

Execution time: 2.262 seconds

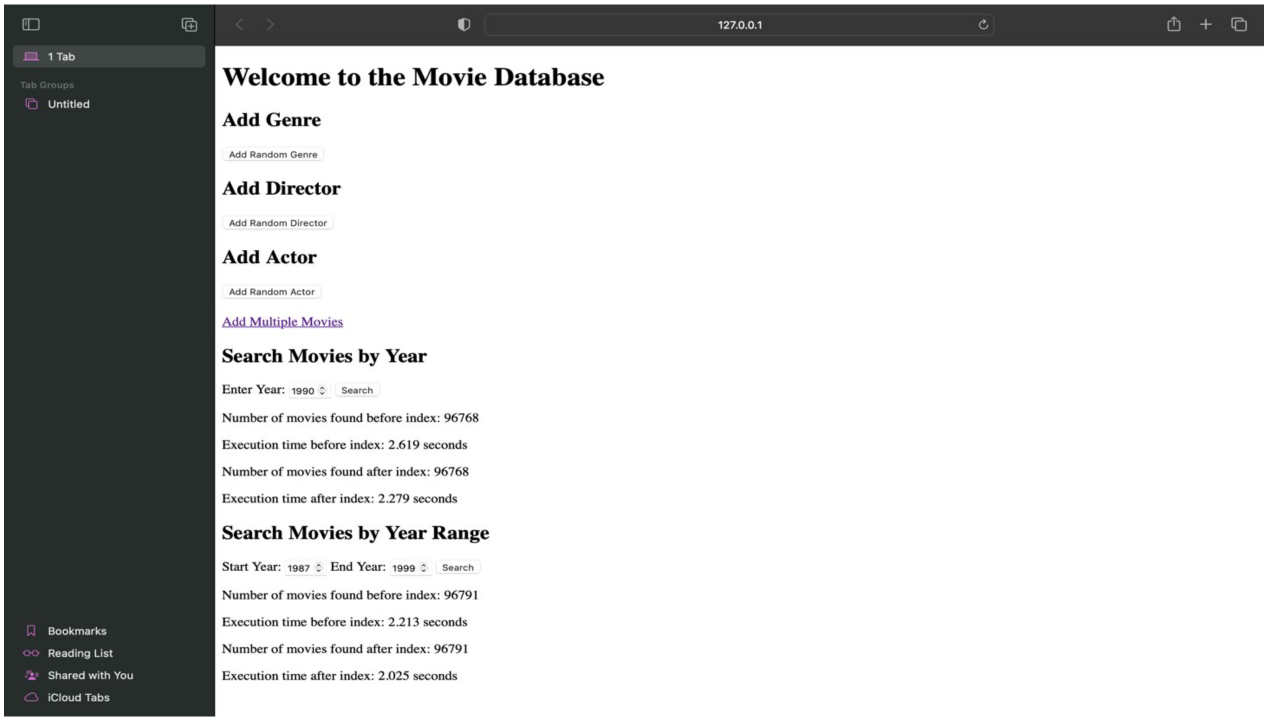
In my case I added 10M random movies to see the search Cme in seconds, and the size of the database grew to 1GB. You must keep adding movies unCl you see that the Cme to search for movies by year is at least **2 seconds**. Depending on the computer/HDD you have, it may be enough to add less than a million movies to see that the search is taking at least 2 seconds. **Again, add movies progressively.**

c. Add the suitable index to reduce the search Cme. Report the Cme before adding the index and the Cme aWer adding the index for every query as shown in the table below.

Remember to add enough movies, so the :me before adding the index is at least 2 seconds.

Ans: I created an index on the release year column on movie table.

SQL Query	Before the index	AAer the index
Search Movies by Year	2.619seconds	2.279seconds
Searcy Movies by Year Range	2.213seconds	2.025 seconds



Welcome to the Movie Database

Add Genre

Add Director

Add Actor

Add Multiple Movies

Search Movies by Year

Enter Year: 1990

Number of movies found before index: 96768

Execution time before index: 2.619 seconds

Number of movies found after index: 96768

Execution time after index: 2.279 seconds

Search Movies by Year Range

Start Year: 1987 End Year: 1999

Number of movies found before index: 96791

Execution time before index: 2.213 seconds

Number of movies found after index: 96791

Execution time after index: 2.025 seconds

Submit the following:

- A PDF report with clear screenshots and SQL scripts for every quesCon above.
- The source code compressed as a ZIP file.
- Record a video up to 5 minutes showing the following details:
    - You have many movies (select count (*) from movies).
    - ○ Search by movies is working and show the count of movies returned and the execuCon Cme.
    - Search movies by year range and show the count of movies returned and the execuCon Cme.