Port Number: 8635
Hosted Link: http://128.238.64.108:8635

Movie Recommendation System

TEAM DETAILS

- Akshit Gandhi (amg9556)
- Tamanna Manek (tm3734)

DESCRIPTION

The wide yet easy availability of data over the internet has proliferated the building of recommendation systems. Leveraging that, we are going to build a movie recommendation system that will primarily suggest movies based on the user's interest in particular genre. However, the aim of this project is not just limited to providing movie recommendations, but also to project meaningful statistics derived from our collected data. The system will collectively maximize the user's watching experience as well as their cinematic knowledge.

ENTITY SETS

1) Movie
2) Directors
3) Theater
4) Theater owner
5) Spectators
6) Producers
7) Actors
8) Awards
9) Genre
10) Scenes(Weak Entity)

RELATIONSHIP SETS

1) ***rated_by*** (Movies are rated by Spectators)
2) ***viewed_by*** (Movies are viewed by Spectators)
3) ***produced_by*** (Movies are produced by Producers)
4) ***directed_by*** (Movies are directed by Directors)
5) ***acted_by*** (Movies are acted by Actors)
6) ***receive*** (Movies receive Awards, Actors receive awards)
7) ***belong_to*** (Movies belong to Genres)
8) ***shown_at*** (Movies shown at Theaters)
9) ***owned_by*** (Theaters are owned by Owners)
10) ***have_Scenes*** (Movies have scenes, where 'Scenes' is the weak entity set which depends on 'Movies')

ATTRIBUTES

1) Movie (Movie_id **(primary key)**, Movie_name **(not null)**, Movie_language, country, release date **(not null)**)
2) Director (Director_id **(primary key)**, name **(not null)**, age, language, gender)
3) Theaters (Theater id **(primary key)**, name **(not null),** location)
4) Theater_Owner (Owner_id **(primary key)**, name **(not null),** age)
5) Spectators (Spectator_id **(primary key)**, name **(not null)**, age, language, gender)
6) Producer (Producer_id **(primary key)**, name **(not null)**, age, language, gender)
7) Actors (Actor_id **(primary key)**, name **(not null),** age, language, gender)
8) Award (**primary key** (year,type))
9) Genre (Genre_id **(primary key)**, type)
10) Scenes (Scene_id, type, timestamp) - It will be a weak entity, i.e. dependent on movies

BUSINESS RULES

MOVIES

- Spectators rate and view movies.
- Movies are directed by Directors. Each Movie is directed by exactly one Director. **(key + participation constraint)**
- Movies are shown in Theaters. Each Movie should be shown in at least one Theater. **(participation constraint)**
- Movies are produced by Producers. Each Movie is produced by at least one Producer. **(participation constraint)**
- Movies are acted by Actors. Each Movie should have at least one actor. **(participation constraint)**
- Movies belong to Genre. Each Movie belongs to at least one Genre. **(participation constraint)**
- Movies receive Awards. A Movie may or may not have awards.
- Movies have scenes. Scenes will be a weak entity with respect to Movies. Consequently, the Scenes entity set cannot exist without the prevalence of the Movies entity set.

DIRECTORS

- Directors may or may not direct any movies.

THEATERS

- Each Theater should screen at least one Movie. **(participation constraint)**
- Theater is owned by Owner. Each Theater is owned by exactly one owner. **(key+participation constraint)**

THEATER OWNERS

- Theater Owners own Theaters. Each owner should own at least one theater. **(participation constraint)**

PRODUCER

- Producer produces Movies. Each producer should have produced at least one movie. **(participation constraint)**

ACTORS

- Actors act in movies. Actors may or may not perform in movies.
- Actors receive Awards. Actors may and may not receive awards.

AWARD

- Awards can be awarded to Movies or Actors.
- Each award can be awarded to exactly one movie. **(key+participation constraint)**
- Each award can be awarded to exactly one actor. **(key+participation constraint)**

GENRE

- There should be at least one movie for a particular genre. **(participation constraint)**

DATA FOR OUR APPLICATION

- We have synthesized our database by manual entry of tuples in each table.
- The manual entries were added keeping in mind about the interesting output that each of our functionality would return subsequently.

HOW WILL THE USER INTERACT WITH OUR DATABASE

Our primary focus is to build a user-friendly system that will enable users to easily utilize various functionalities that we are going to incorporate in the project. There are various ways in which a user could interact with our system, some of which are listed as follows:

1) **Movie Info:** A user can simply enter the name of the movie and our system shall display multiple facts about that movie if found in our database.
    - For instance, if a user types 'Interstellar' in the movie search bar, the system will display its descriptive attributes, such as the cast, language, genre, numeric rating, year of release, language, name of director, producer, and so on.

2) **Distribution of scenes:** A user can select an actor from the dropdown menu. Consequently, the distribution of scenes performed by that actor in all movies (that he/she acted) will be displayed in the form a pie chart. Using this, a user can infer about the types of scenes the actor performed in all movies he/she was part of.

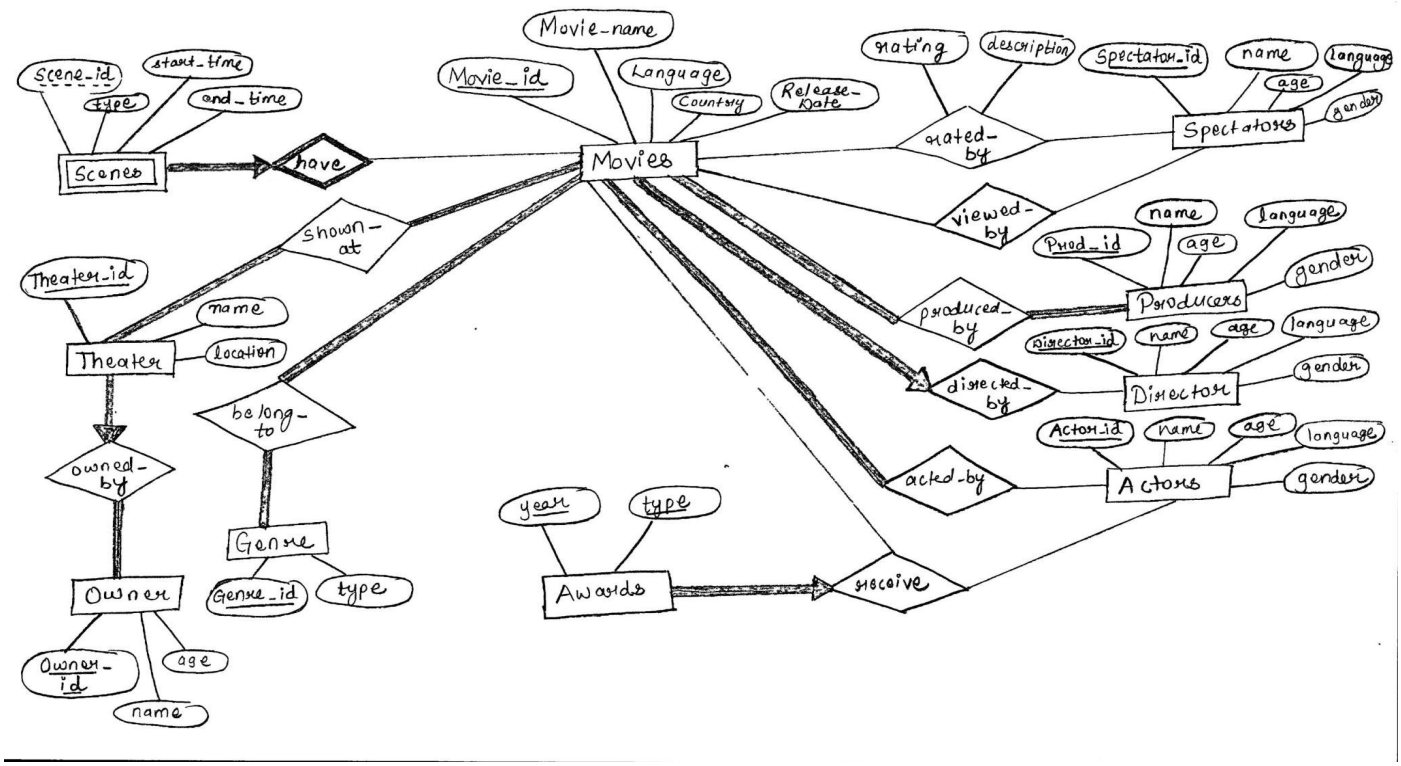
3) **Age wise distribution of genre interests:** A user can select a particular genre from the list. This will display a line chart depicting the number of people of different ages that are interested in this specific genre. This can help user infer about the underlying trends in different age groups for any specific genre.

4) **Genre based recommendation:** A user can simply select a genre from the list and it will display top rated movies of that particular genre. This way, the user can get interesting movie recommendations for any specific genre.

5) **Gender wise distribution of awards received:** A user can select start year, end year and an award category. This will return a meaningful distribution of gender types that received a particular award between a set of years. It will help users to come to a conclusion about which gender type received the most award of a particular category between two years.

6) **Top Viewed Movies:** A user can slide the bar to select a start and an end year. This will display top 5 movies that are viewed the most between a range of years. It will help users to understand which movie was viewed the most between two years and therefore was in the hype at that time.

ER DIAGRAM



HOW DID WE LOAD OUR DATA?

cat csvfilename.csv | psql -U Net_id -d Net_id_db -c "COPY table_name from STDIN CSV HEADER"

- We are using the above command in our terminal to load data from all our csv's into our tables .
- Here,
  csvfilename=Name of our csv file
  Net_id=Our Net_id
  table_name=Name of the table in our schema where we want to load data of the particular csv file
- We are doing this for all CSVs to load data in our different tables

SCHEMA.SQL

```
CREATE TABLE Genre (
Genre_id integer primary key,
Type varchar(128)
);

CREATE TABLE Theater_owner (
Owner_id integer primary key,
Age integer,
Name varchar (128)
);

CREATE TABLE Actors (
Actor_id integer primary key,
Age integer,
Name varchar (128),
Language varchar (128),
Gender varchar(128)
);

CREATE TABLE Director (
Director_id integer primary key,Age integer,
Name varchar (128),
Language varchar (128),
Gender varchar(128)
);

CREATE TABLE Producers (
Producer_id integer primary key,
Age integer,
Name varchar (128),
Language varchar (128),
Gender varchar(128)
);

CREATE TABLE Spectators (
Spectator_id integer primary key,
Age integer,
Name varchar (128),
Language varchar (128),
```

```
Gender varchar(128)
);

CREATE TABLE Movie (
Movie_id integer primary key,
Movie_name varchar(128),
Movie_language varchar(128),
country varchar(128),
release_date integer,
Director_id integer not null,
FOREIGN KEY (Director_id) REFERENCES Director(Director_id)
);

CREATE TABLE Theater_owned_by (
Theater_id integer primary key,
Theater_name varchar(128),
Theater_location varchar (128),
Owner_id integer not null,
FOREIGN KEY (Owner_id) REFERENCES Theater_owner (Owner_id)
);

CREATE TABLE Shown_at (
Movie_id integer,
Theater_id integer,
PRIMARY KEY (Movie_id, Theater_id),
FOREIGN KEY (Movie_id) REFERENCES Movie (Movie_id),
FOREIGN KEY (Theater_id) REFERENCES Theater_owned_by (Theater_id)
);

create table have_scenes(
Movie_Id integer,
Scene_Id integer,
Type varchar(128),
Timestamp integer,
PRIMARY KEY (Movie_id, Scene_id),
FOREIGN KEY (Movie_id) REFERENCES Movie (Movie_id) on delete cascade
);

CREATE TABLE Belong_to(
Movie_id integer,
Genre_id integer,
PRIMARY KEY (Movie_id, Genre_id),
```

```
FOREIGN KEY (Movie_id) REFERENCES Movie (Movie_id),
FOREIGN KEY (Genre_id) REFERENCES Genre (Genre_id)
);

CREATE TABLE Receive_awards (
Year integer,
Type varchar (128),
Movie_id integer not null,
Actor_id integer not null,
PRIMARY KEY (Year, type),
FOREIGN KEY (Movie_id) REFERENCES Movie (Movie_id),
FOREIGN KEY (Actor_id) REFERENCES Actors (Actor_id)
);

CREATE TABLE Produced_by (
Movie_id integer,
Producer_id integer,
PRIMARY KEY (Movie_id, Producer_id),
FOREIGN KEY (Movie_id) REFERENCES Movie (Movie_id),
FOREIGN KEY (Producer_id) REFERENCES Producers (Producer_id)
);

CREATE TABLE Acted_by (
Movie_id integer,
Actor_id integer,
PRIMARY KEY (Movie_id, Actor_id),
FOREIGN KEY (Movie_id) REFERENCES Movie (Movie_id),
FOREIGN KEY (Actor_id) REFERENCES Actors (Actor_id)
);

CREATE TABLE Rated_by (
Movie_id integer,
Spectator_id integer,
Rating decimal,
PRIMARY KEY (Movie_id, Spectator_id),
FOREIGN KEY (Movie_ID) REFERENCES Movie (Movie_id),
FOREIGN KEY (Spectator_id) REFERENCES Spectators (Spectator_id)
);

CREATE TABLE Viewed_by (
Movie_id integer,
Spectator_id integer,
```

```
PRIMARY KEY (Movie_id, Spectator_id),
FOREIGN KEY (Movie_ID) REFERENCES Movie (Movie_id),
FOREIGN KEY (Spectator_id) REFERENCES Spectators (Spectator_id)
);
```