Users

```
Users(
        UserId VARCHAR(255),
        PRIMARY KEY (UserId),
        Password VARCHAR(255),
        FavoriteMovie VARCHAR(255),
        FOREIGN KEY (FavoriteMovie) REFERENCES Titles(primaryTitle)
        );
```

- UID = UserID, UA = {Password, FavoriteMovie}
- FDs: UID -> UA

Titles

```
iMDB_Titles(
        titleID VARCHAR(255),
        PRIMARY KEY (titleID),
        titleType VARCHAR(255),
        primaryTitle VARCHAR(255),
        originalTitle VARCHAR(255),
        isAdult INT,
        startYear INT,
        endYear INT,
        runtimeMinutes INT,
        averageRating FLOAT,
        numVotes INT
        );
```

- TID = titleID and TA = {titleType, primaryTitle, originalTitle, isAdult, startYear, endYear, runtimeMinutes, avgRating, numVotes}
- FDs: TID -> TA

Genres

```
Genres(
        genreName VARCHAR(255),
        titleID VARCHAR(255),
        FOREIGN KEY (titleID) REFERENCES Titles(titleID),
        PRIMARY KEY (genreName, titleID)
        );
```

- GID = genreID and GA = {genreName}
- FDs: GID -> GA

KnownFor (Relationship Table)

```
KnownFor(
        titleID VARCHAR(255),
```

nameID VARCHAR(255),
PRIMARY KEY (titleID, nameID),
FOREIGN KEY (titleID) REFERENCES Titles(titleID),
FOREIGN KEY (nameID) REFERENCES Names(titleID),
);

Names

iMDB_Names(
nameID VARCHAR(255),
PRIMARY KEY (nameID),
nameDetails VARCHAR(255),
primaryName VARCHAR(255),
birthYear: INT,
deathYear: INT,
);

- NID = nameID and NA = {nameDetails, primaryName, birthYear, deathYear}
- FDs: NID -> NA

Professions

Professions(
professionName VARCHAR(255),
PRIMARY KEY (professionName)
);

- PID = professionID and PA = {professionName}
- FDs: PID -> PA

WorksAs (Relationship Table)

WorksAs(
professionName VARCHAR(255),
nameID VARCHAR(255),
PRIMARY KEY(professionName,nameID),
FOREIGN KEY (nameID) REFERENCES iMDB_names(nameID),
FOREIGN KEY(titleID) REFERENCES iMDB_Titles(titleID)
);

SearchSet (Relationship Table)

SearchSet(
VideoId INT,
TrendingDate DATE,
titleID INT,
PRIMARY KEY (VideoId,TrendingDate, titleID),
FOREIGN KEY (VideoId) REFERENCES YT_Trending(VideoId),

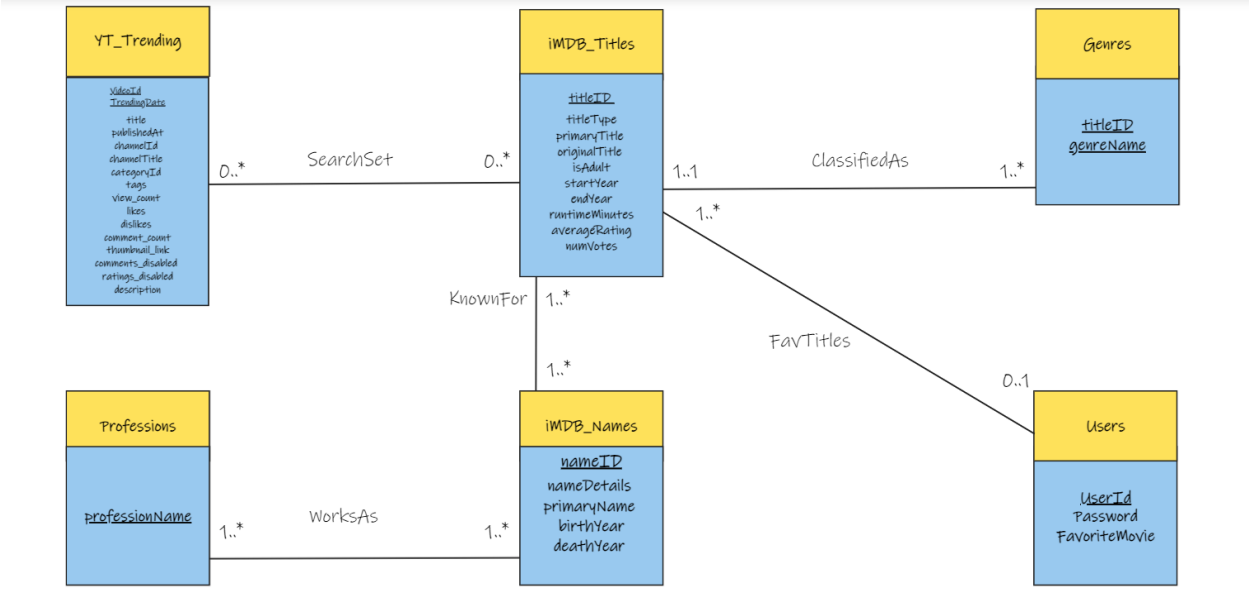
```
        FOREIGN KEY(titleID) REFERENCES iMDB_Titles(titleID),
        FOREIGN KEY (TrendingDate) REFERENCES YT_Trending(TrendingDate),
);
```

Videos

```
YT_Trending (
        VideoId VARCHAR(255),
        TrendingDate DATE,
        PRIMARY KEY(VideoId, TrendingDate),
        title VARCHAR(65536),
        publishedAt DATETIME,
        channelId VARCHAR(255),
        channelTitle VARCHAR(255),
        categoryId VARCHAR(255),
        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)
        )
```

- VID = VideoID, TD = TrendingDate, VA = {title, publishedAt, channelID, channelTitle, categoryID, tags, view_count, likes, dislikes, comment_count, thumbnail_link, comments_disabled, ratings_disables, description}
- FDs: VID, TD -> VA

UML 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. 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.

Detailed list of changes
For each of our relations, there is only one functional dependency, and in each of those functional dependencies the LHS is a key. Therefore, all of our relations are in BCNF.
Detailed list of changes
We added FDs to tables, and described relations between table attributes.
We also added additional tables to represent the many-to-many relations that we have in our schema.
We also changed our ER diagram to more accurately represent the tables in our database.
We also changed some of the datatypes for our primary keys to match the data that we have. For example, VideoId was previously an INT, but now it is a VARCHAR(255).
Lastly, we updated our justification for using BCNF to include the fact that each relation has one FD, and that the LHS is a key.