

Movie Matcher

Andy R., Sean P., Sean N., Caleb P.

- ▼ **MOVIE_ACTOR**
 - ▼ Columns
 - ◆ movielfd
 - ◆ actorId
 - ◆ actorName
 - ◆ ranking
 - ▶ Indexes
 - ▶ Foreign Keys
 - ▶ Triggers
- ▼ **MOVIE_DIRECTOR**
 - ▼ Columns
 - ◆ movielfd
 - ◆ directorId
 - ◆ directorName
 - ▶ Indexes
 - ▶ Foreign Keys
 - ▶ Triggers
- ▼ **MOVIE_GENRE**
 - ▼ Columns
 - ◆ movielfd
 - ◆ genre
 - ▶ Indexes
 - ▶ Foreign Keys
 - ▶ Triggers
- ▼ **MOVIE_TAG**
 - ▼ Columns
 - ◆ movielfd
 - ◆ tagId
 - ◆ tagWeight
 - ▶ Indexes
 - ▶ Foreign Keys
 - ▶ Triggers

- ▼ **Movie_Matcher**
 - ▼ Tables
 - ▼ MOVIE
 - ▼ Columns
 - ◆ id
 - ◆ title
 - ◆ imdbId
 - ◆ spanishTitle
 - ◆ imdbPictureURL
 - ◆ year
 - ◆ rtId
 - ◆ rtAllCriticsRating
 - ◆ rtAllCriticsNumReviews
 - ◆ rtAllCriticsNumFresh
 - ◆ rtAllCriticsNumRotten
 - ◆ rtAllCriticsScore
 - ◆ rtTopCriticsRating
 - ◆ rtTopCriticsNumReviews
 - ◆ rtTopCriticsNumFresh
 - ◆ rtTopCriticsNumRotten
 - ◆ rtTopCriticsScore
 - ◆ rtAudienceRating
 - ◆ rtAudienceNumRatings
 - ◆ rtAudienceScore
 - ◆ rtPictureURL
 - ▶ Indexes
 - ▶ Foreign Keys
 - ▶ Triggers

- ▼ **MOVIE_TAG_LIST**
 - ▼ Columns
 - ◆ id
 - ◆ tag
 - ▶ Indexes
 - ▶ Foreign Keys
 - ▶ Triggers
- ▼ **MOVIE_USER_RATINGS**
 - ▼ Columns
 - ◆ userId
 - ◆ movielfd
 - ◆ rating
 - ◆ dateDay
 - ◆ dateMonth
 - ◆ dateYear
 - ◆ dateHour
 - ◆ dateMinute
 - ◆ dateSecond
 - ▶ Indexes
 - ▶ Foreign Keys
 - ▶ Triggers

Queries

Query 1: See Top popular K movies

This was a simple query that let selected the title, score, year, and ID for a movie and ordered it by the score and by the number of reviews. It used the slider to get the K for the query defaulted to 10.

```
SELECT id, title, year, rtAllCriticsNumReviews, rtAllCriticsScore FROM movie_matcher.movie  
  group by title  
  order by rtAllCriticsScore DESC, rtAllCriticsNumReviews DESC  
 LIMIT K
```

```
A) SELECT id, title, year, rtAllCriticsNumReviews, rtAllCriticsScore FROM movie_matcher.movie  
  group by title  
  order by rtAllCriticsScore DESC, rtAllCriticsNumReviews DESC  
 LIMIT 15
```

Shows the top 15 movies

```
B) SELECT id, title, year, rtAllCriticsNumReviews, rtAllCriticsScore FROM movie_matcher.movie  
  group by title  
  order by rtAllCriticsScore DESC, rtAllCriticsNumReviews DESC  
 LIMIT 100
```

Shows the top 100 movies

Query 2: Search by movie title

A simple search query where we innerjoined the tag list so that you could view the tags associated with the movie. It used substring pattern matching to show all movies that contained or was the movie in the search box.

```
SELECT movie.title, year, rtAudienceScore, GROUP_CONCAT(movie_tag_list.tag) AS Tags
FROM movie
INNER JOIN movie_tag
ON movie.id = movie_tag.movieId
INNER JOIN movie_tag_list
ON movie_tag.tagId = movie_tag_list.id
WHERE movie.title LIKE ?
GROUP BY movie.title
order by rtAllCriticsScore
```

A) Shows all movies with Alien in the title

```
SELECT movie.title, year, rtAudienceScore, GROUP_CONCAT(movie_tag_list.tag) AS Tags
FROM movie
INNER JOIN movie_tag
ON movie.id = movie_tag.movieId
INNER JOIN movie_tag_list
ON movie_tag.tagId = movie_tag_list.id
WHERE movie.title LIKE "%Alien%"
GROUP BY movie.title
order by rtAllCriticsScore
```

B) Shows all movies with cowboy in the title

```
SELECT movie.title, year, rtAudienceScore, GROUP_CONCAT(movie_tag_list.tag) AS Tags
FROM movie
INNER JOIN movie_tag
ON movie.id = movie_tag.movieId
INNER JOIN movie_tag_list
ON movie_tag.tagId = movie_tag_list.id
WHERE movie.title LIKE "%Cowboy%"
GROUP BY movie.title
order by rtAllCriticsScore
```

Query 3: Search by genre

This query selected the top K movies in the specified genre X.
It Inner joined the movie genre table to make the genres visible

```
SELECT id, title, year, rtAllCriticsScore, rtAudienceScore
FROM movie_matcher.movie
INNER JOIN movie_matcher.movie_genre
ON movie.id = movie_genre.movieID
WHERE movie_genre.genre = X
GROUP BY movie.title
ORDER BY rtAllCriticsScore DESC, rtAllCriticsNumReviews DESC
LIMIT K
```

A) Shows the top 25 Adventure movies

```
SELECT id, title, year, rtAllCriticsScore, rtAudienceScore
FROM movie_matcher.movie
INNER JOIN movie_matcher.movie_genre
ON movie.id = movie_genre.movieID
WHERE movie_genre.genre = "Adventure"
GROUP BY movie.title
ORDER BY rtAllCriticsScore DESC, rtAllCriticsNumReviews DESC
LIMIT 25
```

B) Shows the top 50 horror movies

```
SELECT id, title, year, rtAllCriticsScore, rtAudienceScore
FROM movie_matcher.movie
INNER JOIN movie_matcher.movie_genre
ON movie.id = movie_genre.movieID
WHERE movie_genre.genre = "Horror"
GROUP BY movie.title
ORDER BY rtAllCriticsScore DESC, rtAllCriticsNumReviews DESC
LIMIT 50
```


Query 4: Search by Director

Using substring pattern matching it shows the directors that match what was in the search query

```
SELECT directorName, movie.title, year, rtAudienceScore
FROM movie_director
INNER JOIN movie
ON movie.id = movie_director.movieId
WHERE directorName LIKE K
```

```
A)SELECT directorName, movie.title, year, rtAudienceScore
FROM movie_director
INNER JOIN movie
ON movie.id = movie_director.movieId
WHERE directorName LIKE "%Martin%"
```

Shows the directors with Martin in their name

```
B)SELECT directorName, movie.title, year, rtAudienceScore
FROM movie_director
INNER JOIN movie
ON movie.id = movie_director.movieId
WHERE directorName LIKE "%James Cameron%"
```

Shows the directors with James Cameron in their name

Query 5: Search by Actor

Searches by actor using substring pattern matching and shows their name and what movie they were in

```
SELECT actorName, movie.title, year, rtAudienceScore
FROM movie_actor
INNER JOIN movie
ON movie.id = movie_actor.movieId
WHERE actorName LIKE K
```

A) Shows actors with Martin in their name

```
SELECT actorName, movie.title, year, rtAudienceScore
FROM movie_actor
INNER JOIN movie
ON movie.id = movie_actor.movieId
WHERE actorName LIKE "%Martin%"
```

B) Shows actors that have angelina in their name

```
SELECT actorName, movie.title, year, rtAudienceScore
FROM movie_actor
INNER JOIN movie
ON movie.id = movie_actor.movieId
WHERE actorName LIKE "%Angelina%"
```

Query 6: Search by Tag

This query inner joined the tag and tag list tables to show movies that had that tag and used sub string pattern matching to show ones that contained it too.

```
SELECT movie_tag_list.tag, movie.title, year, rtAudienceScore
FROM movie
INNER JOIN movie_tag
ON movie.id = movie_tag.movieId
INNER JOIN movie_tag_list
ON movie_tag.tagId = movie_tag_list.id
WHERE tag LIKE K
order by rtAllCriticsScore
```

A) shows all films that contain the nudity tag

```
SELECT movie_tag_list.tag, movie.title, year, rtAudienceScore
FROM movie
INNER JOIN movie_tag
ON movie.id = movie_tag.movieId
INNER JOIN movie_tag_list
ON movie_tag.tagId = movie_tag_list.id
WHERE tag LIKE "%nudity%"
order by rtAllCriticsScore
```

B) Shows all films that contain the 70mm tag

```
SELECT movie_tag_list.tag, movie.title, year, rtAudienceScore
FROM movie
INNER JOIN movie_tag
ON movie.id = movie_tag.movieId
INNER JOIN movie_tag_list
ON movie_tag.tagId = movie_tag_list.id
WHERE tag LIKE "%70mm%"
order by rtAllCriticsScore
```


Query 7: See Top 10 Directors in K movies

This query inner joined the movie table so that we could sort the results by highest audience score. But essentially it takes the slider value and compares it to the count of movies that the director directed and if it is greater than K it is added to the query with a hard Limit of 10 directors.

```
SELECT count(movieId)as Num_Movies, directorName as Name, rtAudienceScore
  from movie_matcher.movie_director
  INNER JOIN movie
    ON movie.id = movie_director.movieID
  group by directorID
  having count(movie_director.movieId) >= K
  ORDER BY rtAudienceScore DESC, rtAudienceNumRatings DESC, count(movieID) DESC
  LIMIT 10;
```

A)Shows the top 10 directors that directed 25 movies

```
SELECT count(movieId)as Num_Movies, directorName as Name, rtAudienceScore
  from movie_matcher.movie_director
  INNER JOIN movie
    ON movie.id = movie_director.movieID
  group by directorID
  having count(movie_director.movieId) >= 25
  ORDER BY rtAudienceScore DESC, rtAudienceNumRatings DESC, count(movieID) DESC
  LIMIT 10;
```

B) Shows the top 10 directors that directed 15 movies

```
SELECT count(movieId)as Num_Movies, directorName as Name, rtAudienceScore
  from movie_matcher.movie_director
  INNER JOIN movie
    ON movie.id = movie_director.movieID
  group by directorID
  having count(movie_director.movieId) >= 15
  ORDER BY rtAudienceScore DESC, rtAudienceNumRatings DESC, count(movieID) DESC
  LIMIT 10;
```

Query 8: See Top 10 Actors in K movies

This query inner joined the movie table so that we could sort the results by highest audience score. But essentially it takes the slider value and compares it to the count of movies that the actor is in and if it is greater than K it is added to the query with a hard Limit of 10 actors.

```
SELECT count(movieId)as Num_Movies, actorName as Name, rtAudienceScore
from movie_matcher.movie_actor
INNER JOIN movie
ON movie.id = movie_actor.movieID
group by actorID
having count(movie_actor.movieId) >= K
ORDER BY rtAudienceScore DESC, rtAudienceNumRatings DESC, count(movieID) DESC
LIMIT 10;
```

```
A)SELECT count(movieId)as Num_Movies, actorName as Name, rtAudienceScore
from movie_matcher.movie_actor
INNER JOIN movie
ON movie.id = movie_actor.movieID
group by actorID
having count(movie_actor.movieId) >= 20
ORDER BY rtAudienceScore DESC, rtAudienceNumRatings DESC, count(movieID) DESC
LIMIT 10;
```

Shows the top 10 actors that appeared in 20 movies

```
B)SELECT count(movieId)as Num_Movies, actorName as Name, rtAudienceScore
from movie_matcher.movie_actor
INNER JOIN movie
ON movie.id = movie_actor.movieID
group by actorID
having count(movie_actor.movieId) >= 50
ORDER BY rtAudienceScore DESC, rtAudienceNumRatings DESC, count(movieID) DESC
LIMIT 10;
```

Shows the top 10 actors that appeared in 50 movies

Query 9: See time line and graph for a user

Using the user ID provided in the search box this one does two queries one for the graph and one for the timeline.

The timeline one is a simple list of all movies they have reviewed and orders it by Date nothing too special about it.

The Pie Chart one was a bit more involved. It selects the distinct genres so there is only 1 result per genre and it inner joins a subquery that takes the count of each genre and limits it by the user ID.

```
Select concat(dateMonth,'/', dateDay,'/', dateYear,' at ', dateHour,':', dateMinute,':', dateSecond) as Date, movie.title, rating
  From movie_user_ratings
  inner join movie
  on movie.id = movie_user_ratings.movieId
 where userId = K
 group by title
 order by Date ASC
```

```
SELECT distinct S.genre, C.cnt
  FROM movie_genre S
  INNER JOIN (SELECT genre, count(genre) as cnt
  FROM movie_genre as C
  INNER JOIN movie_user_ratings
  ON c.movieID = movie_user_ratings.movieId
  where userId = ? GROUP BY genre) C ON S.genre = C.genre
```

A) Shows the timeline of movie reviews for User ID 75

```
Select concat(dateMonth,'/', dateDay,'/', dateYear,' at ', dateHour,':', dateMinute,':', dateSecond) as Date, movie.title, rating
  From movie_user_ratings
  inner join movie
  on movie.id = movie_user_ratings.movieId
 where userId = 75
 group by title
 order by Date ASC
```

B) Shows the timeline of user reviews for user ID 170

```
Select concat(dateMonth,'/', dateDay,'/', dateYear,' at ', dateHour,':', dateMinute,':', dateSecond) as Date, movie.title, rating
  From movie_user_ratings
  inner join movie
  on movie.id = movie_user_ratings.movieId
 where userId = 170
 group by title
 order by Date ASC
```

C) Shows the Genre and the amount watched for user ID 75

```
SELECT distinct S.genre, C.cnt
  FROM movie_genre S
  INNER JOIN (SELECT genre, count(genre) as cnt
  FROM movie_genre as C
  INNER JOIN movie_user_ratings
  ON c.movieID = movie_user_ratings.movieId
  where userId = 75 GROUP BY genre) C ON S.genre = C.genre
```

D) Shows the Genre and the amount watched for user ID 170

```
SELECT distinct S.genre, C.cnt
  FROM movie_genre S
  INNER JOIN (SELECT genre, count(genre) as cnt
  FROM movie_genre as C
  INNER JOIN movie_user_ratings
  ON c.movieID = movie_user_ratings.movieId
  where userId = 170 GROUP BY genre) C ON S.genre = C.genre
```

Query 10: See movies containing the same tags
 This one was hard and we had to limit it to EXACT movie titles. This takes the movie in the search box and shows all of its tags ordered alphabetically just like the tag query. It limits it by the subquery where all movies that contain any of the specified movies tags. It goes 1 step further and from the tags that those movies have in common it only shows the ones that have at least all the tags in common.

```
SELECT movieId,title, GROUP_CONCAT(movie_tag_list.tag ORDER BY tag ASC separator ', ') as Tags
FROM MOVIE_TAG
INNER JOIN Movie
ON movie.id = movie_tag.movieId
INNER JOIN movie_tag_list
ON movie_tag_list.id = tagId
WHERE movie_tag.tagId in ( SELECT tagId from MOVIE_TAG
INNER JOIN Movie
ON movie.id = movie_tag.movieId
WHERE movie.title = ?)
GROUP BY movieId
having count(distinct MOVIE_TAG.tagId) >= (SELECT COUNT(tagId) from MOVIE_TAG
INNER JOIN Movie
ON movie.id = movie_tag.movieId
WHERE movie.title = ?)
```

A) Shows all movies that have all the tags of Hellraiser: Bloodline

```
SELECT movieId,title, GROUP_CONCAT(movie_tag_list.tag ORDER BY tag ASC separator ', ') as Tags
FROM MOVIE_TAG
INNER JOIN Movie
ON movie.id = movie_tag.movieId
INNER JOIN movie_tag_list
ON movie_tag_list.id = tagId
WHERE movie_tag.tagId in ( SELECT tagId from MOVIE_TAG
INNER JOIN Movie
ON movie.id = movie_tag.movieId
WHERE movie.title = "Hellraiser: Bloodline")
GROUP BY movieId
having count(distinct MOVIE_TAG.tagId) >= (SELECT COUNT(tagId) from MOVIE_TAG
INNER JOIN Movie
ON movie.id = movie_tag.movieId
WHERE movie.title = "Hellraiser: Bloodline")
```

B)Shows all the movies that have all the tags of Aliens

```
SELECT movieId,title, GROUP_CONCAT(movie_tag_list.tag ORDER BY tag ASC separator ', ') as Tags
FROM MOVIE_TAG
INNER JOIN Movie
ON movie.id = movie_tag.movieId
INNER JOIN movie_tag_list
ON movie_tag_list.id = tagId
WHERE movie_tag.tagId in ( SELECT tagId from MOVIE_TAG
INNER JOIN Movie
ON movie.id = movie_tag.movieId
WHERE movie.title = "Aliens")
GROUP BY movieId
having count(distinct MOVIE_TAG.tagId) >= (SELECT COUNT(tagId) from MOVIE_TAG
INNER JOIN Movie
ON movie.id = movie_tag.movieId
WHERE movie.title = "Aliens")
```

Workload

Andy R. - Responsible for the GUI and SQL Statement 9

Sean P. - Responsible for Big Data import/processing and SQL Statement 10

Sean N. - Responsible for statements 1 - 4

Caleb P. Responsible for statements 5 - 8

Learning Outcome

We all learned a great deal about manipulating and querying data with SQL during this project. It was much easier to jump right in and have things come together in a project rather than small demonstrations.

We also all learned more about *Git* and *Github* together as well as learning how to be a contributing member of a remote team.