

**IMPORTANT NOTE:** My responses for Iteration 5 of this project are typed in blue.

## **Project Direction Overview**

I am developing an app that makes cross-media entertainment recommendations to users. User data is collected by the app based on the entertainment a user consumes (a movie, a book series, a video game, etc.) and uses that data to recommend other entertainment that may belong to a different medium. For example, if someone has watched a zombie film, the app might recommend the TV series *The Walking Dead*. If someone listens to Frank Sinatra's music, the app might recommend a film he appears in, or a television series set in the 1960s.

I believe a project like this would be best focused on a company that already offers a variety of entertainment across different mediums, so I will focus on Amazon's offerings (Prime Video, Amazon Music, Twitch). If the product is (theoretically) created for Amazon's media only, it can tap into Amazon's existing databases for all their entertainment content (film titles, genres, release dates, video game data from Twitch, TV series data from Prime Video). Amazon already has a version of this (as do most streaming services) where the user is recommended TV shows and films based on their watch history. I would like this product to expand on that by also incorporating music and video games, possibly even books.

The database will store the user's information such as first name, last name, email, address, and payment information, as well as their entertainment viewing/listening history while logged into their Amazon account (through Prime Video, Twitch, or Amazon Music). The entertainment data will be broken into medium, genre, subgenre, and release date (possibly just a year). I included "subgenre" as genres can often be too broad. For example, "The Bachelor" and "Ice Road Truckers" are both in the reality show genre, but someone who enjoys "The Bachelor" will not necessarily enjoy "Ice Road Truckers" just because it is another reality show. A subgenre of "dating show" or "competition show" would need to be applied to be able to recommend relevant content to the user.

My personal background is in the entertainment industry, and I am interested in the future of entertainment as it affects the creator, the distributor, and the consumer. I believe streaming platforms will replace traditional cable television and box office sales, and I think distributors and creators alike (i.e. film studios, directors, game developers, authors) would benefit greatly from the data that companies like Amazon, Netflix, or Disney can mine from their users via their streaming services.

## Use Cases and Fields

### Category 1: Account Info and Purchases:

#### 1. Account Signup / Installation Use Case

1. The user visits Amazon's website or downloads a mobile application
2. The application asks them to create an account
3. The user enters their identifying information (Name, Email, Address, Payment)
4. Then the user can begin viewing video products with Prime Video, listening to music with Amazon Music, or watching streamers on Twitch, all using the same main account.

Field	What It Stores	Why It's Needed
FirstName	User's first name	Need to identify the user, and use their first name for personalized communications (emails, recommendations)
LastName	User's last name	Need to identify the user
UserEmail	User's email	Needed as an ID# for the account and for communications
UserAddress	User's address	Needed for payment billing/shipping
UserPayment	User's payment information	For Amazon Prime billing, media subscriptions, media purchases (like renting or buying a movie)

#### 2. Purchase Tracking Use Case

1. The user purchases or rents a film on Prime Video, buys a video game, subscribes to a Twitch streamer, purchases a subscription to Amazon Music or a premium TV network
2. Amazon records the purchase and updates the database with the medium, genre, subgenre and release date of the media that was purchased.

Field	What It Stores	Why It's Needed
PurchaseDate	A date a purchase was made	To know when a user interacted with certain media, particularly by making a purchase, so that a timely recommendation can be made.
PurchasePrice	The price of that purchase	To know how much a user will typically spend on media
PurchaseProduct	What the user bought	May tie in with genre, subgenre, etc. What the user is interacting with is important to know to make relevant recommendations to them.

## Category 2: Viewing/Streaming/Listening Activity

### 3. Prime Video View Tracking Use Case

1. The user interacts with media via Prime Video (a film or series) for a specified period of time (maybe 30 minutes or more)
2. Amazon records the activity and updates the database with the medium, genre, subgenre and release date of the media that the user interacted with for at least 30 minutes.

### 4. Twitch Stream View Tracking / Streaming Use Case

1. The user streams or watches a stream via Twitch (a video game) for a specified period of time (maybe 30 minutes or more)
2. Amazon records the activity and updates the database with the medium, genre, subgenre and release date of the media that the user interacted with for at least 30 minutes.

### 5. Amazon Music Streaming Use Case

1. The user listens to music on Amazon Music for a specified period of time (maybe 30 minutes or more)
2. Amazon records the activity and updates the database with the medium, genre, subgenre and release date of the media that the user interacted with for at least 30 minutes.

Field	What It Stores	Why It's Needed
Platform	Which Amazon platform the media was accessed on (PrimeNow, Twitch, Amazon Music)	Need to know where the interaction came from to see where the user is accessing content the most
Media	Which medium (movie, series, game, music)	Must know what medium the user first interacts with to be able to make a cross-medium recommendation
Genre	The genre of the medium	Helps us make relevant recommendations based on the genre
Subgenre	The subgenre of the medium	Helps us make even more relevant recommendations based on the more specific subgenre
RelDate	What year the media that was accessed was originally released	Helps to make recommendations for other media that was released around the same time (ex. a Star Wars game and a Star Wars film that release in the same year)
IntTime	Time the user interacted with the media	If the user interacts with the media for more than 30 minutes, we can infer that they are enjoying it, and thus can make recommendations based on what they enjoy

### Category 3: Recommendation

#### 6. Recommendation Use Case

1. The user logs into an Amazon landing page (maybe Amazon's home page, or the Prime Video landing page, Twitch, Amazon Music, Alexa app, etc.)
2. A widget appears on the landing page making recommendations to the user based on their past activity
  - a. For example, if the user is logging into Prime Video, there may be recommendations there based on a Twitch stream they watched.
3. If the user interacts with recommended media, then the data from that interaction (if it exceeds 30 minutes) will loop back into the database, and more recommendations will be made based on that data

4. Perhaps there should be functionality here with a directive of what to do if a user does not interact with a recommendation for a specified period of time (maybe a few days or a week)

Field	What It Stores	Why It's Needed
RecPlatform	Which Amazon platform the media is recommended for (PrimeNow, Twitch, Amazon Music)	We need to know where we are sending the user (i.e. if they watch a lot of Twitch, maybe we want to recommend a movie, so they go and check out Prime Video)
RecMedia	Which medium (movie, series, game, music) is being recommended	This is the medium we are sending the user to from their original "Media" input
RecGenre	The genre of the medium	This is the genre we are recommending to the user based on their media history
RecSubgenre	The subgenre of the medium	This is the subgenre we are recommending to the user based on their media history
RecRelDate	What year the media that is being recommended was originally released	Release dates are as useful as genre for making relevant recommendations based on prior view / listening history

## Structural Database Rules

### Category 1: Account Info and Purchases:

#### 1. Account Signup / Installation Use Case

1. The user visits Amazon's website or downloads a mobile application
2. The application asks them to create an account
3. The user enters their identifying information (Name, Email, Address, Payment)
4. Then the user can begin viewing video products with Prime Video, listening to music with Amazon Music, or watching streamers on Twitch, all using the same main account.

This use case is all about identifying the user. Because Amazon has so many platforms for one user to interact with, the relationship of a user to their media streaming platforms would be **one-to-many**.

Other identifying use cases are, typically, **one-to-one**. Examples of this include: one user to one address, one user to one credit card number, one user to one email.

It is possible that multiple users may live at the same address, or use the same Prime Video account (such as when you have one Netflix account for the household, but multiple individual users). For the purposes of this project, we will assume a **one-to-one** relationship between a user and all of their identifying information (address, email, payment).

#### To Summarize:

- One user may have many streaming accounts (Prime Video, Amazon Music, Twitch), in which case **this relationship is one-to-many (1:M)**
- For the purposes of this project, we will assume that one user has only one email, one address, and one form of payment. **This relationship will be one-to-one (1:1).**

#### Structural Database Rules from this Use Case:

1. Each Amazon Prime account is associated with one user
2. Within an Amazon Prime account, a user may have many streaming accounts (Prime Video, Twitch, Amazon Music)

#### Entities:

- User
  - Subentities: Identifying information (address, payment, email)
- Accounts
  - Subentities: Prime Video, Twitch, Amazon Music

#### 2. Purchase Tracking Use Case

1. The user purchases or rents a film on Prime Video, buys a video game, subscribes to a Twitch streamer, purchases a subscription to Amazon Music or a premium TV network
2. Amazon records the purchase and updates the database with the medium, genre, subgenre and release date of the media that was purchased.

Similar to the above use case, one user may have many streaming accounts under the Amazon Prime umbrella. Further, within each of these streaming accounts, a user will almost certainly be streaming or purchasing more than one unit of media (i.e., renting multiple movies on Prime Video, subscribing to multiple streamers on Twitch, purchases multiple premium TV network subscriptions, etc.)

**This creates a sort-of nested rule, where one user may have many streaming accounts, and each of those streaming accounts may have many purchases associated with them.**

Further, the purchases will also likely span an array of media, genres, and subgenres. **Each purchase will be assigned to one set of medium, genre, subgenre, and release date.**

#### **Structural Database Rules from this Use Case:**

1. Each Amazon Prime User may have many streaming accounts, and each of those accounts may have many purchases associated with it.
2. Each purchase will be associated with one set of data comprised of medium, genre, subgenre, and release date.

#### **Entities:**

- Subaccount (Twitch, Prime Video, Amazon Music)
  - Subentity: Purchases
    - Subentities: Medium, genre, subgenre, release date (as a set)

### **3. Prime Video View Tracking Use Case**

3. The user interacts with media via Prime Video (a film or series) for a specified period of time (maybe 30 minutes or more)
4. Amazon records the activity and updates the database with the medium, genre, subgenre and release date of the media that the user interacted with for at least 30 minutes.

### **4. Twitch Stream View Tracking / Streaming Use Case**

3. The user streams or watches a stream via Twitch (a video game) for a specified period of time (maybe 30 minutes or more)
4. Amazon records the activity and updates the database with the medium, genre, subgenre and release date of the media that the user interacted with for at least 30 minutes.

## 5. Amazon Music Streaming Use Case

3. The user listens to music on Amazon Music for a specified period of time (maybe 30 minutes or more)
4. Amazon records the activity and updates the database with the medium, genre, subgenre and release date of the media that the user interacted with for at least 30 minutes.

The above three use cases track streaming activity for a certain duration of time (30 minutes or more). As stated in Iteration 1, the reasoning for this is that it can be inferred that a user is enjoying the content they are streaming if they remain invested in it for at least 30 minutes, as opposed to starting a movie, for example, and deciding they don't like it and turning it off after 15 minutes. This ensures that recommendations made to the user (Use Case #6) are relevant to their interests.

Similar to purchases, **each user will generate many records of streaming activity across many platforms. This will be a nested structure, where one user → one streaming account → many occurrences of streaming activity. Some of these occurrences will be over 30 minutes, and some will be under. Like with purchases, each occurrence of streaming activity will be assigned to one dataset of medium, genre, subgenre and release date.**

### Structural Database Rules from this Use Case:

1. Streaming occurrences will either exceed 30 minutes or not.
2. Each Amazon Prime User may have many streaming accounts, and each of those accounts may have many streaming occurrences associated with it.
3. Each streaming occurrence will be associated with one set of data comprised of medium, genre, subgenre, and release date.

### Entities:

- Sub-account (Twitch, Prime Video, Amazon Music)
  - Subentity: Streaming Occurrence
    - EITHER Exceeds 30 minutes OR not
    - Subentity: Medium, genre, subgenre, release date

## 6. Recommendation Use Case

1. The user logs into an Amazon landing page (maybe Amazon's home page, or the Prime Video landing page, Twitch, Amazon Music, Alexa app, etc.)
2. A widget appears on the landing page making recommendations to the user based on their past activity
  - a. For example, if the user is logging into Prime Video, there may be recommendations there based on a Twitch stream they watched.



3. If the user interacts with recommended media, then the data from that interaction (if it exceeds 30 minutes) will loop back into the database, and more recommendations will be made based on that data

Perhaps there should be functionality here with a directive of what to do if a user does not interact with a recommendation for a specified period of time (maybe a few days or a week)

Each set of recommendations to the user will be based on one purchase OR streaming occurrence exceeding 30 minutes. This would be another nested one-to-many relationship, where one purchase/streaming occurrence generates → many cross-platform recommendations → many recommendations.

For example, if someone were to stream 30+ minutes of Frank Sinatra on Amazon Music, that would be one streaming occurrence. Then, the app would make recommendations across many other streaming platforms, such as Prime Video and Twitch. Within each of these platforms would be many recommendations (such as a few films Sinatra appeared in, or films from that era, or games set in that era such as L.A. Noire or The Godfather: The Game).

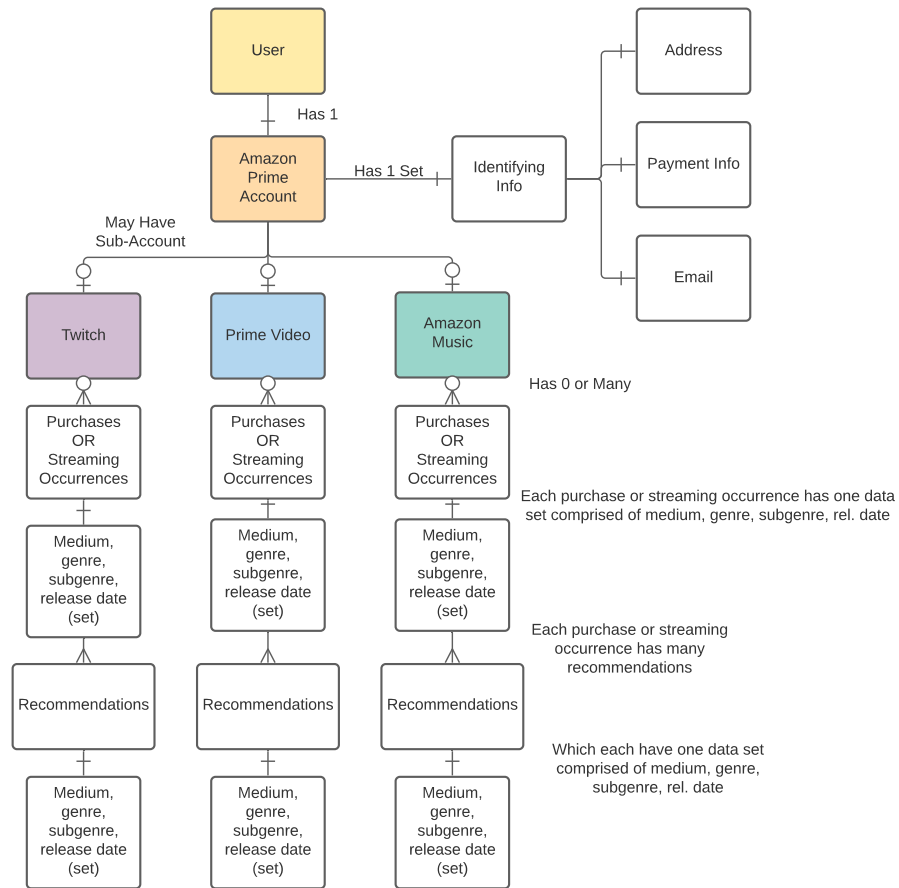
#### **Structural Database Rules from this Use Case:**

1. A purchase OR streaming occurrence exceeding 30 minutes generates many recommendations across many platforms.

#### **Entities:**

- Streaming Occurrence
  - EITHER Exceeds 30 minutes OR not
  - Subentity: Medium, genre, subgenre, release date (set)
  - Subentity: Recommendations
    - Subentity: Medium, genre, subgenre, release date (set)

## Conceptual Entity Relationship Diagram:



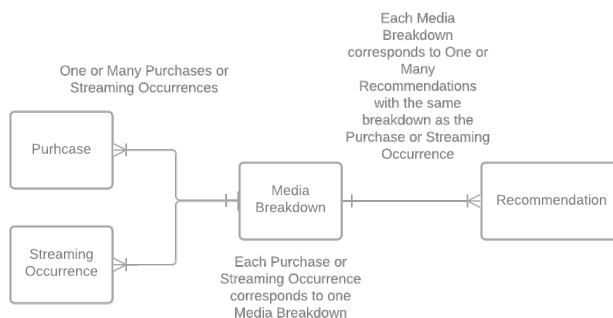
## Initial DBMS Physical ERD

The associated relationships in my conceptual ERD are:

- User/Amazon Prime Account (1:1)
- Amazon Prime Account to Sub-Accounts (Twitch, Prime Video, Amazon Music; 1:M)
- Sub-Account to Purchases OR Streaming Occurrences (1:M)
- Purchases OR Streaming Occurrences to Media Breakdown (medium, genre, subgenre, rel. date; M:1)
- Purchases OR Streaming Occurrences to Recommendations (M:N)
- Recommendations to Recommendation Media Breakdown (medium, genre, subgenre, rel. date)

I can identify a few changes that need to be made for the Initial DBMS Physical ERD:

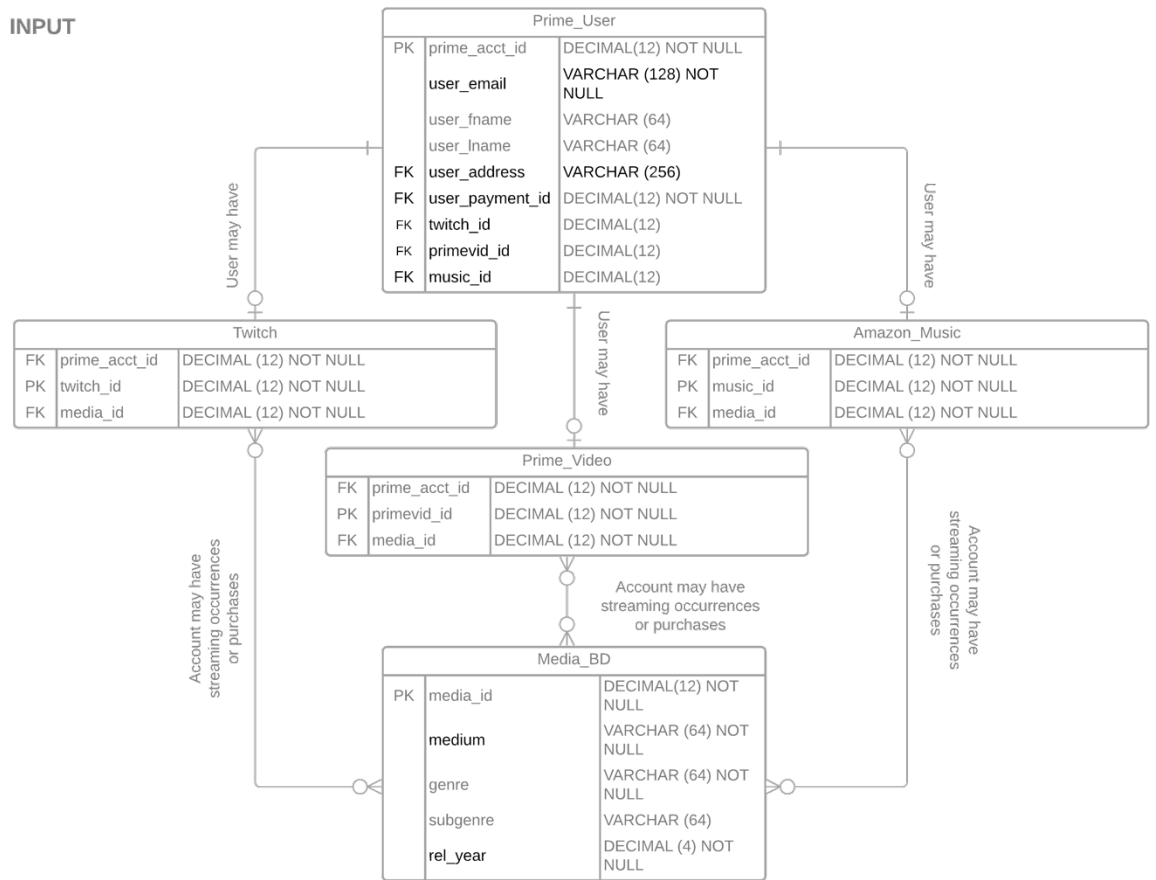
- **I believe the “User” entity and “Amazon Prime Account” entity should be combined into one “Amazon Prime User” entity.** This application is useless to anyone who does not have an Amazon Prime account, so there would be no need to have the user entity separate from the account entity. They must be one in the same for this particular application. I would use an Amazon Prime ID number as the primary key in the user table, and then have the user’s first and last name, address, payment, and email in the table. It is possible that one Amazon Prime account can be associated with many addresses and payment methods, but for the sake of simplicity, I am going to assume only one address and payment method for the user in this ERD, as adding more addresses and payment methods does not serve the functionality of the application.
- What I am calling the “Media Breakdown” (medium, genre, subgenre, release date) is a “Has” entity which connects streaming occurrences OR purchases to a Breakdown, which then connects to recommendations. See a visualization of this concept below:



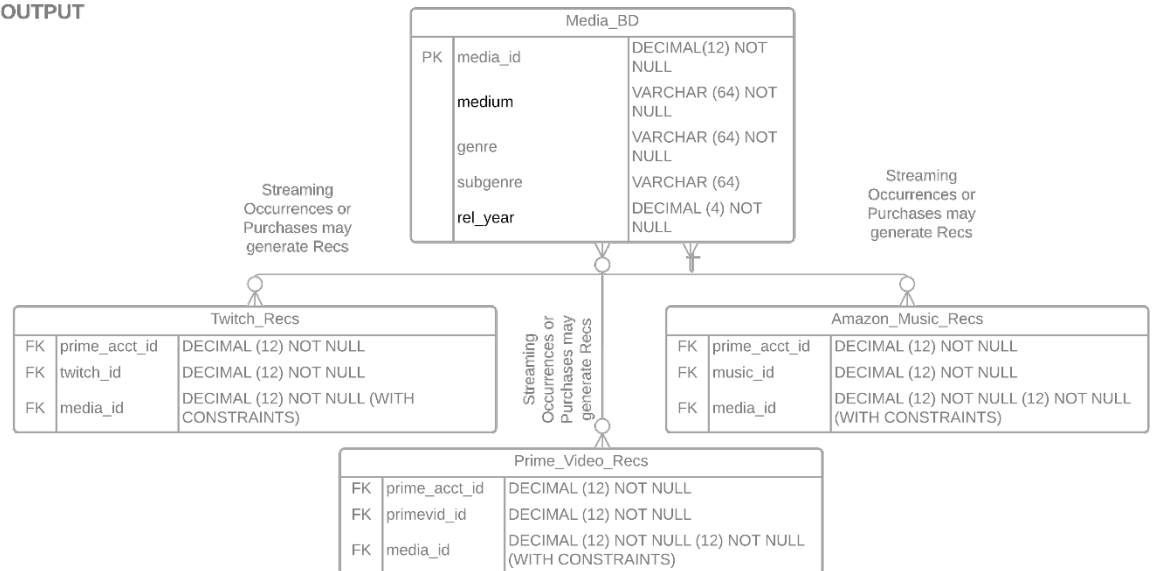
Ultimately, the relationship is between the purchase or streaming occurrence and the recommendation (which is a M:N relationship). But, the instances of streaming and purchases must pass through a “filter” with a breakdown of the media before the application can make relevant recommendations.

## Physical ERD:

### INPUT



### OUTPUT



## Physical ERD Summary

My application has two “macro” level functions: data input and data output. The “input” portion of the application pertains to the user’s streaming and purchase data on any of Amazon Prime’s entertainment platforms. The “output” refers to the cross-platform recommendations that the app will make for the user across Amazon’s media landscape.

To simplify the database, any time a user streams content for more than 30 minutes OR makes a purchase, that is recorded as one instance under “media\_id” in the table corresponding to the platform the instance occurred on (Twitch, Prime Video, or Amazon Music).

In theory, the “media\_id” would correspond to pre-existing databases at Amazon which have records of all movies, TV shows, music, video games, Twitch streamers, and so on with information on the genre, subgenre, etc. of the media. For the purposes of this project, it would be impractical to try to reference every movie, TV show, song, etc. in a database, so that information will be assumed to already exist in Amazon’s databases (and likely in reality, it does).

The ERD begins with the “User” table, which stores information about the Amazon Prime subscriber such as their full name, email, address, and any sub-account ID’s they may have (an Amazon Prime subscriber may have all or none of these sub-accounts for media streaming). For the purposes of simplicity for this project, the user\_payment\_id will reference a theoretical database which stores the user’s payment information in an encrypted table. For the scope of this project, it does not contribute to the functionality of the application to include this table.

As stated earlier, any streaming occurrences (>30min) or purchases made on any of the sub-accounts that an Amazon Prime user may have will be recorded as one instance under “media\_id” in the table corresponding to the sub-account it originated on.

Then, all streaming occurrences or purchases filter into a “Media\_BD” (Media Breakdown) table, which puts them all together and arranges them by medium and genre. Having all this data in one table will be important for the “output” step, where cross-platform recommendations can be pulled based on the user’s multiplatform media consumption habits.

Those recommendations then filter back out to the sub-accounts. For example, a music recommendation may be made on a user’s Twitch homepage. In that case, the media\_id for that song, artist, or album will appear in the Twitch\_Recs table.

The application will then pull data from the “Recs” tables for each of the applicable sub-accounts, and show the user their personalized cross-platform recommendations.

## Attributes

Table	Attribute	Datatype	Reasoning
Prime_User	prime_acct_id (PK)	DECIMAL(12) NOT NULL	The foremost identification of the user. A unique, numeric, synthetic attribute that identifies the user in the system.
Prime_User	user_email	VARCHAR(128) NOT NULL	The second-most important identification for the user is an email address. This cannot be null.
Prime_User	user_fname	VARCHAR (64)	The user's first name
Prime_User	user_lname	VARCHAR (64)	The user's last name
Prime_User	user_address (FK)	VARCHAR (256)	The user's billing/mailling address. Foreign key to be retrieved from the (theoretical for this project) payment table.
Prime_User	user_payment_id (FK)	DECIMAL(12) NOT NULL	A unique, synthetic attribute that corresponds as a foreign key to a (theoretical for this project) encrypted payment table which stores the user's payment information
Prime_User	twitch_id (FK)	DECIMAL(12)	A Prime subscriber may or may not have a Twitch account associated with their parent Prime account. If they do, that account is recorded in the Prime_User table

			with a unique, numeric, synthetic attribute identifying that user's Twitch account.
Prime_User	primevid_id (FK)	DECIMAL(12)	A Prime subscriber may or may not have a Prime Video account associated with their parent Prime account. If they do, that account is recorded in the Prime_User table with a unique, numeric, synthetic attribute identifying that user's Prime Video account.
Prime_User	music_id (FK)	DECIMAL(12)	A Prime subscriber may or may not have an Amazon Music account associated with their parent Prime account. If they do, that account is recorded in the Prime_User table with a unique, numeric, synthetic attribute identifying that user's Amazon Music account.
Twitch	prime_acct_id (FK)	DECIMAL(12) NOT NULL	Connects the sub-account to the parent user account through foreign key
Twitch	twitch_id (PK)	DECIMAL(12) NOT NULL	A unique, numerical, synthetic attribute identifying the Twitch account. This attribute is a composite key with the prime_acct_id

Twitch	media_id (FK)	DECIMAL(12) NOT NULL	A unique, numeric, synthetic attribute that populates in the table when a user streams content for more than 30 minutes OR makes a purchase. The ID number corresponds to a (theoretical for this project) master media database that contains genre, subgenre, medium, release date information for the media that was streamed or purchased.
Prime_Video	prime_acct_id (FK)	DECIMAL(12) NOT NULL	Connects the sub-account to the parent user account through foreign key
Prime_Video	primevid_id (PK)	DECIMAL(12) NOT NULL	A unique, numerical, synthetic attribute identifying the Prime Video account. This attribute is a composite key with the prime_acct_id
Prime_Video	media_id (FK)	DECIMAL(12) NOT NULL	A unique, numeric, synthetic attribute that populates in the table when a user streams content for more than 30 minutes OR makes a purchase. The ID number corresponds to a (theoretical for this project) master media database that contains genre, subgenre, medium,



			release date information for the media that was streamed or purchased.
Amazon_Music	prime_acct_id (FK)	DECIMAL(12) NOT NULL	Connects the sub-account to the parent user account through foreign key
Amazon_Music	music_id (PK)	DECIMAL(12) NOT NULL	A unique, numerical, synthetic attribute identifying the Amazon Music account. This attribute is a composite key with the prime_acct_id
Amazon_Music	media_id (FK)	DECIMAL(12) NOT NULL	A unique, numeric, synthetic attribute that populates in the table when a user streams content for more than 30 minutes OR makes a purchase. The ID number corresponds to a (theoretical for this project) master media database that contains genre, subgenre, medium, release date information for the media that was streamed or purchased.
Media_BD	media_id (PK / or FK?)	DECIMAL(12) NOT NULL	The instances of streaming/purchases across all sub-platforms are recorded in this table, where the media_id is the primary key, and the

			table is populated with information regarding the medium, genre, subgenre, and release year of the media that was consumed.
Media_BD	medium	VARCHAR(64) NOT NULL	What medium the consumed media belongs to (Movie, TV Show, Video Game, Song, Album, etc.)
Media_BD	genre	VARCHAR(64) NOT NULL	What genre the consumed media belongs to (Horror, Comedy, Action, Jazz, etc.)
Media_BD	subgenre	VARCHAR(64)	What subgenre (if applicable) the consumed media belongs to (Slasher, Romantic Comedy, Spy Film, etc.)
Media_BD	rel_year	DECIMAL (4) NOT NULL	The year in which the media was released. This helps to make relevant recommendations for other media belonging to that time period.
Twitch_Recs	prime_acct_id (FK)	DECIMAL (12) NOT NULL	Connects the sub-account to the parent user account through foreign key
Twitch_Recs	twitch_id (FK)	DECIMAL (12) NOT NULL	Identifies the Twitch account. Composite with the prime_acct_id

Twitch_Recs	media_id (FK)	DECIMAL (12) NOT NULL WITH CONSTRAINTS	This will be the recommended media for the user. This attribute will populate based on constraints to only include media available on Twitch (games) that has common genre, subgenre, and release date attributes with previously streamed or purchased media on other platforms.
Prime_Video_Recs	prime_acct_id (FK)	DECIMAL (12) NOT NULL	Connects the sub-account to the parent user account through foreign key
Prime_Video_Recs	primevid_id (FK)	DECIMAL (12) NOT NULL	Identifies the Prime Video account. Composite with the prime_acct_id
Prime_Video_Recs	media_id (FK)	DECIMAL (12) NOT NULL WITH CONSTRAINTS	This will be the recommended media for the user. This attribute will populate based on constraints to only include media available on Prime Video (movies, TV) that has common genre, subgenre, and release date attributes with previously streamed or purchased media on other platforms.
Amazon_Music_Recs	prime_acct_id (FK)	DECIMAL (12) NOT NULL	Connects the sub-account to the parent user account through foreign key

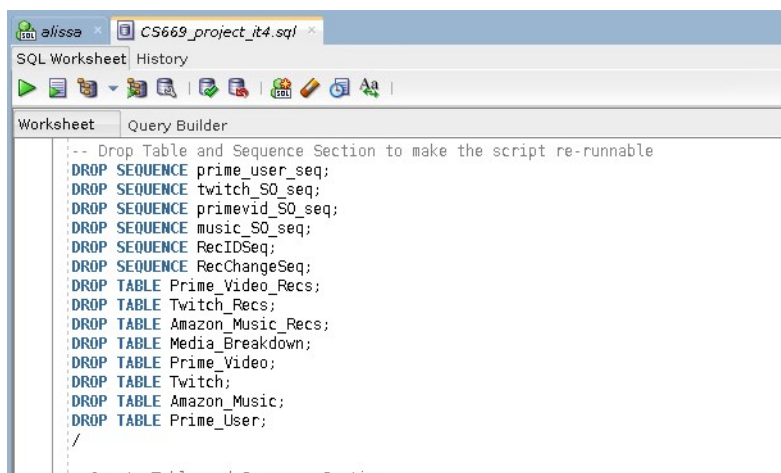
Amazon_Music_Recs	music_id (FK)	DECIMAL (12) NOT NULL	Identifies the Amazon Music account. Composite with the prime_acct_id
Amazon_Music_Recs	media_id (FK)	DECIMAL (12) NOT NULL WITH CONSTRAINTS	This will be the recommended media for the user. This attribute will populate based on constraints to only include media available on Amazon Music (music) that has common genre, subgenre, and release date attributes with previously streamed or purchased media on other platforms.

## Normalization

I can see that the attribute “media\_id” appears across 7 different tables. Theoretically, each numerical media\_id would correspond to a row of data in a massive Amazon media database spanning all Amazon’s available Movies, TV shows, Music, and Video Games that are streamed on Twitch. This table is theoretical because I cannot create such a vast database on my own in the scope of this project.

## Create Script

Below is the creation script for my tables and sequences using Oracle SQL Developer for Linux. Beginning with the DROP SEQUENCE/TABLE commands:

A screenshot of the Oracle SQL Developer interface. The title bar shows a user named 'alissa' and a file named 'CS669\_project\_it4.sql'. The main window is titled 'SQL Worksheet: History' and contains a 'Worksheet' tab. The script in the worksheet is as follows:

```
-- Drop Table and Sequence Section to make the script re-runnable
DROP SEQUENCE prime_user_seq;
DROP SEQUENCE twitch_S0_seq;
DROP SEQUENCE primevid_S0_seq;
DROP SEQUENCE music_S0_seq;
DROP SEQUENCE RecIDSeq;
DROP SEQUENCE RecChangeSeq;
DROP TABLE Prime_Video_Recs;
DROP TABLE Twitch_Recs;
DROP TABLE Amazon_Music_Recs;
DROP TABLE Media_Breakdown;
DROP TABLE Prime_Video;
DROP TABLE Twitch;
DROP TABLE Amazon_Music;
DROP TABLE Prime_User;
/
```

```

--Create Table and Sequence Section
CREATE SEQUENCE prime_user_seq START WITH 1;
CREATE SEQUENCE twitch_SO_seq START WITH 1;
CREATE SEQUENCE primevid_SO_seq START WITH 1;
CREATE SEQUENCE music_SO_seq START WITH 1;
/
CREATE TABLE Prime_User (
Prime_Acct_ID DECIMAL(12) NOT NULL PRIMARY KEY,
User_Email VARCHAR(128) NOT NULL,
User_Fname VARCHAR(64) NOT NULL,
User_LName VARCHAR(64) NOT NULL,
User_Address VARCHAR(256) NOT NULL,
User_Payment_ID DECIMAL(12) NOT NULL);

CREATE TABLE Prime_Video (
Prime_Acct_ID DECIMAL(12) NOT NULL,
FOREIGN KEY (Prime_Acct_ID) REFERENCES Prime_User(Prime_Acct_ID),
PrimeVid_SO_ID DECIMAL (12) NOT NULL PRIMARY KEY,
Media_ID DECIMAL (12) NOT NULL,
PrimeVid_SO_Date DATE NOT NULL);

CREATE TABLE Twitch (
Prime_Acct_ID DECIMAL(12) NOT NULL,
FOREIGN KEY (Prime_Acct_ID) REFERENCES Prime_User(Prime_Acct_ID),
Twitch_SO_ID DECIMAL (12) NOT NULL PRIMARY KEY,
Media_ID DECIMAL (12) NOT NULL,
Twitch_SO_Date DATE NOT NULL);

CREATE TABLE Amazon_Music (
Prime_Acct_ID DECIMAL(12) NOT NULL,
FOREIGN KEY (Prime_Acct_ID) REFERENCES Prime_User(Prime_Acct_ID),
Music_SO_ID DECIMAL (12) NOT NULL PRIMARY KEY,
Media_ID DECIMAL (12) NOT NULL,
Music_SO_Date DATE NOT NULL);

CREATE TABLE Media_Breakdown (
Media_ID DECIMAL(12) NOT NULL PRIMARY KEY,
Medium VARCHAR (64) NOT NULL,
Genre VARCHAR (64) NOT NULL,
Subgenre VARCHAR (64),
Title VARCHAR (128) NOT NULL,
Rel_Year DECIMAL(4) NOT NULL);

CREATE TABLE Recommendations (
Rec_ID DECIMAL (12) NOT NULL PRIMARY KEY,
Media_ID DECIMAL (12),
FOREIGN KEY (Media_ID) REFERENCES Media_Breakdown(Media_ID),
Title VARCHAR (128),
Date_Recommended DATE NOT NULL);

CREATE SEQUENCE RecIDSeq START WITH 1;

```

## History Table and Trigger

I ran into some errors when creating the trigger to record the history table. My intent was to create a table to record the history of recommendations made to the user, and on what date they were made. Theoretically, the app would make new recommendations to the user every week, and the history table would track which recommendations were already made, so that the same media wouldn't be recommended over and over.

See below for the creation script of the change table and trigger:

```
-- Change History Table and Trigger for Recommendation Dates
CREATE TABLE Rec_History (
  RecChange_ID DECIMAL(12) NOT NULL PRIMARY KEY,
  OldRecID DECIMAL (12) NOT NULL,
  NewRecID DECIMAL (12) NOT NULL,
  Old_Media_ID DECIMAL (12) NOT NULL,
  New_Media_ID DECIMAL (12) NOT NULL,
  ChangeDate DATE NOT NULL);

CREATE SEQUENCE RecChangeSeq START WITH 1;

--Error with this Trigger
CREATE OR REPLACE TRIGGER RecDateTrigger
BEFORE UPDATE OF Media_ID ON Recommendations
FOR EACH ROW
BEGIN
  INSERT INTO Rec_History (RecChange_ID, OldRecID, NewRecID, Old_Media_ID, New_Media_ID, ChangeDate)
  VALUES (RecChangeSeq.nextval,
    :OLD.Rec_ID,
    :NEW.Rec_ID,
    :OLD.Media_ID,
    :NEW.Media_ID,
    trunc(sysdate));
END;
/

--Updates to Recommendations to test Rec_History and History Trigger
INSERT INTO Recommendations
VALUES (RecIDSeq.nextval, 2001, 'Halloween', CAST('04-MAR-2021' AS DATE));

--Error with this command
UPDATE Recommendations
SET
  Media_ID = 2002,
  Title = 'A Nightmare on Elm Street',
  Date_Recommended = CAST('05-MAR-2021' AS DATE)
WHERE
  Rec_ID = 5;

--Indexes
```

And see below for the errors that resulted from attempting to update the Recommendations table with a “new” recommendation as a test of the trigger.

Trigger RECDATETRIGGER compiled

LINE/COL ERROR

-----  
2/5 PL/SQL: SQL Statement ignored  
3/5 PL/SQL: ORA-00947: not enough values  
Errors: check compiler log

Error starting at line : 100 in command -

UPDATE Recommendations

SET

Media\_ID = 2002,

Title = 'A Nightmare on Elm Street',

Date\_Recommended = CAST('05-MAR-2021' AS DATE)

WHERE

Rec\_ID = 5

Error at Command Line : 100 Column : 8

Error report -

SQL Error: ORA-04098: trigger 'ALISSA.RECDATETRIGGER' is invalid and failed re-validation  
04098. 00000 - "trigger '%s.%s' is invalid and failed re-validation"

\*Cause: A trigger was attempted to be retrieved for execution and was  
found to be invalid. This also means that compilation/authorization  
failed for the trigger.

\*Action: Options are to resolve the compilation/authorization errors,  
disable the trigger, or drop the trigger.

Trigger RECDATETRIGGER compiled

LINE/COL ERROR

-----  
2/5 PL/SQL: SQL Statement ignored  
3/5 PL/SQL: ORA-00947: not enough values  
Errors: check compiler log

Trigger RECDATETRIGGER compiled

Oracle SQL tells me that the trigger was compiled, but with errors. Then, when I attempt to run it, it says the trigger is invalid.



## UPDATE TO TRIGGERS (ITERATION 5)

I have made edits to the Rec\_History table and trigger commands, and the History table now works. See below:

```
-- Change History Table and Trigger for Recommendation Dates
CREATE TABLE Rec_History (
  RecChange_ID DECIMAL(12) NOT NULL PRIMARY KEY,
  OldRecID DECIMAL(12) NOT NULL,
  NewRecID DECIMAL(12) NOT NULL,
  Old_Media_ID DECIMAL(12) NOT NULL,
  New_Media_ID DECIMAL(12) NOT NULL,
  ChangeDate DATE NOT NULL);

CREATE SEQUENCE RecChangeSeq START WITH 1;

CREATE OR REPLACE TRIGGER RecDateTrigger
BEFORE UPDATE OF Media_ID ON Recommendations
FOR EACH ROW
BEGIN
  INSERT INTO Rec_History (RecChange_ID, OldRecID, NewRecID, Old_Media_ID, New_Media_ID, ChangeDate)
  VALUES (RecChangeSeq.nextval,
    :OLD.Rec_ID,
    :NEW.Rec_ID,
    :OLD.Media_ID,
    :NEW.Media_ID,
    trunc(sysdate));
END;
```

```
--Updates to Recommendations to test Rec_History and History Trigger
INSERT INTO Recommendations
VALUES (RecIDSeq.nextval, 2001, 'Halloween', CAST('04-MAR-2021' AS DATE));

UPDATE Recommendations
SET
  Media_ID = 2002,
  Title = 'A Nightmare on Elm Street',
  Date_Recommended = CAST('05-MAR-2021' AS DATE)
WHERE
  Rec_ID = 5;

INSERT INTO Recommendations
VALUES (RecIDSeq.nextval, 2009, 'The Boys', CAST('04-APR-2021' AS DATE));

UPDATE Recommendations
SET
  Media_ID = 2008,
  Title = 'Wonder Woman: 1984',
  Date_Recommended = CAST('05-APR-2021' AS DATE)
WHERE
  Rec_ID = 21;

INSERT INTO Recommendations
VALUES (RecIDSeq.nextval, 2003, 'Stranger Things', CAST('09-APR-2021' AS DATE));

UPDATE Recommendations
SET
  Media_ID = 1001,
  Title = 'Dead by Daylight',
  Date_Recommended = CAST('10-APR-2021' AS DATE)
WHERE
  Rec_ID = 41;

--Proof the trigger works for updating Rec History
SELECT *
FROM Rec_History;
```

Script Output x Query Result x

All Rows Fetched: 1 in 0.005 seconds

RECCHANGE...	OLDRECID	NEWRECID	OLD_MEDIA_ID	NEW_MEDIA_ID	CHANGEDATE
1	1	5	5	2001	2002 20- APR- 21

## Indexes

I identified the following Primary Keys which are already Indexed. The foreign keys in my script reference only Primary Keys, and thus do not require dedicated indexes.

- Prime\_User.Prime\_Acct\_ID
- Prime\_Video.PrimeVid\_SO\_ID \*note that "SO" here stands for "Streaming Occurrence"
- Twitch.Twitch\_SO\_ID
- Amazon\_Music.Music\_SO\_ID
- Media\_Breakdown.Media\_ID
- Recommendations.Rec\_ID
- Rec\_History.RecChange\_ID

There are a few other attributes that I believe an analyst would want to index separately, particularly regarding dates of streaming occurrences and recommendations. Those attributes are detailed below:

- Prime\_Video.PrimeVid\_SO\_Date
- Twitch.Twitch\_SO\_Date
- Amazon\_Music.Music\_SO\_Date
- Recommendations.Date\_Recommended

SQL Index Creation Commands below:

```
--Indexes
CREATE INDEX PrimeVidSODate
ON Prime_Video(PrimeVid_SO_Date);

CREATE INDEX TwitchSODate
ON Twitch(Twitch_SO_Date);

CREATE INDEX MusicSODate
ON Amazon_Music(Music_SO_Date);

CREATE INDEX DateRecIdx
ON Recommendations(Date_Recommended);
```

Index PRIMEVIDSODATE created.

Index TWITCHSODATE created.

Index MUSICSODATE created.

Index DATERECIDX created.

## Stored Procedure/Transaction

I added a "Add Prime User" transaction into my database structure. I was unable to get it to run in the second part, however, so I did end up adding my users in manually.

```
CREATE OR REPLACE PROCEDURE AddPrimeUser (Prime_Acct_ID IN DECIMAL, User_Email IN VARCHAR, User_FName IN VARCHAR,
User_LName IN VARCHAR, User_Address IN VARCHAR, User_Payment_ID IN DECIMAL)
AS
BEGIN
    INSERT INTO Prime_User(Prime_Acct_ID, User_Email, User_FName, User_LName, User_Address, User_Payment_ID)
    VALUES(Prime_Acct_ID, User_Email, User_FName, User_LName, User_Address, User_Payment_ID);

    INSERT INTO Prime_Video(Prime_Acct_ID)
    VALUES(Prime_Acct_ID);

    INSERT INTO Twitch(Prime_Acct_ID)
    VALUES(Prime_Acct_ID);

    INSERT INTO Amazon_Music(Prime_Acct_ID)
    VALUES(Prime_Acct_ID);
END;
/

BEGIN
    AddPrimeUser(2, 'blight@dbd.com', 'Blight', 'Monster', '123 Daylight Way', 2);
    COMMIT;
END;
/
```

Error message:

```
Error starting at line : 170 in command -
BEGIN
    AddPrimeUser(2, 'blight@dbd.com', 'Blight', 'Monster', '123 Daylight Way', 2);
    COMMIT;
END;
Error report -
ORA-01400: cannot insert NULL into ("ALISSA"."PRIME_VIDEO"."PRIMEVID_SO_ID")
ORA-06512: at "ALISSA.ADDPRIMEUSER", line 8
ORA-06512: at line 2
01400. 00000 - "cannot insert NULL into (%s)"
*Cause:      An attempt was made to insert NULL into previously listed objects.
*Action:     These objects cannot accept NULL values.
```

## Queries:

### First Query:

This query retrieves data from the Rec\_History table, ordered by date:

```
--Queries
--First Query
SELECT New_Media_ID, ChangeDate FROM Rec_History
ORDER BY ChangeDate ASC;
```

NEW_MEDIA_ID	CHANGEDAT
2002	20-APR-21

### Second Query:

This set of queries connects the Prime\_User table with the sub-accounts (Prime Video, Twitch, and Amazon Music) to show which users have interactions in which sub-account.

```
--Second Query (for all sub-accounts)
SELECT Prime_User.Prime_Acct_ID, Prime_User.User_Fname, Prime_User.User_LName
FROM Prime_User
RIGHT JOIN Prime_Video ON Prime_User.Prime_Acct_ID=Prime_Video.Prime_Acct_ID;

SELECT Prime_User.Prime_Acct_ID, Prime_User.User_Fname, Prime_User.User_LName
FROM Prime_User
RIGHT JOIN Twitch ON Prime_User.Prime_Acct_ID=Twitch.Prime_Acct_ID;

SELECT Prime_User.Prime_Acct_ID, Prime_User.User_Fname, Prime_User.User_LName
FROM Prime_User
RIGHT JOIN Amazon_Music ON Prime_User.Prime_Acct_ID=Amazon_Music.Prime_Acct_ID;
```

PRIME_ACCT_ID	USER_FNAME	USER_LNAME
1	Michael	Myers
6	Nurse	Blink
7	Trickster	Singer
9	Bella	Spirit
10	Ghost	Face

### Third Query:

This set of queries show streaming occurrences on Prime Video, Twitch, and Amazon Music and displays the Media Breakdown for each.

```
--Third Query
SELECT *
FROM Prime_Video
LEFT JOIN Media_Breakdown ON Prime_Video.Media_ID=Media_Breakdown.Media_ID;

SELECT *
FROM Twitch
LEFT JOIN Media_Breakdown ON Twitch.Media_ID=Media_Breakdown.Media_ID;

SELECT *
FROM Amazon_Music
LEFT JOIN Media_Breakdown ON Amazon_Music.Media_ID=Media_Breakdown.Media_ID;

Select *
FROM Rec_History
JOIN Media_Breakdown ON Rec_History.New_Media_ID=Media_Breakdown.Media_ID;
```

PRIME_ACCT_ID	PRIMEVID_SQ_ID	MEDIA_ID	PRIMEVID_	MEDIA_ID	MEDIUM	GENRE
6	100004	2001	30-APR-21	2001	Feature Film	Horror
7	100003	2002	08-APR-21	2002	Feature Film	Horror
1	100001	2003	12-APR-21	2003	TV Series	Horror
10	100005	2003	10-APR-21	2003	TV Series	Horror
9	100002	2008	04-APR-21	2008	Feature Film	Action
PRIME_ACCT_ID	TWITCH_SQ_ID	MEDIA_ID	TWITCH_SQ	MEDIA_ID	MEDIUM	GENRE
4	200001	1001	01-APR-21	1001	Video Game	Horror
5	200002	2007	21-APR-21	2007	Video Game	Horror
PRIME_ACCT_ID	MUSIC_SQ_ID	MEDIA_ID	MUSIC_SQ_	MEDIA_ID	MEDIUM	GENRE
2	300001	2004	24-APR-21	2004	Song	Pop
3	300002	2005	26-APR-21	2005	Song	Rock
8	300003	2010	11-APR-21	2010	Song	Pop

#### Fourth Query:

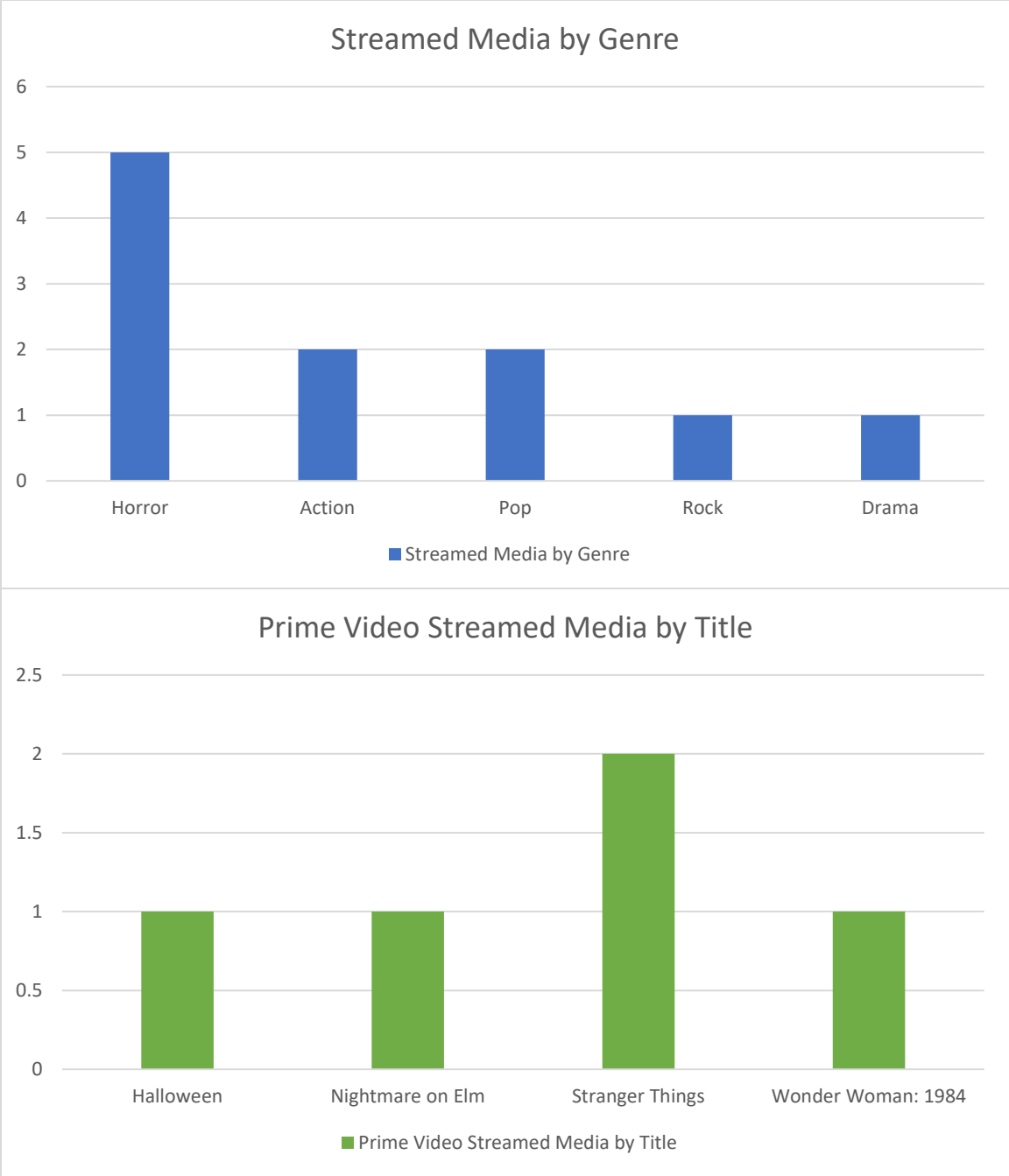
In group 1 of this query, I show all media belonging to the “Horror” Genre, across platforms. In group 2, I show which media has more than one streaming occurrence (in this case, it is Media ID 2003, which is the TV Show *Stranger Things*)

--Fourth Query		
--Group 1		
SELECT *		
FROM Media_Breakdown		
WHERE Genre = 'Horror';		
--Group 2		
SELECT Prime_Video.Media_ID		
FROM Prime_Video		
LEFT JOIN Media_Breakdown ON Prime_Video.Media_ID=Media_Breakdown.Media_ID		
GROUP BY Prime_Video.Media_ID		
HAVING COUNT(Prime_Video.Media_ID) > 1;		
MEDIA_ID	MEDIUM	GENRE
1001	Video Game	Horror
2001	Feature Film	Horror
2002	Feature Film	Horror
2003	TV Series	Horror
2007	Video Game	Horror
MEDIA_ID		
2003		

## Data Visualization

I chose to display how many streamed units of media belong to each genre in the first query, and show how many times media on Prime Video had been streamed in the second.

<pre>--Data Visualization SELECT Genre, COUNT(Media_ID) FROM Media_Breakdown GROUP BY Genre ORDER BY Count(Media_ID) DESC; /  SELECT Media_Breakdown.Media_ID, Media_Breakdown.Title, Prime_Video.PrimeVid_SO_ID FROM Prime_Video INNER JOIN Media_Breakdown ON Media_Breakdown.Media_ID = Prime_Video.Media_ID ORDER BY Media_ID;</pre>	
GENRE	COUNT(MEDIA_ID)
-----	-----
Horror	5
Action	2
Pop	2
Drama	1
Rock	1
MEDIA_ID	TITLE
-----	-----
2001	Halloween
2002	A Nightmare on Elm Street
2003	Stranger Things
2003	Stranger Things
2008	Wonder Woman: 1984





## Summary and Reflection

I developed a database for an application to be used primarily for Amazon's line of entertainment platforms that will make cross-platform recommendations to the end user. I believe a tool like this will be useful as most streaming services only provide same-platform recommendations, when a cross-platform recommendation may also be relevant to the user, and can keep them interacting with Amazon's products even if they are consuming entertainment belonging to a different medium. Currently, the system that I personally use for cross-platform recommendations is word-of-mouth from friends, so I think it would be nice if I could have these recommendations tailored to me and automated with a database and an application.

Any version of this application would require access to several massive media databases containing information for all movies, TV series, games, music, etc. that a platform may carry. Clearly, I will not be able to focus on that aspect in my database project for this class. Instead, I will be working with what I assume will have to be hypothetical data mapped to hypothetical recommendations. The execution of this project for the scope of this class is what I will have some questions on.

That said, the rapidly-changing entertainment landscape of today is fascinating to me, and as it is work I wish to carry on with post-grad, I am excited to work on this project and hopefully use it as a case example of my work.

--

Now that I have completed a very nascent version of this database, I have found that in some ways, having a database of all users and their streaming activities makes the function of the application easier, but also far more complex, as a real-world version of the database would require access to a master database of all media offered on Amazon's platforms (which is a LOT of media and therefore, a lot of data!)

I still believe that an application like this would be beneficial to Amazon's retention rate on their platforms, keeping users engaged in the Amazon streaming ecosystem and making recommendations to them beyond just similar movies to a movie they've already seen. As mediums become more blended, having a cross-medium selection of media to consume based on one's interests would be beneficial to both the company and the end users.