Name: Vincent Nguyen

**Table of Contents**

[Project Direction Overview 2](#_Toc73971270)

[Use Cases and Fields 2](#_Toc73971271)

[Structural Database Rules 6](#_Toc73971272)

[Conceptual Entity-Relationship Diagram 12](#_Toc73971273)

[Full DBMS Physical ERD 14](#_Toc73971274)

[Stored Procedure Execution and Explanations 27](#_Toc73971275)

[Question Identification and Explanations 30](#_Toc73971276)

[Query Executions and Explanations 30](#_Toc73971277)

[Index Identification and Creations 33](#_Toc73971278)

[History Table Demonstration 37](#_Toc73971279)

[Data Visualizations 44](#_Toc73971280)

[Summary and Reflection 46](#_Toc73971281)

# Project Direction Overview

I would like to develop a web app that can track all the movies and shows when someone watches movies and shows through streaming services such as Amazon Prime, Disney+ or Netflix etc. by web extension. It’s named “MyWatchList”. It is truly a modern problem that people can’t remember all the movies or shows which people have watched on the particularly streaming platforms when people use multiple streaming services. Usually, when people visit one streaming service to watch movies or show. They can only see their movies or shows history within that same platform. It would be beneficial to have an app to combine and view all these histories where, when and what they have watched or have not finished. MyWatchList will store histories of the movies/ shows which they have seen through all streaming services, making it easier for people to search and continue to watch those movies and shows again.

The following is some examples of how people would use the app and how the app become helpful for them. Mike is using several streaming services such as Amazon Prime, Netflix, Disney +. He usually searches through all of 3 streaming platform to find good movies or shows to watch in his free time. Sometimes, he watched the movie and skipped it right after if he found another interesting thing, or he watched it but could not finish to watch it in one time. Day by day, the list keeps increasing and it becomes hard to remember what he has and has not finished, and those movies/ shows belong to which streaming platforms. Using a web extension, MyWatchList automatically recorded all movies and shows information. Last week, his coworkers discussed about ‘Man vs. Bee’ movie and they wondered which streaming service is offering this movie, Mike remembered he already watched this movie, but he could not remember where it belongs to, so he opened the app to get information and forward to his coworkers. Using MyWatchList would be much easier to double check when or what platforms he has seen a movie or show.

In fact, I will have to make a lot of programming components in order to make MyWatchList to be commercial. In this project, I will only develop on database which stores names, streaming services etc. of movies and shows. It will record the details of movies and shows like movie released year, when to watch it, finished to watch or not, rating, actor name, director name.

The reason I am interested in it because I am also a big fan of watching movies and shows on difference streaming services. Firstly, I always look for new and interesting movies or shows to watch but can’t finished it in one time. Secondly, it always takes a lot of time for me to locate the location of which movie or show I have watched. Thirdly, the current streaming services only have ‘Recently views’ option, so it encourages me to make something that can track and retrieve all movies and shows which I have watched.

# Use Cases and Fields

One important usage of the database is when someone signs up for an account and install the app.

***Use case #1: Account signup & installation***

1. User visits app store, download and install “MyWatchList” app.
2. The “MyWatchList” ask user to create an account when it’s first use.
3. The user enters their information, and the account is created in the database.
4. The application prompt user to install browser extension so that their movies can be automatically tracked when they watch or click them.

From a database of first use case, this use case requires storing information about user’s accounts (in step#2 and #3). The step #1 and # 4 apply to the user and application but not the database directly.

Significant fields for an account for this application are listed in the able below

|  |  |  |
| --- | --- | --- |
| Field | What is Stored | Why it’s Need |
| UserName | This is a unique summary name associated with each account. | The user will have the option to log in with their username in order to use app. |
| FirstName | This is the first name of the account holder. | This is necessary for displaying the person’s name within the application and addressing them when sending them emails or other communications. |
| LastName | This is the last name of the account holder. | This is necessary for displaying the person’s name within the application and addressing them when sending them emails or other communications. |
| Email | This is the email of account holder. | This is using for communication to the user. (Such as verification, notification). |
| PhoneNumber | This is user’s phone number. | This is using for communication to the user. (Such as verification, notification). |
| User\_DOB | This is user’s date of birth. | This is using for verifying user. |

Second important usage of the database is when users watch movies/shows on streaming platforms and automatically recorded in MyWatchList by web extension.

***Use case #2: Watched Content Tracking***

1. The person visits a streaming service and starts to consume content.
2. The “MyWatchList” browser extension will start to record the relevant information in the database such as service name, title of content, released date, length etc. when user watch into content.

The database will store the relevant information about movies/shows in step#2.

Significant fields are listed below.

|  |  |  |
| --- | --- | --- |
| Field | What is Stored | Why it’s Need |
| Service\_Name | This is the name of streaming service. | This is useful so the person knows which site movie/show belongs to. |
| Title | This is the name of movie or show. | This is useful so the person can search the movie/show by title. |
| Released\_Date | This is the released date of movie or show. | This is useful to tell when movie/show is made so the person knows if the version is old or new. |
| Length | This is the length of movie show. | This is useful to tell user how long it would take to complete the movie/show. |
| URL | This is the URL where movie or show belong to. | This is necessary so the person knows which website movie/show is provided. |
| Status | This is the status of movie or show. It shows if movie completed or not. | This is useful to tell user if movie/show already completed or not. |
| Mov\_ID | This is the ID of movie/show. | This is useful so the person can search the movie/show by ID. |
| DirName | This is the director’s name of the movie/show | This is useful so the person can search the movie/show by director’s name. |
| ActorName | This is the actor’s name in the movie/show. | This is useful so the person can search the movie/show by actor’s name. |
| Genre | This is the category of the movie/show. | This is useful so the person can search the movie/show by category/genre. |
| Subtitle | This is saying if movie/show supports subtitle. | This is useful so the person knows if the movie/show supports subtitle. |
| Studio | This is the name of studio. | This is useful so the person can search the movie/show by producing studio. |

Third important usage is when a person downloads movie/ show and automatically recorded in MyWatchList by web extension.

***Use case #3: Download Content.***

1. The user visits streaming service site and look for movies/shows.
2. The user download movie/show to watch offline.
3. The MyWatchList web extension will automatically records all relevant information in the database which listed in the use case number 2 and in addition with downloaded status record.

The database will store the relevant information about movies/shows in step#3. The database will record all fields which listed in the use case number 2 and in addition the field listed below.

|  |  |  |
| --- | --- | --- |
| Field | What is Stored | Why it’s Need |
| DownloadedStt | This is the download status of the movie/show. | This is useful so the people can know which movie/show they can watch offline. |

Fourth important usage of database is when user pays subscription fee and automatically recorded in MyWatchList by web extension.

***Use case #4: Payment tracking.***

1. The user visit streaming service site.
2. The user login in streaming service site.
3. The user chooses pay membership option and pays the subscription fee.
4. The MyWatchList web extension will automatically record all relevant information in the database such as account streaming ID, payment amount, payment date.

The database will store the relevant information about payment in step#2 and #4. The database will record all fields listed below.

|  |  |  |
| --- | --- | --- |
| Field | What is Stored | Why it’s Need |
| Streaming\_ID | This is ID account of streaming service. | This is useful so the people can remember their account ID easily when uses several streaming services. |
| PaymentDate | This is that payment date for the service. | This is useful so the people can remember their recent membership payment date. |
| Amount | This is the amount paid. | This is useful so the people can keep on track how much they paid for the membership. |
| Next\_Payment\_Date | This is that next payment date | This is useful so the people can remember their next membership payment date. |

Another important usage of the database is when user wants to rate the movie/show.

***Use case #5: Rate Content.***

1. The user visit streaming service site.
2. The user login in streaming service site.
3. The user clicks to rate movie/show.
4. The MyWatchList web extension will automatically record all relevant information in the database such as Streaming ID, Movie/show ID, rating ID, rating stars and number of ratings.

The database will store the relevant information about rate content in step #4. The database will record all fields listed below.

|  |  |  |
| --- | --- | --- |
| Field | What is Stored | Why it’s Need |
| Rating\_ID | This is rating ID. | This is useful so the people can remember search their rating. |
| Rating\_star | This is that rating stars for movie/show. | This is useful so the people know their rating star for the movie/show. |
| Avg\_rating\_stars | This is that average of rating stars for movie/show. | This is useful so the people know how movie/show popular. |
| Number\_of\_ratings | This is the number of ratings for movie/show. | This is useful so the people know how is the movie/show popular. |

# Structural Database Rules

***Use case #1: Account signup & installation***

1. User visits app store, download and install “MyWatchList” app.
2. The “MyWatchList” ask user to create an account when it’s first use.
3. The user enters their information, and the account is created in the database.
4. The application prompt user to install browser extension so that their movies can be automatically tracked when they watch or click them.

In this use case. I see that only step #3 starts to be associated with the database. From step#3 makes it clear that there is an Account entity in this use case. Although an Account entity can be broken down into multiple entities with relationships, But I don’t have enough information from this use case alone. So, I keep in mind the Account entity is needed for the database.

The next use case is:

***Use case #2: Watched Movies/ Shows Tracking***

1. The person visits a streaming service and starts to consume content.
2. The “MyWatchList” browser extension will start to record the relevant information in the database such as service name, title of content, released date, length etc. when user watch into content.

In this use case, there are several entities can be identified such as Streaming Service, Content (Movie/Show), Director, Actor, Studio, Genre. MyWatchList tracks the contents are watched and for which service name and director, actor, studio and genre of the content and those are associated with the Account entity which we discovered in the first use case. I now have enough information to create some structural database rules.

The structural database rules are:

1. **Each content is associated with many accounts, each account could be associated with many contents.**

I create this structural rule because I infer from the use case that each content is associated with many accounts. While an account may watch zero or many contents. The reason I make account to content optional because the account won’t have any contents associate with it in the database when its first created or if someone may install the application and create an account but never record any contents. Conversely, an account has a lot of contents associate with it when user watches a lot of contents.

1. **Each content is associated with one or many streaming services, each streaming service must be associated with many contents.**

For this structural business rule, it indicates that every content can have many streaming services associated with it, because many streaming services offer the same content in the real-world if that content is not exclusive, I make content to streaming service mandatory since content won’t be stored in my database if it is not on the streaming services. Also, based on real world example, streaming service must have many and variety contents in order to attract the customer, the business will be not good if its platform only offer one content, therefore I indicate streaming service to content mandatory.

1. **Each director directs one or many contents, each content is directed by one or more directors.**

This rule indicates that each director can have many contents associated with it, since a director may direct several contents at once, but within this use case, a director must associate with at least one content in order to be recorded, so I make director to content mandatory. And of course, each content may have multiples directors, but content must have at least one director, I indicate content to director mandatory since content can’t be made without having at least one director, and this rule ensures that every content can be traced back to its director.

1. **Each content is associated with many actors, each actor is associated with one or many contents.**

This rule indicates that every content always has many actors on their cast, there is no content that has one actor and no supporting cast, so I indicate content to actor mandatory. Since each actor can be associated in many contents, but actor must associate in at least one content within this database, therefore I indicate actor to content mandatory.

1. **Each content is associated with one or many studios, each studio associated with one or many contents.**

This rule indicates that every content can linked to many studios, and that of course each studio could have many contents associated to it. I make content to studio mandatory since a content can be produced by many studios which are cooperated, but it is mandatory that a content is to be produced by at least one studio. I make studio to content mandatory because studio can produce many contents, but studio won’t be stored in my database unless a content is produced from it.

1. **Each genre is assigned to one or many contents, each content is assigned by one or many genres.**

This rule indicates that each genre can be assigned to many contents, and of course each content can have many genres. I make genre to content mandatory since every genre is linked to at least one piece of contents, so that users can easily find the content that interests them. I indicate content to genre mandatory since each content always be categorized in some genres, sometimes it is categorized in one specific genre or many genres but will never has zero genre base on criteria of the producing studio and this is how streaming services provider can organize their contents for user to search based on their interests and get more user friendly.

The next use case is:

***Use case #3: Download Movie/ Show.***

1. The user visits streaming service site and look for movies/shows.
2. The user download movie/show to watch offline.
3. The MyWatchList web extension will automatically records all relevant information in the database which listed in the use case number 2 and in addition with downloaded status record.

In this use case, Because the date is to be stored in database similar with use case #2, I don’t see any extra entities here.

The next use case is:

***Use case #4: Payment tracking.***

1. The user visit streaming service site.
2. The user login in streaming service site.
3. The user chooses pay membership option and pays the subscription fee.
4. The MyWatchList web extension will automatically record all relevant information in the database such as account Streaming ID, payment amount, payment date.

From this payment tracking user case, the database will store the relevant information about payment in step#2 and #4, I see on entity – Subscription. MyWatchList track all of data of Subscription such as Streaming ID, payment date, payment amount and next payment date. Though not explicitly mentioned, it stands to reason that each Subscription is associated with Streaming service and MyWatchList account. The entity has a relationship with the streaming service shown as below.

1. **Each subscription is associated with one streaming service, each streaming service may be associated with many subscriptions.**

I create this structural rule because I infer from the use cases that each subscription is associated with one streaming service, and of course, since each streaming service has so many customers, so a streaming service has many subscriptions. I indicate subscription to streaming service mandatory because a customer must subscribe in order to watch content or using a streaming service in the real world. Every streaming service business must have so many customers in order to keep business running, but some streaming services may not have subscription yet, therefore, I indicate streaming service to subscription optional.

1. **Each subscription is associated with an account, each account may be associated with many subscriptions.**

I create this structural because I infer from the use case that each subscription is associated with an account. While an account may associate with many subscriptions. The reason I make account to subscription optional because the account won’t have any subscription associate with it in the database when its first created or if someone may install the application and create an account but never subscribe any streaming services to watch content. Conversely, an account has many subscriptions associate with it when user uses several streaming services.

The next use case is:

***Use case #5: Rate Content.***

1. The user visit streaming service site.
2. The user login in streaming service site.
3. The user clicks to rate movie/show.
4. The MyWatchList web extension will automatically record all relevant information in the database such as Streaming ID, Movie/show ID, rating ID, rating stars and number of ratings.

From this rate content use case, because all of this use case’s data will use to keep track of users – content – rating, so I don’t see any extra entity for this use case

So, from the 5 use cases I have thus far, I have these 8 structural database rules as following.

1. **Each content is associated with many accounts, each account could be associated with many contents.**
2. **Each content is associated with one or many streaming services, each streaming service must be associated with many contents.**
3. **Each director directs one or many contents, each content is directed by one or more directors.**
4. **Each content is associated with many actors, each actor is associated with one or many contents.**
5. **Each content is associated with one or many studios, each studio associated with one or many contents.**
6. **Each genre is assigned to one or many contents, each content is assigned by one or many genres.**
7. **Each subscription is associated with one streaming service, each streaming service may be associated with many subscriptions.**
8. **Each subscription is associated with an account, each account may be associated with many subscriptions.**

**Adding Specialization-generalization structural database rule**

1. Firstly, I had a look again previous use cases. After some carefully thought, the first use case I am interested in for this purpose.

***Use case #2: Watched Content Tracking***

1. The person visits a streaming service and starts to consume content.
2. The “MyWatchList” browser extension will start to record the relevant information in the database such as service name, tittle of content, released date, length etc. when user watch into content.

In this use case, I created several entities but the entity – Content caught my attention when I was thinking about specialization – generalization, it also became plain to me is that I would like to treat different kinds of Content differently. There are several kinds of contents such as movie, show, documentary.

To support the different kinds of contents, I modify it as follows.

***Use case #2: Watched Content Tracking (New)***

1. The person visits a streaming service and starts to consume content.
2. The “MyWatchList” browser extension will start to record the relevant information in the database such as whether the content is a movie, show, or documentary, service name, tittle of content, released date, length etc. when user watch into content.

Notice that #2 now mentions the movie, show and documentary content types.

From this modification, I derive the 9th structural database rule.

1. **A content is a movie, a show, a documentary, several of these or none of these.**

This relationship allows my database to have all three types, but I am not sure that these are the only types available. So, I made this relationship partially complete. That’s the reason why I add “or none of these” at the end of the 9th structural database rule. Also, according to Wikipedia that a documentary is a broad term to describe a non-friction movie that in some way “documents” or captures reality and some streaming services organized documentary as one type. So, I made the relationship overlapping in case user watch a documentary (also known as non-friction movie type). Therefore, I used the phrase “several of these”.

1. The second use case related to specialization-generalization is use case #4 Payment tracking.

***Use case #4: Payment tracking.***

1. The user visit streaming service site.
2. The user login in streaming service site.
3. The user chooses pay membership option and pays the subscription fee.
4. The MyWatchList web extension will automatically record all relevant information in the database such as account Streaming ID, payment amount, payment date.

Considering the specialization-generalization relationship in this use case, I would like to treat different kind of subscriptions differently. There are several kinds of subscriptions such as monthly subscription, annual subscription.

To support the different kinds of contents, I modify it as follows.

***Use case #4: Payment tracking (New)***

1. The user visits specific streaming service site.
2. The user login in streaming service site.
3. The user chooses pay membership option (such as monthly or annual subscription) and pays the subscription fee.
4. The MyWatchList web extension will automatically record all relevant information in the database such as whether subscription is monthly or annual subscription, account Streaming ID, payment amount, payment date.

Notice that step #3 and #4 now mentions the monthly and annual subscription types.

I now derive 10th structural database rule from this change to the use case #4

1. **A subscription is a monthly subscription, an annual subscription, or none of these.**

This relationship allows my database to have both types, but I also not sure if there are any other types of subscription available. So, I made this relationship partially complete. That’s why I used the phrase “none of these” at the end of 10th structural database rule. Also, As the person who is using several streaming services, I know that many streaming services have both monthly subscription and annual subscription. When you subscribe, you only can choose either monthly or annual at the same time, so the relationship is disjoint.

Now I have 10 structural database rules, including my previous 8 associative, plus the two I just created.

1. **Each content is associated with many accounts, each account could be associated with many contents.**
2. **Each content is associated with one or many streaming services, each streaming service must be associated with many contents.**
3. **Each director directs one or many contents, each content is directed by one or more directors.**
4. **Each content is associated with many actors, each actor is associated with one or many contents.**
5. **Each content is associated with one or many studios, each studio associated with one or many contents.**
6. **Each genre is assigned to one or many contents, each content is assigned by one or many genres.**
7. **Each subscription is associated with one streaming service, each streaming service may be associated with many subscriptions.**
8. **Each subscription is associated with an account, each account may be associated with many subscriptions.**
9. **A content is a movie, a show, a documentary, several of these or none of these.**
10. **A subscription is a monthly subscription, an annual subscription or none of these.**

# Conceptual Entity-Relationship Diagram

Here are the associative structural database rules I came up, relist below.

1. **Each content is associated with many accounts, each account could be associated with many contents.**
2. **Each content is associated with one or many streaming services, each streaming service must be associated with many contents.**
3. **Each director directs one or many contents, each content is directed by one or more directors.**
4. **Each content is associated with many actors, each actor is associated with one or many contents.**
5. **Each content is associated with one or many studios, each studio associated with one or many contents.**
6. **Each genre is assigned to one or many contents, each content is assigned by one or many genres.**
7. **Each subscription is associated with one streaming service, each streaming service may be associated with many subscriptions.**
8. **Each subscription is associated with an account, each account may be associated with many subscriptions.**
9. **A content is a movie, a show, a documentary, several of these or none of these.**
10. **A subscription is a monthly subscription, an annual subscription, or none of these.**

# Break down individual associative structural database rule for ERD purpose.

1. **Structure database rule # 1.**

Each content is associated with many accounts ***(Mandatory participation, Plural)***.

Each account could be associated with many contents ***(Optional participation, Plural).***

1. **Structure database rule # 2.**

Each content is associated with one or many streaming services ***(Mandatory participation, Plural).***

Each streaming service must be associated with many contents ***(Mandatory participation, Plural).***

1. **Structure database rule # 3.**

Each director directs one or many contents ***(Mandatory participation, Plural).***

Each content is directed by one or more directors ***(Mandatory participation, Plural).***

1. **Structure database rule # 4.**

Each content is associated with many actors ***(Mandatory participation, Plural).***

Each actor is associated with one or many contents ***(Mandatory participation, Plural).***

1. **Structure database rule # 5.**

Each content is associated with one or many studios ***(Mandatory participation, Plural).***

Each studio associated with one or many contents ***(Mandatory participation, Plural).***

1. **Structure database rule # 6.**

Each genre is assigned to one or many contents ***(Mandatory participation, Plural).***

Each content is assigned by one or many genres ***(Mandatory participation, Plural).***

1. **Structure database rule # 7.**

Each subscription is associated with one streaming service ***(Mandatory participation, Singular).***

Each streaming service maybe associated with many subscriptions ***(Optional participation, Plural).***

1. **Structure database rule # 8.**

Each subscription is associated with an account ***(Mandatory participation, Singular).***

Each account may be associated with many subscriptions ***(Optional participation, Plural).***

1. **Structure database rule # 9.**

A content is a movie, a show, a documentary, several of these or none of these.

***(Partial complete and overlapping)***

1. **Structure database rule # 10.**

A subscription is a monthly subscription, an annual subscription, or none of these.

***(Partial complete and disjoint)***

Here is the ERD I came up with for these rules. I use Crow’s Foot diagram to make it. As you can see, The Content entity is associated with seven other entities such as Account, Director, Studio, Actor, Genre, Streaming Service, just as it was the one of the entities in seven structural database rules. In addition, there is Subscription entity is associated with two other entities such as Account and Streaming Service. The participation and plurality constraints reflect what is in each of the structural database rules. For instance,

The  between Account and Content indicates that each Account may be associated with many contents. Likewise, the  indicates that each Content must be associated with an Account.

For the specialization-generalization diagram, I used Crow’s foot. The symbol  here represent the partial complete and overlapping. The symbol  here represent the partial complete and disjoint

Diagram

Description automatically generated

With the Conceptual ERD above. There are 2 specialization-generalization structural database rules has been added. As you can see, I defined 3 subtypes (Movie, Show, Documentary) under supertype (Content) with partial complete and overlapping relationship. Also 3 subtypes (Monthly\_Subscription, Annual\_Subcription) under supertype (Subscription) with partial and disjoint relationship.

# Full DBMS Physical ERD

From the initial 10 structural database rules which I created above. There are 8 associative relationships in my conceptual ERD such as Account/Content, Streaming Service/Content, Director/Content, Actor/Content, Studio/Content, Genre/Content, Subscription/Streaming Service, Subscription/Account. There are also 2 specialization-generalization structural database rules such as Content entity, Subscription entity.

1. **Account/Content relationship is M:N.** Each content is associated with many accounts, and each account could be associated with many contents.
2. **Streaming Service/Content relationship is M:N.** Each content is associated with one or many streaming services, and each streaming service must be associated with many contents.
3. **Director/Content relationship is M:N.** Each director directs one or many contents, and each content is directed by one or more directors.
4. **Actor/Content relationship is M:N.** Each content is associated with many actors, and each actor is associated with one or many contents.
5. **Studio/Content relationship is M:N.** Each content is associated with one or many studios, and each studio associated with one or many contents.
6. **Genre/Content relationship is M:N.** Each genre is assigned to one or many contents, and each content is assigned by one or many genres.
7. **Streaming/Subscription Service relationship is 1:M.** Each streaming service may be associated with many subscriptions, but each subscription is associated with one streaming service.
8. **Account/Subscription relationship is 1:M.** Each account may be associated with many subscriptions, but each subscription is associated with an account.
9. **Content entity relationship is partial complete and overlapping.** A content is a movie, a show, a documentary, several of these or none of these.
10. **Subscription entity relationship is partial complete and disjoint.** A subscription is a monthly subscription, an annual subscription, or none of these.

I followed the best practice of creating synthetic keys for all tables, and opted to make primary keys of the DECIMAL(12) datatype. This allow for a lot of data to be stored in my database.

Here are list synthetic keys:

1. Account table: AccountID
2. Content table: ContentID
3. Director table: DirectorID
4. Studio table: StudioID
5. Actor table: ActorID
6. Genre table: GenreID
7. StreamingService table: StreamingID
8. Subscription table: SubscriptionID
9. ***Mapping Associative Relationships.***

* Since relation model does not support the M:N relationship directly, the most the relational model supports is a 1:M relationship. So, it is necessary for me to create a bridging entity to support the relationship. There are 6 M:N relationships such as Account/Content, Streaming Service/Content, Director/Content, Actor/Content, Studio/Content, Genre/Content.
  + Bridging entity of Account/Content is Account\_Content which has foreign keys to both Account and Content in the DBMS physical ERD, resulting in two 1:M relationships between Account\_Content and Account and Content.
  + Bridging entity of Streaming Service/Content is StreamingService\_Content which has foreign keys to both Streaming Service and Content in the DBMS physical ERD, resulting in two 1:M relationships between StreamingService\_Content and Streaming Service and Content.
  + Bridging entity of Director/Content is Director\_Content which has foreign keys to both Director and Content in the DBMS physical ERD, resulting in two 1:M relationships between Director\_Content and Director and Content.
  + Bridging entity of Actor/Content is Actor\_Content which has foreign keys to both Actor and Content in the DBMS physical ERD, resulting in two 1:M relationships between Actor\_Content and Actor and Content.
  + Bridging entity of Studio/Content is Studio\_Content which has foreign keys to both Studio and Content in the DBMS physical ERD, resulting in two 1:M relationships between Studio\_Content and Studio and Content.
  + Bridging entity of Genre/Content is Genre\_Content which has foreign keys to both Genre and Content in the DBMS physical ERD, resulting in two 1:M relationships between Genre\_Content and Genre and Content.
* There are two 1:M relationships such as Account/Subscription and Streaming Service/ Subscription. Since Account/Subscription and Streaming Service/Subscription are 1:M relationship, I retained the entities from conceptual ERD, and placed foreign keys in Subscription (AccountID, StreamingID) since Subscription entity that associate at most one Account and at most one Streaming Service.

1. ***Mapping Specialization-Generalization Relationships.***

Because the specialization-generalization relationship is known as the “is a” relationship, so the primary key of each subtype is a foreign key to the primary key of the supertype.

* Additional entities under Content (supertype) are Movie, Show and Documentary (subtypes) each of which have a primary key and foreign key of ContentID which reference the primary key of Content.
* Additional entities under Subscription (supertype) are Monthly\_Subscription and Annual\_Subscription (subtype), each of which have a primary key and foreign key of SubscriptionID which reference the primary key of Subscription.
* Mapping Conceptual ERDs to DBMS Physical ERDs.

Diagram

Description automatically generated

# Adding Attributes to DBMS Physical ERD

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Attribute** | **Datatype** | **Reasoning** |
| Account | UserName | VARCHAR(64) | Every account has a username associated with it, which is need to login to MyWatchList. I allowed usernames to be up to 64 characters. |
| Account | E\_Password | VARCHAR(255) | Every account has a password. It will be stored in encrypted text format in the database. I allow up to 255 characters. |
| Account | FirstName | VARCHAR(255) | This is the first name of the account  holder, up to 255 characters of the  name. |
| Account | LastName | VARCHAR(255) | This is the last name of the account holder, up to 255 characters of the name. |
| Account | Email | VARCHAR(255) | This is the email of account holder. I allow up to 255 characters of the email. |
| Account | Phone | DECIMAL(16) | This is the phone number of account holder. I allow up to 16 digits due to some countries has a long phone number. |
| Account | DOB | DATE | This is the date of birth of account holder, this is using for verifying user. |
| Content | Title | VARCHAR(255) | Every content has a title, I allow up to 255 characters in case the content has long title. |
| Content | Released\_date | DATE | This is the released data of the content |
| Content | Language | VARCHAR(50) | This is the language of the content; I allow up 50 characters for language |
| Content | Avg\_Rating\_Stars | DECIMAL(1) | The content is always be rated, so I create avg\_rating\_star attribute which is number of stars of the content. Usually there is 5 stars, so I allow 1 digit. |
| Content | Is\_movie | BOOLEAN | This attribute indicate Content is a movie or not. True is movie, False is not |
| Content | Is\_show | BOOLEAN | This attribute indicate Content is a show or not. True is show, False is not |
| Content | Is\_documentary | BOOLEAN | This attribute indicate Content is documentary or not. True is documentary, False is not |
| Show | Season\_Name | VARCHAR(255) | A show has multiple season, this is the name of the season, I allow up to 255 characters. |
| Show | Season\_Date | DATE | This is the released date of the season. |
| Show | S\_Episode | DECIMAL(4) | This is the episode ‘number of the show, It is very rare if show has 9999 episodes, so I allow up 4 digits. |
| Show | S\_Episode\_Title | VARCHAR(255) | Beside the main Show’ title, each episode has it’ own title. |
| Show | S\_Episode\_Date | DATE | This is the date when the episode of the show released. |
| Documentary | D\_Type | VARCHAR(255) | This is the documentary type of the content. I allow up to 255 characters. |
| Director | D\_fname | VARCHAR(255) | This is director’s first name; I allow up to 255 characters. |
| Director | D\_lname | VARCHAR(255) | This is director’s last name; I allow up to 255 characters. |
| Studio | Studio\_Name | VARCHAR(255) | This is studio name which content is made; I allow up to 255 characters. |
| Actor | A\_fname | VARCHAR(255) | This is actor’s first name; up to 255 characters should be enough. |
| Actor | A\_lname | VARCHAR(255) | This is actor’s last name; up to 255 characters should be enough. |
| Actor\_Content | Role | VARCHAR(150) | This is type of acting role of the actor in the content. I allow 150 characters that should be enough. |
| Genre | Gen\_Title | VARCHAR(150) | This is the genre’ title of the content, up to 150 characters should be enough. |
| StreamingService | S\_Name | VARCHAR(150) | This is the streaming name, which is used by the user, I allow up to 150 characters. |
| StreamingService | URL | VARCHAR(1024) | This is the website URL path of streaming service. I allow up to 1024 characters and this should be enough. |
| Subscription | S\_Account | VARCHAR(150) | This is a particular streaming’s account, and this account is used to subscribe membership, so I allow up to 150 characters. Due to security of all streaming service in the real world, so I don’t record the password of the streaming account into my database. |
| Subscription | PaymentDate | DATE | This is the payment date when account paid for the subscription. |
| Subscription | Amount | DECIMAL(7,2) | This is the amount that user pay for subscription. I allow for up to 7 digits, though it will likely never get near this high. |
| Subscription | Next\_Payment\_Date | DATE | This is the date when account need to pay fee for renewal the subscription. |
| Subscription | Subscr\_type | VARCHAR(1) | This is the subscription type of the account. “M” is monthly. “A” is annual. |
| Annual\_Subscription | Discount\_Fee | DECIMAL(4) | This is a discount amount dollar fee which is offer to the annual subscription. Up to 4 digits would be enough. |

I think that I may capture all the necessary fundamental attributes for the MyWatchList in the table above. I see that I could be more detailed with more attributes in some entities but given the use cases and structural database rules I’ve developed thus far, these attributes included.

Diagram

Description automatically generated

# Each of the attributes have been added to their respective entities in the ERD. The previously added primary and foreign keys have also been retained. There is a comment that I did not identify any attributes necessary for some entities such as Account\_Content, Director\_Content, Studio\_Content, Movie, Genre\_Content, StreamingService\_Content and Monthly\_Subscription. I may identify some as the application is further developed.

I feel that this is a solid DBMS physical ERD for the use cases and structural database rules I have added thus far in the design.

# Normalizing DBMS Physical ERD

When I look at my DBMS physical ERD, I notice that there is redundancy in my physical ERD, and that is in the Show entity (Season\_Name 🡪 S\_Episode, S\_Episode\_Title, S\_Eposide\_Date).

This is my ERD with the normalized.

Diagram

Description automatically generated

There are two addition entities after normalization – Season and Episode. By moving the primary season and episode information into its own entity, I already remove dependency of the entity Show.

Below are my structural database rules modified to reflect the new entity. The new two are italicized.

1. **Each content is associated with many accounts, each account could be associated with many contents.**
2. **Each content is associated with one or many streaming services, each streaming service must be associated with many contents.**
3. **Each director directs one or many contents, each content is directed by one or more directors.**
4. **Each content is associated with many actors, each actor is associated with one or many contents.**
5. **Each content is associated with one or many studios, each studio associated with one or many contents.**
6. **Each genre is assigned to one or many contents, each content is assigned by one or many genres.**
7. **Each subscription is associated with one streaming service, each streaming service may be associated with many subscriptions.**
8. **Each subscription is associated with an account, each account may be associated with many subscriptions.**
9. **A content is a movie, a show, a documentary, several of these or none of these.**
10. **A subscription is a monthly subscription, an annual subscription, or none of these.**
11. ***Each show is associated with one or many seasons, each season is associated with one show.***
12. ***Each season is associated with one or many episodes, each episode is associated with one season.***

# Below is my new conceptual ERD to reflect the new entity (Episode).

# Diagram Description automatically generated

# The Show entity are now included in the conceptual ERD, and the conceptual ERD is sync with the structural database rules and the DBMS physical ERD.

# Tables Create Script

The following screenshots are the scripts for creating the tables, sequences, drop table, drop sequences. I put the DROP TABLE commands at the top so that the script is re-runnable, then followed by DROP SEQUENCE, CREATE TABLE and CREATE SEQUENCE. All columns, constraints and sequences are included as illustrated in the ERD. Note that I am using PostgreSQL for MyWatchList.

Table

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

A picture containing text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

# Stored Procedure Execution and Explanations

The first use case for MyWatchList is account signup & installation use case listed below

***Use case #1: Account signup & installation***

1. User visits app store, download and install “MyWatchList” app.
2. The “MyWatchList” ask user to create an account when it’s first use.
3. The user enters their information, and the account is created in the database.
4. The application prompt user to install browser extension so that their movies can be automatically tracked when they watch or click them.

For this use case, I will implement a transaction that create an account, using Postgresql.

Below is screenshot of my stored procedure definition.

Graphical user interface, text, application, email

Description automatically generated

I name the store procedure “ADD\_ACCOUNT” and give it parameters that correspond to the Account table. Since I used nextval(‘account\_seq’) to create AccountID, so I do not need the parameter for that. There is one insert statement to insert into Account table.

Below is screenshot of my stored procedure execution

Graphical user interface, text, application

Description automatically generated with medium confidence

I nested the stored procedure call between transaction control statements to ensure the transaction is committed.

# Question Graphical user interface, application Description automatically generated

***Use case #2: Watched Content Tracking***

1. The person visits a streaming service and starts to consume content.
2. The “MyWatchList” browser extension will start to record the relevant information in the database such as whether the content is a movie, show, or documentary, service name, tittle of content, released date, length etc. when user watch into content.

# For this use case, I will implement a transaction that input movie content data, using Postgresql.

Here is the screenshot of my stored procedure definition.

Graphical user interface, text, application, email

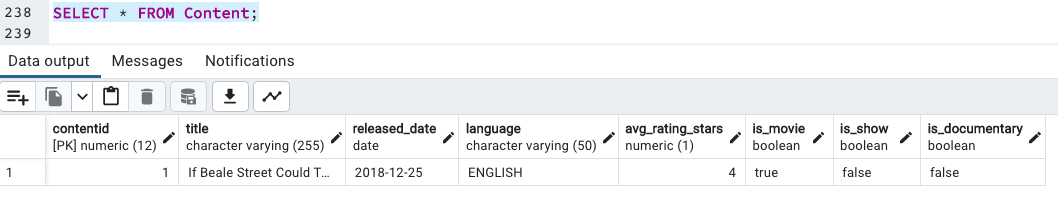
Description automatically generated

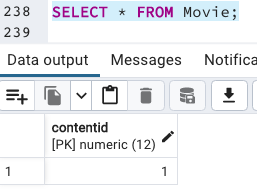
I name the stored procedure “ADD\_MOVIE\_CONTENT” and give it parameters that correspond to the Content and Movie tables. Instead of hardcoding ContentID, I use nextval(‘content\_seq’) to get automatedly ContentID in the Content table and I also use currval(‘content\_seq’) for ContentID in the Movie table because Content and Movie is specialization-generalization relationship.

Graphical user interface, text, application

Description automatically generated

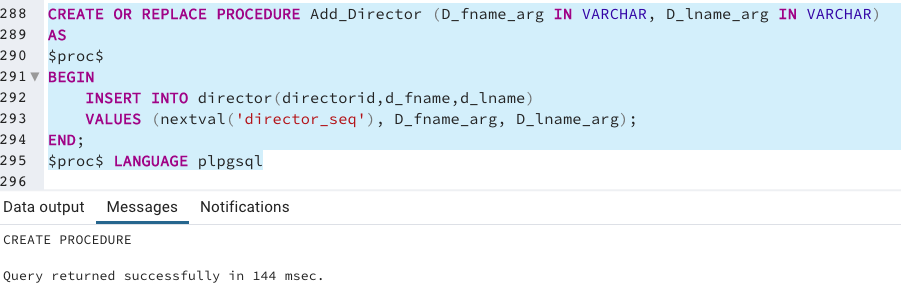
I nested the stored procedure call between transaction control statements to ensure the transaction is committed.

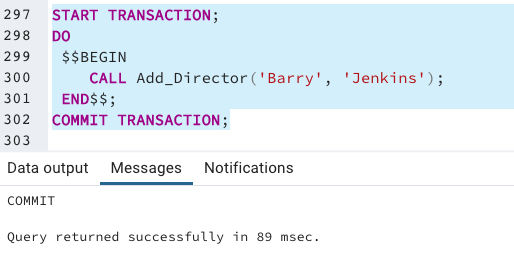


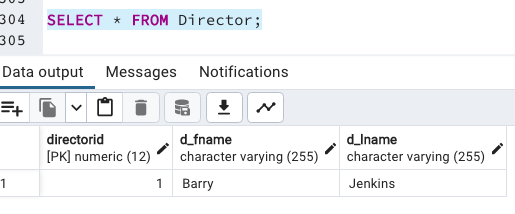


In this use case, I want to implement one more transaction that input the Director’ information into Director table.

Here is the screenshot of the stored procedure.







# Question Identification and Explanations

Here is my question useful to the core operation of MyWatchList:

1. how many accounts watch one content? Who have directed this content?

First, I explain why this question is useful. The answer can be used to give the user an idea what content is the most popular now and how many people are watching it. It also helps the user update the trending content because this database list out who many people are watching this content across the platforms. In addition, the user also knows who directed this content, and help the director to be well known because it also lists out director along with the content’ title.

1. What is different between monthly subscription and annual subscription? What streaming service does each account subscribe?

I explain why this question is useful. The answer can be useful to help the user know how many streaming services they are using now and give them the idea that how much they must pay for the subscription in term of monthly or annual subscription. And it also tells them what benefit they can get if their subscription is annual.

1. How many account subscriptions and contents does each streaming service has?

The question is useful because the answer to this question can be useful to help the user know which streaming service has the highest number of contents, also let them know how many people is subscribing to each streaming service, therefore, they can have better decision when they choose new streaming service.

How many streaming services does the MyWatchList? And how much do they need to pay for each service?

# Query Executions and Explanations

1. Query 1 for question 1.

Below is a screenshot of the query I use.

Table

Description automatically generated with medium confidence

To get the result, First, I join the Account to the Account\_Content by using AccountID. Second, I continue join to Content by using ContentID. Third, I continue join to Director\_Content by using ContentID, then continue to join Director by using DirectorID, Then I use Group By function to group by Contenid, title, director\_name, and order the result by descending order of Number\_of\_User\_Watch.

Upon inspection the result, you see that the content name The Friends and It’s director – David Crane is the most popular and having 5 users watch.

1. Query 2 for question 2.

Table

Description automatically generated

To get the result, I join the Account to Subscription table by using AccountID, then I use left join instead of join (inner joint) to continue to join to Annual\_Subscription in order to get Discount subscription fee and not lose the value for the row which does not have discount fee, Then I join to StreamingService table and Then I order the result by AccountID and s\_account( subscription user name).

Upon inspection above result, you can see that the different between Monthly subscription and annual subscription is that all streaming service has discount fee for annual subscription. And we can also see which streaming services is each MyWatchList user using. For example: the username ‘nvui’ is using Netflix and Disney Plus at the same time, and he gets discount fee for $20 for Disney Plus service due to he is subscribing annually.

# Query for question 3.

Graphical user interface, text, application, email

Description automatically generated

Table

Description automatically generated

To get this result, First, I create a view name Number\_Content\_In\_StreamingService. In this view’ result, I join Content to StreamingService\_content by using contented, I continue join to StreamingService table, and then I group the result by using Group By function to group streamingid and S\_name, and then I order the result by using Order By function to order number\_of\_content descending.

Then I use the view which I created above into the query. I join the view

(Number\_Content\_In\_StreamingService) to Subscription table by using StreamingID, then I group the result by using Group By function and then I use order by function to order the results descending.

Upon inspection the result, you see that there are 5 streaming services such as Netflix, Disney Plus, Hulu, Amazon Prime, Apple TV+ which are using by MyWatchList accounts. We also see that Netflix is the most popular service and contribute highest number of contents.

# Index Identification and Creations

As far as primary keys which are already indexed, below is the list.

Account.AccountID

Content.ContentID

Director.DirectorID

Studio.StudioID

Actor.ActorID

Actor\_Content.Actor\_ContentID

Movie.ContentID

Show.ContentID

Season.Season\_ID

Episode.Episode\_ID

Documentary.ContentID

Genre.GenreID

StreamingService.StreamingID

Subscription.SubscriptionID

Monthly\_Subscription.SubscriptionID

Annual\_Subscription.SubscriptionID

As far as foreign keys, I know all of them need an index. Below is a table identifying each foreign key column, whether the index should be unique or not, and why.

|  |  |  |
| --- | --- | --- |
| **Column** | **Unique?** | **Description** |
| Account\_Content.AccountID | Not Unique | The foreign key in Account\_Content referencing Account is not unique because the same account can be associated with many contents |
| Account\_Content.ContentID | Not Unique | The foreign key in Account\_Content referencing Content is not unique because the same Content can be associated with many accounts |
| Director\_Content.DirectorID | Not Unique | The foreign key in Director\_Content referencing Director is not unique because the same Director can direct many Contents |
| Director\_Content.ContentID | Not Unique | The foreign key in Director\_Content referencing Content is not unique because the same Content can be directed by many Directors. |
| Studio\_Content.StudioID | Not Unique | The foreign key in Studio\_Content referencing Studio is not unique because the same Studio is associated with many Contents. |
| Studio\_Content.ContentID | Not Unique | The foreign key in Studio\_Content referencing Content is not unique because the same Content is associated with many Studios. |
| Actor\_Content.ContentID | Not Unique | The foreign key in Actor\_Content referencing Content is not unique because the same Content is associated with many actors. |
| Actor\_Content.ActorID | Not Unique | The foreign key in Actor\_Content referencing Actor is not unique because the same Actor is associated with many contents. |
| Season.ContentID | Not Unique | The foreign key in Season referencing Content is not unique because the same Content is associated with many Seasons. |
| Episode.Season\_ID | Not Unique | The foreign key in Episode referencing Season is not unique because the same Season is associated with many Episodes. |
| Genre\_Content.GenreID | Not Unique | The foreign key in Genre\_Content referencing Genre is not unique because the same Genre is associated with many contents. |
| Genre\_Content.ContentID | Not Unique | The foreign key in Genre\_Content referencing Content is not unique because the same Content is associated with many Genres. |
| StreamingService\_Content.StreamingID | Not Unique | The foreign key in StreamingService\_Content referencing StreamingService is not unique because the same StreamingService is associated with many contents. |
| StreamingService\_Content.ContentID | Not Unique | The foreign key in StreamingService\_Content referencing Content is not unique because the same Content is associated with many StreamingServices. |
| Subscription.AccountID | Not Unique | The foreign key in Subscription referencing Account is not unique because the same Account is associated with many Subscriptions. |
| Subscription.StreamingID | Not Unique | The foreign key in Subscription referencing StreamingService t is not unique because the same StreamingService is associated with many Subscriptions. |

- As far as the three query driven indexes, I spotted three fairly easily by predicting what columns will commonly be queried. For instance, its reasonable that there will be many queries that limit by S\_name (streaming service), to see which accounts register to the particular streaming service or to see which contents are offering in the particular streaming service. So, I select StreamingService.S\_Name to be indexed. This would be a non-unique index because many subscriptions could associate to one streaming service.

- it’s also reasonable that the average rating stars (Avg\_Rating\_Stars) will be a limiting column in queries because the user will commonly want to limit their searching by average rating stars (Avg\_Rating\_Stars). So I select Content.Avg\_Rating\_Stars to index. This would be non-unique index because many contents can have the same average rating stars (Avg\_Rating\_Stars).

- Lastly, it’s reasonable that the released date of Content will be a limiting column for some queries, such as queries that want to see how many contents were released in a certain date range. So, I select Content.Released\_date to index. This would be non-unique index because many contents can be released on the same day.

Here is a screenshot demonstrating of a primary key index Which are listed above for MyWatchList

A picture containing graphical user interface

Description automatically generated

Below is screenshot for the foreign key index.

Table

Description automatically generated with medium confidence

Note that, the screenshot for primary and foreign key index above is not all, there will be more in the sql script.

Below is a screenshot demonstrating creation of a query-driven index.

Graphical user interface, text, application, email

Description automatically generated

# History Table Demonstration

Because my database’s purpose is mainly tracking the Content’s data, So, in reviewing my DBMS physical ERD, on piece of data that would obviously benefit from a historical record a content’s Avg\_Rating\_Stars in the Content table. Such a history would help the user update the popular and rating of the content when more and more people rate it (the content’s average rating can be higher or lower or keep the same) and Content can be judged better, so it will help user to decide to continue to watch the content or not if the content’s time is long.

First, my new structural database rule is: ***Each Content may have many average rating stars change; Each average rating stars change is for a Content.***

My updated conceptual ERD is below.

Diagram

Description automatically generated

I added the Avg\_Rating\_Stars\_Change entity and related it to Content entity. My updated DBMS physical is below.

Diagram

Description automatically generated

The Avg\_Rating\_Stars\_Change entity is present and linked to Content. Below are the attributes I added and why.

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| Stars\_ChangeID | This is the primary key of the history table. It is a DECIMAL (12) to allow many values. |
| Old\_Stars | This is the average rating stars of the Content before change. The datatype mirrors the Avg\_Rating\_Stars datatype in the Content table. |
| New\_Stars | This is the average rating stars of the Content after change. The datatype mirrors the Avg\_Rating\_Stars datatype in the Content table. |
| ContentID | This is foreign key to the Content table, a reference to the Content table that had the change in Avg\_Rating\_Stars. Datatype is DECIMAL (12). |
| ChangeDate | This is the Date the average rating stars change occurred, with DATE datatype. |

Below is a screenshot of my table and sequence creation, which has all of the same attributes and datatypes as indicated in the DBMS physical ERD.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Below is a screenshot of my trigger creation which will maintain the Avg\_Rating\_Stars\_Change table.

Graphical user interface, text, application, email

Description automatically generated

I explain it here line by line.

|  |  |
| --- | --- |
| **CODE** | **DESCRIPTION** |
| CREATE OR REPLACE FUNCTION StarsChangeFunction()  RETURNS TRIGGER LANGUAGE plpgsql | This starts the definition of a function named “StarsChangeFunction” that will be executed when the trigger fires. The language used is Postgres’ version of PL/SQL. |
| AS $trigfunc$  BEGIN | This is part of syntax starting the function block. |
| INSERT INTO avg\_rating\_stars\_change(stars\_changeid, old\_stars, new\_stars, contentid, changedate)  VALUES (nextval('Avg\_Rating\_Stars\_Change\_seq'), OLD.Avg\_Rating\_Stars, NEW.Avg\_Rating\_Stars, NEW.ContentID, CURRENT\_DATE); | This is the insert statement that records the average rating stars change by adding a row into the avg\_rating\_stars\_change table. The stars\_changeid colume is genrerated by using the Avg\_Rating\_Stars\_Change\_seq. The old\_stars and new\_stars are accessed through the NEW and OLD pseudo tables provided in plpsql triggers. The Content ID is extracted from the NEW pseudo table. The built-in variable current\_date obtains the current date. |
| RETURN NEW;  END;  $trigfunc$; | This ends the function definition. |
| CREATE TRIGGER StarsChangeTrigger  BEFORE UPDATE OF Avg\_Rating\_Stars ON Content  FOR EACH ROW | This indicate that a trigger name “StarsChangeTrigger” is being defined, to be triggered whenever the Avg\_Rating\_Stars column is updated in the Content table. The trigger is to run for each row updated. |
| EXECUTE PROCEDURE StarsChangeFunction(); | This indicates that the trigger executes the function StarsChangeFunction() whenerver it is executed. |

I have a screenshot about the Content before I am doing any updates.

Table

Description automatically generated

Next, I update the avg\_rating\_stars several times.

Graphical user interface, text, application, email

Description automatically generated

Last, I verify that the Avg\_Rating\_Stars\_Change table has a record of these average rating stars changes in the screenshot below.

Table

Description automatically generated

As you can see that there are two 6 rows which records the old average rating stars and new average rating stars Ex: change from 4 stars to 5 stars of the contentid 1. The old and new average rating stars are now tracked with a trigger and a history table.

Besides of the core operation’s question, now we can ask useful question from the history table.

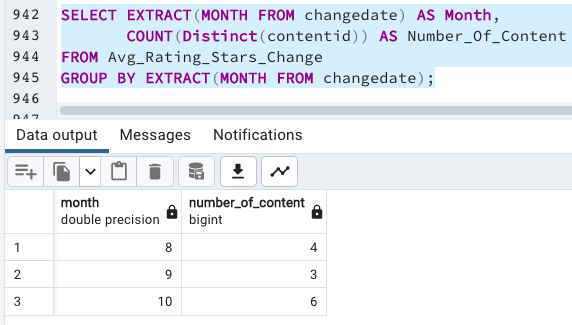
*How many contents had their average rating stars changed each month?* This question is useful because the user can look at it and have an idea of the trending content base on people rating.

In order to provide more data for this, I added a couple more stars change in different months. Below is what the Avg\_Rating\_Stars\_Change tale looks like after these changes.

Table

Description automatically generated

There are 4 content’s Average rating stars change in August (8), 3 content’s average rating stars change in September (9) and 6 content’s average rating stars change in October (10). Below is a screenshot of the query and its results.



Below is an explanation of the query

|  |  |
| --- | --- |
| **CODE** | **DESCRIPTION** |
| SELECT EXTRACT(MONTH FROM changedate) AS Month, COUNT(Distinct(contentid)) AS Number\_Of\_Content | This code obtain the Month of the year and number of the contents. EXTRACT(MONTH FROM changedate) – I extract the month of the column changedate, COUNT(Distinct(contentid))— I only count each contentid by using Distinct(contentid). |
| FROM Avg\_Rating\_Stars\_Change  GROUP BY EXTRACT(MONTH FROM changedate); | This is obtain the rows from Avg\_Rating\_Stars\_Change and I group the result with EXTRACT(MONTH FROM changedate) by the GROUP BY function. |

# Data Visualizations

# *The first visualization.*

# My app is about tracking the content, so I think the most important question is about content and I come up with the question “how many contents does user consume on each streaming service?”. Answer this question will give the user an idea which streaming platform has the highest number of contents consuming and help them to decide to renew their subscription in the future more effectively.

To answer this question, I develop and execute the query.

# Graphical user interface, application Description automatically generated

I exported data to CSV and changed the column names with more human readable.

# Table Description automatically generated

Since the results only have one measure, I plan to use simple pie chart to visualize the results.

# What story does this visualization tell us? I observe several things. The chart illustrates that there is a small percentage of content which consumes by the user, it is about 9% which is only 2 Contents. In contrast of Hulu, Netflix is the streaming service which has the highest contents, which were consumed by user, it’s exactly 44% - 10 Contents. Follow by Amazon Prime and Apple TV+, those has same popularity with 17% - 4 contents. Disney Plus follows by 13% - Contents.

This information could be used in a variety way. But this most useful way is to give the user a roughly idea to choose the best streaming service and the most popular one.

***The second visualization*** is coming from data in my history table.

Below is a screenshot of my query.

Table

Description automatically generated

After exportin to a CSV and change columns to more human readable and creating bar chart in Excel, I see the following result.

Chart, bar chart

Description automatically generated

The bar chart illustrates the number of contents which has average rating change in the month of August, September, and October.

As we can see that October has the highest number of Contents which has average rating change with 6 Contents. In contrast, there is lowest number of contents which has average rating change in September – 3 Contents. Meanwhile, the number of Contents is 4 contents in August.

This information could help the user have an overview in which Contents has change the rating base on the reviews of other users, Therefore I could save them time to decide to continue watch that contents or not.

# Summary and Reflection

The database I built is utilized in the MyWatchList application which keeps track of all movies/shows information are from various streaming platforms in one place, giving user ability to view the movies/ shows history and provide the user with a reminder or an idea when they paid subscription fee, amount payment and the date of payments. Generally, people may use multiple service providers to watch movies/ shows, and it can be difficult to remember which movies/shows belong to specific streaming platform when they watch several movies/ shows at once every day. Additionally, it might be difficult to keep on track of subscription fees, and The MyWatchList will offer a one interface for all movies/shows that user watches or downloads. The user inputting, searching, and even analyzing their movies/shows from all streaming services must be supported by the database.

As I reflect on the database and my accomplishment, I can say it’s been a long but rewarding road. It is amazing to see a real database in live.