Sci-Fi Movie Collection

IST-659

Toby Anderson

03/26/2021

Contents

[Summary 2](#_Toc67746491)

[Stakeholders 3](#_Toc67746492)

[Business Rules 3](#_Toc67746493)

[Data Questions 3](#_Toc67746494)

[Conceptual Model 4](#_Toc67746495)

[Logical Model 5](#_Toc67746496)

# Summary

Artsforge.com keeps a curated list of the top 100 Sci-Fi films. These films are submitted by users then reviewed by the website owner before being added to their “Best Sci-Fi Films” list. However, currently the movies are kept in a plain HTML list. There are no attributes collected on the movies, such as release date or length, and there are no connections between the movie and the writer, author, or actors. By structuring this movie list into a database we will be able to gain more insight into these movies and possibly discover what we should be watching next!

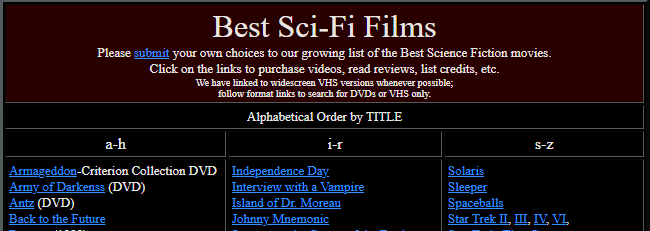


Figure 1 – Sample of Artsforge list of Best Sci-Fi movies

## Stakeholders

The stakeholder to the project is the author, who will benefit from the project by gaining new movie recommendations.

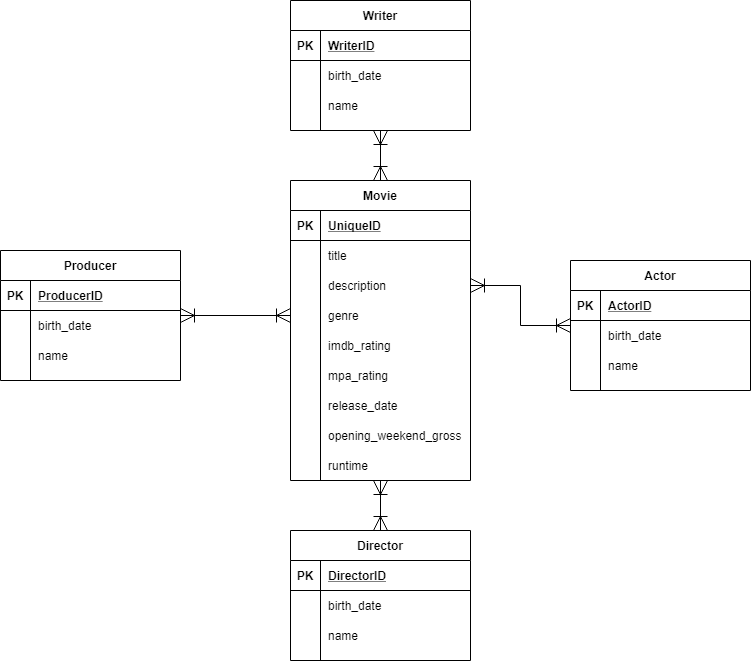
## Business Rules

* A movie is written by one or more writers.
* A writer writes one or more movies.
* A movie is directed by one or more directors
* A director directs one or more movies.
* A movie has one or more actors.
* An actor stars in one or more movies.
* A movie belongs to one or more genres.
* A genre includes one or more movies.

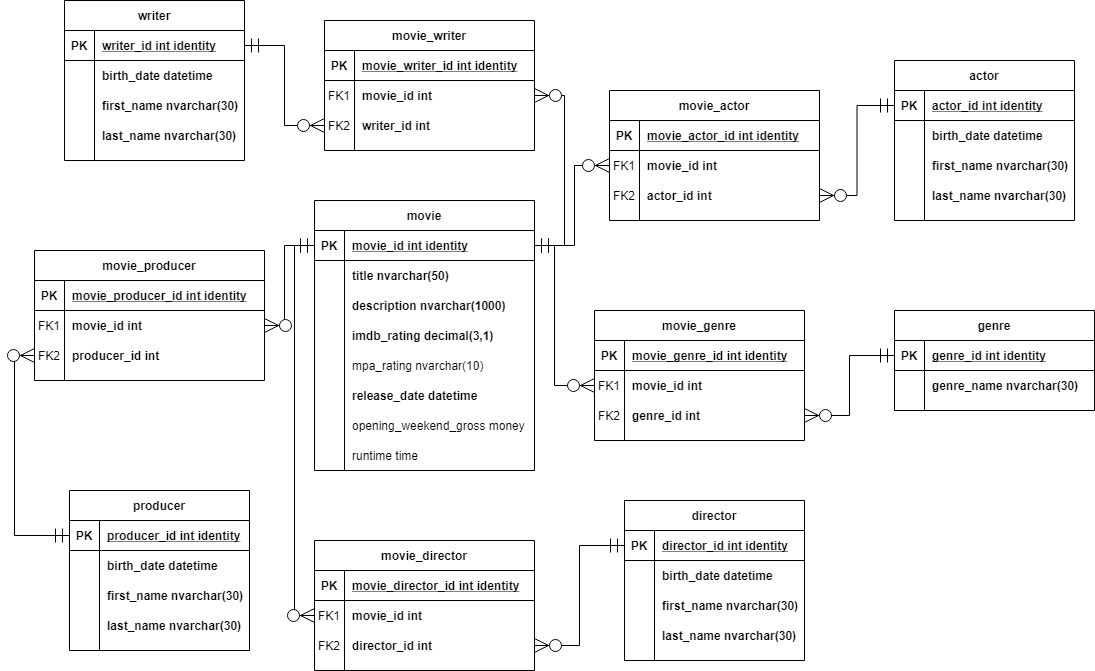
## Data Questions

* Which actor has starred in the most “top 100” sci-fi movies?
* Besides “Sci-Fi”, what additional genres are prevalent in this list?
* Which director has the highest average IMDB rating on their movies?
* If the stakeholder likes a particular movie, are there any movies by the same writer?
* Do older or newer movies have a higher average IMDB rating?

## Conceptual Model

****

## Logical Model

****

## Physical Database Design

--Creating primary tables

IF OBJECT\_ID('dbo.sf\_director') IS NOT NULL

DROP TABLE dbo.sf\_director

CREATE TABLE [dbo].[sf\_director](

[director\_ID] [int] identity primary key,

[director\_name] [nvarchar](50) NOT NULL,

)

IF OBJECT\_ID('dbo.sf\_actor') IS NOT NULL

DROP TABLE dbo.sf\_actor

CREATE TABLE [dbo].[sf\_actor](

[actor\_ID] [int] identity primary key,

[actor\_name] [nvarchar](50) NOT NULL,

)

IF OBJECT\_ID('dbo.sf\_genre') IS NOT NULL

DROP TABLE dbo.sf\_genre

CREATE TABLE [dbo].[sf\_genre](

[genre\_ID] [int] identity primary key,

[genre\_name] [nvarchar](50) NOT NULL,

)

IF OBJECT\_ID('dbo.sf\_production') IS NOT NULL

DROP TABLE dbo.sf\_production

CREATE TABLE [dbo].[sf\_production](

[production\_ID] [int] identity primary key,

[production\_name] [nvarchar](50) NOT NULL,

)

IF OBJECT\_ID('dbo.sf\_writer') IS NOT NULL

DROP TABLE dbo.sf\_writer

CREATE TABLE [dbo].[sf\_writer](

[writer\_ID] [int] identity primary key,

[writer\_name] [nvarchar](50) NOT NULL,

)

IF OBJECT\_ID('dbo.sf\_movie') IS NOT NULL

DROP TABLE dbo.sf\_movie

CREATE TABLE [dbo].[sf\_movie](

[imdbID] [nvarchar](50) primary key,

[Title] [nvarchar](255) NOT NULL,

[Year] [nvarchar](10) NOT NULL,

[Rated] [nvarchar](50) NOT NULL,

[Plot] [nvarchar](1000) NOT NULL,

[Metascore] [nvarchar](10) NULL,

[imdbVotes] [int] NULL,

[imdbRating] [nvarchar](10) NULL,

)

--Creating associative tables

IF OBJECT\_ID('dbo.sf\_movie\_actor') IS NOT NULL

DROP TABLE dbo.sf\_movie\_actor

CREATE TABLE [dbo].[sf\_movie\_actor](

[ActorMovieID] [int] NOT NULL,

[imdbID] [nvarchar](50) NOT NULL,

[actor\_id] [int] NULL,

CONSTRAINT [PK\_sf\_actor] PRIMARY KEY CLUSTERED ([ActorMovieID]),

CONSTRAINT [FK\_actor\_id] FOREIGN KEY([actor\_id]) REFERENCES [dbo].[sf\_actor] ([actor\_ID])

)

IF OBJECT\_ID('dbo.sf\_movie\_director') IS NOT NULL

DROP TABLE dbo.sf\_movie\_director

CREATE TABLE [dbo].[sf\_movie\_director](

[DirectorMovieID] [int] NOT NULL,

[imdbID] [nvarchar](50) NOT NULL,

[director\_id] [int] NULL,

CONSTRAINT [PK\_sf\_director] PRIMARY KEY CLUSTERED ([DirectorMovieID]),

CONSTRAINT [FK\_director\_id] FOREIGN KEY([director\_id]) REFERENCES [dbo].[sf\_director] ([director\_ID]))

IF OBJECT\_ID('dbo.sf\_movie\_genre') IS NOT NULL

DROP TABLE dbo.sf\_movie\_genre

CREATE TABLE [dbo].[sf\_movie\_genre](

[GenreMovieID] [int] NOT NULL,

[imdbID] [nvarchar](50) NOT NULL,

[genre\_id] [int] NULL,

CONSTRAINT [PK\_sf\_genre] PRIMARY KEY CLUSTERED ([GenreMovieID]),

CONSTRAINT [FK\_genre\_id] FOREIGN KEY([genre\_id]) REFERENCES [dbo].[sf\_genre] ([genre\_ID]))

IF OBJECT\_ID('dbo.sf\_movie\_production') IS NOT NULL

DROP TABLE dbo.sf\_movie\_production

CREATE TABLE [dbo].[sf\_movie\_production](

[ProductionMovieID] [int] NOT NULL,

[imdbID] [nvarchar](50) NOT NULL,

[production\_id] [int] NULL,

CONSTRAINT [PK\_sf\_production] PRIMARY KEY CLUSTERED ([ProductionMovieID]),

CONSTRAINT [FK\_production\_id] FOREIGN KEY([production\_id]) REFERENCES [dbo].[sf\_production] ([production\_ID]))

IF OBJECT\_ID('dbo.sf\_movie\_writer') IS NOT NULL

DROP TABLE dbo.sf\_movie\_writer

CREATE TABLE [dbo].[sf\_movie\_writer](

[WriterMovieID] [int] NOT NULL,

[imdbID] [nvarchar](50) NOT NULL,

[writer\_id] [int] NULL,

CONSTRAINT [PK\_sf\_writer] PRIMARY KEY CLUSTERED ([WriterMovieID]),

CONSTRAINT [FK\_writer\_id] FOREIGN KEY([writer\_id])

REFERENCES [dbo].[sf\_writer] ([writer\_ID]))

## Data Creation

--Actual data as imported as flat files after being scraped from the web.

--Below are some sample insertion queries if the data is added programmatically

INSERT INTO [dbo].[sf\_actor] ([actor\_name])

VALUES

('Adrian Brody'),('CJ Adams')

INSERT INTO [dbo].[sf\_director] ([director\_name])

VALUES

('Peter Weir'),('Luc Besson')

INSERT INTO [dbo].[sf\_genre] ([genre\_name])

VALUES

('Action'),('Documentary')

INSERT INTO [dbo].[sf\_production] ([production\_name])

VALUES

('Alliance'),('Silver Pictures')

INSERT INTO [dbo].[sf\_writer] ([writer\_name])

VALUES

('David Brin'),('Eric Roth')

INSERT INTO [dbo].[sf\_movie]

([imdbID]

,[Title]

,[Year]

,[Rated]

,[Plot]

,[Metascore]

,[imdbVotes]

,[imdbRating])

VALUES

('tt0021814'

,'Dracula'

,'1931'

,'PG13'

,'After a naive real estate agent succumbs to the will of Count Dracula, the two head to London where the vampire sleeps in his coffin by day and searches for potential victims by night.'

,'71'

,47025

,'7.5')

--Adding connections in the Associative tables

--To make the perfect movie

INSERT INTO [dbo].[sf\_movie\_actor]

([imdbID],[actor\_id])

VALUES

((SELECT imdbID from dbo.sf\_movie where title='Dracula')

,(SELECT actor\_id from dbo.sf\_actor where actor\_name='Gary Oldman')

--What'chu talkin' 'bout Dracula?

INSERT INTO [dbo].[sf\_movie\_director]

([imdbID],[director\_id])

VALUES

((SELECT imdbID from dbo.sf\_movie where title='Dracula')

,(SELECT director\_id FROM dbo.sf\_director WHERE director\_name='Michael Bay'))

--“Add another CGI scene where the bat blows up an office building”

INSERT INTO [dbo].[sf\_movie\_genre]

([imdbID],[genre\_id])

VALUES

((SELECT imdbID from dbo.sf\_movie where title='Dracula')

,(SELECT genre\_id from dbo.sf\_movie\_genre where genre\_name='Comedy'))

--Why else would we cast Gary Coleman?

INSERT INTO [dbo].[sf\_movie\_production]

([imdbID],[production\_id])

VALUES

((SELECT imdbID from dbo.sf\_movie where title='Dracula')

,(SELECT production\_id from dbo.sf\_production where production\_name='Walt Disney Pictures'))

--We can finally have that Dracula, Luke Skywalker, Spiderman crossover we've been waiting for

INSERT INTO [dbo].[sf\_movie\_writer]

([imdbID],[writer\_id])

VALUES

((SELECT imdbID from dbo.sf\_movie where title='Dracula')

,(SELECT writer\_id FROM dbo.sf\_writer where writer\_name='Gene Roddenberry'))

--I hope you're ready to go where no vampire has gone before

## Data Manipulation

--Some examples of fixing data errors

--Politician was inserted as an actor

DELETE FROM dbo.sf\_actor

WHERE actor\_name = 'Bill Clinton'

--Actor's name was mispelled

UPDATE dbo.sf\_actor

SET actor\_name = 'Gary Coleman'

WHERE actor\_name = 'Bary Coleman'

--Gene Roddenberry didn't actually write Dracula

DELETE from dbo.sf\_movie\_writer

WHERE w.writer\_name = 'Gene Roddenberry'

AND m.Title = 'Dracula'

JOIN dbo.sf\_writer w

ON w.writer\_id = dbo.sf\_movie\_writer.writer\_id

JOIN dbo.sf\_movie m

ON m.imdbID = dbo.sf\_movie\_writer.imdbID

## Answering Data Questions

-- Which actors has starred in the most sci-fi movies?

SELECT TOP 5 a.actor\_name,

count(m.[imdbID]) num\_movies

FROM [IST659\_M400\_tander10].[dbo].[sf\_movie] m

JOIN dbo.sf\_movie\_actor ma

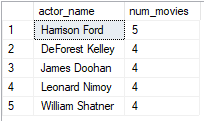
ON ma.imdbID = m.imdbID

JOIN dbo.sf\_actor a

ON a.actor\_ID = ma.actor\_id

GROUP BY a.actor\_name

ORDER BY num\_movies desc



-- Besides “Sci-Fi”, what additional genres are prevalent in this list?

SELECT TOP 5 g.genre\_name,

count(m.[imdbID]) num\_movies

FROM [IST659\_M400\_tander10].[dbo].[sf\_movie] m

JOIN dbo.sf\_movie\_genre mg

ON mg.imdbID = m.imdbID

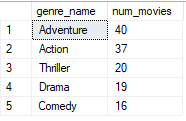
JOIN dbo.sf\_genre g

ON g.genre\_id = mg.genre\_id

WHERE genre\_name <> 'Sci-Fi'

GROUP BY g.genre\_name

ORDER BY num\_movies desc



-- Which director has the highest average IMDB rating on their movies?

SELECT TOP 5 d.director\_name,

avg(CAST(m.[imdbRating] AS float)) avg\_rating

FROM [IST659\_M400\_tander10].[dbo].[sf\_movie] m

JOIN dbo.sf\_movie\_director md

ON md.imdbID = m.imdbID

JOIN dbo.sf\_director d

ON d.director\_id = md.director\_id

GROUP BY d.director\_name

ORDER BY avg\_rating desc



-- I sure love "The Time Machine", any movies by the same writer?

SELECT DISTINCT m.Title

FROM [IST659\_M400\_tander10].[dbo].[sf\_movie] m

JOIN dbo.sf\_movie\_writer mw

ON mw.imdbID = m.imdbID

JOIN

(SELECT mw.writer\_id FROM dbo.sf\_movie\_writer mw

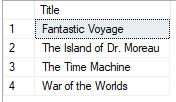
JOIN dbo.sf\_movie m

ON mw.imdbID = m.imdbID

WHERE m.Title = 'The Time Machine'

) mw2

ON mw2.writer\_id = mw.writer\_id



-- Do older or newer movies have a higher average IMDB rating?

SELECT year\_range, avg(CAST(imdbRating As float)) as avg\_rating

FROM (

SELECT [year], imdbRating,

CASE

WHEN CAST([year] AS int) < 1961 THEN '1930-1960'

WHEN CAST([year] AS int) < 1980 THEN '1960-1980'

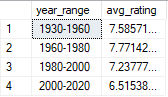
WHEN CAST([year] AS int) < 2000 THEN '1980-2000'

ELSE '2000-2020' END AS year\_range

FROM dbo.sf\_movie

) T1

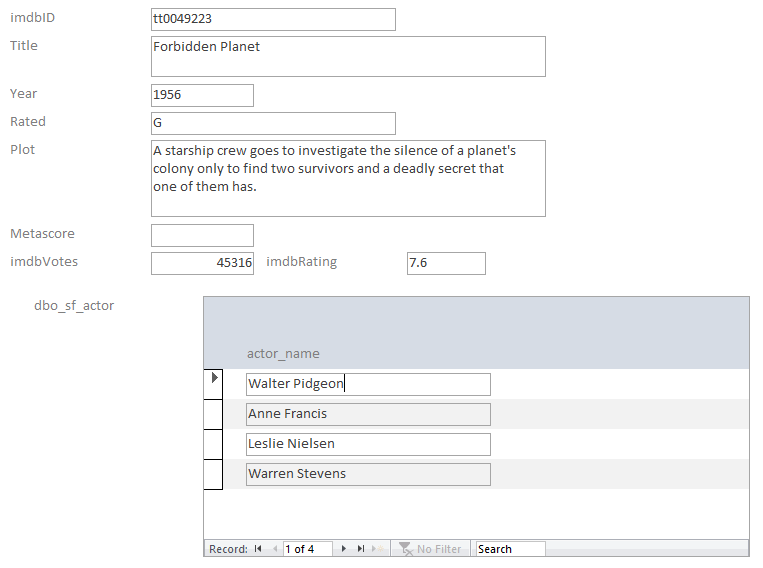
GROUP by year\_range



--Looks like 2000s started going down hill

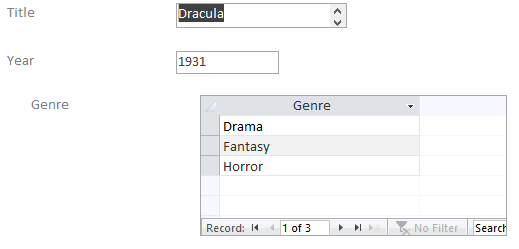
## Implementation

Here we have a form to insert or update movies and actors

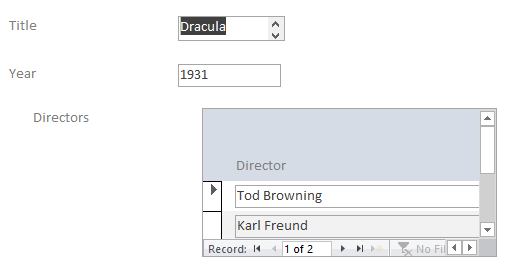


We then have separate forms for modifying director, genrea, production, and writer relationships to the movies.

Genre form:



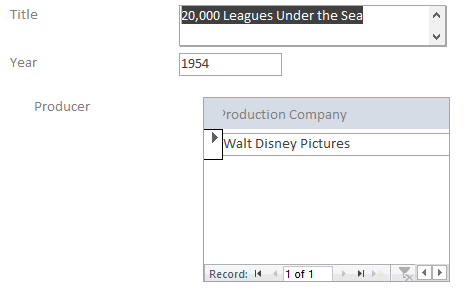
Director form:



Writer form:



Producer form:



## Reflection

In retrospect, I should have chosen a different dataset. I picked the top 100 Sci-Fi movies because it was ripe for analysis, but I think work on such a static dataset is better suited for R or Python.

Creating a database is needed for dynamic data with multiple users inserting, updating, and querying. Picking data around a process would have fit this assignment better, because I would have a more sensible end-user and realistic transactions in terms of forms and procedures.

A static dataset does not require users to insert and update frequently, so the front-end implementation doesn’t really make sense. Additionally, this made 3rd normal form less useful, since the risk of insertion anomalies is negligible on a static dataset. This design decision needlessly complicated the database and the query process.

## Summary

We were able to use SQL to analyze complex data, which included relationships between multiple entities, including writers, directors, actors, and producers. By using SQLs built-in capabilities to link together tables using primary and foreign keys, we were able to seamlessly join together multiple data sources to get insightful answers into our favorite movies.

This solution could potentially be scaled up to include several thousand movies, and be used by a recommender system that tracked users favorite movies, and provided similar movies they may like.