ALEX GEIGER

“Every Portrait that is painted with feeling is a portrait of the artist, not of the sitter.” Oscar Wilde

IST 658

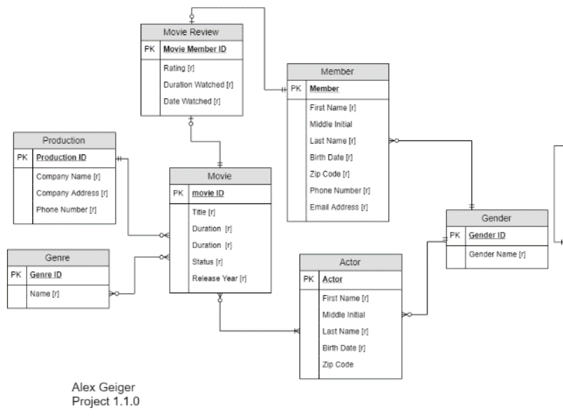
Contents

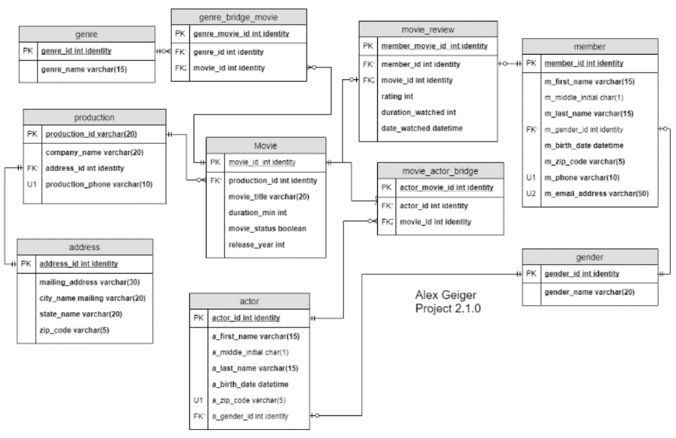
Root Path to SQL Code *SYR Project\sql code*

File Name	Purpose	Run Number
FULL SQL.sql	Runs all sections	1,2,3
Physical_Database_Design.sql	Generate tables	1
Data_Creation.sql	Insert data	2
Data_Manipulation.sql	Applies inserted data	3

Physical Database Design

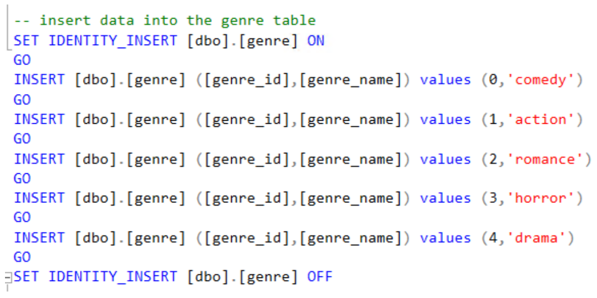
The DDL code to build the tables, and objects such as view procedures and functions are combined into one file called **FULL SQL** for the individual files to help you identify the code more readily you wish to grade see the file directory above. The file names, purpose and run number are listed in the table. To generate just the tables and create the physical database run the **Physical_Database_Design.sql**





Data Creation

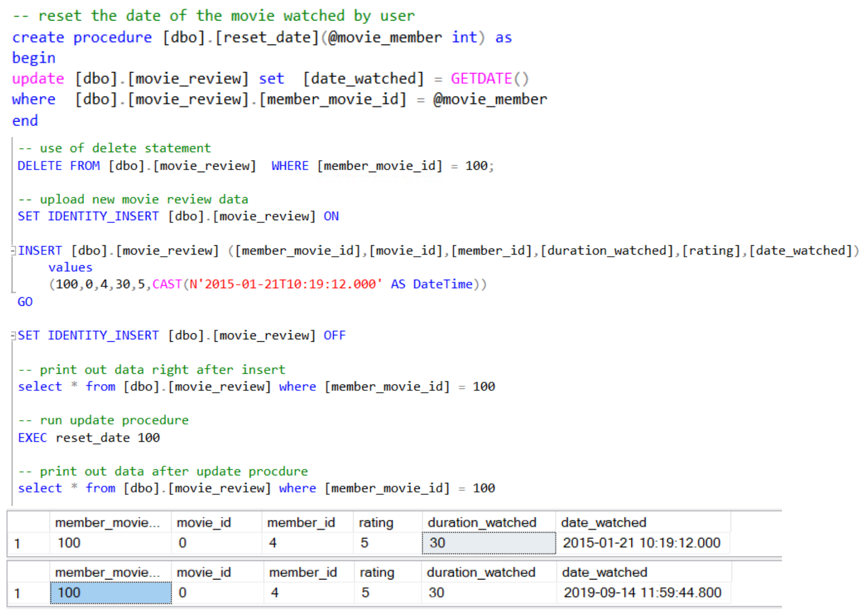
Data was created using several INSERT statements one of which is shown in the example below. To ensure best practice to the best of my knowledge. For instance, I used a set identity insert as shown below and brackets to specify the table. If there is any room for improvement, please provide any feedback as to how I may. For more information on the SQL insert commands see the SQL file titled **Data_Creation.sql** for more information.



```
-- insert data into the genre table
SET IDENTITY_INSERT [dbo].[genre] ON
GO
INSERT [dbo].[genre] ([genre_id],[genre_name]) values (0,'comedy')
GO
INSERT [dbo].[genre] ([genre_id],[genre_name]) values (1,'action')
GO
INSERT [dbo].[genre] ([genre_id],[genre_name]) values (2,'romance')
GO
INSERT [dbo].[genre] ([genre_id],[genre_name]) values (3,'horror')
GO
INSERT [dbo].[genre] ([genre_id],[genre_name]) values (4,'drama')
GO
SET IDENTITY_INSERT [dbo].[genre] OFF
```

Data Manipulation

One data manipulation example is when I had deleted the member movie reviews where the member movie review was equal to 100. After I inserted a new member movie review with the primary key set to 100. Then I ran the procedure which updated the field date_watch. The field is defined the date the user had watched the movie which would reset the date to the current time. To show the procedure had worked I wrote SQL code which printed in out before and after I ran the procedure reset date as shown below. For more information see the SQL script **Data_Manipulation.SQL**. In addition these are were the SQL questions are answered



```sql
-- reset the date of the movie watched by user
create procedure [dbo].[reset_date](@movie_member int) as
begin
update [dbo].[movie_review] set  [date_watched] = GETDATE()
where  [dbo].[movie_review].[member_movie_id] = @movie_member
end
```

```sql
-- use of delete statement
DELETE FROM [dbo].[movie_review]  WHERE [member_movie_id] = 100;

-- upload new movie review data
SET IDENTITY_INSERT [dbo].[movie_review] ON

INSERT [dbo].[movie_review] ([member_movie_id],[movie_id],[member_id],[duration_watched],[rating],[date_watched])
    values
    (100,0,4,30,5,CAST(N'2015-01-21T10:19:12.000' AS DateTime))
GO

SET IDENTITY_INSERT [dbo].[movie_review] OFF

-- print out data right after insert
select * from [dbo].[movie_review] where [member_movie_id] = 100

-- run update procedure
EXEC reset_date 100

-- print out data after update procdure
select * from [dbo].[movie_review] where [member_movie_id] = 100
```

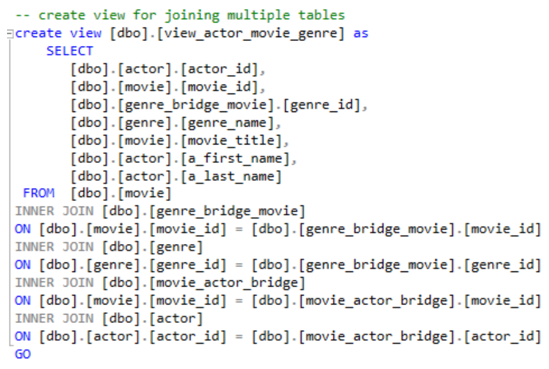
```
   member_movie...  movie_id  member_id  rating  duration_watched  date_watched
1  100              0         4          5       30                2015-01-21 10:19:12.000

   member_movie...  movie_id  member_id  rating  duration_watched  date_watched
1  100              0         4          5       30                2019-09-14 11:59:44.800
```

Answering Data Questions

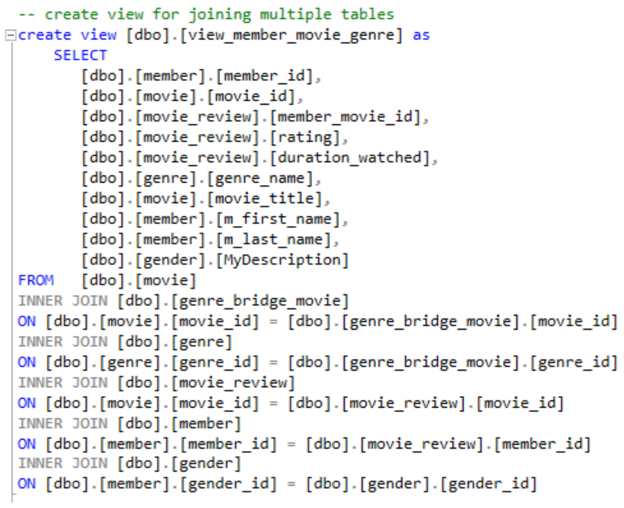
Creating View for Questions

The following section will cover views that were used to make answering the data questions easier and more streamline than ever. The first view combined data across the genre, movie, and actor tables to un normalize the data. The follow view was used in both questions 1.



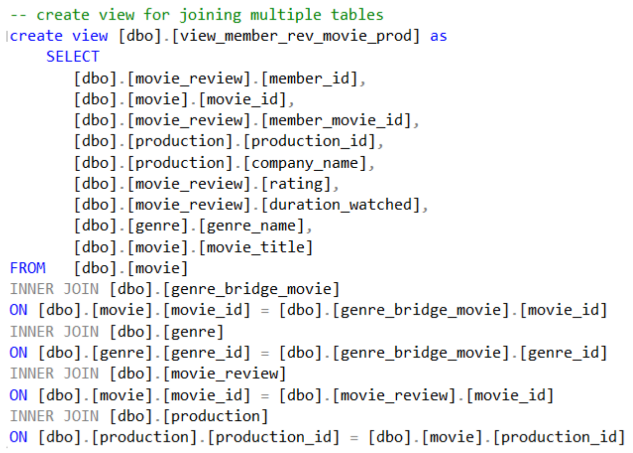
```sql
-- create view for joining multiple tables
create view [dbo].[view_actor_movie_genre] as
    SELECT
        [dbo].[actor].[actor_id],
        [dbo].[movie].[movie_id],
        [dbo].[genre_bridge_movie].[genre_id],
        [dbo].[genre].[genre_name],
        [dbo].[movie].[movie_title],
        [dbo].[actor].[a_first_name],
        [dbo].[actor].[a_last_name]
 FROM   [dbo].[movie]
INNER JOIN [dbo].[genre_bridge_movie]
ON [dbo].[movie].[movie_id] = [dbo].[genre_bridge_movie].[movie_id]
INNER JOIN [dbo].[genre]
ON [dbo].[genre].[genre_id] = [dbo].[genre_bridge_movie].[genre_id]
INNER JOIN [dbo].[movie_actor_bridge]
ON [dbo].[movie].[movie_id] = [dbo].[movie_actor_bridge].[movie_id]
INNER JOIN [dbo].[actor]
ON [dbo].[actor].[actor_id] = [dbo].[movie_actor_bridge].[actor_id]
GO
```

The next view was used to combine the member, movie, genre and gender tables. The view was used in questions two and question three. It allows easy access to assess the relationship between members and movies.



```sql
-- create view for joining multiple tables
create view [dbo].[view_member_movie_genre] as
    SELECT
        [dbo].[member].[member_id],
        [dbo].[movie].[movie_id],
        [dbo].[movie_review].[member_movie_id],
        [dbo].[movie_review].[rating],
        [dbo].[movie_review].[duration_watched],
        [dbo].[genre].[genre_name],
        [dbo].[movie].[movie_title],
        [dbo].[member].[m_first_name],
        [dbo].[member].[m_last_name],
        [dbo].[gender].[MyDescription]
FROM    [dbo].[movie]
INNER JOIN [dbo].[genre_bridge_movie]
ON [dbo].[movie].[movie_id] = [dbo].[genre_bridge_movie].[movie_id]
INNER JOIN [dbo].[genre]
ON [dbo].[genre].[genre_id] = [dbo].[genre_bridge_movie].[genre_id]
INNER JOIN [dbo].[movie_review]
ON [dbo].[movie].[movie_id] = [dbo].[movie_review].[movie_id]
INNER JOIN [dbo].[member]
ON [dbo].[member].[member_id] = [dbo].[movie_review].[member_id]
INNER JOIN [dbo].[gender]
ON [dbo].[member].[gender_id] = [dbo].[gender].[gender_id]
```

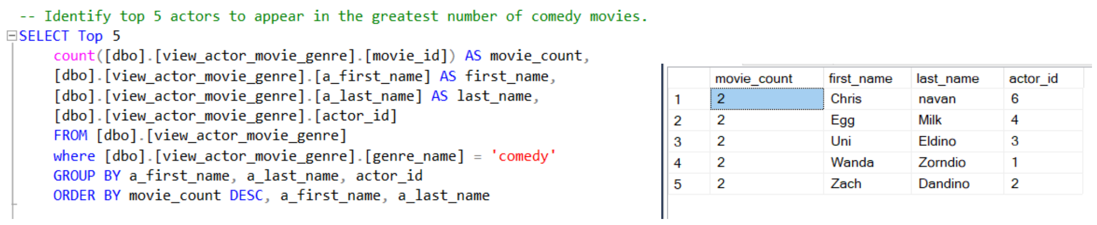
The next view was used to combine the member movie review, movie and production tables. The view was used in questions two and question three. It allows easy access to assess the relationship between members and movies.



```sql
-- create view for joining multiple tables
create view [dbo].[view_member_rev_movie_prod] as
    SELECT
        [dbo].[movie_review].[member_id],
        [dbo].[movie].[movie_id],
        [dbo].[movie_review].[member_movie_id],
        [dbo].[production].[production_id],
        [dbo].[production].[company_name],
        [dbo].[movie_review].[rating],
        [dbo].[movie_review].[duration_watched],
        [dbo].[genre].[genre_name],
        [dbo].[movie].[movie_title]
FROM    [dbo].[movie]
INNER JOIN [dbo].[genre_bridge_movie]
ON [dbo].[movie].[movie_id] = [dbo].[genre_bridge_movie].[movie_id]
INNER JOIN [dbo].[genre]
ON [dbo].[genre].[genre_id] = [dbo].[genre_bridge_movie].[genre_id]
INNER JOIN [dbo].[movie_review]
ON [dbo].[movie].[movie_id] = [dbo].[movie_review].[movie_id]
INNER JOIN [dbo].[production]
ON [dbo].[production].[production_id] = [dbo].[movie].[production_id]
```

Question 1

For question one I built a select statement which found the top 5 actors to appear in the greatest number of comedy movies. The select statement references a view view_member_movie_genre which I created to help me more quickly reference data. The top 5 actors were Chris Navan, Egg Milk, Uni Eldino and Zach Dandino all of which had a movie count of 2. All other actors had appeared in just one movie. If a tie had occurred the data is ranked by first then last name in ascending order.

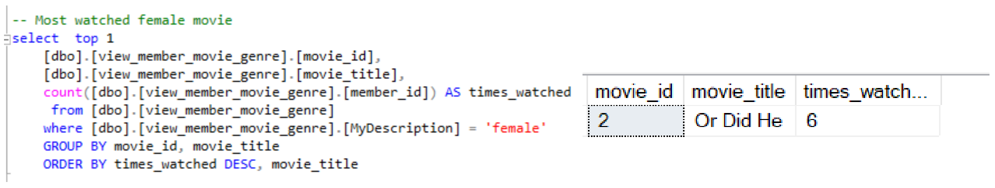


```
-- Identify top 5 actors to appear in the greatest number of comedy movies.
SELECT Top 5
    count([dbo].[view_actor_movie_genre].[movie_id]) AS movie_count,
    [dbo].[view_actor_movie_genre].[a_first_name] AS first_name,
    [dbo].[view_actor_movie_genre].[a_last_name] AS last_name,
    [dbo].[view_actor_movie_genre].[actor_id]
    FROM [dbo].[view_actor_movie_genre]
    where [dbo].[view_actor_movie_genre].[genre_name] = 'comedy'
    GROUP BY a_first_name, a_last_name, actor_id
    ORDER BY movie_count DESC, a_first_name, a_last_name
```

	movie_count	first_name	last_name	actor_id
1	2	Chris	navan	6
2	2	Egg	Milk	4
3	2	Uni	Eldino	3
4	2	Wanda	Zorndio	1
5	2	Zach	Dandino	2

Question 2

The next select statement calculated which movie had the most amount of female views. The movie the select statement returned was Or Did He. The hard stop thriller drew more than six members to have acknowledged they had watched it. If a tie had occurred the data would be ranked by ascending movie title

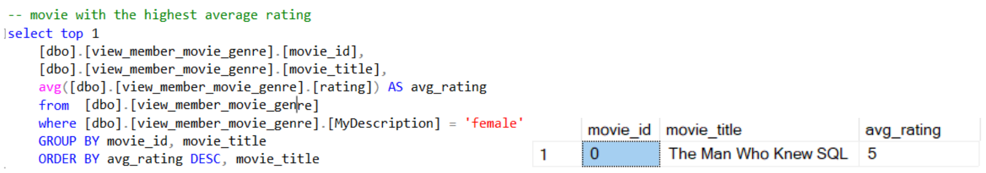


```
-- Most watched female movie
select  top 1
    [dbo].[view_member_movie_genre].[movie_id],
    [dbo].[view_member_movie_genre].[movie_title],
    count([dbo].[view_member_movie_genre].[member_id]) AS times_watched
     from [dbo].[view_member_movie_genre]
    where [dbo].[view_member_movie_genre].[MyDescription] = 'female'
    GROUP BY movie_id, movie_title
    ORDER BY times_watched DESC, movie_title
```

movie_id	movie_title	times_watch...
2	Or Did He	6

Question 3

The third question interrogated the data until it returned the most popular female movie genre. There were a significant of joins which had to be done to make the data talk. The movie title returned was the man who knew SQL which had a five star rating! If the movie was tied the data would be ranked by the movie title name in ascending order.

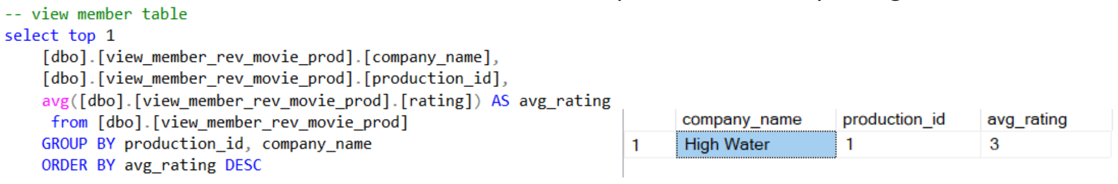


```
-- movie with the highest average rating
select top 1
    [dbo].[view_member_movie_genre].[movie_id],
    [dbo].[view_member_movie_genre].[movie_title],
    avg([dbo].[view_member_movie_genre].[rating]) AS avg_rating
    from  [dbo].[view_member_movie_genre]
    where [dbo].[view_member_movie_genre].[MyDescription] = 'female'
    GROUP BY movie_id, movie_title
    ORDER BY avg_rating DESC, movie_title
```

	movie_id	movie_title	avg_rating
1	0	The Man Who Knew SQL	5

Question 4

Lastly, the select statement created returned which production company had the highest rating. In the following case the company High Water was the highest rated coming in with a rating of three. To make the select statement easier I created a view to encapsulate the heavy lifting.



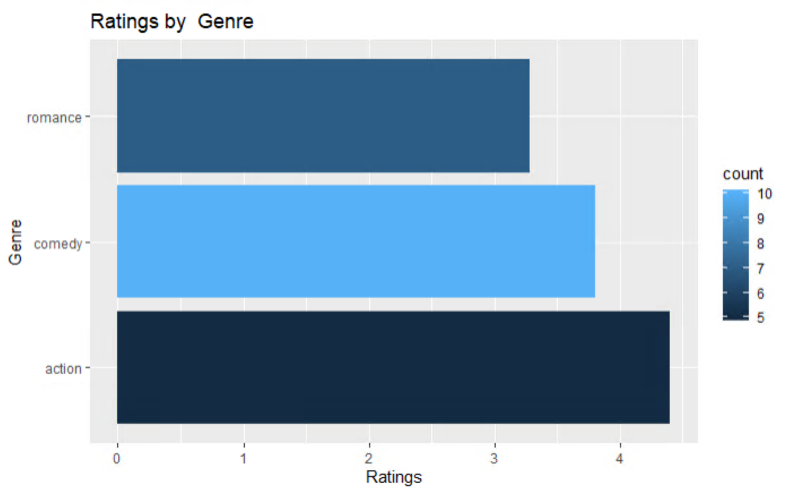
```
-- view member table
select top 1
    [dbo].[view_member_rev_movie_prod].[company_name],
    [dbo].[view_member_rev_movie_prod].[production_id],
    avg([dbo].[view_member_rev_movie_prod].[rating]) AS avg_rating
     from [dbo].[view_member_rev_movie_prod]
    GROUP BY production_id, company_name
    ORDER BY avg_rating DESC
```

	company_name	production_id	avg_rating
1	High Water	1	3

Implementation

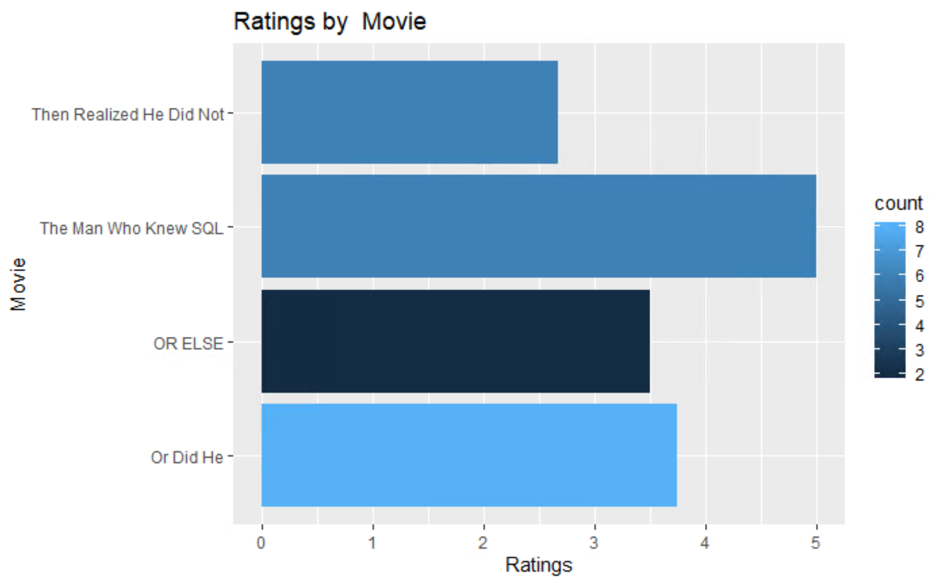
Ratings by Genre

The following graph depicts the seemingly unlike outcome of an individual deciding on which genre is the highest. The data suggests the member demographic prefers action above all else, even that of which the human condition craves most of all romance.



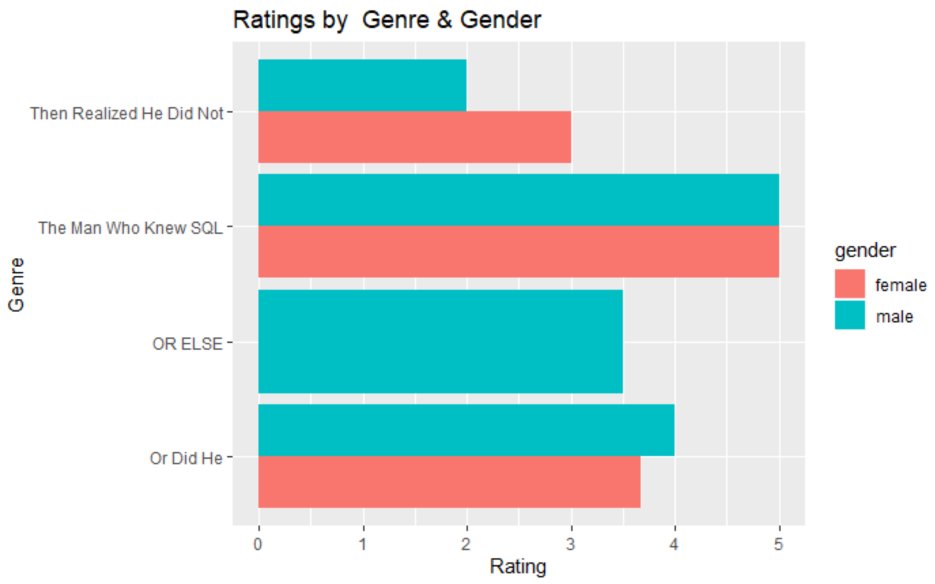
Ratings by Movie

The seemingly antiquated collections of words strung together in titles have been rated. The rating has been aggregated to a bar graph representation. We see the highest performing movie to date is the man who knew SQL with a score of five stars. Followed by the most widely watched film amongst the members *Or Did He* with a total of eight views.



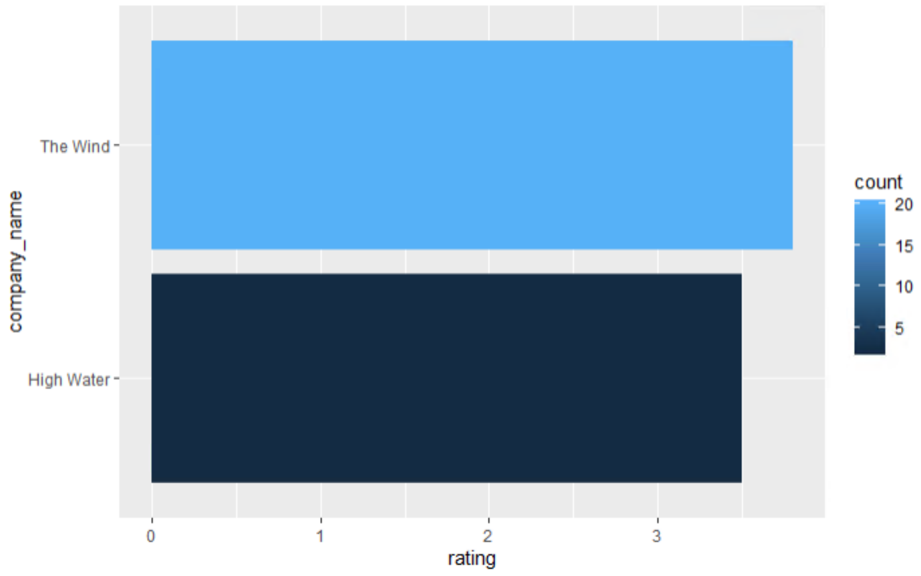
Ratings by Genre & Gender

The following graph depicts average movie ratings of a gender. The bar graph conveys disparity between movie rating for both male and females. The biggest disparity between male and female members appeared in the cinema feature *Then Realized He Did Not*. An epic tale of self-doubt which is followed up by Or Did He.



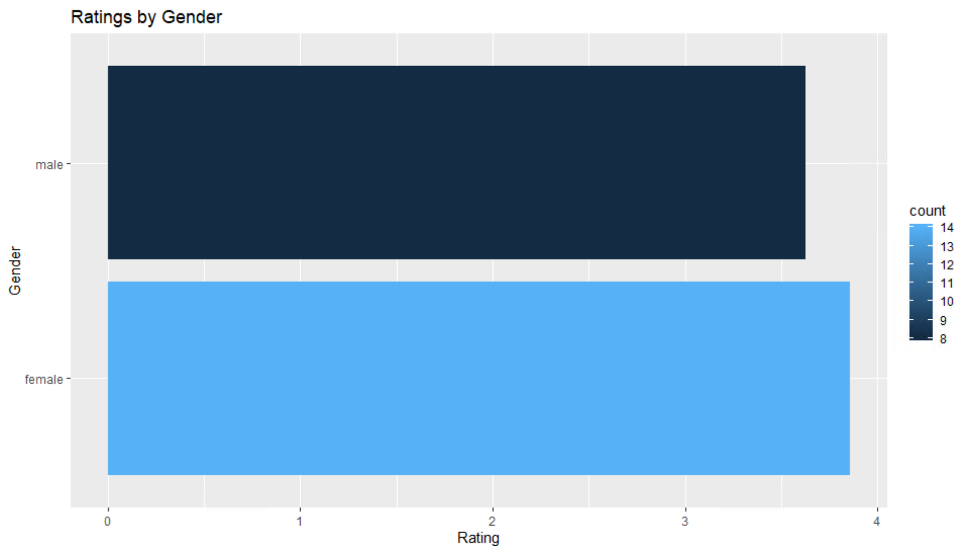
Ratings by Production Company

Next, the production company average rating across all movies weighted by number of members votes. Looking at the visualization it is evident *The Wind* have a higher rating and more member engagement. Whereas the other cinema features High water under preformed its counterpart.



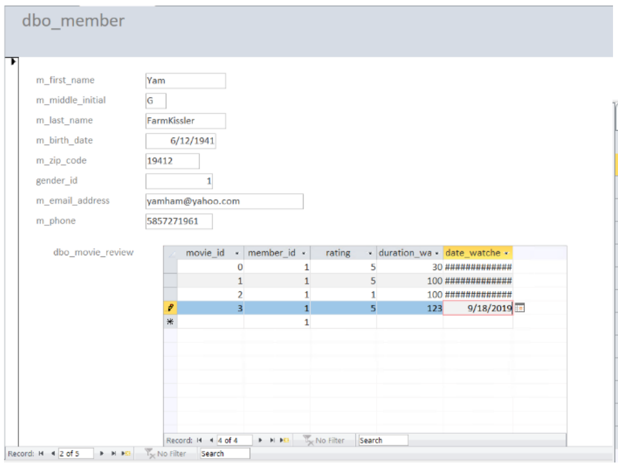
Ratings by Gender

Following the above graph, the ratings by gender is depicted across all cinema features and production companies. Female engagement was significantly higher than their male counter parts. In addition, females had higher ratings as well.



Access Forms

The following form allows users of the database backend team to more readily edit data. However, Access is highly speculated whether it is optimal while handling large amounts of data. Given the current state of the database the Microsoft access form front end interface is perfect for just a set of data as shown below.



dbo_member

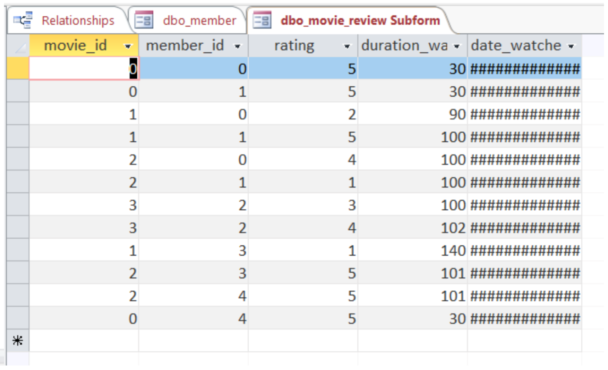
m_first_name	Yam
m_middle_initial	G
m_last_name	FarmKissler
m_birth_date	6/12/1941
m_zip_code	19412
gender_id	1
m_email_address	yamham@yahoo.com
m_phone	5857271961

dbo_movie_review

movie_id	member_id	rating	duration_wa	date_watche
0	1	5	30	#############
1	1	5	100	#############
2	1	1	100	#############
3	1	5	123	9/18/2019
	1			

Record: 4 of 4 No Filter Search

Record: 2 of 5 No Filter Search



movie_id	member_id	rating	duration_wa	date_watche
0	0	5	30	#############
0	1	5	30	#############
1	0	2	90	#############
1	1	5	100	#############
2	0	4	100	#############
2	1	1	100	#############
3	2	3	100	#############
3	2	4	102	#############
1	3	1	140	#############
2	3	5	101	#############
2	4	5	101	#############
0	4	5	30	#############

Reflection

One of the assumptions made during the initialization of the project was whether it would be relatively easy to make my database and if the primary keys were not unique integers. Historically, I used a random ID generator. However, in this instance I had realized it would be more optimal to just use unique integers. Therefore, when I officially had built the database, I had converted all the primary keys to integers.

Regarding my approach to data as an information professional I have made great strides. Most notably, I would consider myself to being more open when building procedures and functions in SQL rather than Python and R. I believe, I will still do the heavy lifting in R and Python. However, I will be more apt to do it in SQL, nonetheless.

In conclusion, the class had led me down a path which has guided me on an unexpected journey. The road I am on to becoming a better data scientist is long and hard one. From moving away from all I've ever known, to meeting new people, not all of good intention is hard. I wish we could query the optimal decisions moving forward.

Summary

The movie database MD provides insight into movies by collecting data through an application called *video insight* or VI developed by VI Solutions. The app allows people to rate movies and provide meta-data associated with their experience. The primary application of the data is aimed to facilitate member engagement. To facilitate member engagement the MD captures insight into market sentiments of movies across a breadth demographics in the film industry.

High level rules for the database include the database must be in third normal form. In addition, all foreign and primary keys are required in the database except for the gender ID. Third, any non-required unknown data string value must be NULL except for the. Lastly, all primary keys are an identity integer.

The most important stakeholder is our members, the VI product allows a medium for them to voice their opinion of movies they have seen. A steering committee comprised of regulators, information security, marketing and the database architect team will monitor project quality and ensure our member data is stored securely. The biggest influencer of the project will be the marketing department within the VI solutions because they will be using the data.

In conclusion, the MD surpassed all goals set forth and had developed visualizations to graphically depict insight in the data. Secondly, Interactive forms allowed our team to adjust the data in a professional streamlined approach. Most importantly, all the questions set forth in the original introduction. The most notable awarded by a hypothetical meeting was the most watch female movie *Or Did He.*

Glossaries

FIELD NAME	Data Type	Definition
genre_id	**int identity**	Primary key for genre table
genre_name	varchar(15)	Name of genre ex. comedy
genre_movie_id	**int identity**	Genre movie bridge table primary key
movie_id	**int identity**	Movie primary key
production_id	**int identity**	Production company primary key
company_name	varchar(20)	Production company name
production_phone	varchar(10)	Production company phone number
movie_title	varchar(20)	Title of movie
address_id	**int identity**	Primary address key to a location
duration_min	int	Duration of movie in minutes
movie_status	Boolean	Is movie still showing in theaters?
release_year	int	First day the movie came to the box office
member_id	varchar(20)	Member primary key
movie_id	varchar(20)	Movie primary key
rating	int	Member rating of movie
m_duration_watched	int	Amount of time member watched movie
date_watched	datetime	Date the member watched the movie
m_first_name	varchar(15)	First name of member
m_middle_initial	char(1)	Middle initial of member
m_last_name	varchar(15)	Last name of member
m_gender_id	varchar(4)	Member gender id
m_birth_date	datetime	Member birth date
m_phone	varchar(10)	Member phone number
m_email_address	varchar(50)	Member email
actor_movie_id	**int identity**	Primary key for the actor movie bridge table
actor_id	**int identity**	Primary key for actors
a_first_name	varchar(15)	First name of actor
a_middle_initial	char(1)	Middle initial of actor
a_last_name	varchar(15)	Actor last name
a_zip_code	varchar(5)	Actor zip code
a_gender_id	**int identity**	Actor gender id
mailing_address	varchar(30)	Mailing address of production company
state_name	varchar(20)	State name of production company
city_name	varchar(20)	City of production company
gender_name	varchar(20)	Name of gender relating to the gender_id primary key
gender_id	**int identity**	Gender id associated with gender name.