Users
Users(UserId:INT [PK],
    Password: VARCHAR(255),
    FavoriteMovie: VARCHAR(255))

Titles
Titles(TitleId: INT [PK],
    titleType: VARCHAR(255),
    primaryTitle: VARCHAR(255),
    originalTitle: VARCHAR(255),
    isAdult: INT,
    startYear: INT,
    endYear: INT,
    runtimeMinutes: INT,
    Genres: VARCHAR(255),
    averageRating: FLOAT,
    numVotes: INT
)

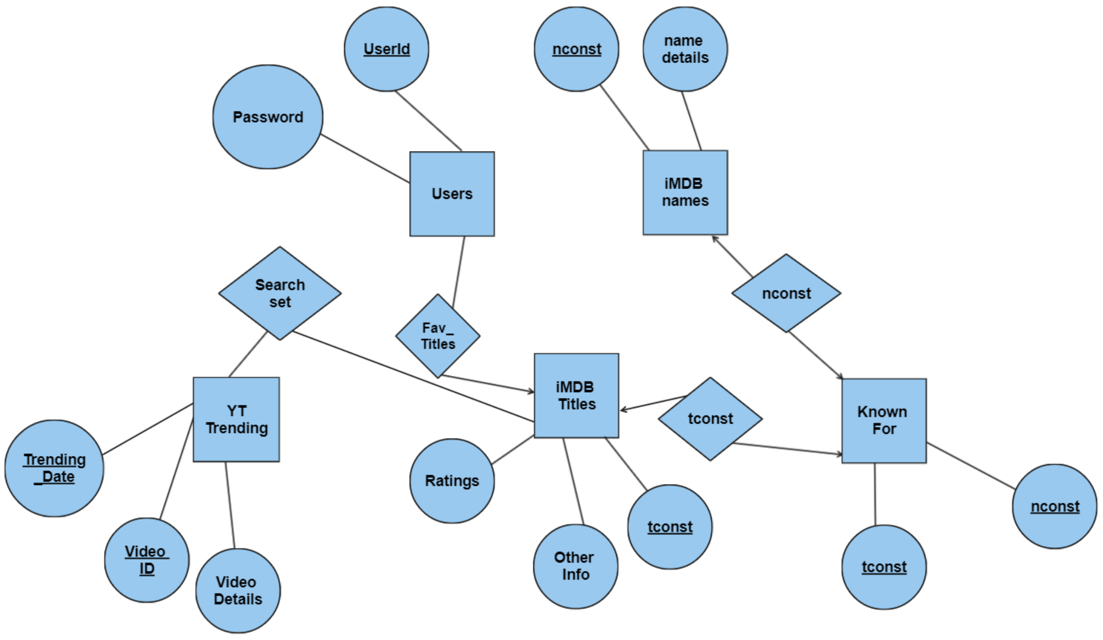
Names
Names(NameId: INT [PK],
    nconst: VARCHAR(255),
    primaryName: VARCHAR(255),
    birthYear: INT,
    deathYear: INT,
    primaryProfession: VARCHAR(255),
    knownForTitles: VARCHAR(255) [FK to Titles.TitleId]
)

KnownFor
KnownFor((NameId,TitleId): (INT,INT) [PK],
    NameId: INT [FK to Names.NameId],
    TitleId: INT [FK to Titles.TitleId])

Videos
Videos((VideoId,TrendingDate): (INT,DATE) [PK],
    title: VARCHAR(65536),
    publishedAt: DATETIME,
    channelId: VARCHAR(255),
    channelTitle: VARCHAR(255),
    categoryId: VARCHAR(255),
    trending_date: DATE,
    tags: VARCHAR(65536),
    view_count: INT,
    likes: INT,
    dislikes: INT,
    comment_count: INT,

```
thumbnail_link: VARCHAR(255),
comments_disabled: BOOL,
ratings_disabled: BOOL,
description: VARCHAR(65535)
)
```

ER Diagram



BCNF Justification

It is useful to employ BCNF in order to do two things:

- Removal of redundancies based on functional dependencies
- Avoid loss of information

For our database schema, we found that using BCNF would be the most effective normalization technique. The main purpose of our project is to allow a user to select for specific movies, actors, or directors in order to generate a set of videos from the YouTube trending list that relate to those topics. To get this information we have the following entities:

- YouTube trending videos
- IMDB movie titles
- IMDB crew names
- User Information
- Known For (movies specific crew are known for being a part of)

If we just used 3NF, we would be able to normalize our schema and remove transitive dependencies. However, by using BCNF, we can remove transitive dependencies as well as non-trivial dependencies.