­

The Fall of the Simpsons Project

Authors: Rusty Gentile, Minal Jajoo, Jean Medina, Scott Nubar, Axander Wilson

Contents

[1 Introduction 1](#_Toc613793)

[2 Architecture 1](#_Toc613794)

[3 Database Design 3](#_Toc613795)

[3.1 Design Process 3](#_Toc613796)

[3.2 Production Database Design 8](#_Toc613797)

[4 Database Insight 8](#_Toc613798)

[4.1 Mining the Database for Insights 8](#_Toc613799)

[5 IMDB-Py Extraction Processes 10](#_Toc613800)

[5.1 Jupyter Notebooks 10](#_Toc613801)

[5.2 Sample Data Scripts 11](#_Toc613802)

[6 Findings and Visualizations 12](#_Toc613803)

List of Figures

[Figure 1‑1 Rating Over Time Plot 1](#_Toc613804)

[Figure 2‑1 Overview of the ETL Process 2](#_Toc613805)

[Figure 2‑2 Application Architecture 2](#_Toc613806)

[Figure 3‑1 Raw .tsv.gz Files 3](#_Toc613807)

[Figure 3‑2 IMDB Documentation 4](#_Toc613808)

[Figure 3‑3 Data Cleansing 5](#_Toc613809)

[Figure 3‑4 Entity Relationship Diagram 6](#_Toc613810)

[Figure 3‑5 Falcon SQL Editor 7](#_Toc613811)

[Figure 3‑6 Production Database 8](#_Toc613812)

[Figure 4‑1 Sample Query 9](#_Toc613813)

[Figure 4‑2 Actor to Actress Bar Graph 9](#_Toc613814)

[Figure 4‑3 Episode Writing by Writer 10](#_Toc613815)

[Figure 5‑1 Series Table Try-Except Logic 11](#_Toc613816)

[Figure 5‑2 Series Table Try-Except Logic 12](#_Toc613817)

[Figure 6‑1 Zombie Simpsons 13](#_Toc613818)

[Figure 6‑2 Rick and Morty Mania 14](#_Toc613819)

[Figure 6‑3 Life of Brian 15](#_Toc613820)

# Introduction

This project was inspired by a YouTube [documentary](https://www.youtube.com/watch?v=KqFNbCcyFkk) titled "The Fall of the Simpsons: How it Happened". It features a figure which trends the average IMDB user rating of each Simpsons episode over time. To limit the scope, we selected 26 series in the genre of animation and built a dashboard around the idea of those figures [Figure 1‑1]. Users can add up to seven series for juxtaposition to see how the IMDB user ratings have reflected in regard to changing social, political, and cultural climates.

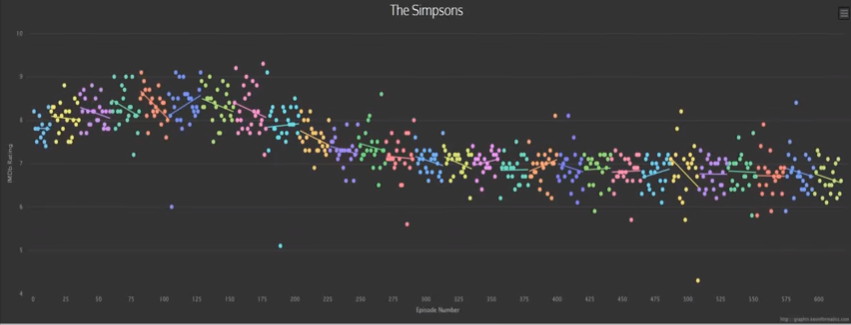


Figure ‑ Rating Over Time Plot

# Architecture

The project uses two sources of data. First, the [.tsv](https://www.imdb.com/interfaces/) files which IMDb releases nightly [Figure 2‑1]. These contain a subset of the actual IMDb database. We analyzed these and derived an entity relationship diagram. However, we found some columns needed to build the site were missing. For example, the original air date of each episode. There was also an issue with resources needed to script and scrape new series to the database as the files were extremely large. To overcome this, the [IMDbPy](https://imdbpy.sourceforge.io/) package was used for information not available in the .tsv files.

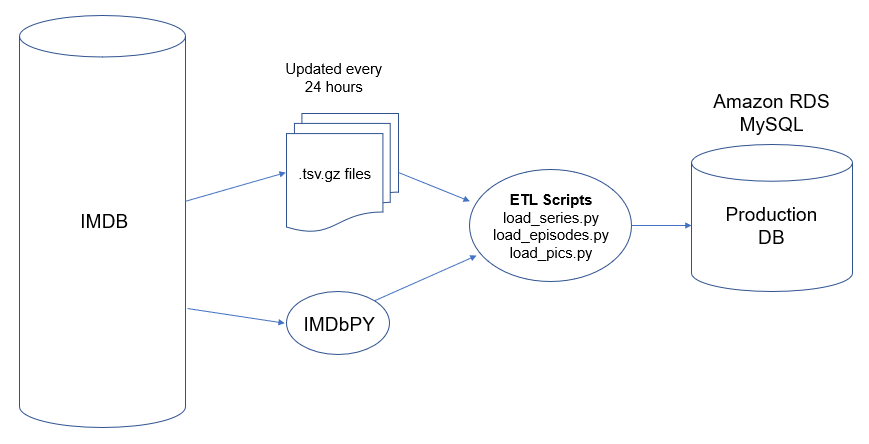


Figure ‑ Overview of the ETL Process

The site’s database is stored using an AWS RDS MySQL database, and Heroku is used to host our Flask application on the back end [Figure 2‑2]. On the front end, we used plotly.js to render most of the figures.

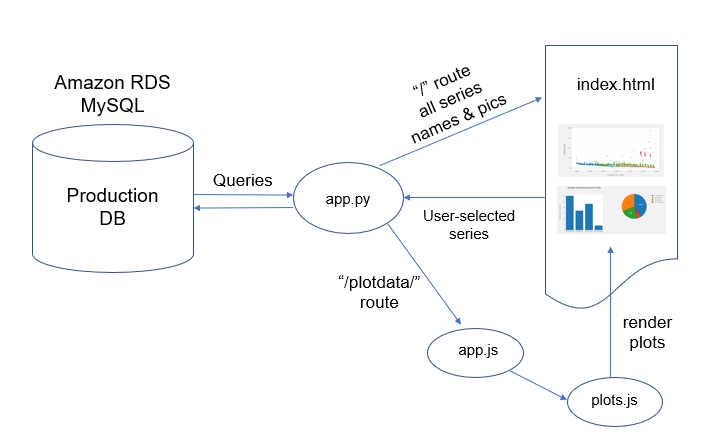


Figure ‑ Application Architecture

# Production Database Design

## Design Process

The first step in designing the database was to extract raw .tv.gz files from IMDB [Figure 3‑1]. The next step was to read IMDB documentation and study the dataset to discover data relationships [Figure 3‑2].

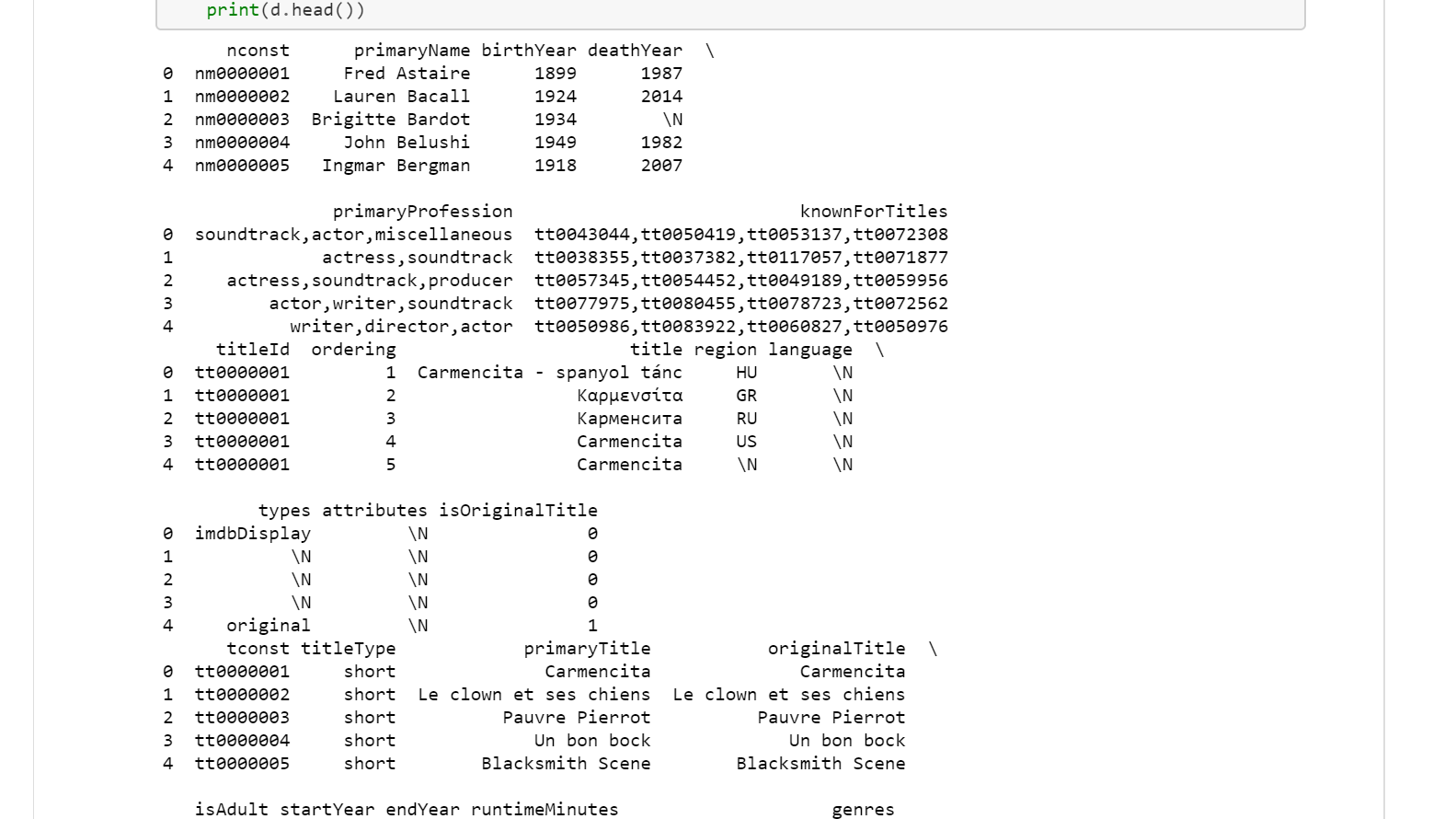


Figure ‑ Raw .tsv.gz Files

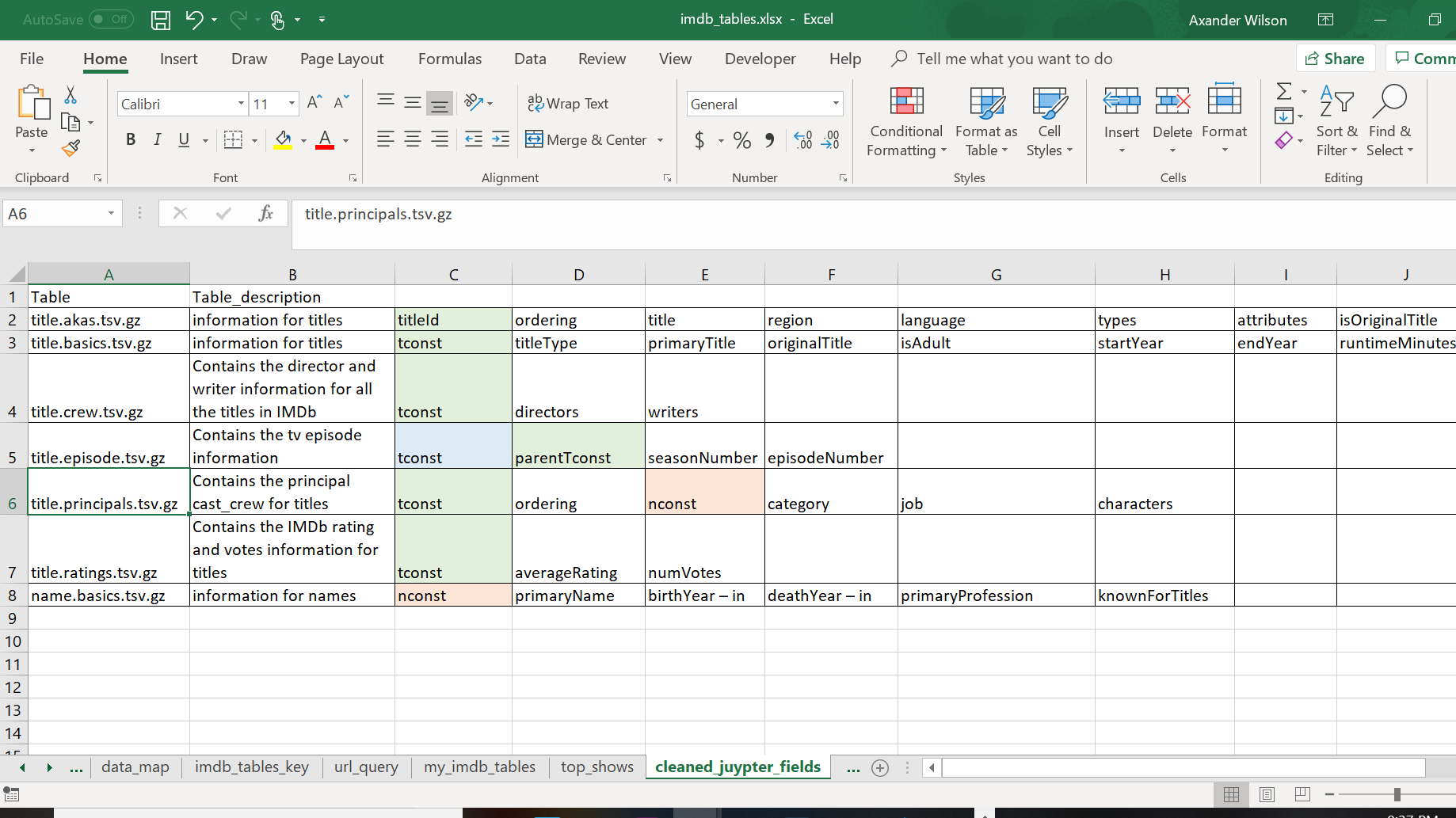


Figure ‑ IMDB Documentation

With an understanding of the dataset, we could then start the cleansing and transformation [Figure 3‑3]. An entity relationship diagram was created to map out the tables, fields and necessary relationships in order to load the database properly [Figure 3‑4­­]. We were able to identify outstanding datapoints not provided in the .tv.gz files such as original air date and plot that we need to extract from IMDbPy.

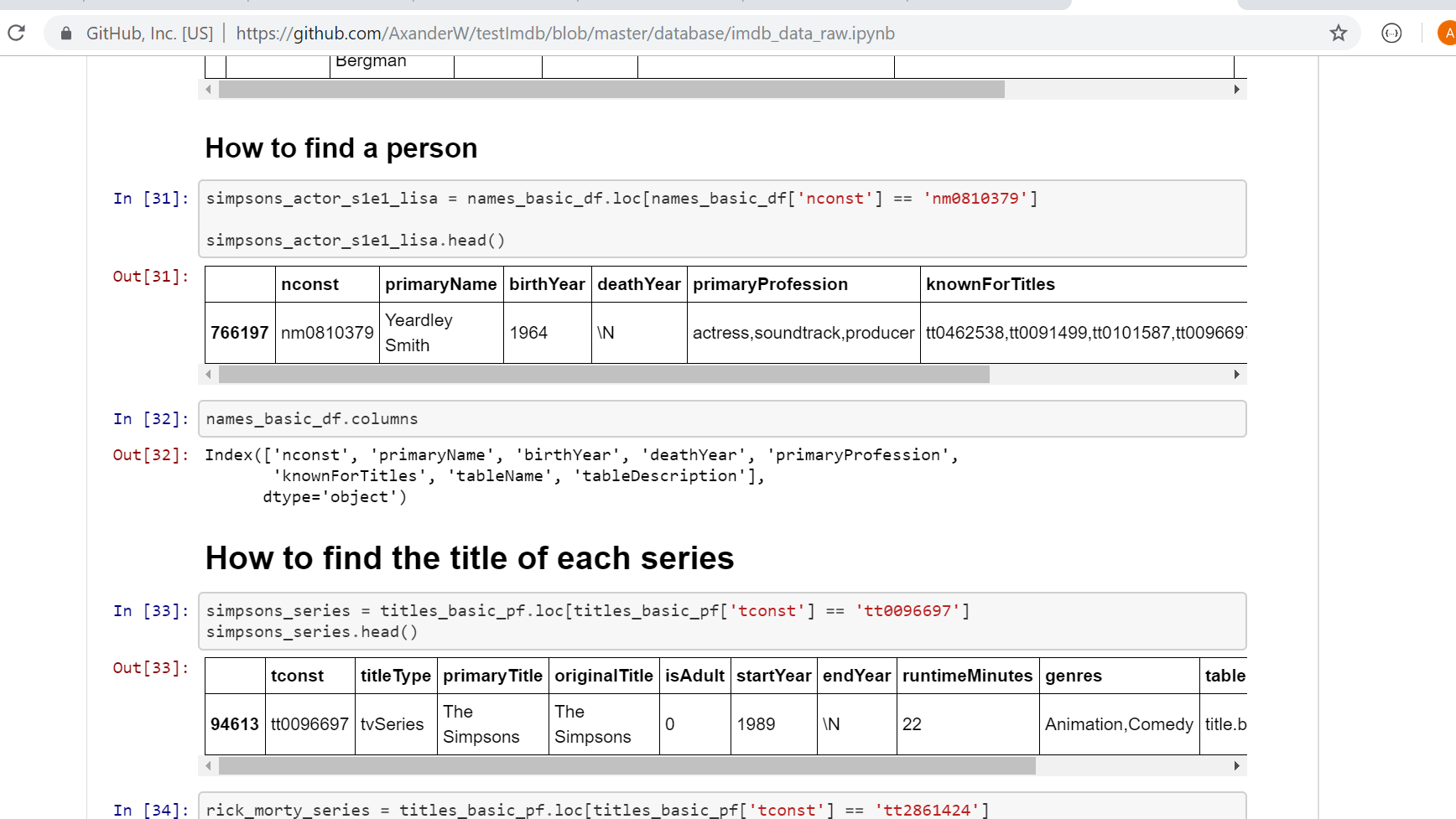


Figure ‑ Data Cleansing

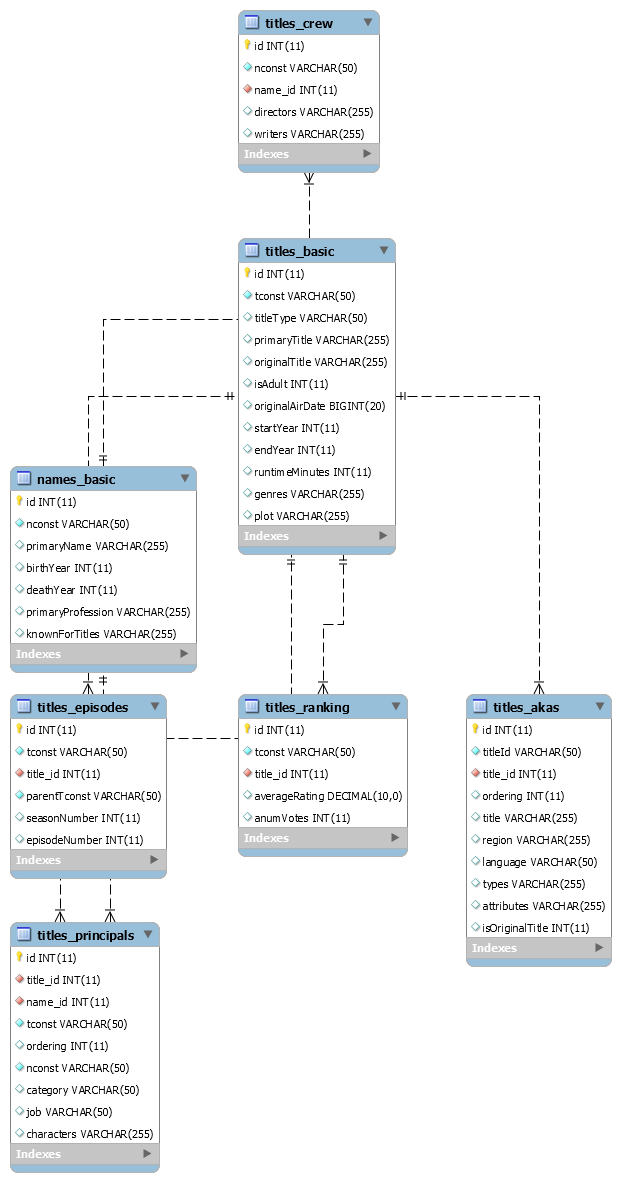


Figure ‑ Entity Relationship Diagram

After we set up our database, we decided to use a SQL editor to get a better idea of how our charts will look [Figure 3-5]. Falcon is a free, open source SQL editor with inline data visualization. With Falcon we were able to connect to our database in the “Connection” tab, run SQL queries in the “Query” tab, and finally export the results as a CSV. The editor was used in conjunction with the AWS database.

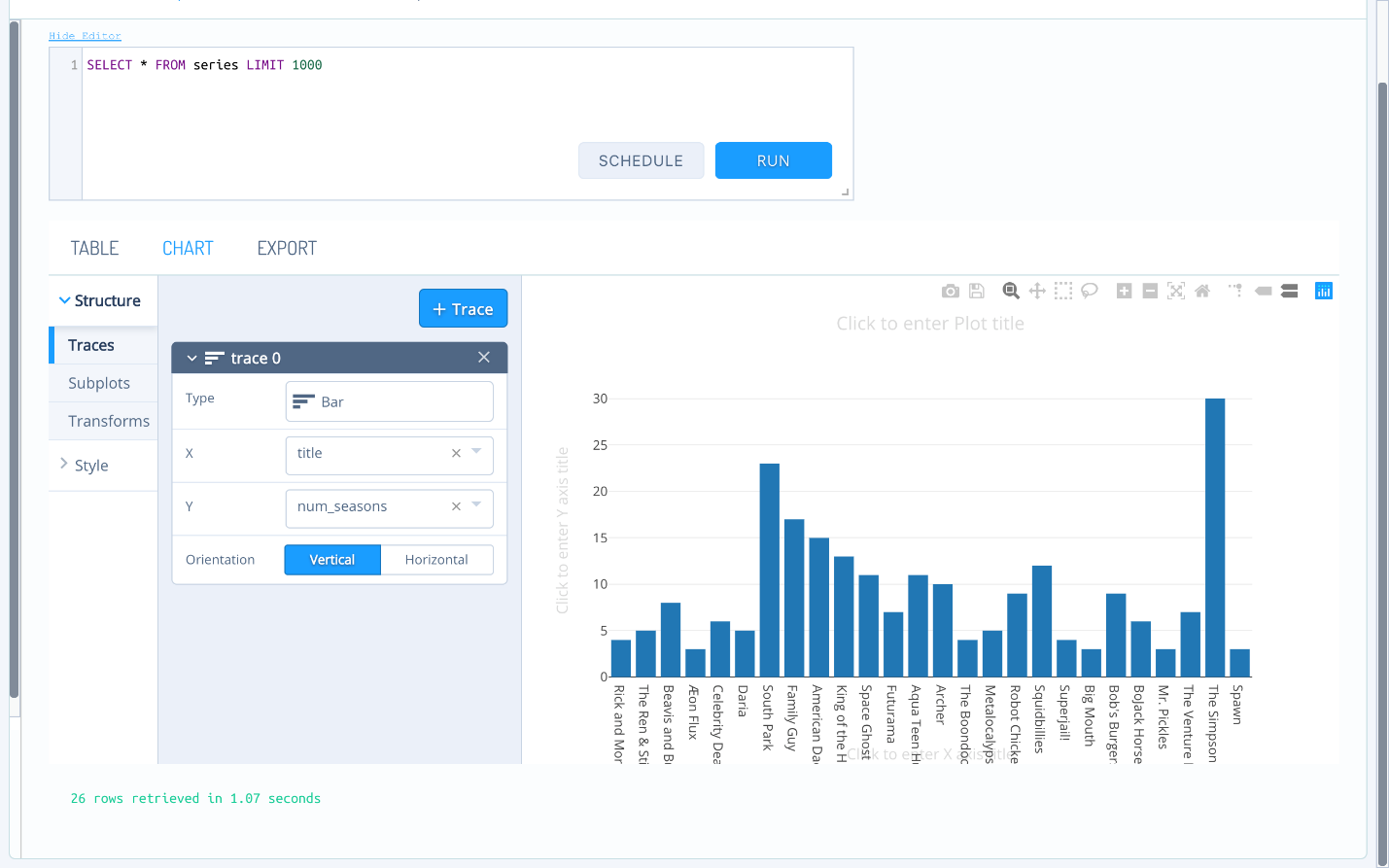


Figure ‑ Falcon SQL Editor

## Lean Database Design

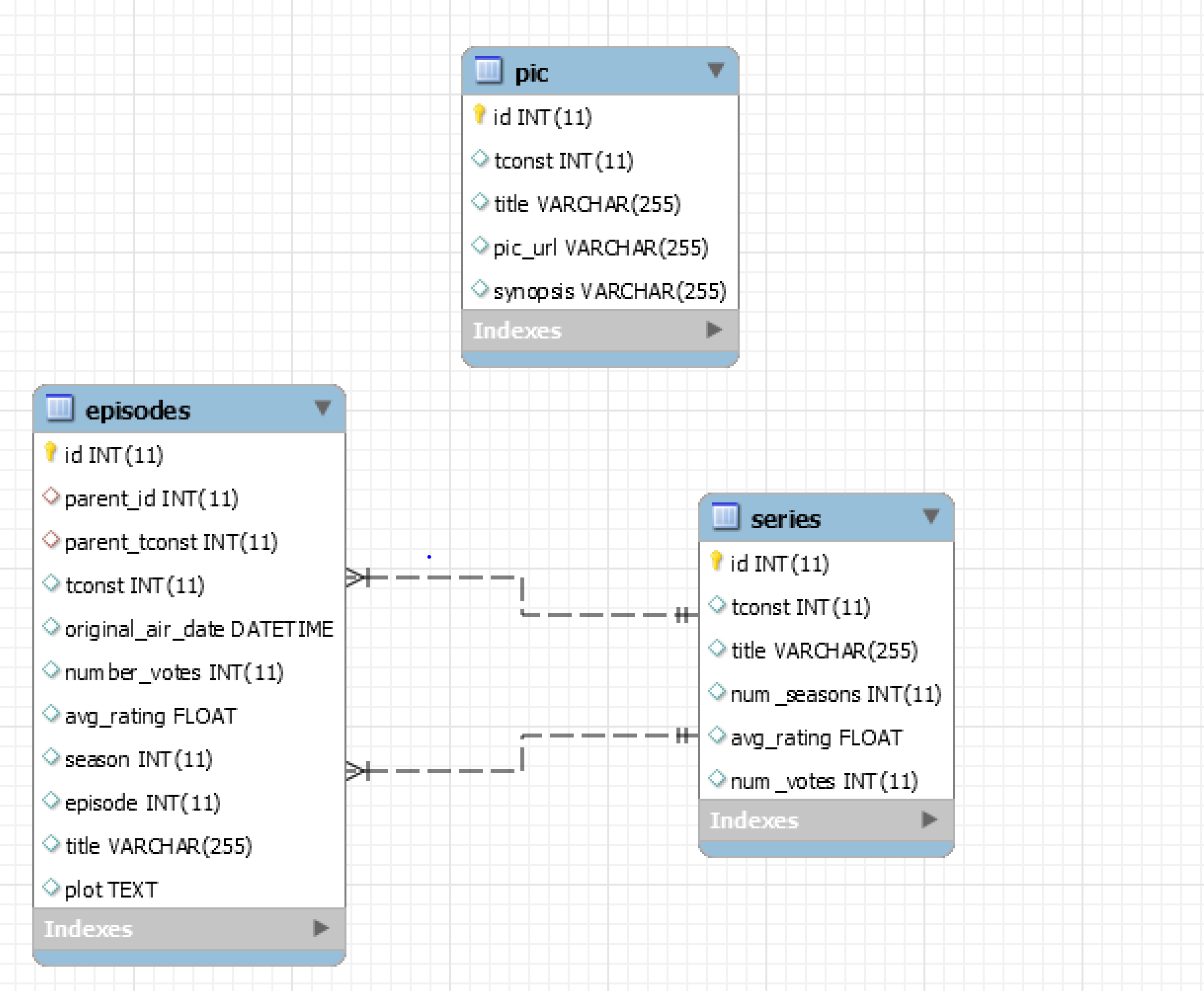


Figure ‑ Lean Database

To overcome the obstacles with the .gz files a lean version of the database was developed to power the visualizations on the site. This is a very light version of the original design in order to capture the required data and complete the graphs. There are three tables in the design: Series, Episodes, and Pic. Series and episode are relational to allow for more advanced plots. The pic table is non-relational as it is designed to power the carousel of our site. In hindsight, and in the future, the pic table will possibly be eliminated as the “cover url” info can be added to the series table. Due to time constraints the update was not made.

# Database Insight

## Mining the Database for Insights

With a fully loaded relationship database, we were able to mine hidden gems located within IMDB database to create data visualization models. Leveraging our structured query language skillsets, we were able to build complex queries [Figure 4‑1] to give us insight on data such as the actor-to-actress ratio across series [Figure 4‑2], the breakdown of job categories, and writers of the highest rated episodes [Figure 4‑3].

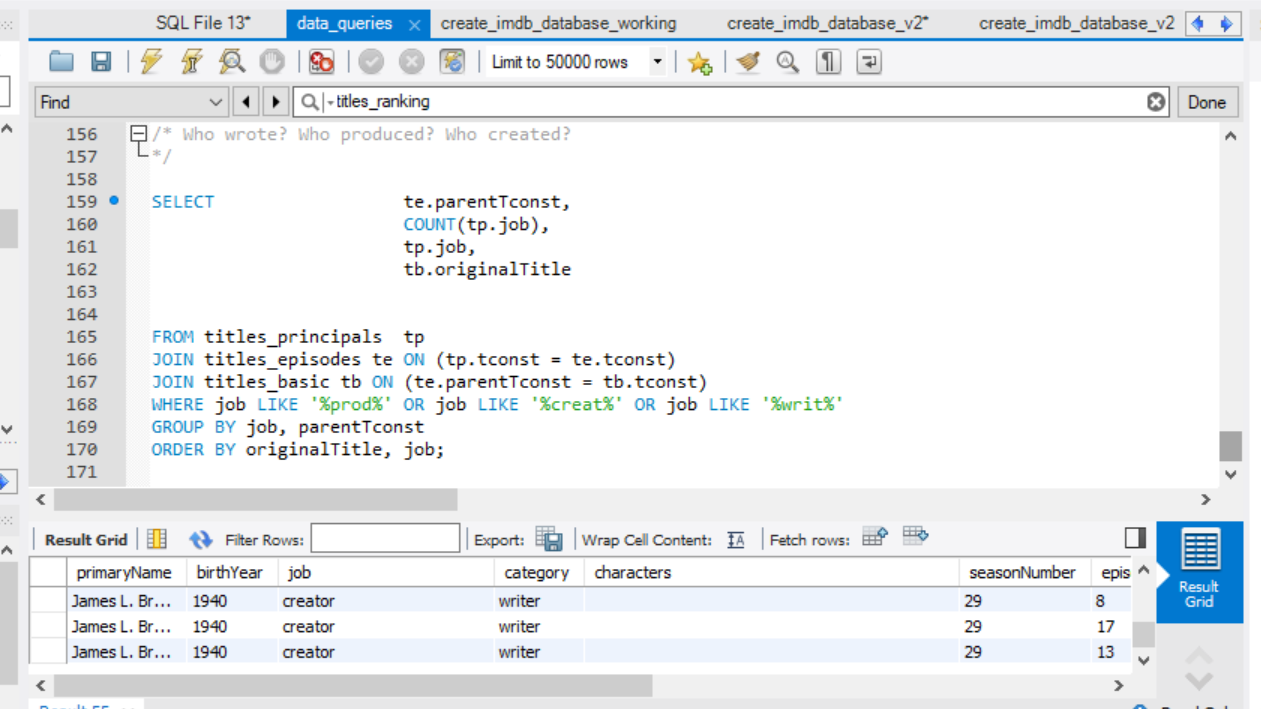


Figure ‑ Sample Query

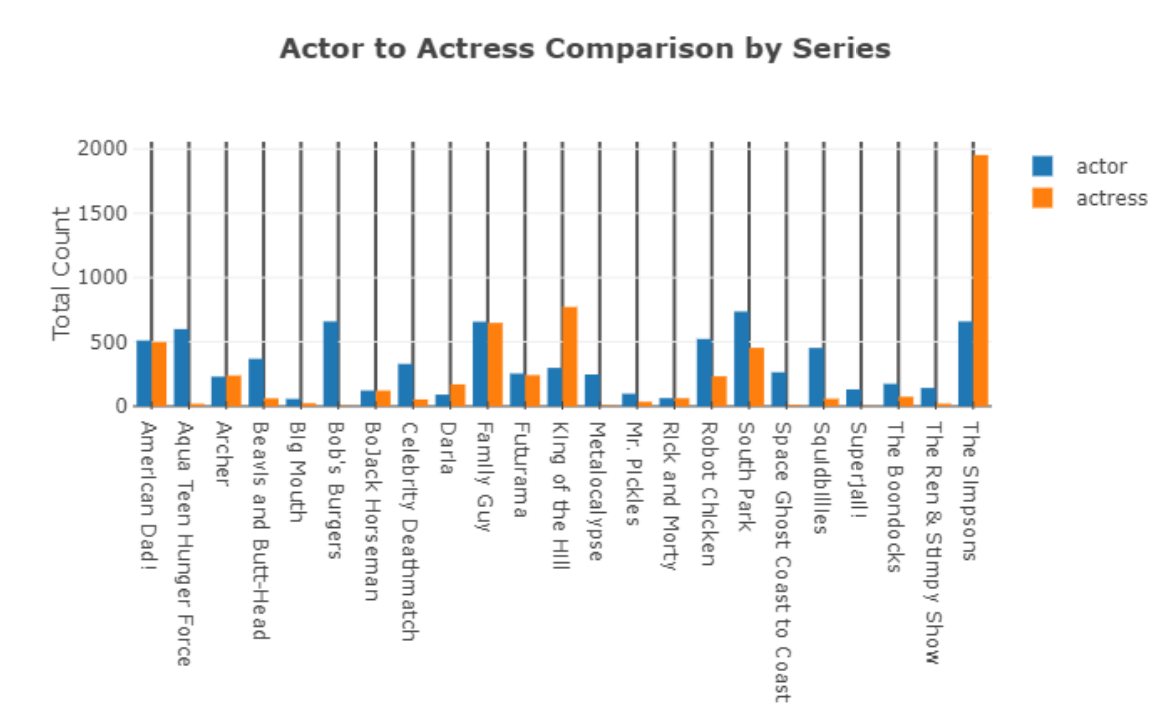


Figure ‑ Actor to Actress Bar Graph

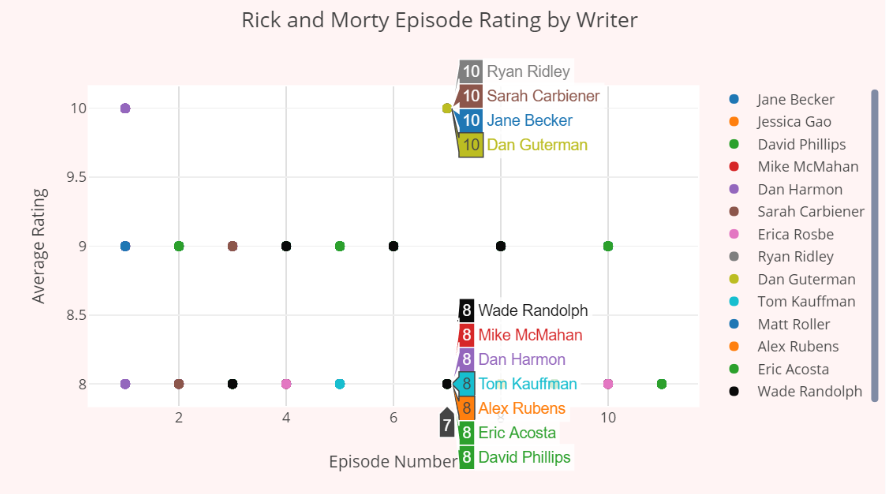


Figure ‑ Episode Writing by Writer

# IMDB-Py Extraction Processes

## Jupyter Notebooks

To begin our project the Jupyter notebooks were used to test out the use and functionality of IMDbPY and the extraction of the .gz files. “Playground” files were created for our series, episodes, and pic tables in order to grab the necessary info from the IMDb API. Once table data was stored in data frames we then exported the data as .csv files for testing.



Figure ‑ Series Table Try-Except Logic

In Figure 5‑1 we can see an example of the logic used to pull info from the IMDb API. Try and except blocks were used in order to account for NaN or “/n” values. The except blocks replaced those values with something easier to store in a database. Most often this was a “none” value which left the cell blank. One issues encountered in the .gz files were a large amount of ‘/n’ values which sat in the spots where no data was provided via IMDb’s crowd sourced information.

## Sample Data Scripts

Sample data scripts were written for series and episodes tables. These scripts output the series data and episodes data into csv for testing.

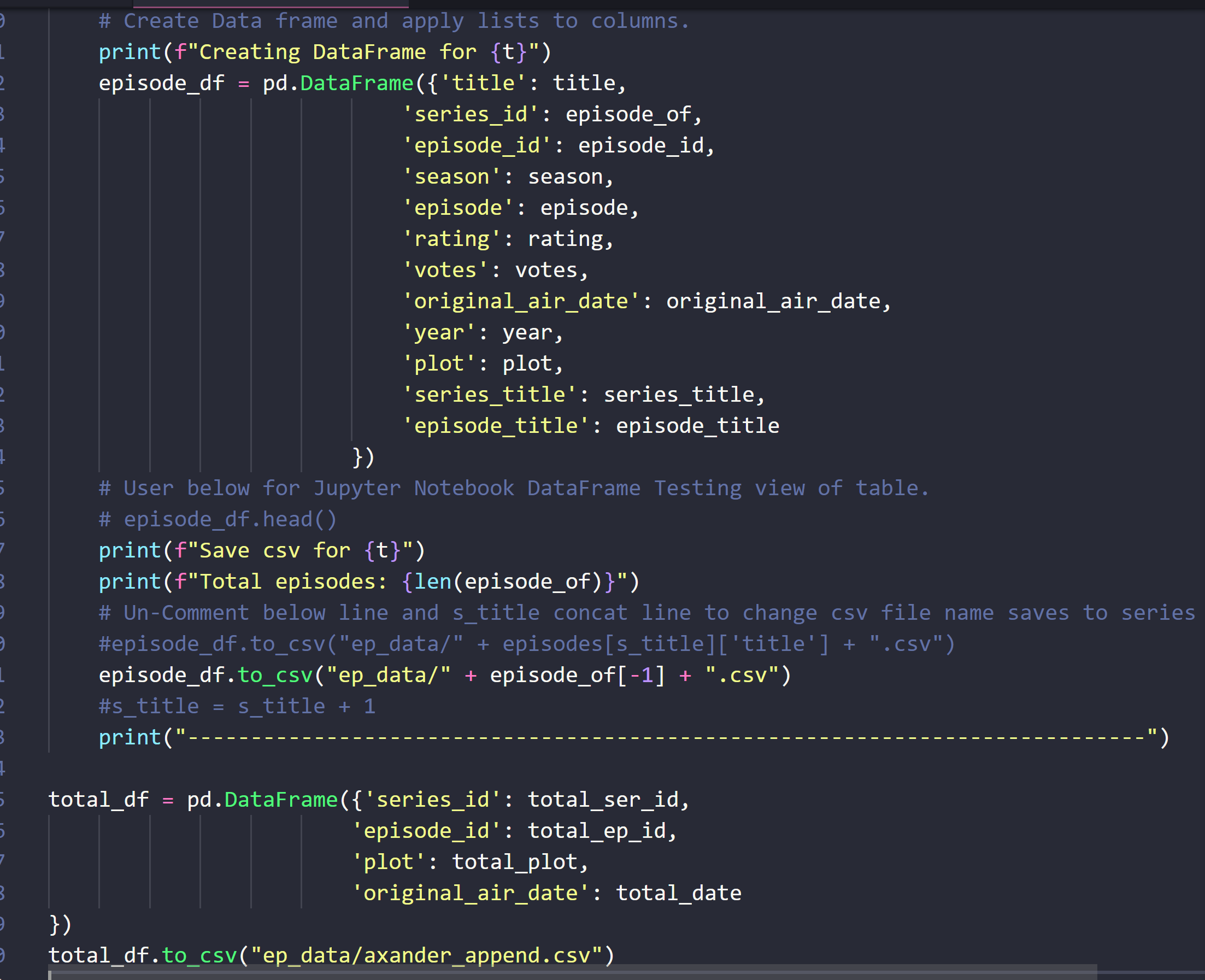


Figure ‑ Series Table Try-Except Logic

In Figure 5‑2 we can see a snippet of the episodes table script which grabbed the original air date of each episode as well as other information necessary for the table. Something to note is that each date captured from the API was originally pulled as a string. When the sample data was originally exported it was proven impossible to plot these dates as strings could be plotted. A library was imported called parser.parse to convert the dates to date-time objects and the table column was declared as a date-time type when uploading to MySQL.

# 

# Findings and Visualizations

The mid 90s marked the height of popularity for the Simpsons. User ratings and number of votes show a clear peak during these years [Figure 6‑1]. While many fans refer to more recent episodes as "Zombie Simpsons". Gone is the sharply timed, layered humor and poignant social commentary. Filmmaker John Walsh cited a changing writing staff as a large contributing factor for the decline in quality.

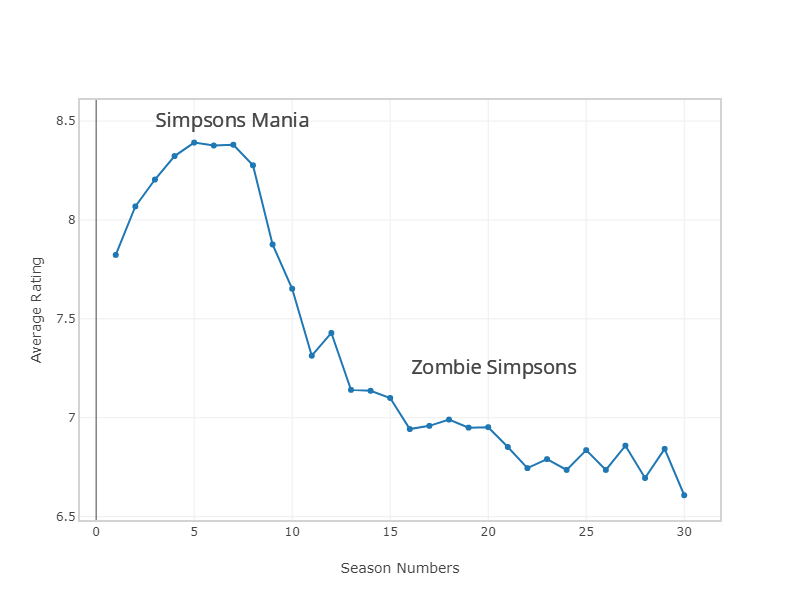


Figure ‑ Zombie Simpsons

Comparing *The Simpsons* to its main competitors, *South Park* and *Family Guy*, a similar trend can be seen in ratings [Figure 6‑2]. Though the drop is not as sharp as *The Simpsons*' Season 8. All of these series were successful in part because they pushed boundaries. And all of them have waned or declined in popularity after several years.

A more modern example would be the enormously popular *Rick and Morty*. As of this writing, the show is due to be renewed for its fourth season and more. *Adult Swim* has ordered 70 new episodes from its creators. Though it's impossible to predict how long or how successful a run the series will have, it's safe to say that Rick and Morty mania won't last forever.

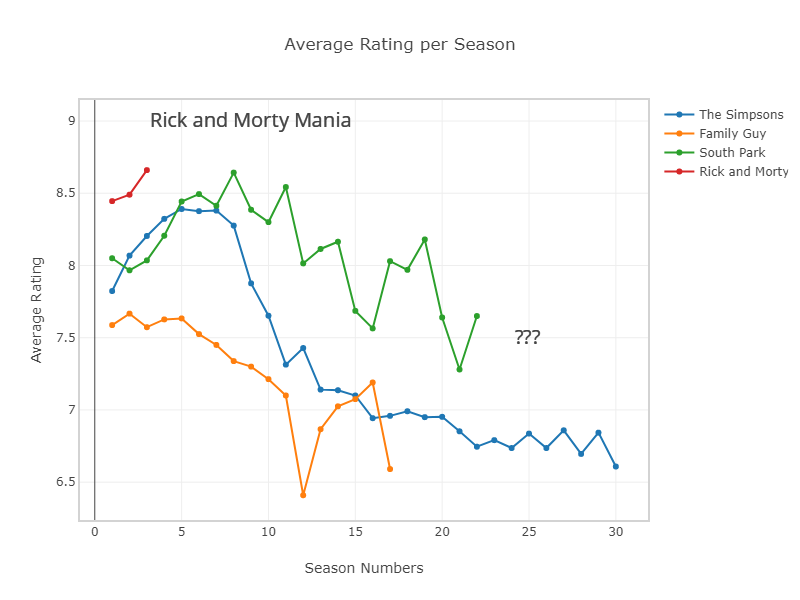


Figure ‑ Rick and Morty Mania

"Life of Brian" is *Family Guy's* lowest IMDb user-rated episode [Figure 6‑3]. It is the sixth episode of the twelfth season and originally aired November 24, 2013. Generally well received by critics, the episode generated a severe backlash from fans. It featured some touching moments where the beloved Brian is killed and the rest of the family is forced to hold a tearful funeral. Fans responded by starting a petition on *Change.org* to have the character revived. The petition succeeded and two episodes later, Brian is rescued from death, by Stewie, using a time machine in "Christmas Guy".

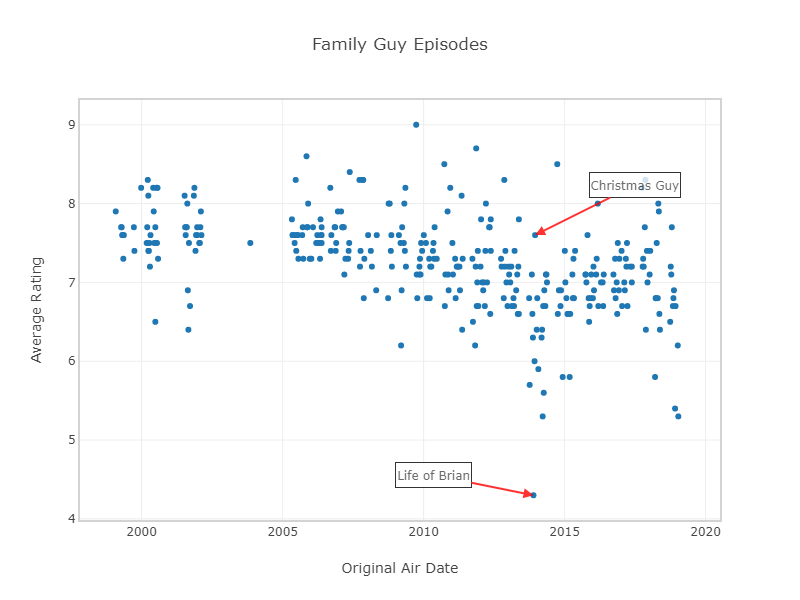


Figure ‑ Life of Brian