

**Table of Contents**

[**Part One** 3](#_Toc90329074)

[**Summary** 3](#_Toc90329075)

[**Stakeholders** 3](#_Toc90329076)

[**Business Rules** 3](#_Toc90329077)

[**Data Questions** 3](#_Toc90329078)

[**Conceptual Model** 4](#_Toc90329079)

[**Logical Model** 5](#_Toc90329080)

[**Part Two** 6](#_Toc90329081)

[**Physical Database Design** 6](#_Toc90329082)

[**Data Creation** 8](#_Toc90329083)

[**Data Manipulation** 14](#_Toc90329084)

[**Answering Data Questions** 16](#_Toc90329085)

[**Implementation** 21](#_Toc90329086)

[**Reflection** 22](#_Toc90329087)

[**Summary** 22](#_Toc90329088)

# **Part One**

# **Summary**

Friday movie night is a ritual in my household but identifying the movie that gains consensus is difficult. Currently, a list is generated by incorporating a reasonable number of everyone’s wishes and then each individual takes turns eliminating the movie they least prefer. Finding options that everyone enjoys and then retaining the list of acceptable movies from week to week requires efforts beyond what is currently done.

Additionally, my 10 year old daughter believes that there is foul play and that her selections never win and are instead supplanted by my husband who prefers period dramas and war movies.

# **Stakeholders**

* Current Core Family Members
  + Includes myself, my husband, and my daughter
* Future Core Family Members
  + Includes the potential for future children or eventual introduction of “in laws” for relationships by my daughter
* Visiting extended family or friends identified as significant
  + A selection of visiting immediate family identified as parents, grandparents, aunts, uncles or cousins as well as “best friends” of my daughter considered for inclusion in the family

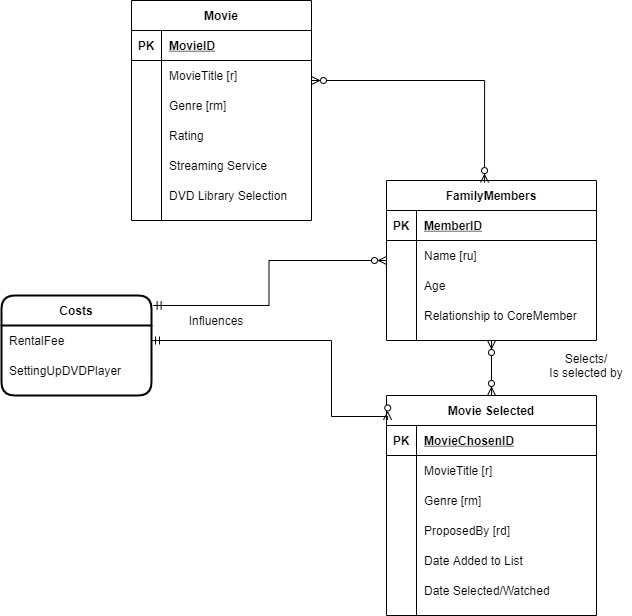
# **Business Rules**

* Movies will include feature length movies or documentaries available on existing streaming services or within available DVDs
* Selections must be appropriate for all ages and can be eliminated if the governing body of the movie night facilitation, myself, deems it inappropriate for children
* Rental fees may be taken into consideration for inclusion on the list
* Certain times per year may necessitate a thematic search, for example Halloween or Christmas movies may be a focused theme during that time of the year.
* Other business rules will be defined in later iterations of the database.

# **Data Questions**

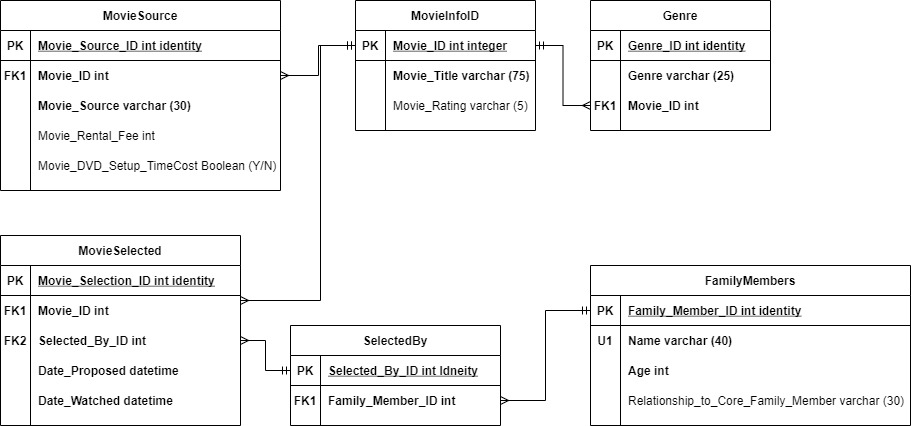
* What is the current list of movies nominated for watching?
* What genre of movie is predominantly selected?
* What is the distribution of movie watched to who selected it?
* Do selected movies originate predominantly from specific streaming services?

# **Conceptual Model**



# 

# **Logical Model**



Date \_Watched datetime

# **Part Two**

# **Physical Database Design**

/\*

Author: Jennifer Williams

Title: ProjectPart2

Database Formation

\*/

--Deleting existing copies of the Table

DROP TABLE MovieInfoID;

--Creating the MovieInfoID Table

CREATE OR ALTER TABLE MovieInfoID(

--Columns for the MovieInfoID Table

Movie\_ID int identity,

Movie\_Title varchar (75) NOT NULL,

Movie\_Rating varchar (5),

--Constraints on the MovieInfoID Table

CONSTRAINT PK\_MovieInfoID PRIMARY KEY (Movie\_ID)

)

/\*

This table establishes parameters that assumes that movie rating

is not required as some movies are not rated. Some corrections needed

to be made as well because the movie title could not contain punctuation

that SQL would try to recognize as commands. The Movie\_ID is an int identity

field that will automatically populate and advance. It and Movie\_Title are

required but the title does not need to be unique.

\*/

--End creating the MovieInfoID Table

--Deleting existing copies of the Table

DROP TABLE FamilyMembers;

--Creating the FamilyMembers Table

CREATE OR ALTER TABLE FamilyMembers(

--Columns for the FamilyMembers Table

Family\_Member\_ID int identity,

Family\_Member\_Name varchar (40) NOT NULL,

Age int NOT NULL,

Relationship\_to\_Core\_Family\_Member varchar (30),

--Constraints on the FamilyMembers Table

CONSTRAINT PK\_FamilyMembers PRIMARY KEY (Family\_Member\_ID),

CONSTRAINT U1\_FamilyMembers UNIQUE (Family\_Member\_Name),

)

/\*

This table establishes parameters that assumes that relationship to a core

Family member is not required. The Family Member ID is an int identity

field that will automatically populate and advance. Modifications needed to be

made to the title of the name field, changing it to Family\_Member\_Name as it was

related to a SQL codeword already. The ID and Family Member Name are

required and must be unique.

\*/

--End creating the FamilyMembers Table

--Deleting existing copies of the Table

DROP TABLE MovieSource;

--Creating the MovieSource Table

CREATE OR ALTER TABLE MovieSource(

--Columns for the MovieSource Table

Movie\_Source\_ID int identity,

Movie\_ID int NOT NULL,

Movie\_Source varchar(30) NOT NULL,

Movie\_Rental\_Fee int,

Movie\_DVD\_Setup\_TimeCost BIT,

--Constraints on the MovieSource Table

CONSTRAINT PK\_Movie\_Source\_ID PRIMARY KEY (Movie\_Source\_ID),

CONSTRAINT FK1\_MovieSource FOREIGN KEY (Movie\_ID) REFERENCES MovieInfoID(Movie\_ID)

)

/\*

This table establishes parameters that assumes that Movie Rental Fee and DVD

Setup costs are not required. Movie Source ID is an int identity field that

will automatically populate and advance. The Movie Source ID, Movie ID and Movie Source are required but that Movie Source does not need to be unique.

\*/

--End Creating MovieSource Table

--Deleting existing copies of the Table

DROP TABLE Genre;

--Creating the Genre Table

CREATE OR ALTER TABLE Genre(

--Columns for the Genre Table

GenreID int identity,

Genre varchar(20) NOT NULL,

Movie\_ID int NOT NULL,

--Constraints on the Genre Table

CONSTRAINT PK\_Genre PRIMARY KEY (GenreId),

CONSTRAINT FK1\_Genre FOREIGN KEY (Movie\_ID) REFERENCES MovieInfoID(Movie\_ID),

)

/\*

This table establishes parameters for creation of this table. Genre ID is an int identity field that will automatically populate and advance. The Genre ID, Movie ID, and Genre are required but that Genre does not need to be unique.

\*/

--End creating the Genre Table

--Deleting existing copies of the Table

DROP TABLE SelectedBy;

--Creating the SelectedBy Table

CREATE OR ALTER TABLE SelectedBy(

--Columns for the SelectedBy Table

Selected\_By\_ID int identity,

Family\_Member\_ID int NOT NULL,

--Constraints on the SelectedBy Table

CONSTRAINT PK\_Selected PRIMARY KEY (Selected\_By\_ID),

CONSTRAINT FK1\_SelectedBy FOREIGN KEY (Family\_Member\_ID) REFERENCES FamilyMembers(Family\_Member\_ID)

)

/\*

This table establishes parameters for creation of this table. Selected By ID is an int identity field that will automatically populate and advance. The Selected By ID and Family Member ID are required but that Genre and unique.

\*/

--End creating the SelectedBy Table

--Deleting existing copies of the Table

DROP TABLE MovieSelected;

--Creating the MovieSelected Table

CREATE OR ALTER TABLE MovieSelected(

--Columns for the MovieSelected Table

Movie\_Selection\_ID int identity,

Movie\_ID int NOT NULL,

Selected\_By\_ID int NOT NULL,

Date\_Proposed datetime NOT NULL,

Date\_Watched datetime,

--Constraints on the MovieSelected Table

CONSTRAINT PK\_Movie\_Selection\_ID PRIMARY KEY (Movie\_Selection\_ID),

CONSTRAINT FK1\_MovieSelected FOREIGN KEY (Movie\_ID) REFERENCES MovieInfoID(Movie\_ID),

CONSTRAINT FK2\_MovieSelected FOREIGN KEY (Selected\_By\_ID) REFERENCES SelectedBy(Selected\_By\_ID)

)

/\*

This table establishes parameters that assumes that Date Watched is not a required field. Movie Selection ID is an int identity field that will automatically populate and advance. The Movie Selection ID and Movie ID are required and unique while Selected By ID and Date Proposed are required and not unique and finally, Date Watched is not required and not required to be unique.

\*/

--End Creating MovieSelected Table

# **Data Creation**

/\*

Author: Jennifer Williams

Title: ProjectPart2

Data Creation and Manipulation

\*/

--Inserting Data into Family Members Table

INSERT INTO FamilyMembers(Family\_Member\_Name, Age, Relationship\_to\_Core\_Family\_Member)

VALUES

('Jennifer',42, ' '),

('Nathan',41, ' '),

('Reagan',10, ' '),

('Kyah',10,'BestFriend')

SELECT\*FROM FamilyMembers

/\*

Had the 'Family\_Member\_ID column not been a int identity automatically

completed column, that would have also been included int he insert statement

found on line 8 here. This is the similar approach to the other table creation

\*/

--End Inserting Data into Family Members Table

--Inserting Data into SelectedBy Table

INSERT INTO SelectedBy(Family\_Member\_ID)

VALUES

(1),

(2),

(3),

(4)

SELECT\*FROM SelectedBy

--End Inserting Data into Selected Table

--Inserting Data into MovieInfoID Table

INSERT INTO MovieInfoID(Movie\_Title, Movie\_Rating)

VALUES

('News of the World','PG-13'),

('The Ballad of Buster Scruggs','R'),

('The Dig','PG-13'),

('The Trial of the Chicago 7','R'),

('The Two Popes','PG-13'),

('Inglorious Bastards','R'),

('Stranger Things','TV-14'),

('Bridge Over the River Kwai','PG'),

('The Mitchells vs. the Machines','PG'),

('The Simpsons Movie','PG-13'),

('The Irishman','R'),

('Beasts of No Nation','TV-MA'),

('Deadpool','R'),

('Deadpool2','R'),

('Venom','PG-13'),

('Hamilton','PG-13'),

('Free Guy','PG-13'),

('Fantastic Mr. Fox','PG'),

('The Lighthouse','R'),

('Sound of Metal','R'),

('Snatch','R'),

('The Lost City of Z','PG-13'),

('Knives Out','PG-13'),

('I,Tonya','R'),

('La La Land','PG-13'),

('Ken Burns Baseball','TV-14'),

('Ken Burns The Civil War','TV-14'),

('Ken Burns Vietnam War','TV-14'),

('Gone Girl','R'),

('Napoleon Dynamite','R'),

('The Revenant','R'),

('Parasite','R'),

('The Fifth Element','PG-13'),

('Black Widow','PG-13'),

('Avatar','PG-13'),

('Wind','PG-13'),

('The Rocketeer','PG'),

('Guardians of the Galaxy','PG-13'),

('Isle of Dogs','PG-13'),

('Loki','TV-14'),

('Princess Bride','PG'),

('Free Solo','PG-13'),

('The Meg','PG-13'),

('Zero Dark Thirty','R'),

('Black Hawk Down','R'),

('The Dutchess','R'),

('Three Musketeers','PG-13'),

('Invictus','PG-13'),

('Outpost','R'),

('Atomic Blond','R'),

('A Quiet Place','R'),

('Tombstone','R'),

('Little Women','PG'),

('Hancock','PG-13'),

('Click','PG'),

('Cant Buy Me Love','PG-13'),

('Jurassic Park','PG-13')

SELECT\*FROM MovieInfoID

--End Inserting Data into MovieInfoID Table

--Inserting Data into MovieSource Table

INSERT INTO MovieSource (Movie\_ID, Movie\_Source, Movie\_Rental\_Fee, Movie\_DVD\_Setup\_TimeCost)

VALUES

(1,'Netflix',0,0),

(2,'Netflix',0,0),

(3,'Netflix',0,0),

(4,'Netflix',0,0),

(5,'Netflix',0,0),

(6,'Netflix',0,0),

(7,'Netflix',0,0),

(8,'Netflix',0,0),

(9,'Netflix',0,0),

(10,'Disney',0,0),

(11,'Netflix',0,0),

(12,'Netflix',0,0),

(13,'Hulu',0,0),

(14,'Hulu',0,0),

(15,'Disney',0,0),

(16,'Disney',0,0),

(17,'Netflix',0,0),

(18,'Amazon',3.99,0),

(19,'Amazon',0,0),

(20,'Amazon',0,0),

(21,'Amazon',0,0),

(22,'Amazon',0,0),

(23,'Amazon',0,0),

(24,'Hulu',0,0),

(25,'Hulu',0,0),

(26,'DVD',0,1),

(27,'DVD',0,1),

(28,'DVD',0,1),

(29,'Hulu',0,0),

(30,'Hulu',0,0),

(31,'Hulu',0,0),

(32,'Hulu',0,0),

(33,'Hulu',0,0),

(34,'Disney',0,0),

(35,'Disney',0,0),

(36,'Amazon',0,0),

(37,'Amazon',2.99,0),

(38,'Disney',0,0),

(39,'Disney',0,0),

(40,'Disney',0,0),

(41,'Disney',0,0),

(42,'Disney',0,0),

(43,'Hulu',0,0),

(44,'Amazon',0,0),

(45,'Amazon',0,0),

(46,'Amazon',0,0),

(47,'Amazon',0,0),

(48,'Netflix',0,0),

(49,'Amazon',0,0),

(50,'Amazon',0,0),

(51,'Netflix',0,0),

(52,'Amazon',0,0),

(53,'Amazon',3.99,0),

(54,'Disney',0,0),

(55,'Netflix',0,0),

(56,'Amazon',3.99,0),

(57,'Amazon',0,0)

SELECT\*FROM MovieSource

--End Inserting Data into MovieSource Table

--Inserting Data into MovieSelected Table

INSERT INTO MovieSelected(Movie\_ID, Selected\_By\_ID, Date\_Proposed, Date\_Watched)

VALUES

(3,1,44470,' '),

(4,1,44470,' '),

(1,2,44470,' '),

(6,2,44470,44470),

(2,3,44470,' '),

(9,3,44470,' '),

(5,1,44477,' '),

(7,1,44477,' '),

(8,2,44477,' '),

(11,2,44477,' '),

(10,3,44477,44477),

(18,4,44477,' '),

(12,1,44484,' '),

(13,1,44484,44484),

(19,2,44484,' '),

(21,2,44484,' '),

(17,3,44484,' '),

(23,3,44484,' '),

(14,1,44491,' '),

(15,1,44491,44491),

(22,2,44491,' '),

(26,2,44491,' '),

(25,3,44491,' '),

(33,3,44491,' '),

(16,1,44498,' '),

(20,1,44498,' '),

(27,2,44498,44498),

(28,2,44498,' '),

(35,3,44498,' '),

(39,3,44498,' '),

(24,1,44505,' '),

(32,1,44505,' '),

(29,2,44505,44505),

(30,2,44505,' '),

(41,3,44505,' '),

(43,3,44505,' '),

(34,1,44512,44512),

(38,1,44512,' '),

(31,2,44512,' '),

(36,2,44512,' '),

(54,3,44512,' '),

(55,3,44512,' '),

(40,1,44519,' '),

(42,1,44519,' '),

(37,2,44519,' '),

(44,2,44519,' '),

(56,3,44519,44519),

(46,1,44526,' '),

(47,1,44526,' '),

(45,2,44526,44526),

(49,2,44526,' '),

(57,3,44526,' '),

(48,1,44533,44533),

(53,1,44533,' '),

(50,2,44533,' '),

(51,2,44533,' '),

(52,2,44540,' ')

SELECT\*FROM MovieSelected

--End Inserting Data into MovieSelected Table

--Inserting Data into Genre Table

INSERT INTO Genre(Genre, Movie\_ID)

VALUES

('Action',1),

('Comedy',2),

('Biography',3),

('Drama',4),

('Comedy',5),

('Adventure',6),

('Fantasy',7),

('Adventure',8),

('Animation',9),

('Animation',10),

('Biography',11),

('Drama',12),

('Sci-Fi',13),

('Sci-Fi',14),

('Sci-Fi',15),

('Biography',16),

('Action',17),

('Animation',18),

('Drama',19),

('Drama',20),

('Comedy',21),

('Adventure',22),

('Comedy',23),

('Biography',24),

('Comedy',25),

('Documentary',26),

('Documentary',27),

('Documentary',28),

('Drama',29),

('Comedy',30),

('Drama',31),

('Horror',32),

('Sci-Fi',33),

('Sci-Fi',34),

('Fantasy',35),

('Action',36),

('Adventure',37),

('Sci-Fi',38),

('Animation',39),

('Fantasy',40),

('Adventure',41),

('Documentary',42),

('Action',43),

('Drama',44),

('Drama',45),

('Biography',46),

('Adventure',47),

('Biography',48),

('Action',49),

('Action',50),

('Horror',51),

('Action',52),

('Drama',53),

('Action',54),

('Comedy',55),

('Comedy',56),

('Action',57)

SELECT\*FROM Genre

--End Inserting Data into Genre Table

--Now I’m going to create a stored procedure for one of my tables to optimize data control

SET NOCOUNT ON

INSERT INTO MovieSelcted

(MovieSelected.Movie\_ID,

MovieSelected.Selected\_By\_ID,

MovieSelected.Date\_Proposed,

MovieSelected.Date\_Watched)

VALUES

(@Movie\_ID int,

@Selected\_By\_ID int,

@Date\_Proposed datetime,

@Date\_Watched datetime)

GO

EXECUTE INSERT\_dbo\_MovieSelected

@Movie\_ID='60',

@Selected\_By\_ID='3',

@Date\_Proposed = ‘15 Dec 2021’,

@Date\_Watched)= ’15 Dec 2021’,

# GO

# **Data Manipulation**

--Creating a view that effectively joins my disparate tables

--This view will allow me to control security and run queries

CREATE VIEW TotalPicture

AS SELECT MovieInfoID.Movie\_Title, MovieInfoID.Movie\_Rating, MovieSource.Movie\_Source, SelectedBy.Family\_Member\_ID,

FamilyMembers.Family\_Member\_Name, MovieSelected.Date\_Proposed, MovieSelected.Date\_Watched,

MovieSource.Movie\_Rental\_Fee, MovieSource.Movie\_DVD\_Setup\_TimeCost

FROM MovieInfoID, MovieSource, SelectedBy, FamilyMembers, MovieSelected

WHERE SelectedBy.Family\_Member\_ID=FamilyMembers.Family\_Member\_ID

AND MovieSelected.Selected\_By\_ID=SelectedBy.Selected\_By\_ID

AND MovieSource.Movie\_ID=MovieInfoID.Movie\_ID

AND MovieSource.Movie\_ID=MovieSelected.Movie\_ID

SELECT \* FROM TotalPicture

--Complete at making the joined view

--Creating User Accounts for my primary family members

--This will allow me to then control and limit their access to the database for security purposes

CREATE USER Reagan FOR LOGIN Reagan

CREATE USER Nate FOR LOGIN Nate

--Complete in creating the user accounts for primary family

--Grant UPDATE only permissions to newly created user accounts to total picture view

GRANT UPDATE ON TotalPicture to Reagan

GRANT UPDATE ON TotalPicture to Nate

--Complete in creating permissions associated with the accounts.

--Reviewing the business rules clearly indicate that the selection must be

--either a movie or documentary therefore, some of the items are in violation

SELECT \* FROM MovieInfoID WHERE MovieInfoID.Movie\_Rating LIKE '%TV%'

UPDATE MovieInfoID

SET Movie\_Title = 'NOT AUTHORIZED'

WHERE MovieInfoID.Movie\_Rating LIKE '%TV%'

--That appears to have removed our 6 affected rows and sadly removed loki and stranger things from our list of options

--Now I am going to attempt to create a procedure to update the date watched to

--a movie already in the database

--Therefore this is a procedure to update the MovieSelected.Date\_Watched based on

--the selected movie title

CREATE PROCEDURE FinallyWatched(@Movie\_Title varchar (75), @Date\_Watched datetime)

AS

BEGIN

UPDATE TotalPicture SET Date\_Watched=@Date\_Watched

WHERE Movie\_Title=@Movie\_Title

END

GO

--Now I'm "testing" the execution of the procedure with an execute update

EXEC FinallyWatched 'Black Widow', '14 Dec 21'

--With the message that one row has been updated, I verify by selecting all

--from the aforementioned db that meet my changed criteria.

SELECT \* FROM TotalPicture WHERE Movie\_Title='Black Widow'

--huzzah it worked.

--Now I'm going to update my security measures to allow my users to

--update on this stored procedure only

GRANT EXECUTE ON FinallyWatch TO Reagan

GRANT EXECUTE ON FinallyWatch TO Nate

# **Answering Data Questions**

* What is the current list of movies nominated for watching?

/\*

Author: Jennifer Williams

Title: ProjectPart2

Data Questions

\*/

--What is the current list of movies nominated for watching?

/\*

This is question that requires a select all statement followed

by a parameter that excludes the movies that have not yet been watched.

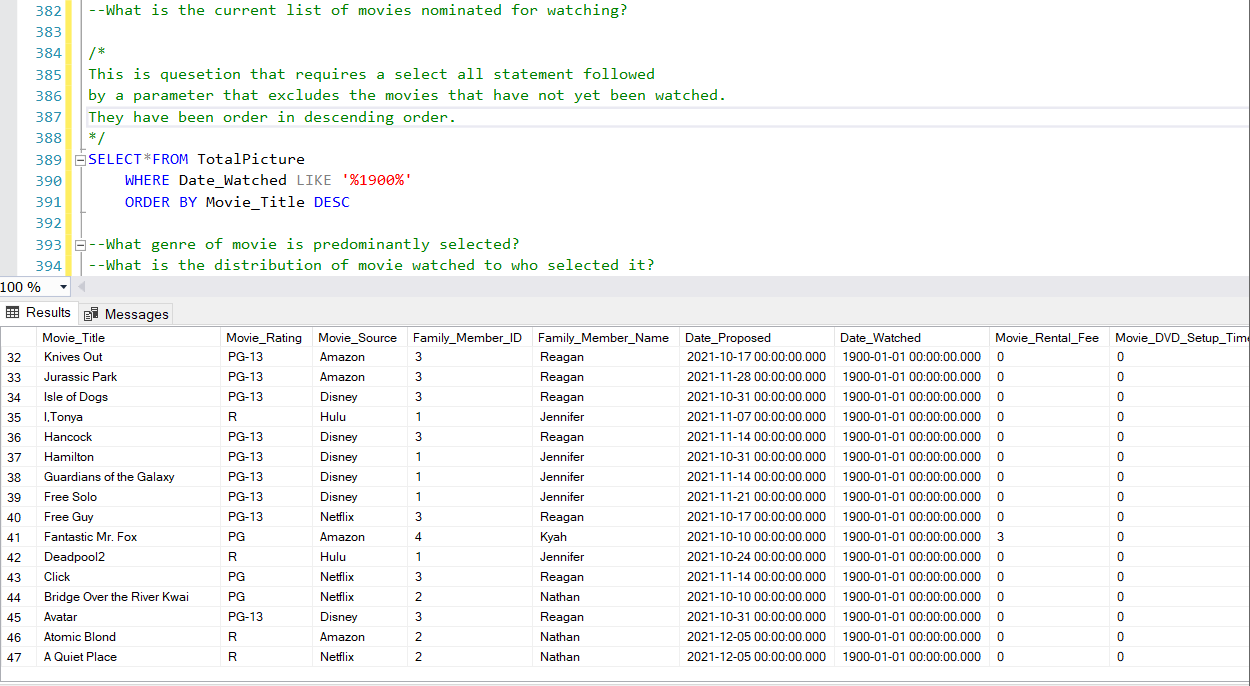
They have been order in descending order.

\*/

SELECT\*FROM TotalPicture

WHERE Date\_Watched LIKE '%1900%'

ORDER BY Movie\_Title DESC



* What genre of movie is predominantly selected?

--What genre of movie is predominantly selected?

--Create a statement to count the number of selected movies by Genre

--This capitalizes on the group by clause to make the count easier

SELECT

Genre,

COUNT(\*)

FROM

MovieGenre

GROUP BY

Genre

ORDER BY

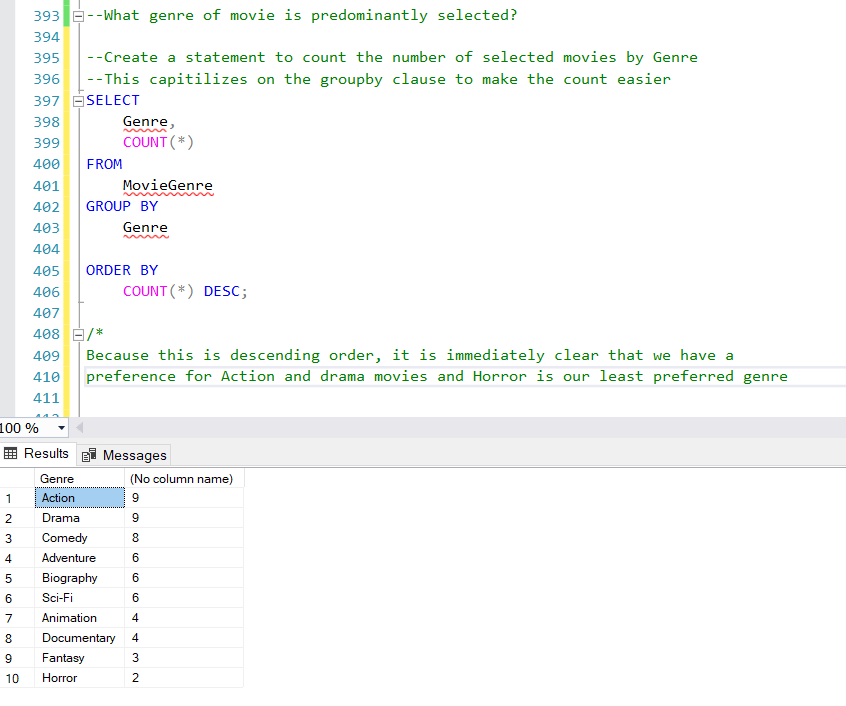
COUNT(\*) DESC;

/\*

Because this is descending order, it is immediately clear that we have a

preference for Action and drama movies and Horror is our least preferred genre

\*/



* What is the distribution of movie watched to who selected it?

--Create a function to count the number of selected movies by who selected them

CREATE FUNCTION dbo.WhoPicks (@FamilyMemberID int)

RETURNS int AS

--Treating COUNT as an integer therefore an integer is what will be returned

BEGIN

DECLARE @returnValue int

--it's an int value as that is what we expect the output to be

/\*

The code the parameters. Get the count of the Family Member ID for the selected movies and

assign that output a value or int for the return value.

\*/

SELECT @returnValue=COUNT (TotalPicture.Family\_Member\_ID) FROM TotalPicture

WHERE Family\_Member\_ID=@FamilyMemberID

--Request that return value returned

RETURN @returnValue

END

GO

--Then write the statement to code and execute and produce the results.

SELECT TOP 50

\*

,dbo.WhoPicks(Family\_Member\_ID) as WhoPicks

FROM TotalPicture

ORDER BY Family\_Member\_Name DESC

--Now I'm going to run a select statement that runs a count for each of the four family members to compare

SELECT dbo.WhoPicks(1)

SELECT dbo.WhoPicks(2)

SELECT dbo.WhoPicks(3)

SELECT dbo.WhoPicks(4)

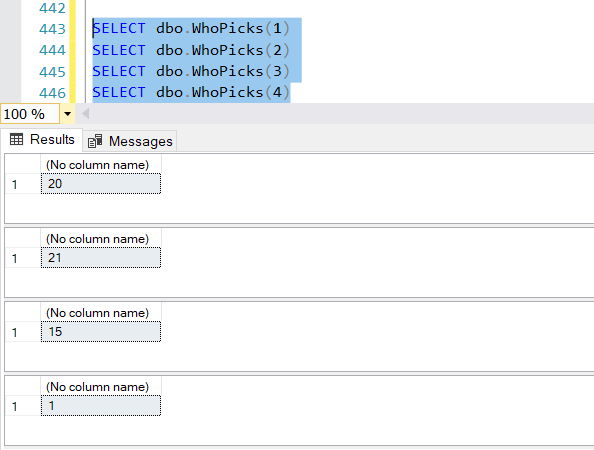
/\*

This shows that I have 20 selected movies, my husband has 21, my daughter

has 15 and her friend has 1. That confirms her affirmation that her father

gets more movie options into consideration than she does.

\*/



--A quick look confirms that when it comes to which movies are watched the disparity is even greater.

SELECT TOP 50

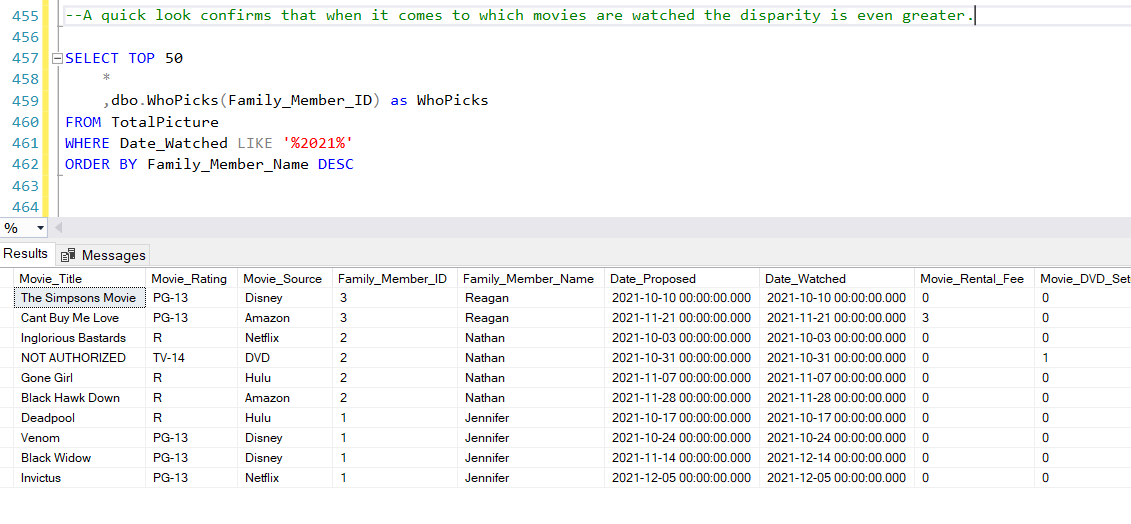
\*

,dbo.WhoPicks(Family\_Member\_ID) as WhoPicks

FROM TotalPicture

WHERE Date\_Watched LIKE '%2021%'

ORDER BY Family\_Member\_Name DESC



--This affirmation if inequity in movie watching prompted me to create a view

--that uses the WhoPicks Function and selects it in descending order so those

--who have had fewer of thier choices viewed will be loaded at the top

CREATE OR ALTER VIEW AndTheWinnerIs AS

--This view takes the top 200 records using all the columns of the specified table but adds the who pics count to the end

SELECT TOP 200

\*

,dbo.WhoPicks(Family\_Member\_ID) AS WhoPicks

FROM TotalPicture

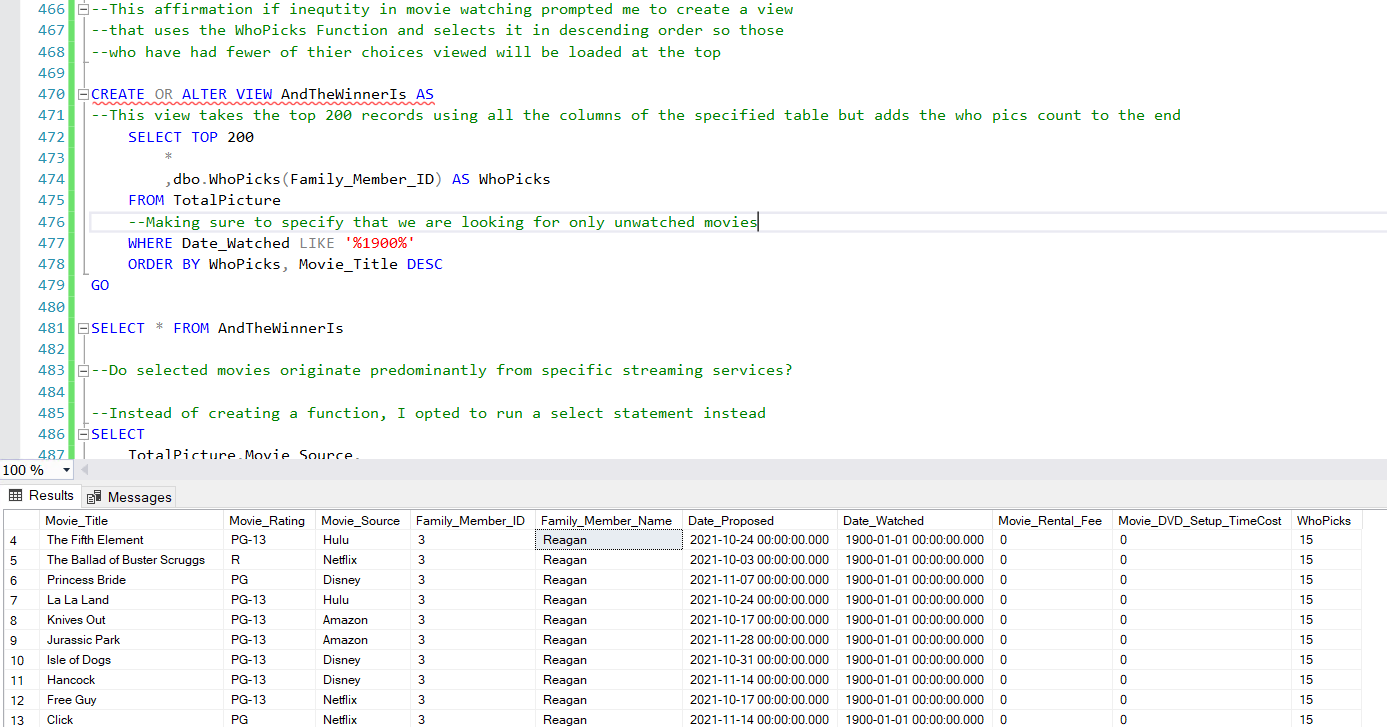
--Making sure to specify that we are looking for only unwatched movies

WHERE Date\_Watched LIKE '%1900%'

ORDER BY WhoPicks, Movie\_Title DESC

GO

SELECT \* FROM AndTheWinnerIs



* Do selected movies originate predominantly from specific streaming services?

--Do selected movies originate predominantly from specific streaming services?

--Instead of creating a function, I opted to run a select statement instead

SELECT

TotalPicture.Movie\_Source,

COUNT(\*)

FROM

TotalPicture

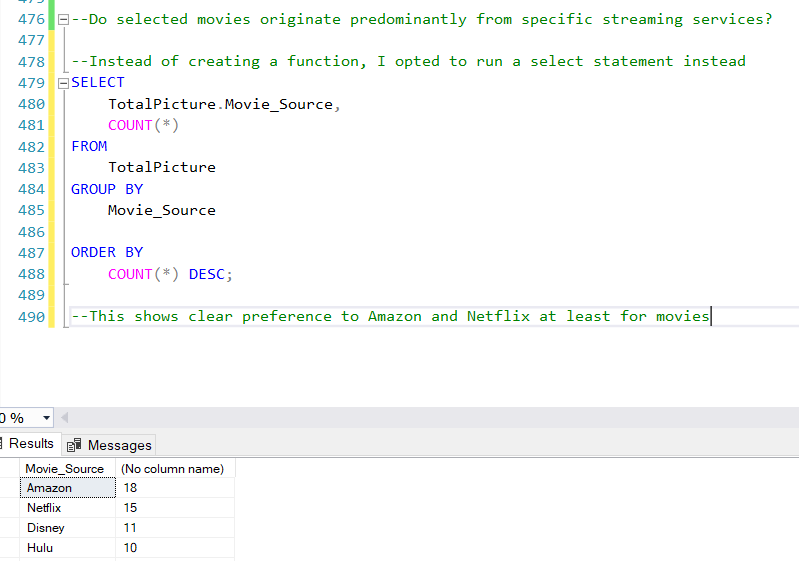
GROUP BY

Movie\_Source

ORDER BY

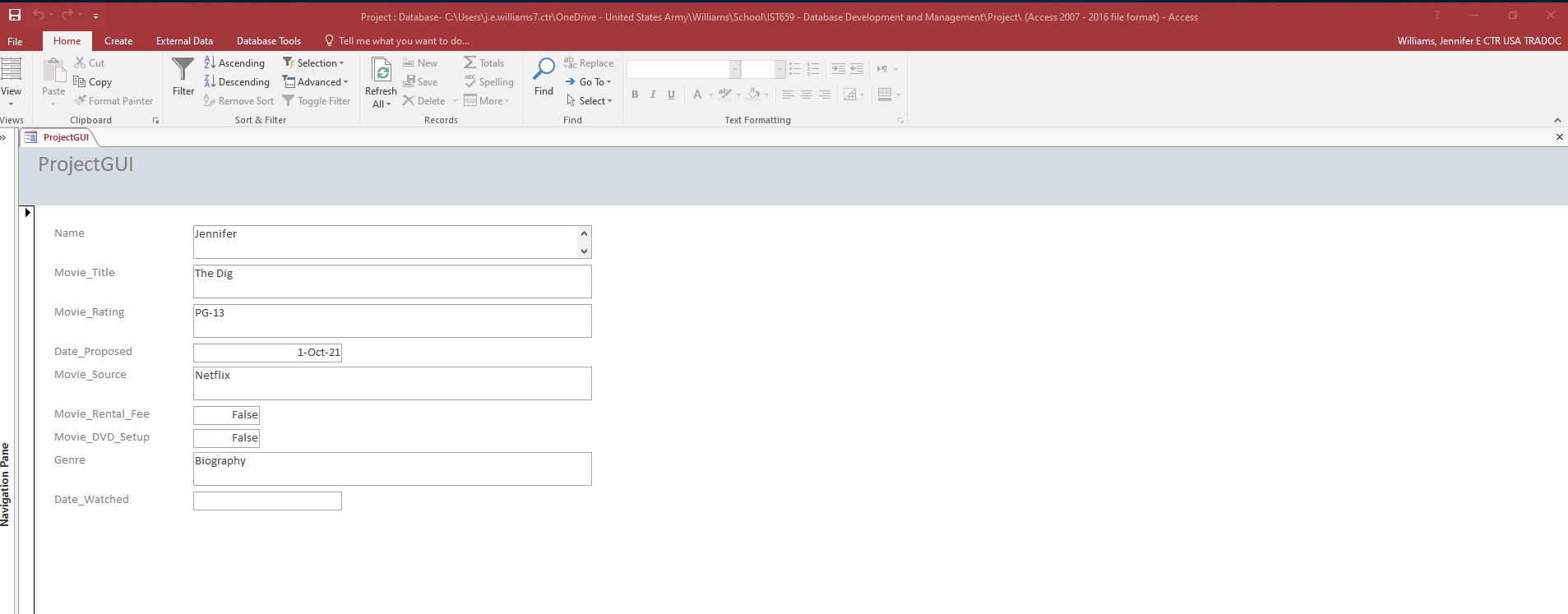
COUNT(\*) DESC;

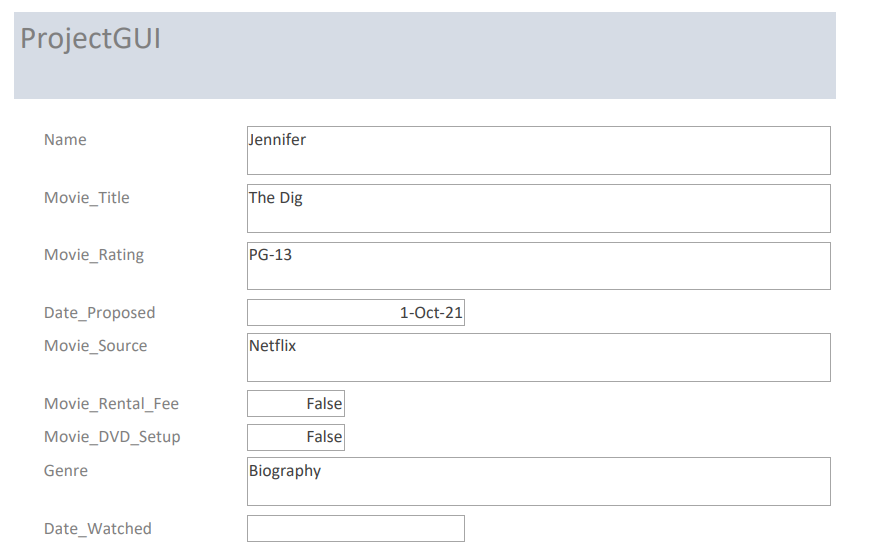
--This shows clear preference to Amazon and Netflix at least for movies



# **Implementation**

For the front end interface, I opted somewhat simplistic option that would pull key points from different tables that were joined in SQL. This would allow family members to go in and select themselves from the Name field and then input the remaining fields with their preferred movie. It would also enable updating of records once movies are watched.





# **Reflection**

When going through this process, I believe the data question that I started with were not necessarily the same questions that I would develop now. I attribute this to a greater understanding and comprehension of the capabilities that could provide enhanced queries, view and procedures. I believe that really addresses the assumptions that changed throughout the processes and how I assumed a degree of either system or user/operator limitations. If I were to complete this again, I would make much more refined and sophisticated data questions.

Looking back at the database and the model components that went into this, I still believe in the concept and the elements that made it up. I don’t believe I would change the architecture of the database, just what I would do with the data once inserted.

Moving forward, this will shape how I can parse, understand and visualize the information available in my database (or in a database I’m accessing). The ability to customize the views and outputs of select statements allows greater control over manipulation of the data. Possibilities are endless. I believe linkages between the SQL and R could further magnify the visualization possibilities

# **Summary**

This single document represents the combined parts 1 and 2. The only update from part 1 is on the Logical Model making the field Date\_Watched not required (change reflected on page 5).

I was able to answer all my data questions. I now have a running list of existing proposed movies to select from for future movie nights. I also know that we as a family prefer action and drama movies primarily from Amazon Prime and Netflix. If I were in marketing, I could use this information in an exploitative manner to refine and tailor content. I also identified that there is disparity in who proposes the movies and an disproportionate ratio of who’s movie’s we end up watching. I can use this to inform my husband that he can propose fewer movies and that we need to watch more movies that my daughter wants.

The implementation is a rather simplistic view that would enable a straight forward approach for family members to both input new and update existing records. This was created using MS Access linked to the SQL tables. Optimally, it would ultimately be a web-based interface using this same form to ensure the greatest accessibility.

I must admit that I actually had fun creating this. While I found some challenges in re-creating one or two of the labs, when working with my own data to achieve my own objectives, working the syntax of the code was much more intuitive.