PHASE REPORT

Group members:

1. Jahnavi Chowdary Tharigopula, 112078393
2. Vamsikrishna Kodumuru Meesala, 112006882

Environmental Information:

1. Database - MySQL
2. Front-End - HTML, PHP
3. Language of Front End - HTML, PHP
4. Server - Apache2 Ubuntu

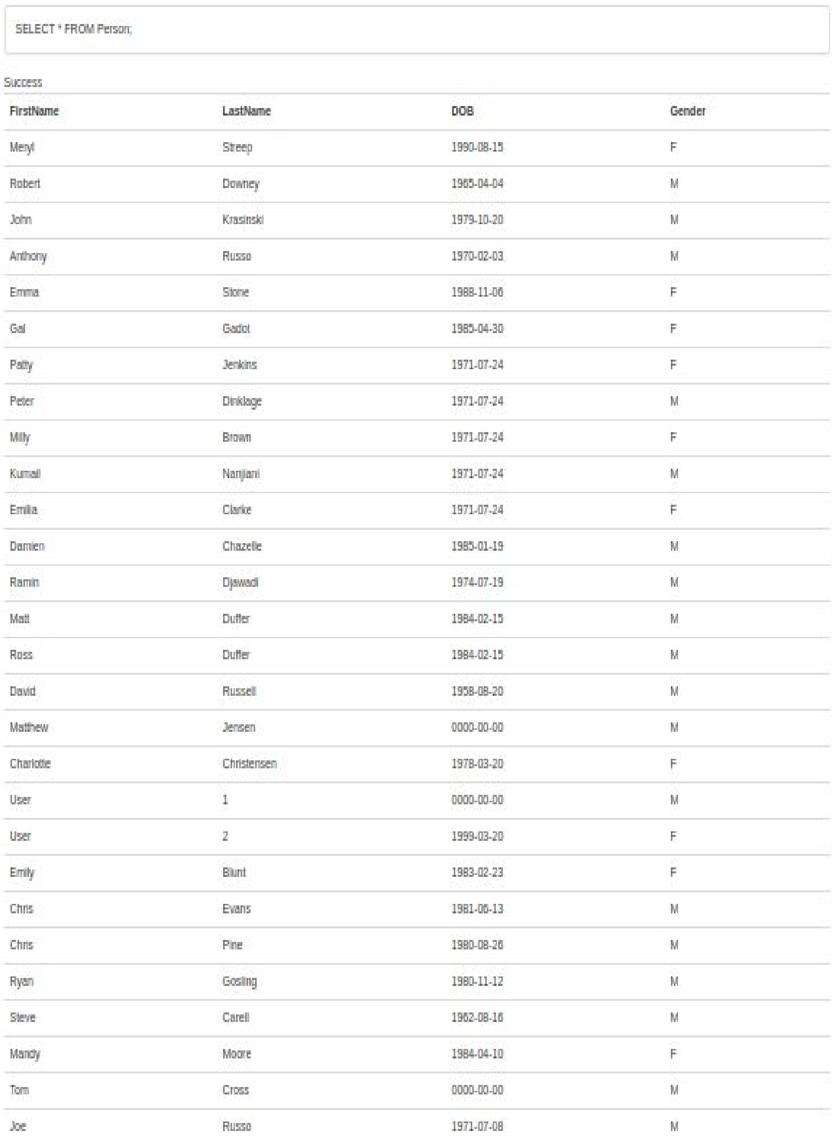
Transactions Description:

The transactions that we have implemented can broadly be divided into 3 parts.

1. Transactions to 'View the existing Database'
2. Transactions to 'Search the Database'
3. Transactions to allow User to 'Add new entries into the Database'

Transactions to 'View the existing Database'

1. View Persons
    a. This allows us to view all the People present in the Database.
    b. This is done by selecting all entires from
        i. The 'Person' Table - which contains information about every single Person present in the database
    c. SQL Code:
    ```
    SELECT * FROM Person
    ```
    d. Result:

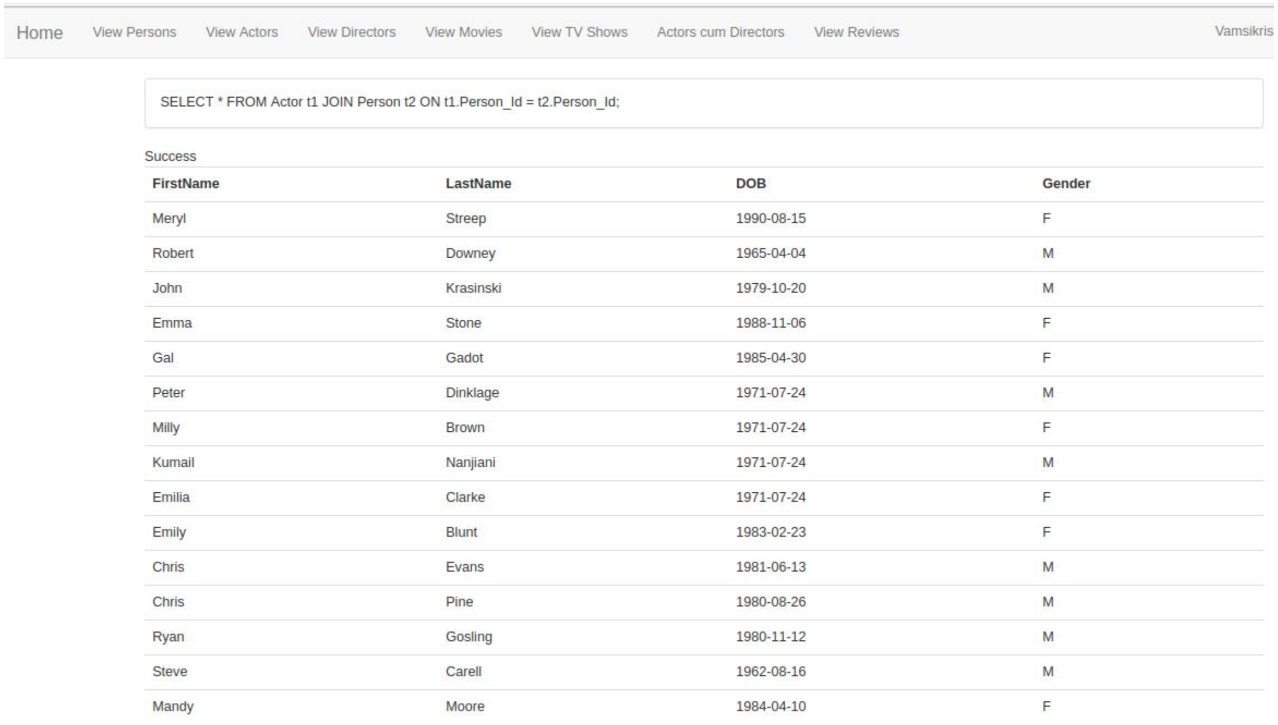


```
SELECT * FROM Person;
```

Success

FirstName	LastName	DOB	Gender
Meryl	Streep	1990-08-15	F
Robert	Downey	1965-04-04	M
John	Krasinski	1979-10-20	M
Anthony	Russo	1970-02-03	M
Emma	Stone	1988-11-06	F
Gal	Gadot	1985-04-30	F
Patty	Jenkins	1971-07-24	F
Peter	Dinklage	1971-07-24	M
Milly	Brown	1971-07-24	F
Kumail	Nanjiani	1971-07-24	M
Emilia	Clarke	1971-07-24	F
Damien	Chazelle	1985-01-19	M
Ramin	Djawadi	1974-07-19	M
Matt	Duffer	1984-02-15	M
Ross	Duffer	1984-02-15	M
David	Russell	1958-08-20	M
Matthew	Jensen	0000-00-00	M
Charlotte	Christensen	1978-03-20	F
User	1	0000-00-00	M
User	2	1999-03-20	F
Emily	Blunt	1983-02-23	F
Chris	Evans	1981-06-13	M
Chris	Pine	1980-08-26	M
Ryan	Gosling	1980-11-12	M
Steve	Carell	1962-08-16	M
Mandy	Moore	1984-04-10	F
Tom	Cross	0000-00-00	M
Joe	Russo	1971-07-08	M

2. View Actors
    a. This allows us to view all the Actors present in the Database.
    b. This is done by performing a Join on
        i. The ‘Actor’ table - which contains information about who among the Persons are Actors,
        ii. The ‘Person’ Table - which contains information about every single Person present in the database
    c. SQL Code:
       SELECT * FROM Actor t1 JOIN Person t2
       ON t1.Person_Id = t2.Person_Id;
    d. Result:

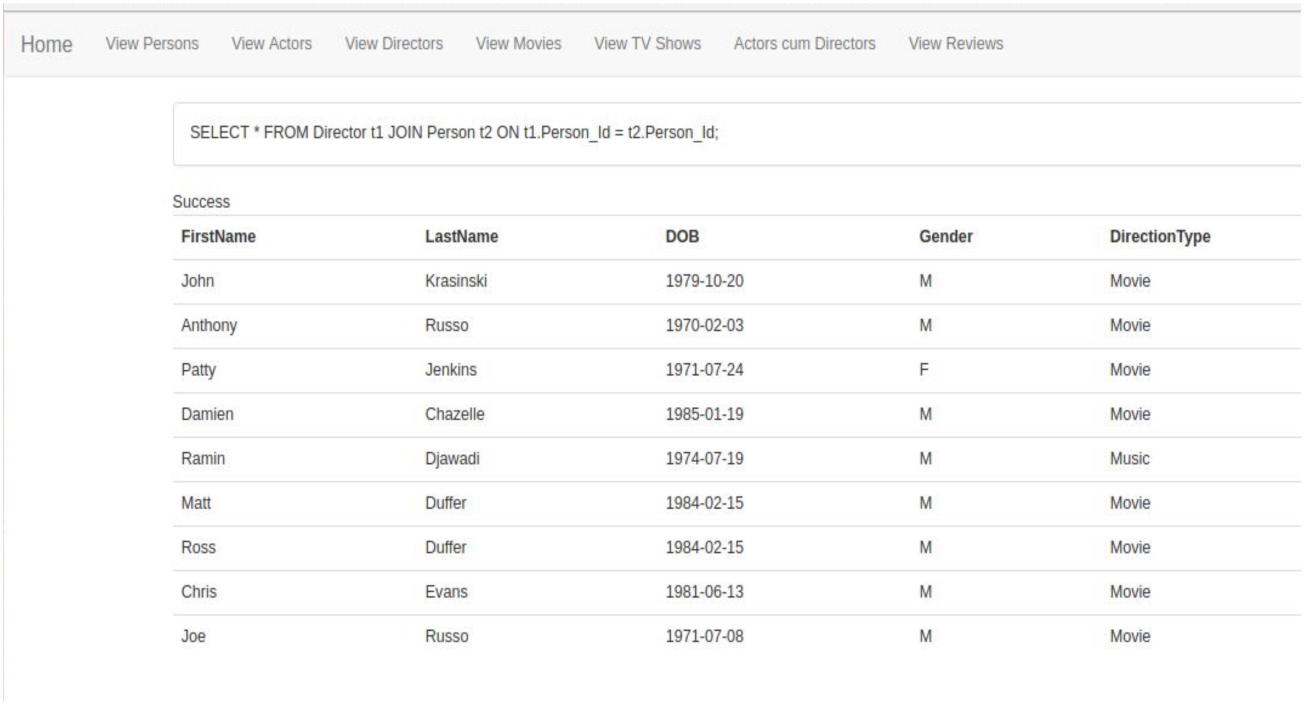


```
SELECT * FROM Actor t1 JOIN Person t2 ON t1.Person_Id = t2.Person_Id;
```

Success

FirstName	LastName	DOB	Gender
Meryl	Streep	1990-08-15	F
Robert	Downey	1965-04-04	M
John	Krasinski	1979-10-20	M
Emma	Stone	1988-11-06	F
Gal	Gadot	1985-04-30	F
Peter	Dinklage	1971-07-24	M
Milly	Brown	1971-07-24	F
Kumail	Nanjiani	1971-07-24	M
Emilia	Clarke	1971-07-24	F
Emily	Blunt	1983-02-23	F
Chris	Evans	1981-06-13	M
Chris	Pine	1980-08-26	M
Ryan	Gosling	1980-11-12	M
Steve	Carell	1962-08-16	M
Mandy	Moore	1984-04-10	F

3. View Directors
   a. This allows us to view all the Directors present in the Database.
   b. This is done by performing a Join on
      i. The ‘Director’ table - which contains information about who among the Persons are Directors,
      ii. The ‘Person’ Table - which contains information about every single Person present in the database
   c. SQL Code:
      SELECT * FROM Director t1 JOIN Person t2
      ON t1.Person_Id = t2.Person_Id;
   d. Result:

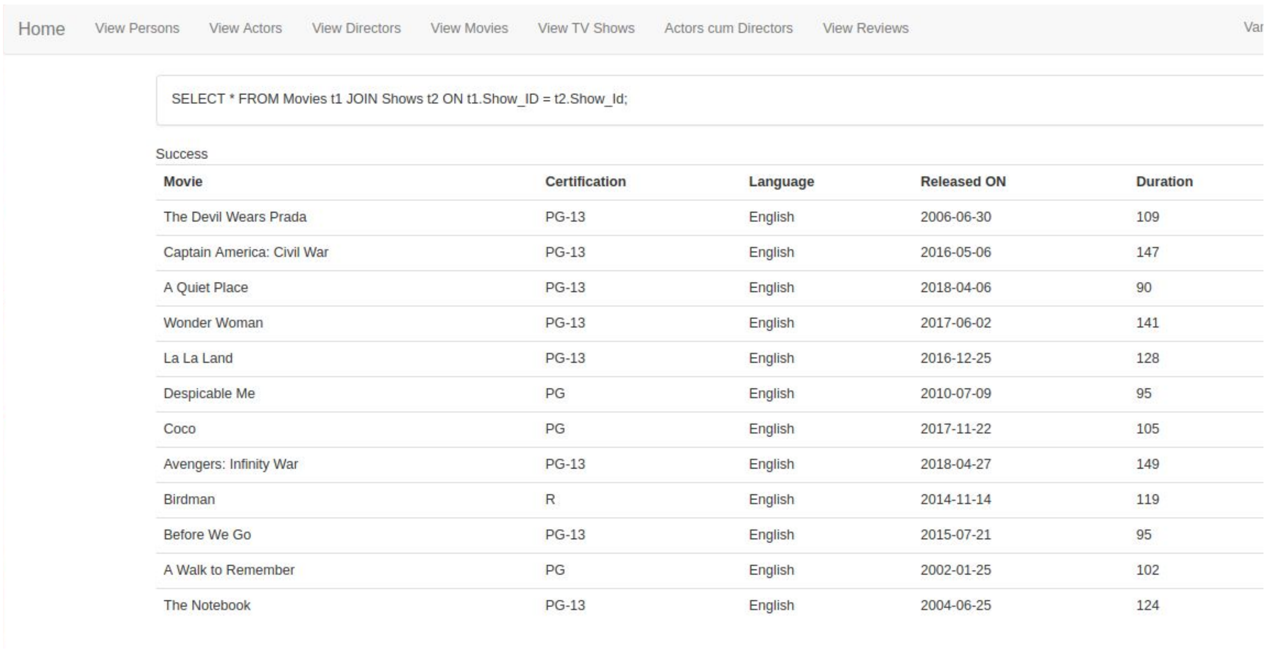


```
SELECT * FROM Director t1 JOIN Person t2 ON t1.Person_Id = t2.Person_Id;
```

Success

FirstName	LastName	DOB	Gender	DirectionType
John	Krasinski	1979-10-20	M	Movie
Anthony	Russo	1970-02-03	M	Movie
Patty	Jenkins	1971-07-24	F	Movie
Damien	Chazelle	1985-01-19	M	Movie
Ramin	Djawadi	1974-07-19	M	Music
Matt	Duffer	1984-02-15	M	Movie
Ross	Duffer	1984-02-15	M	Movie
Chris	Evans	1981-06-13	M	Movie
Joe	Russo	1971-07-08	M	Movie

4. View Movies
    a. This allows us to view all the Movies present in the Database.
    b. This is done by performing a Join on
        i. The ‘Movie’ table - which contains information about which among the Shows are Movies,
        ii. The ‘Shows’ Table - which contains information about every single Show present in the database, Show includes both Movies + TV Series
    c. SQL Code:
    ```
    SELECT * FROM Movies t1 JOIN Shows t2
    ON t1.Show_ID = t2.Show_Id;
    ```
    d. Result:

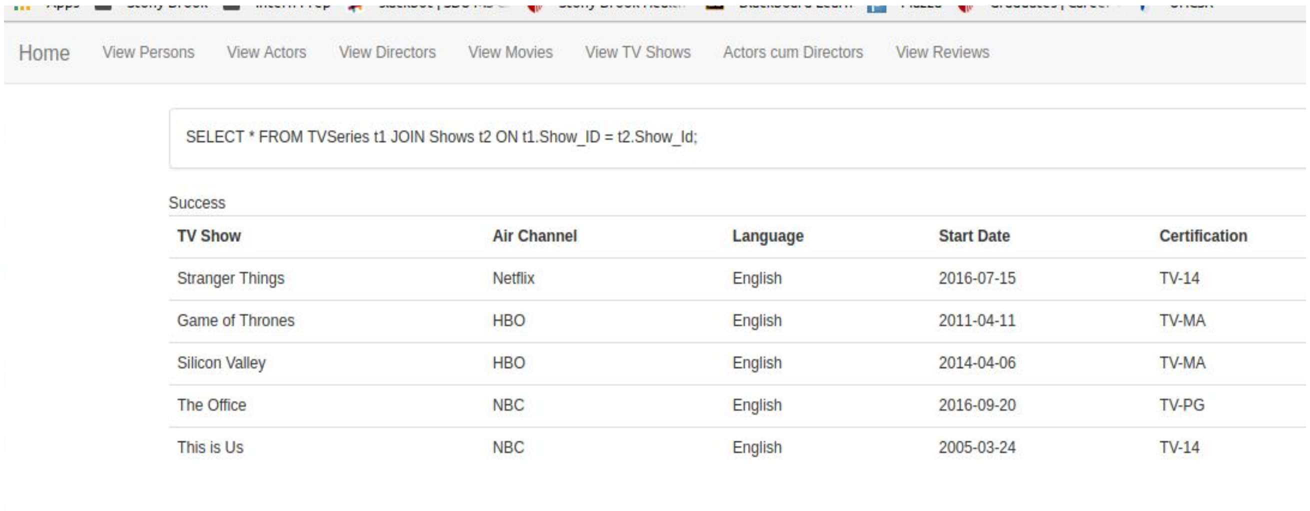


```
SELECT * FROM Movies t1 JOIN Shows t2 ON t1.Show_ID = t2.Show_Id;
```

Success

Movie	Certification	Language	Released ON	Duration
The Devil Wears Prada	PG-13	English	2006-06-30	109
Captain America: Civil War	PG-13	English	2016-05-06	147
A Quiet Place	PG-13	English	2018-04-06	90
Wonder Woman	PG-13	English	2017-06-02	141
La La Land	PG-13	English	2016-12-25	128
Despicable Me	PG	English	2010-07-09	95
Coco	PG	English	2017-11-22	105
Avengers: Infinity War	PG-13	English	2018-04-27	149
Birdman	R	English	2014-11-14	119
Before We Go	PG-13	English	2015-07-21	95
A Walk to Remember	PG	English	2002-01-25	102
The Notebook	PG-13	English	2004-06-25	124

5. View TV Shows
    a. This allows us to view all the TV Shows present in the Database.
    b. This is done by performing a Join on
        i. The ‘TV Series’ table - which contains information about which among the Shows are TV Series,
        ii. The ‘Shows’ Table - which contains information about every single Show present in the database, Show includes both Movies + TV Series
    c. SQL Code:
    SELECT * FROM TVSeries t1 JOIN Shows t2
    ON t1.Show_ID = t2.Show_Id;
    d. Result:

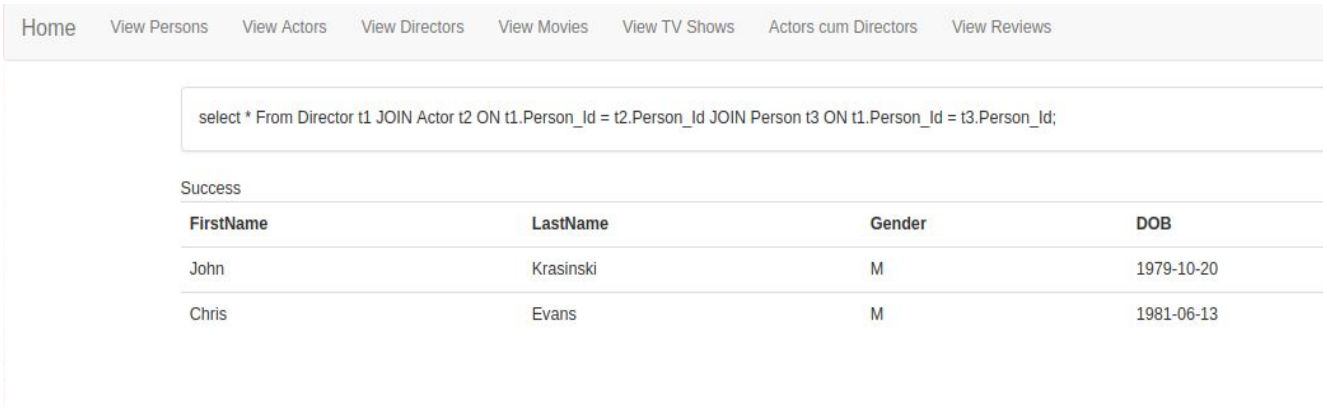


```
SELECT * FROM TVSeries t1 JOIN Shows t2 ON t1.Show_ID = t2.Show_Id;
```

Success

TV Show	Air Channel	Language	Start Date	Certification
Stranger Things	Netflix	English	2016-07-15	TV-14
Game of Thrones	HBO	English	2011-04-11	TV-MA
Silicon Valley	HBO	English	2014-04-06	TV-MA
The Office	NBC	English	2016-09-20	TV-PG
This is Us	NBC	English	2005-03-24	TV-14

6. Search People who are both Actors and Directors
    a. This allows us to view all the people who are both Actors and Directors.
    b. This is done by performing a Join on
        i. The ‘Director’ table - which contains information about who among the Persons are Directors,
        ii. The ‘Actor’ table - which contains information about who among the Persons are Actors,
        iii. The ‘Person’ Table - which contains information about every single Person present in the database.
    c. SQL Code:
    SELECT * From Director t1 JOIN Actor t2
    ON t1.Person_Id = t2.Person_Id JOIN Person t3
    ON t1.Person_Id = t3.Person_Id;
    d. Result:
        i. The results gives 2 entries
            1. John Krasinski acted and directed ‘A Quiet Place’.
            2. Chris Evans acted in ‘Captain America:Civil War’, ‘Avengers Infinity War’ and directed ‘Before We Go’.

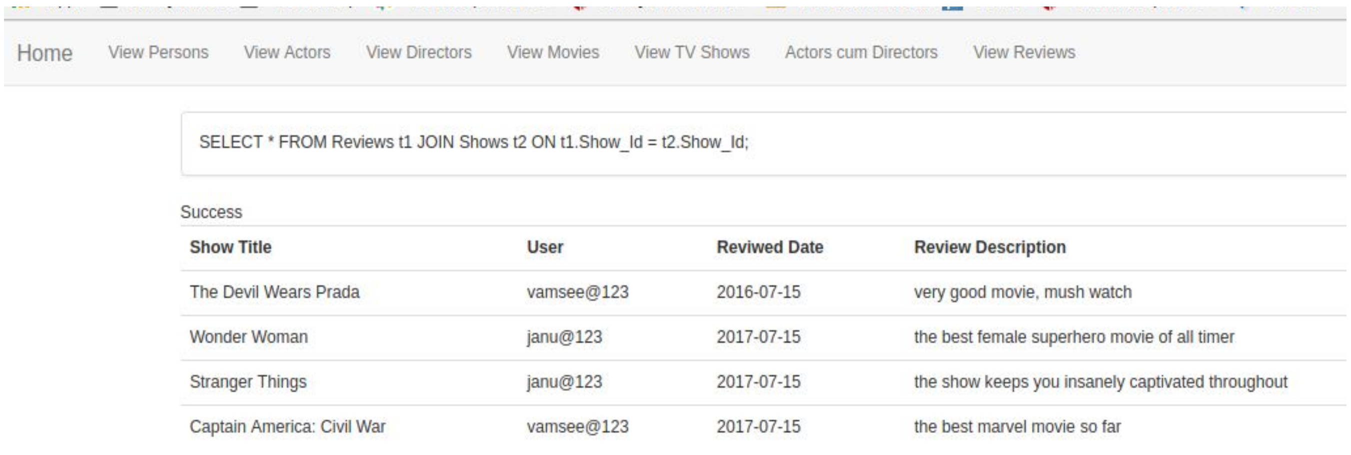


```
select * From Director t1 JOIN Actor t2 ON t1.Person_Id = t2.Person_Id JOIN Person t3 ON t1.Person_Id = t3.Person_Id;
```

Success

FirstName	LastName	Gender	DOB
John	Krasinski	M	1979-10-20
Chris	Evans	M	1981-06-13

7. View Reviews
    a. This allows us to view all the Reviews provided by the Users for the Shows existing in the Database.
    b. This is done by performing a Join on
        i. The ‘Reviews’ table - which contains information about the Reviews provided by the Users,
        ii. The ‘Shows’ Table - which contains information about every single Show present in the database, Show includes both Movies + TV Series
    c. SQL Code:
    SELECT * FROM Reviews t1 JOIN Shows t2
    ON t1.Show_Id = t2.Show_Id;
    d. Result:



```
SELECT * FROM Reviews t1 JOIN Shows t2 ON t1.Show_Id = t2.Show_Id;
```

Success

Show Title	User	Reviwed Date	Review Description
The Devil Wears Prada	vamsee@123	2016-07-15	very good movie, mush watch
Wonder Woman	janu@123	2017-07-15	the best female superhero movie of all timer
Stranger Things	janu@123	2017-07-15	the show keeps you insanely captivated throughout
Captain America: Civil War	vamsee@123	2017-07-15	the best marvel movie so far

Transactions to 'Search the Database'

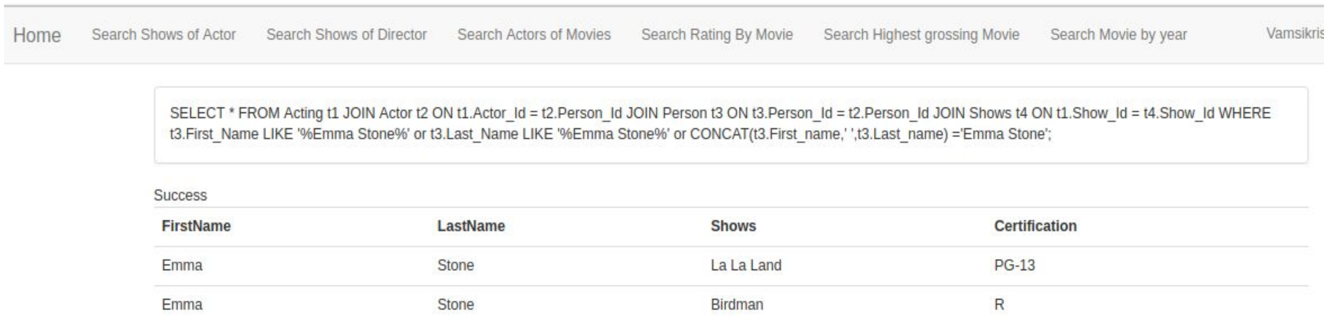
1. Search Shows of Actors
    a. This allows us to Search the Database for all the Shows (Movies + TV Series) of a particular actor. The input for the Actor is provided by the User.
    b. This is done by performing a Join on
        i. The 'Acting' Table - which contains the Shows and Actor pairs, i.e which actors acted in which Shows,
        ii. The 'Actor' table - which contains information about who among the Persons are Actors,
        iii. The 'Person' Table - which contains information about every single Person present in the database,
        iv. The 'Shows' Table - which contains information about every single Show present in the database, Show includes both Movies + TV Series.
    c. SQL Code:

```
SELECT * FROM Acting t1 JOIN Actor t2
ON t1.Actor_Id = t2.Person_Id JOIN Person t3
ON t3.Person_Id = t2.Person_Id JOIN Shows t4
ON t1.Show_Id = t4.Show_Id
WHERE t3.First_Name LIKE '%Robert%' or t3.Last_Name LIKE '%Robert%';
```

The LIKE operator here is used to retrieve information even if someone types in half the name, say 'Rober' instead of 'Robert'. This could have been a typo, hence we have considered this scenario.

Also, the search can be done based on just First Name, just Last Name, or both included.

    d. Result:
        i. Searching Shows of 'Emma Stone'

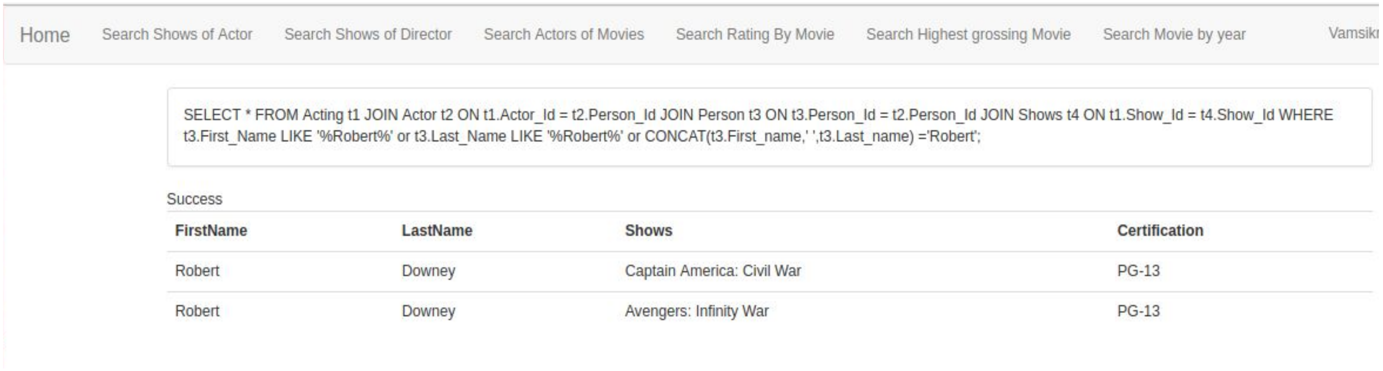


```
SELECT * FROM Acting t1 JOIN Actor t2 ON t1.Actor_Id = t2.Person_Id JOIN Person t3 ON t3.Person_Id = t2.Person_Id JOIN Shows t4 ON t1.Show_Id = t4.Show_Id WHERE
t3.First_Name LIKE '%Emma Stone%' or t3.Last_Name LIKE '%Emma Stone%' or CONCAT(t3.First_name,' ',t3.Last_name) ='Emma Stone';
```

Success

FirstName	LastName	Shows	Certification
Emma	Stone	La La Land	PG-13
Emma	Stone	Birdman	R

ii. Searching Shows of ‘Rober’

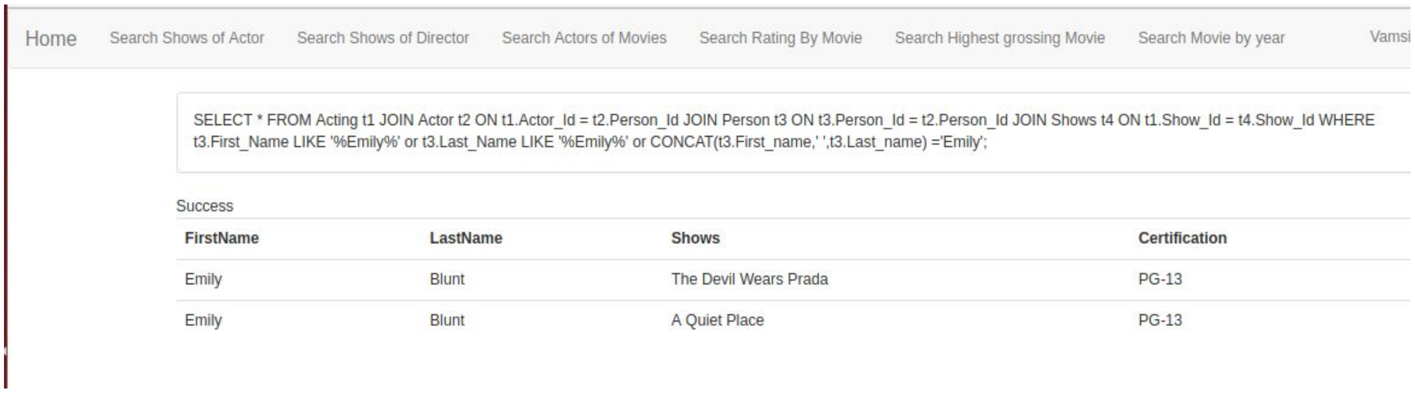


```
SELECT * FROM Acting t1 JOIN Actor t2 ON t1.Actor_Id = t2.Person_Id JOIN Person t3 ON t3.Person_Id = t2.Person_Id JOIN Shows t4 ON t1.Show_Id = t4.Show_Id WHERE
t3.First_Name LIKE '%Robert%' or t3.Last_Name LIKE '%Robert%' or CONCAT(t3.First_name,' ',t3.Last_name) ='Robert';
```

Success

FirstName	LastName	Shows	Certification
Robert	Downey	Captain America: Civil War	PG-13
Robert	Downey	Avengers: Infinity War	PG-13

iii. Searching Shows of ‘Emily Blunt’

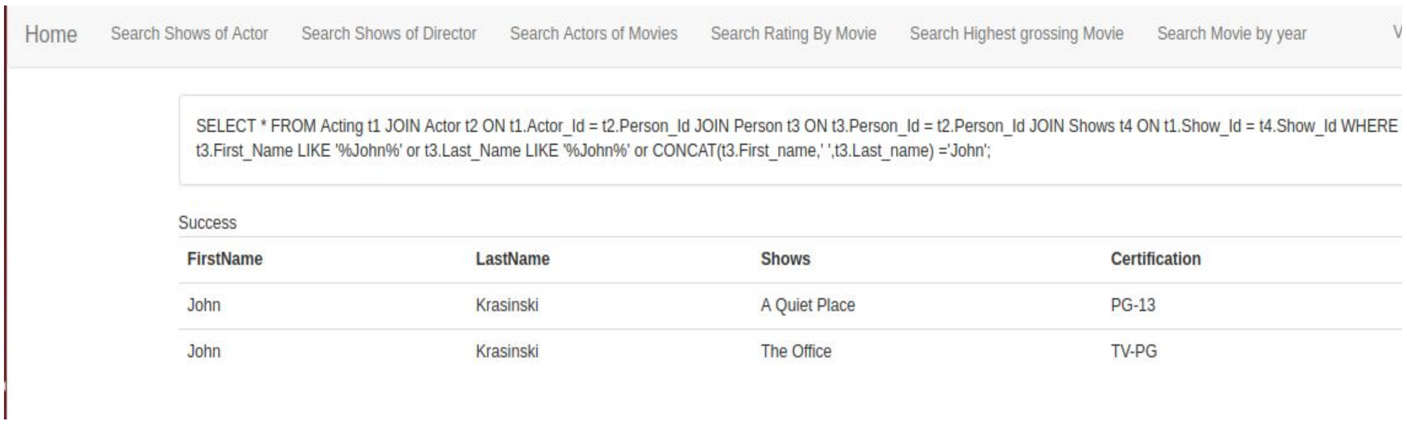


```
SELECT * FROM Acting t1 JOIN Actor t2 ON t1.Actor_Id = t2.Person_Id JOIN Person t3 ON t3.Person_Id = t2.Person_Id JOIN Shows t4 ON t1.Show_Id = t4.Show_Id WHERE
t3.First_Name LIKE '%Emily%' or t3.Last_Name LIKE '%Emily%' or CONCAT(t3.First_name,' ',t3.Last_name) ='Emily';
```

Success

FirstName	LastName	Shows	Certification
Emily	Blunt	The Devil Wears Prada	PG-13
Emily	Blunt	A Quiet Place	PG-13

iv. Searching Shows of ‘John’



```
SELECT * FROM Acting t1 JOIN Actor t2 ON t1.Actor_Id = t2.Person_Id JOIN Person t3 ON t3.Person_Id = t2.Person_Id JOIN Shows t4 ON t1.Show_Id = t4.Show_Id WHERE
t3.First_Name LIKE '%John%' or t3.Last_Name LIKE '%John%' or CONCAT(t3.First_name,' ',t3.Last_name) ='John';
```

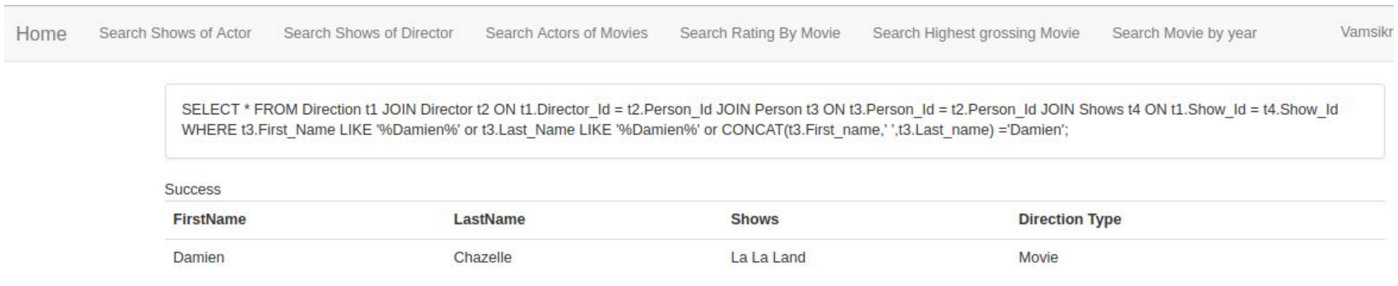
Success

FirstName	LastName	Shows	Certification
John	Krasinski	A Quiet Place	PG-13
John	Krasinski	The Office	TV-PG

2. Search Shows of Directors
    a. Similar to the above case, this allows us to Search the Database for all the Shows of a particular Director. The input for the Director is provided by the User.
    b. This is done by performing a Join on
        i. The ‘Directing’ Table - which contains the Shows and Director pairs, i.e which Directors directed in which Shows,
        ii. The ‘Director’ table - which contains information about who among the Persons are Directors,
        iii. The ‘Person’ Table - which contains information about every single Person present in the database,
        iv. The ‘Shows’ Table - which contains information about every single Show present in the database, Show includes both Movies + TV Series.
    c. SQL Code:

```
SELECT * FROM Direction t1 JOIN Director t2
ON t1.Director_Id = t2.Person_Id JOIN Person t3
ON t3.Person_Id = t2.Person_Id JOIN Shows t4
ON t1.Show_Id = t4.Show_Id
WHERE t3.First_Name LIKE '%John%' or t3.Last_Name LIKE '%John%';
```

    d. Result:
        i. Shows directed by Damien Chazelle

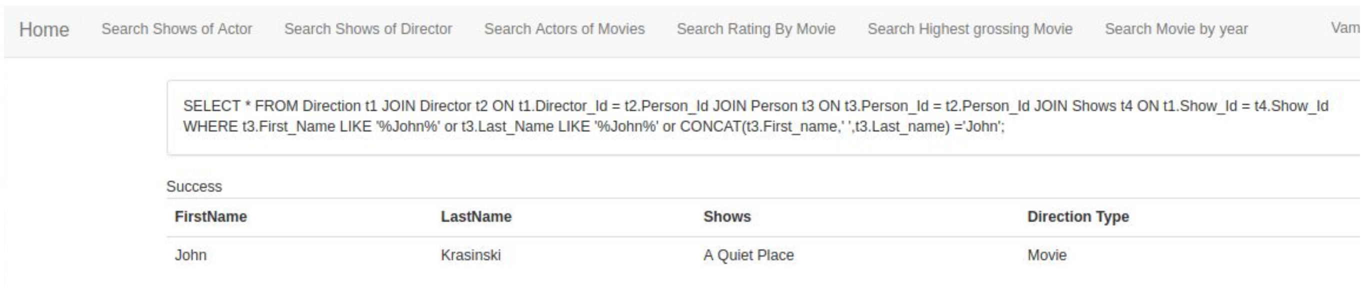


```
SELECT * FROM Direction t1 JOIN Director t2 ON t1.Director_Id = t2.Person_Id JOIN Person t3 ON t3.Person_Id = t2.Person_Id JOIN Shows t4 ON t1.Show_Id = t4.Show_Id
WHERE t3.First_Name LIKE '%Damien%' or t3.Last_Name LIKE '%Damien%' or CONCAT(t3.First_name,' ',t3.Last_name) ='Damien';
```

Success

FirstName	LastName	Shows	Direction Type
Damien	Chazelle	La La Land	Movie

        ii. Shows directed by John Krasinski



```
SELECT * FROM Direction t1 JOIN Director t2 ON t1.Director_Id = t2.Person_Id JOIN Person t3 ON t3.Person_Id = t2.Person_Id JOIN Shows t4 ON t1.Show_Id = t4.Show_Id
WHERE t3.First_Name LIKE '%John%' or t3.Last_Name LIKE '%John%' or CONCAT(t3.First_name,' ',t3.Last_name) ='John';
```

Success

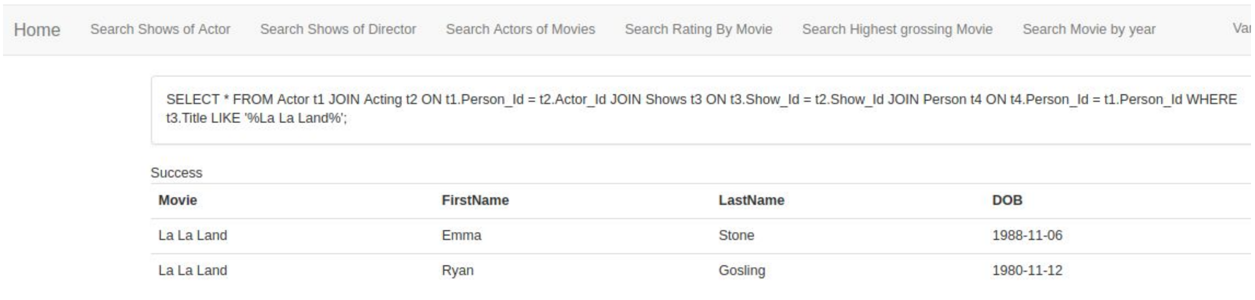
FirstName	LastName	Shows	Direction Type
John	Krasinski	A Quiet Place	Movie

3. Search Actors of Shows

a. This allows us to Search the Database for all the Actors of a particular Show. The input for the Show is provided by the User.
b. This is done by performing a Join on
    i. The ‘Actor’ table - which contains information about who among the Persons are Actors,
    ii. The ‘Acting’ Table - which contains the Shows and Actor pairs, i.e which actors acted in which Shows,
    iii. The ‘Shows’ Table - which contains information about every single Show present in the database, Show includes both Movies + TV Series,
    iv. The ‘Person’ Table - which contains information about every single Person present in the database.
c. SQL Code:

```
SELECT * FROM Actor t1 JOIN Acting t2
ON t1.Person_Id = t2.Actor_Id JOIN Shows t3
ON t3.Show_Id = t2.Show_Id JOIN Person t4
ON t4.Person_Id = t1.Person_Id
WHERE t3.Title LIKE '%La La land%';
```

d. Result:
    i. Actors in La La Land

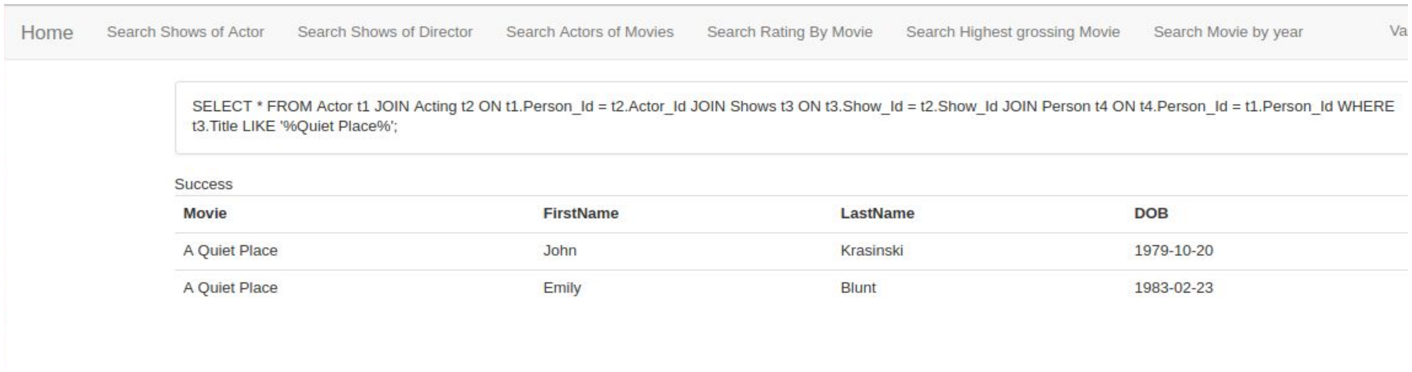


```
SELECT * FROM Actor t1 JOIN Acting t2 ON t1.Person_Id = t2.Actor_Id JOIN Shows t3 ON t3.Show_Id = t2.Show_Id JOIN Person t4 ON t4.Person_Id = t1.Person_Id WHERE
t3.Title LIKE '%La La Land%';
```

Success

Movie	FirstName	LastName	DOB
La La Land	Emma	Stone	1988-11-06
La La Land	Ryan	Gosling	1980-11-12

    ii. Actors in Quiet Place



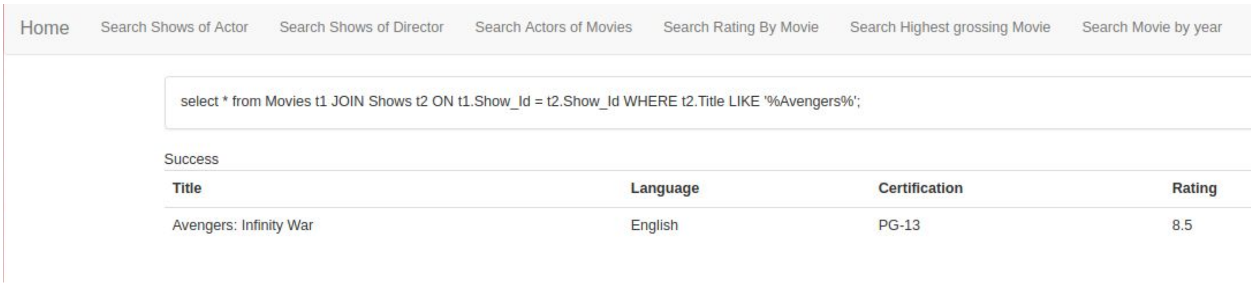
```
SELECT * FROM Actor t1 JOIN Acting t2 ON t1.Person_Id = t2.Actor_Id JOIN Shows t3 ON t3.Show_Id = t2.Show_Id JOIN Person t4 ON t4.Person_Id = t1.Person_Id WHERE
t3.Title LIKE '%Quiet Place%';
```

Success

Movie	FirstName	LastName	DOB
A Quiet Place	John	Krasinski	1979-10-20
A Quiet Place	Emily	Blunt	1983-02-23

4. Search Rating Of Movie
    a. This allows us to Search the Database for the Rating of a particular Movie. The input for the Movie is provided by the User.
    b. This is done by performing a Join on

        i. The ‘Movie’ table - which contains information about which among the Shows are Movies,
        ii. The ‘Shows’ Table - which contains information about every single Show present in the database, Show includes both Movies + TV Series.
    c. SQL Code:
       SELECT * FROM Movies t1 JOIN Shows t2
       ON t1.Show_Id = t2.Show_Id WHERE t2.Title LIKE '%Avengers%';
    d. Result:
        i. Rating of Avengers Infinity War

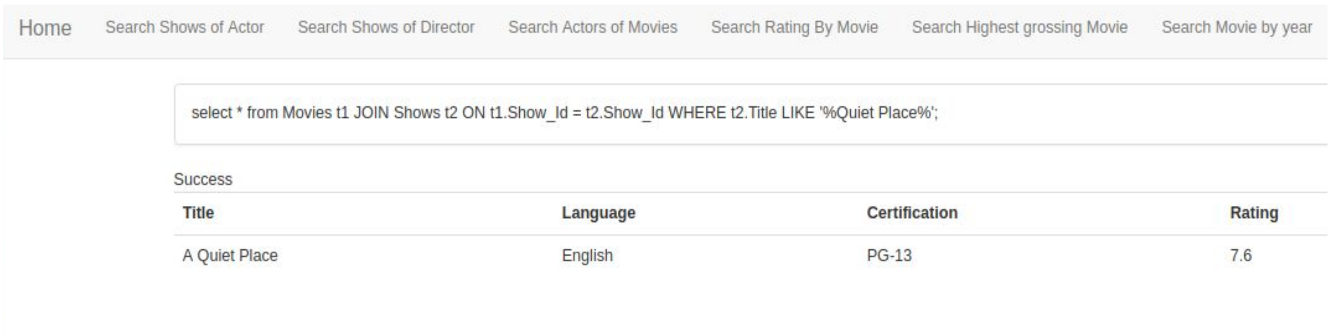


```
select * from Movies t1 JOIN Shows t2 ON t1.Show_Id = t2.Show_Id WHERE t2.Title LIKE '%Avengers%';
```

Success

Title	Language	Certification	Rating
Avengers: Infinity War	English	PG-13	8.5

        ii. Rating of A Quiet Place

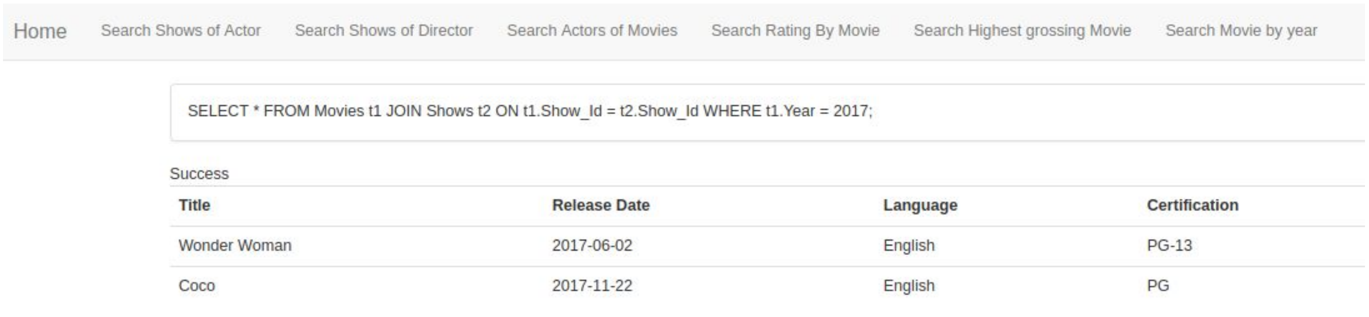


```
select * from Movies t1 JOIN Shows t2 ON t1.Show_Id = t2.Show_Id WHERE t2.Title LIKE '%Quiet Place%';
```

Success

Title	Language	Certification	Rating
A Quiet Place	English	PG-13	7.6

5. Search Movies by Year
    a. This allows us to Search the Database for the Movies that were released in a particular year. The input for the Year is provided by the User.
    b. This is done by performing a Join on
        i. The ‘Movie’ table - which contains information about which among the Shows are Movies,
        ii. The ‘Shows’ Table - which contains information about every single Show present in the database, Show includes both Movies + TV Series.
    c. SQL Code:
       SELECT * FROM Movies t1 JOIN Shows t2
       ON t1.Show_Id = t2.Show_Id
       WHERE t1.Year = 2017;
    d. Result:
        i. Movies released in 2017

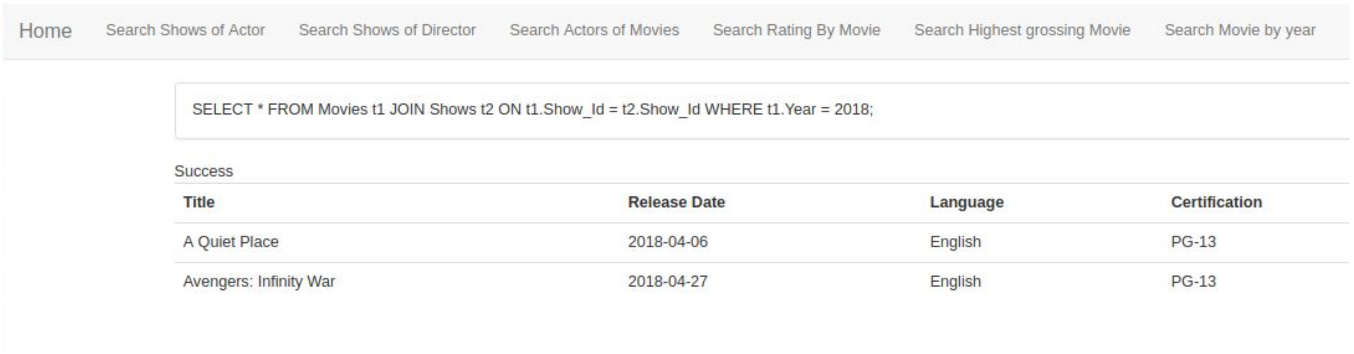


```
SELECT * FROM Movies t1 JOIN Shows t2 ON t1.Show_Id = t2.Show_Id WHERE t1.Year = 2017;
```

Success

Title	Release Date	Language	Certification
Wonder Woman	2017-06-02	English	PG-13
Coco	2017-11-22	English	PG

ii. Movies released in 2018

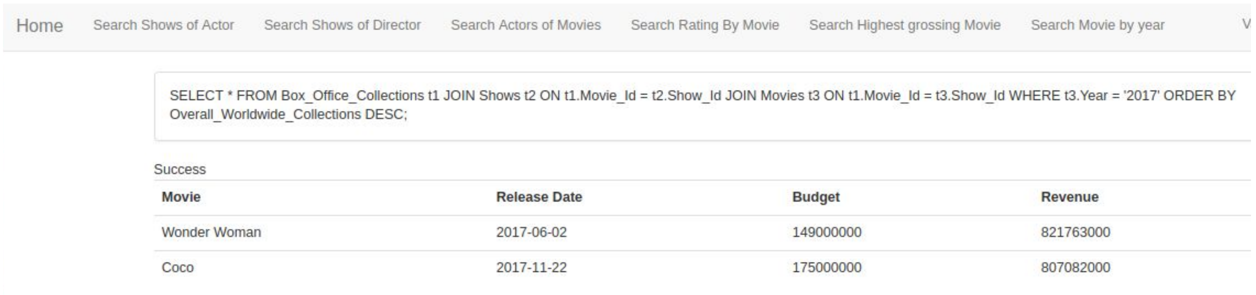


```
SELECT * FROM Movies t1 JOIN Shows t2 ON t1.Show_Id = t2.Show_Id WHERE t1.Year = 2018;
```

Success

Title	Release Date	Language	Certification
A Quiet Place	2018-04-06	English	PG-13
Avengers: Infinity War	2018-04-27	English	PG-13

6. Search Highest Grossing Movie by Year
   a. This allows us to Search the Database for the Highest Grossing Movies of a particular year. The input for the Year is provided by the User.
   b. This is done by performing a Join on
      i. The ‘Box_Office_Collections’ table - which contains information about the Box Office Collections of a particular Movie,
      ii. The ‘Shows’ Table - which contains information about every single Show present in the database, Show includes both Movies + TV Series,
      iii. The ‘Movie’ table - which contains information about which among the Shows are Movies.
   c. The result provides all the Movies released in that year ordered in descending order of their Box Office Collections, the first entry indicating the highest grossing movie of that year.
   d. SQL Code:
      ```
      SELECT * FROM Box_Office_Collections t1 JOIN Shows t2
      ON t1.Movie_Id = t2.Show_Id JOIN Movies t3
      ON t1.Movie_Id = t3.Show_Id WHERE t3.Year = '2017'
      ORDER BY Overall_Worldwide_Collections DESC;
      ```
   e. Result:
      i. Highest Grossing Movies in 2017

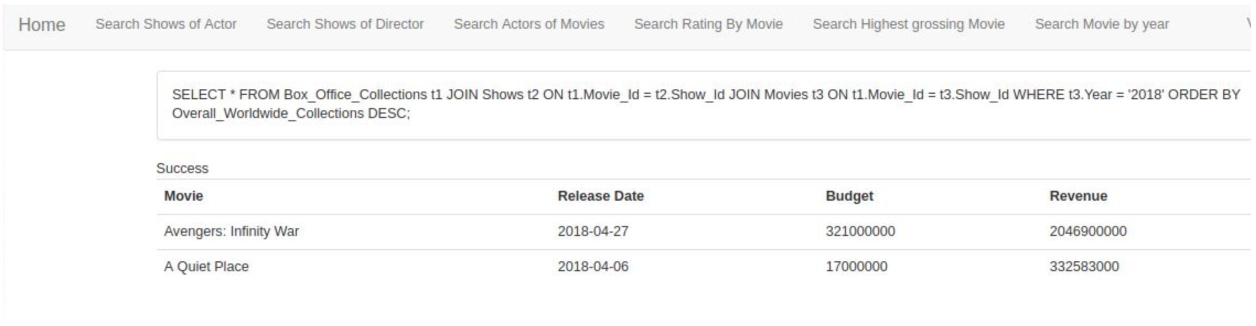


```
SELECT * FROM Box_Office_Collections t1 JOIN Shows t2 ON t1.Movie_Id = t2.Show_Id JOIN Movies t3 ON t1.Movie_Id = t3.Show_Id WHERE t3.Year = '2017' ORDER BY Overall_Worldwide_Collections DESC;
```

Success

Movie	Release Date	Budget	Revenue
Wonder Woman	2017-06-02	149000000	821763000
Coco	2017-11-22	175000000	807082000

ii. Highest Grossing Movies in 2018

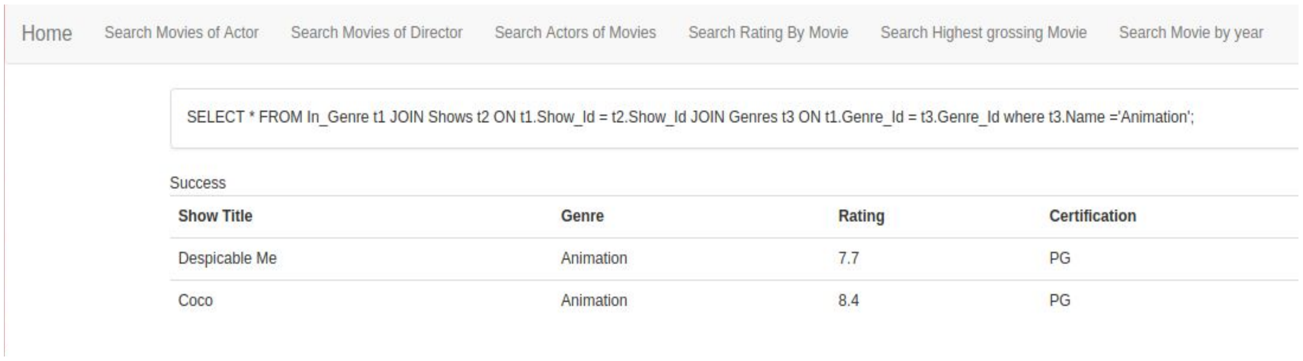


```
SELECT * FROM Box_Office_Collections t1 JOIN Shows t2 ON t1.Movie_Id = t2.Show_Id JOIN Movies t3 ON t1.Movie_Id = t3.Show_Id WHERE t3.Year = '2018' ORDER BY Overall_Worldwide_Collections DESC;
```

Success

Movie	Release Date	Budget	Revenue
Avengers: Infinity War	2018-04-27	321000000	2046900000
A Quiet Place	2018-04-06	17000000	332583000

7. Search Shows by Genre
   a. This allows us to Search the Database for the Shows of a particular genre. The input for the Genre is provided by the User.
   b. This is done by performing a Join on
      i. The 'In_Genre' Table - which contains the Genre and Show pairs, i.e which Show belongs to which Genre,
      ii. The 'Shows' Table - which contains information about every single Show present in the database, Show includes both Movies + TV Series,
      iii. The 'Genres' table - which contains information about all the available Genres.
   c. <u>SQL Code:</u>
      SELECT * from In_Genre t1 JOIN Shows t2
      ON t1.Show_Id = t2.Show_Id JOIN Genres t3
      ON t1.Genre_Id = t3.Genre_Id
      WHERE t3.Name = "Action"
   d. <u>Result:</u>
      i. Animated Movies

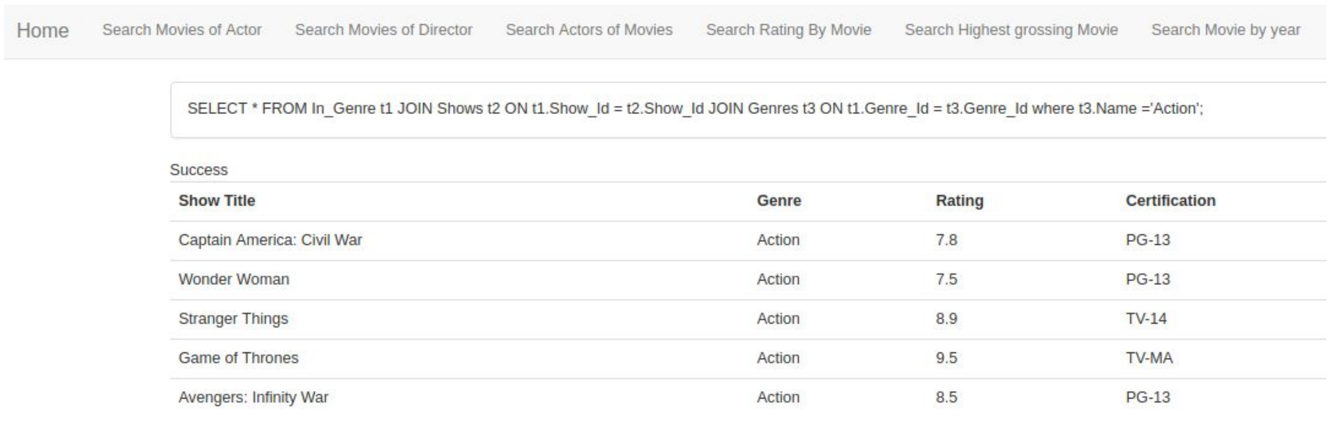


```
SELECT * FROM In_Genre t1 JOIN Shows t2 ON t1.Show_Id = t2.Show_Id JOIN Genres t3 ON t1.Genre_Id = t3.Genre_Id where t3.Name ='Animation';
```

Success

Show Title	Genre	Rating	Certification
Despicable Me	Animation	7.7	PG
Coco	Animation	8.4	PG

ii. Action Movies



```
SELECT * FROM In_Genre t1 JOIN Shows t2 ON t1.Show_Id = t2.Show_Id JOIN Genres t3 ON t1.Genre_Id = t3.Genre_Id where t3.Name ='Action';
```

Success

Show Title	Genre	Rating	Certification
Captain America: Civil War	Action	7.8	PG-13
Wonder Woman	Action	7.5	PG-13
Stranger Things	Action	8.9	TV-14
Game of Thrones	Action	9.5	TV-MA
Avengers: Infinity War	Action	8.5	PG-13

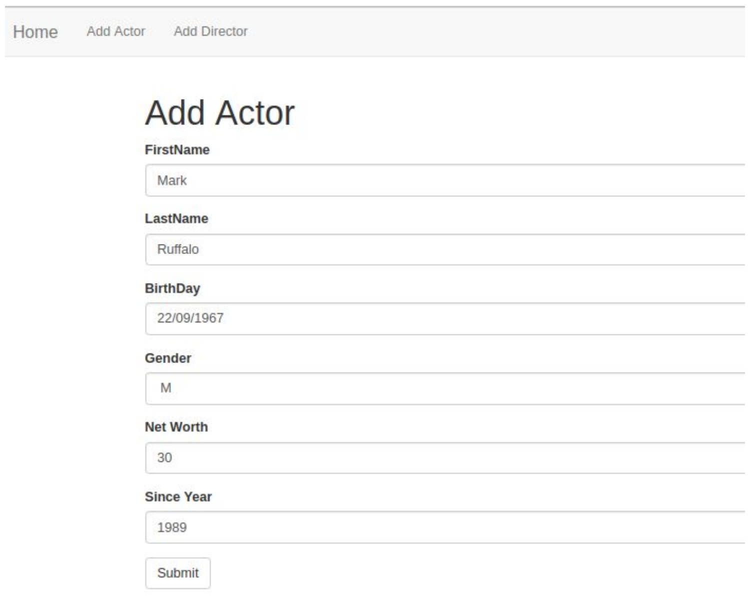
Transactions to allow User to 'Add new entries into the Database'

- Our model allows the User to modify the database by adding new Actors and Directors into the database.
- A request to add a new entry (actor/director) into the database should insert the actor/director in the Person table as well, as the Person table keeps track of all the people existing in the database.
  - If an Actor already exists in the Person and Actor table, and we want to add him as a Director as well, then we first check whether the actor exists in the Person table, and if it does, we insert the entry only in the director table, and vice versa.
  - Whereas if the Actor already exists in the Person and Actor table, and we want to add the same person again as an Actor, then this request will be Failed as the entry already exists in both the tables, and vice versa for the Director as well.
- The following details of the Actor are provided by the User.
  - First Name
  - Last Name
  - DOB
  - Gender
  - Net Worth
  - Working Since Year

- The result of the successful addition of the info in the Person and Actor/Director Tables can be verified by Viewing the Person and Actor/Director tables using the View Database functionality.

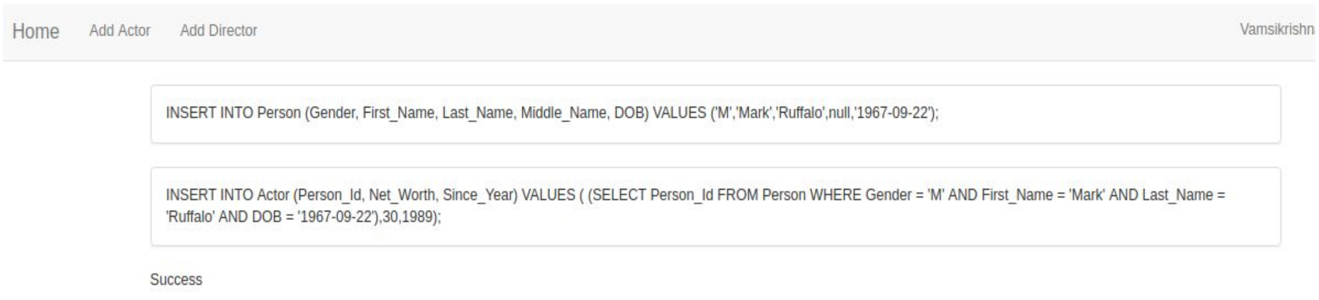
1. Add Actor
    a. This allows us to Add an Actor to the Database. The details of the Actor are provided by the User.
    b. This is done by performing an Insert into both the Person as well as actor table.
        i. The 'Person' Table - which contains information about every single Person present in the database. The Actor ID is obtained from the Person ID after inserting in to the Person Table.
        ii. The 'Actor' table - which contains information about who among the Persons are Actors.
    c. SQL Code:
```
INSERT INTO Person (Gender, First_Name, Last_Name, Middle_Name, DOB)
VALUES ('M','Mark','Ruffalo',null,'1967-09-22');

INSERT INTO Actor (Person_Id, Net_Worth, Since_Year)
VALUES ( (SELECT Person_Id FROM Person WHERE Gender = 'M' AND
First_Name = 'Mark' AND Last_Name = 'Ruffalo' AND DOB =
'1967-09-22'),30,1989);
```
    d. Result:
        i. If doesn't exist in Person, inserts into both Person and Actor tables.
            1. Added Mark Ruffalo in the Actor Table.
            2. He doesn't exist yet in the Person Table, hence he is added to both the Person and Actor Table.



Add Actor

FirstName	Mark
LastName	Ruffalo
BirthDay	22/09/1967
Gender	M
Net Worth	30
Since Year	1989

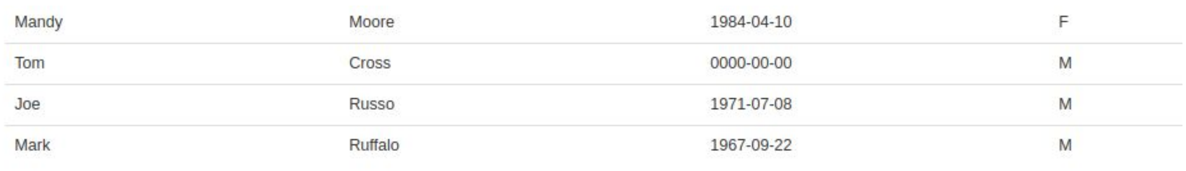

Home Add Actor Add Director Vamsikrishn

```
INSERT INTO Person (Gender, First_Name, Last_Name, Middle_Name, DOB) VALUES ('M','Mark','Ruffalo',null,'1967-09-22');
```

```
INSERT INTO Actor (Person_Id, Net_Worth, Since_Year) VALUES ( (SELECT Person_Id FROM Person WHERE Gender = 'M' AND First_Name = 'Mark' AND Last_Name =
'Ruffalo' AND DOB = '1967-09-22'),30,1989);
```

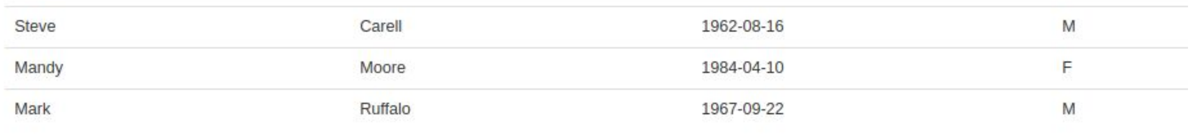
Success

Snapshot of Person Table:



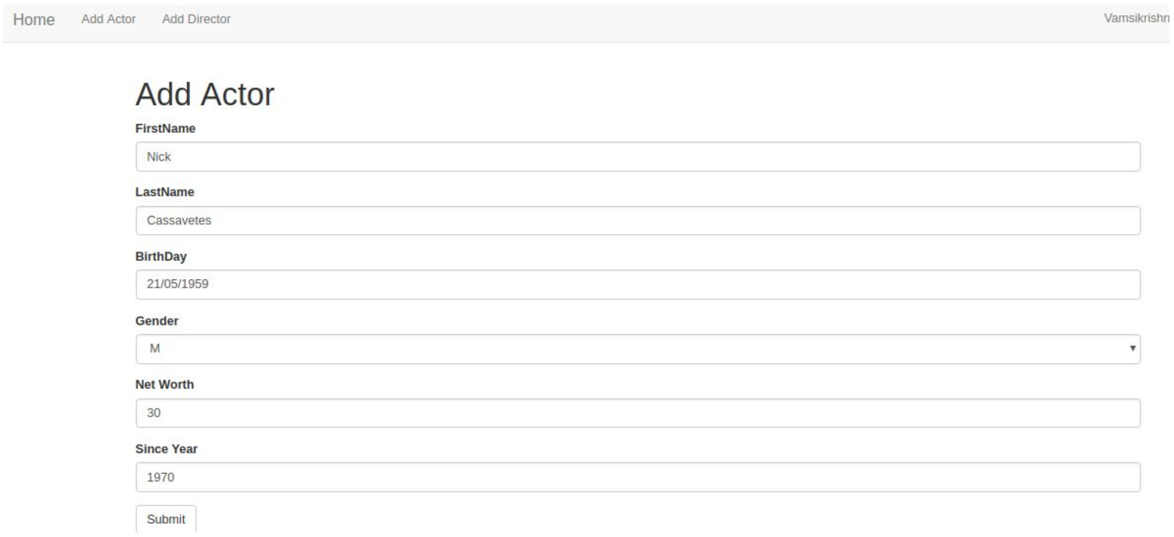
Mandy	Moore	1984-04-10	F
Tom	Cross	0000-00-00	M
Joe	Russo	1971-07-08	M
Mark	Ruffalo	1967-09-22	M

Snapshot of Actor Table:



Steve	Carell	1962-08-16	M
Mandy	Moore	1984-04-10	F
Mark	Ruffalo	1967-09-22	M

ii. If exists in Person but doesn't exist in Actor, inserts only into Actor table.
    1. Then, added Director Nick Cassavetes (Result is shown below). So he exists in the Person Table.
    2. Now added him in the Actor Table as well. He is added successfully in Actor Table.


Home Add Actor Add Director Vamsikrishn

Add Actor

FirstName: Nick

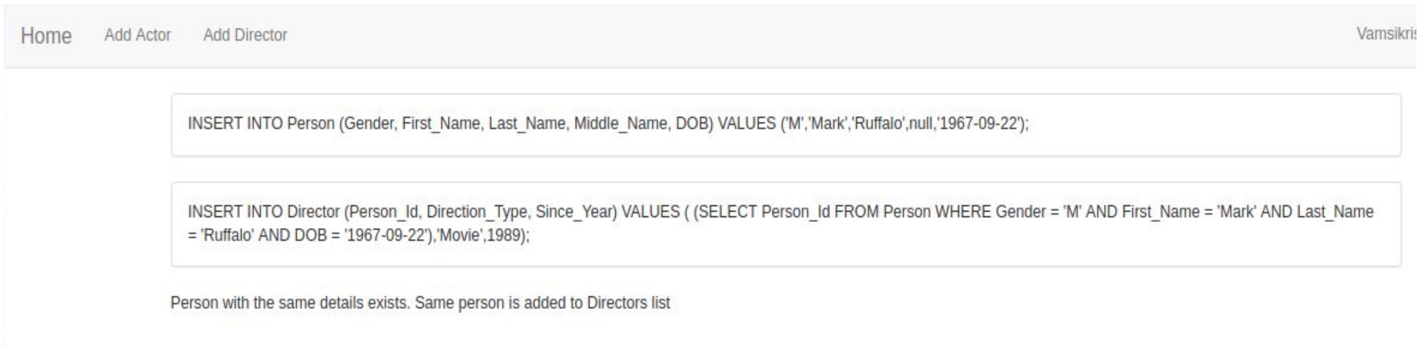
LastName: Cassavetes

BirthDay: 21/05/1959

Gender: M

Net Worth: 30

Since Year: 1970



```
INSERT INTO Person (Gender, First_Name, Last_Name, Middle_Name, DOB) VALUES ('M','Mark','Ruffalo',null,'1967-09-22');
```

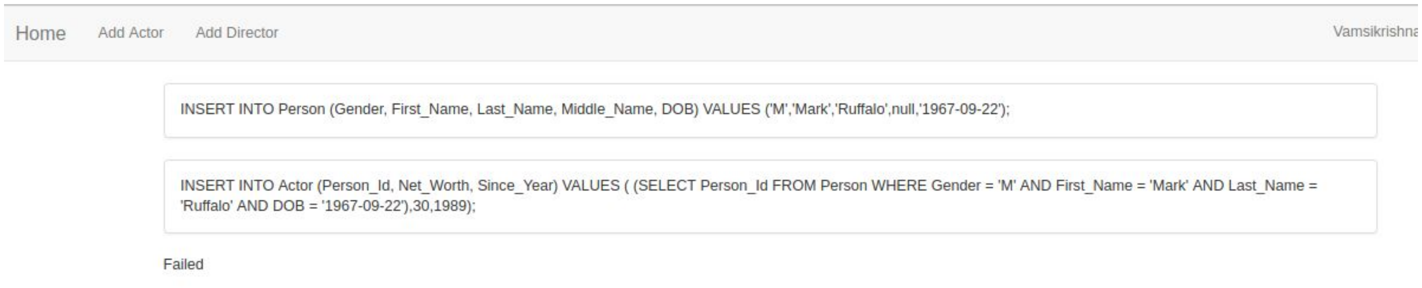
```
INSERT INTO Director (Person_Id, Direction_Type, Since_Year) VALUES ( (SELECT Person_Id FROM Person WHERE Gender = 'M' AND First_Name = 'Mark' AND Last_Name = 'Ruffalo' AND DOB = '1967-09-22'),'Movie',1989);
```

Person with the same details exists. Same person is added to Directors list

Snapshot of Actor Table

Steve	Carell	1962-08-16	M
Mandy	Moore	1984-04-10	F
Mark	Ruffalo	1967-09-22	M
Nick	Cassavetes	1959-05-21	M

iii. If exists in both Person and Actor table, the transaction Fails.
   1. Trying to add Mark Ruffalo again in the Actor Table
   2. This gives an error as he already exists in both Person and Actor Table.



```
INSERT INTO Person (Gender, First_Name, Last_Name, Middle_Name, DOB) VALUES ('M','Mark','Ruffalo',null,'1967-09-22');
```

```
INSERT INTO Actor (Person_Id, Net_Worth, Since_Year) VALUES ( (SELECT Person_Id FROM Person WHERE Gender = 'M' AND First_Name = 'Mark' AND Last_Name = 'Ruffalo' AND DOB = '1967-09-22'),30,1989);
```

Failed

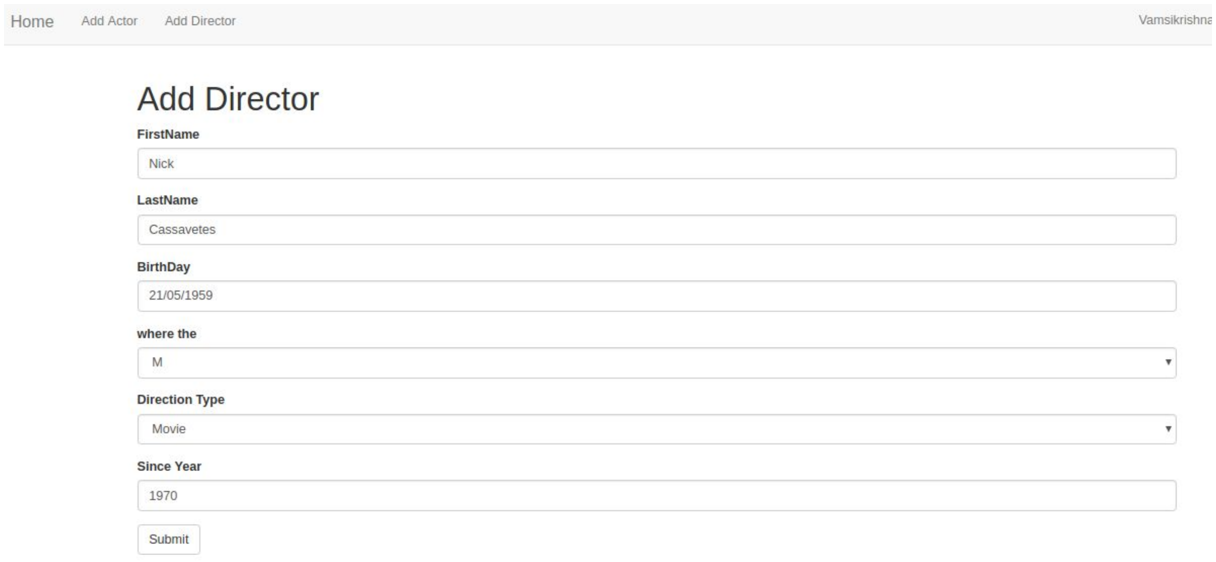
2. Add Director
   a. Similar to the above case, this allows us to Add a Director to the Database. The details of the Director are provided by the User.
   b. This is done by performing an Insert into both the Person as well as Director table.
      i. The 'Person' Table - which contains information about every single Person present in the database. The Director ID is obtained from the Person ID after inserting in to the Person Table.
      ii. The 'Director' table - which contains information about who among the Persons are Directors.
   c. SQL Code:
      INSERT INTO Person (Gender, First_Name, Last_Name, Middle_Name, DOB) VALUES ('M','Nick','Cassavetes',null,'1954-05-21');

INSERT INTO Director (Person_Id, Direction_Type, Since_Year) VALUES ((SELECT Person_Id FROM Person WHERE Gender = 'M' AND First_Name = 'Nick' AND Last_Name = 'Cassavetes' AND DOB = '1954-05-21'),'Movie',1970);

d. Result:

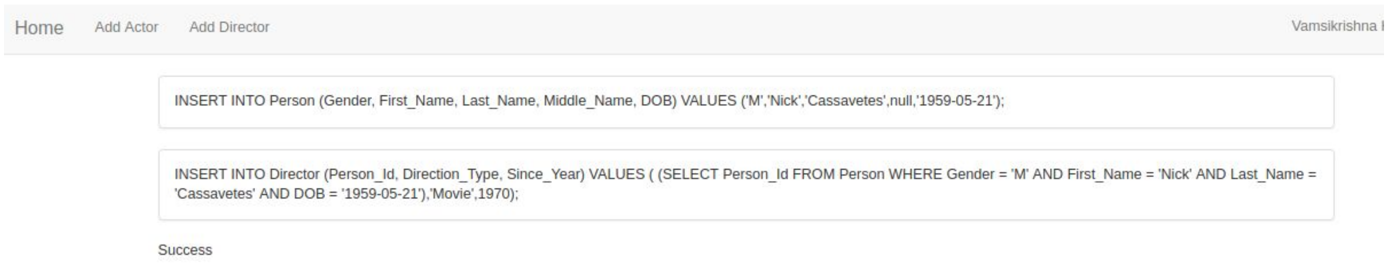
   i. If doesn't exist in Person, inserts into both Person and Director tables.

      1. Added Nick Cassavetes in the Director Table.
      2. He doesn't exist yet in the Person Table, hence he is added to both the Person and Director Table.



Add Director

FirstName	Nick
LastName	Cassavetes
BirthDay	21/05/1959
where the	M
Direction Type	Movie
Since Year	1970

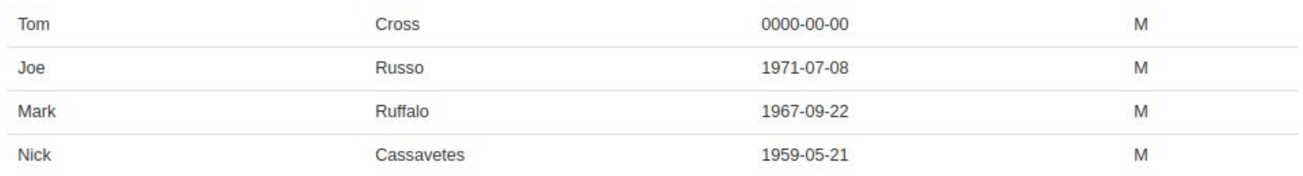


```
INSERT INTO Person (Gender, First_Name, Last_Name, Middle_Name, DOB) VALUES ('M','Nick','Cassavetes',null,'1959-05-21');
```

```
INSERT INTO Director (Person_Id, Direction_Type, Since_Year) VALUES ( (SELECT Person_Id FROM Person WHERE Gender = 'M' AND First_Name = 'Nick' AND Last_Name =
'Cassavetes' AND DOB = '1959-05-21'),'Movie',1970);
```

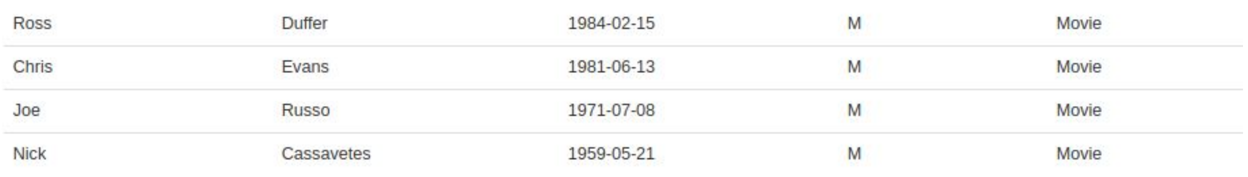
Success

Snapshot of Person Table



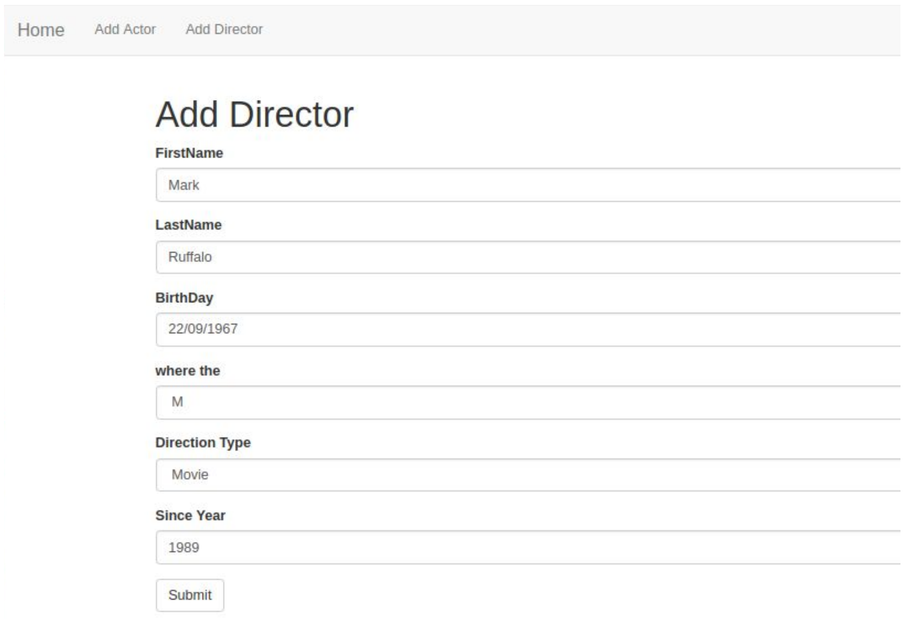
Tom	Cross	0000-00-00	M
Joe	Russo	1971-07-08	M
Mark	Ruffalo	1967-09-22	M
Nick	Cassavetes	1959-05-21	M

Snapshot of Director Table



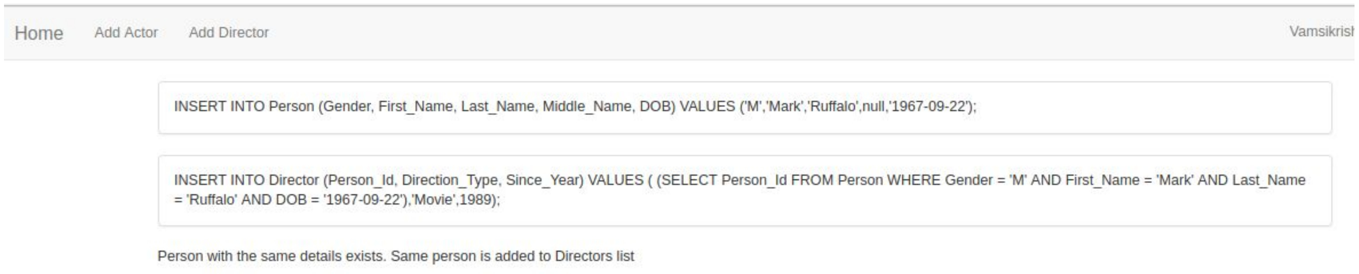
Ross	Duffer	1984-02-15	M	Movie
Chris	Evans	1981-06-13	M	Movie
Joe	Russo	1971-07-08	M	Movie
Nick	Cassavetes	1959-05-21	M	Movie

ii. If exists in Person but doesn't exist in Director, inserts only into Director table.
   1. Then, added Actor Mark Ruffalo. He exists in the Person Table.
   2. Now added him in the Director Table as well. He is added successfully in Director Table.



Add Director

FirstName	Mark
LastName	Ruffalo
BirthDay	22/09/1967
where the	M
Direction Type	Movie
Since Year	1989

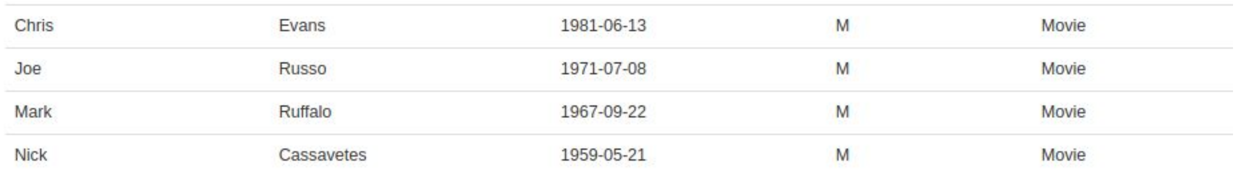


```
INSERT INTO Person (Gender, First_Name, Last_Name, Middle_Name, DOB) VALUES ('M','Mark','Ruffalo',null,'1967-09-22');
```

```
INSERT INTO Director (Person_Id, Direction_Type, Since_Year) VALUES ( (SELECT Person_Id FROM Person WHERE Gender = 'M' AND First_Name = 'Mark' AND Last_Name = 'Ruffalo' AND DOB = '1967-09-22'),'Movie',1989);
```

Person with the same details exists. Same person is added to Directors list

Snapshot of Directors Table



Chris	Evans	1981-06-13	M	Movie
Joe	Russo	1971-07-08	M	Movie
Mark	Ruffalo	1967-09-22	M	Movie
Nick	Cassavetes	1959-05-21	M	Movie

iii. If exists in both Person and Director table, the transaction Fails.
    1. Trying to add Nick Cassavetes again in the Director Table
    2. This gives an error as he already exists in both Person and Director Table.