University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: 10/20/2023

Group Number: 19

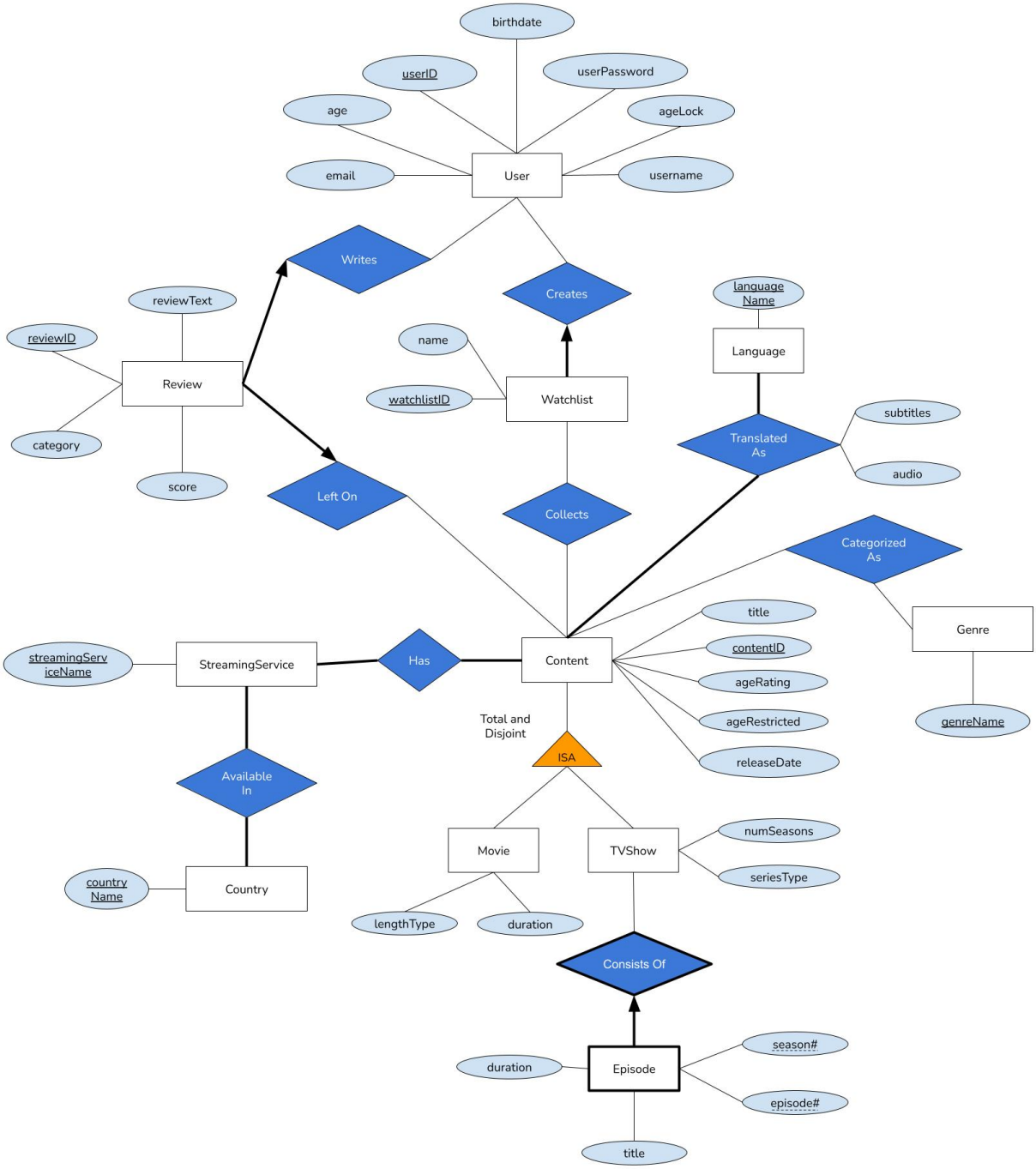
Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Ayush Saldhi	70059464	l7w9w	ayushsaldhi@hotmail.com
Brian Chu	27539022	o2m1v	brianchu3141@gmail.com
Tristan Buckoll	86167715	d1l2q	tman8667@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

Summary

Our application will allow users to track movies and TV shows that are available across different streaming services, and create lists of such content. Data about genre categorization and translated languages of the content will be stored, and users can leave reviews on content.



Notes on ER Diagram:

- Country, Genre, Language, and StreamingService only have one attribute because they only convey meaning with regards to the entity they're in a relationship with and they need to be entity sets because multiple of them can be associated with one of the corresponding entities (i.e. Streaming Service available in multiple countries, Content categorized as multiple genres, etc.)

- UserID is the primary key of User since users may be able to change their username and email, so they will have a fixed ID that will always be used to identify them on the database side

- We have updated the all the id and name attributes from Milestone 1 to more easily distinguish between attributes of different entity sets

- We renamed the text attribute of Review to reviewText because that is a SQL data type

- We renamed the password attribute of User to userPassword because that is a SQL data type

- We have added some attributes to facilitate a more interesting normalization

- Age lock is true if age is <= 13

Relational Schema

Legend:

Primary Key, **Foreign Key**, *Candidate Key* - implies UNIQUE, Not Null

Entities

1. Review(reviewID: int, reviewText: varchar, score: int CHECK (score >=1 AND score <= 10), category: varchar)
2. User(userID: int, *email*: varchar, userPassword: varchar, *username*: char[16], birthdate: date, age: int, ageLock: bool)
3. Content(contentID: int, *title*: varchar, *releaseDate*: date, ageRating: char[10], ageRestricted: bool)
4. Movie(**contentID (ref. Content)**: int, duration: int, lengthType: varchar)
5. TVShow(**contentID (ref. Content)**: int, numSeasons: int, seriesType: varchar)
6. Watchlist(watchlistID: int, name: varchar)
7. Language(languageName: varchar)
8. Genre(genreName: varchar)
9. Country(countryName: varchar)
10. StreamingService(streamingServiceName: varchar)
11. Episode(**contentID (ref. TVShow)**: int, season#: int, episode#: int, duration: int, title: varchar)

Relationships

1. TranslatedAs(**languageName (ref. Language)**: varchar, **contentID (ref. Content)**: int, audio: bool, subtitles: bool)
2. Writes(**userID (ref. User):** int, **reviewID (ref. Review):** int)
3. Creates(**watchlistID (ref. Watchlist):** int, **userID (ref. User):** int)
4. Collects(**watchlistID (ref. Watchlist):** int, **contentID (ref. Content):** int)
5. CategorizedAs(**genreName (ref. Genre)**: varchar, **contentID (ref. Content)**: int)
6. Has(**streamingServiceName (ref StreamingService)**: varchar, **contentID (ref. Content)**: int)
7. AvailableIn(**countryName (ref. Country)**: varchar, **streamingServiceName (ref. StreamingService)**: varchar)
8. LeftOn(**contentID (ref. Content)**: int, **reviewID (ref. Review)**: int)

Note: All varchar variables are limited to a maximum size of 255.

Functional Dependencies

Entities

1. Review(reviewID, reviewText, score, category)
    - reviewID -> reviewText, score, category
    - score -> category

2. User(userID, *email*, userPassword, *username*, birthdate, age, ageLock)
    - userID -> email, userPassword, username, birthdate, age, ageLock
    - email -> userID, username, userPassword, birthdate, age, ageLock
    - username -> userID, email, userPassword, birthdate, age, ageLock
    - birthdate -> age
    - age -> ageLock

3. Content(contentID, *title*, *releaseDate*, ageRating, ageRestricted)
    - contentID-> title, ageRating, releaseDate, ageRestricted
    - title, releaseDate -> contentID, ageRating, ageRestricted
    - ageRating -> ageRestricted

4. Movie(**contentID (ref. Content)**, duration, lengthType)
    - contentID -> duration, lengthType
    - duration -> lengthType

5. TVShow(**contentID (ref. Content)**, numSeasons, seriesType)
    - contentID -> numSeasons, seriesType
    - numSeasons -> seriesType

6. Watchlist(watchlistID, name)
    - watchlistID -> name
    -

7. Language(languageName)

8. Genre(genreName)

9. Country(genreName)

10. StreamingService(streamingServiceName)

11. Episode(**contentID**, season#, episode#, duration, title)
    - contentID, season#, episode# -> duration, title

Relationships

TranslatedAs(**languageName (ref. Language)**, **contentID (ref. Content)**, audio, subtitles)

- languageName, contentID -> subtitles, audio

All other FD's with relationship relations are trivial.

Normalization (BCNF)

1. Review(<u>reviewID</u>, reviewText, score, category)
    - reviewID -> reviewText, score, category
    - score -> category

    reviewID$^+$ = {reviewID, reviewText, score, category}

    score$^+$ = {score, category}

    Notes

    - reviewID is a superkey for the relation since it identifies all attributes.
    - score is not a superkey, therefore the FD score -> category violates BCNF

    Decomposition:

    reviewID, reviewText - score - category

    Review_1(<u>score</u>, category)
    - two attribute relation, thus in BCNF

    Review_2(<u>reviewID</u>, **<u>score (ref. Review_1)</u>**, reviewText)
    - no FD's are applicable, thus in BCNF

2. User(<u>userID</u>, *<u>email</u>*, userPassword, *<u>username</u>*, birthdate, age, ageLock)
    - userID -> email, userPassword, username, birthdate, age, ageLock
    - email -> userID, username, userPassword, birthdate, age, ageLock
    - username -> userID, email, userPassword, birthdate, age, ageLock
    - birthdate -> age
    - age -> ageLock

    $userID^+$ = {userID, email, userPassword, username, birthdate, age, ageLock}

    $email^+$ = {email, userID, username, userPassword, birthdate, age, ageLock}

    $username^+$ = {username, userID, email, userPassword, birthdate, age, ageLock}

    $birthdate^+$= {birthdate, age, ageLock}

    age^+ = {age, ageLock}

    Notes:

    - userID, email, and username are all superkeys for the relation since they uniquely identify all attributes. birthdate and age are not superkeys, hence both violate BCNF

    Decomposition:

    userID, email, userPassword, username, birthdate - age - ageLock

    User_1(<u>age</u>, ageLock)
    - automatically in BCNF

    User(userID, email, userPassword, username, birthdate, age)
    - birthdate is not a super key, needs further decomposition

    userID, email, userPassword, username - birthdate - age

    User_2(<u>userID</u>, **<u>birthdate (ref. User_3)</u>**, *<u>email</u>*, userPassword, *<u>username</u>*)
    - no FD's are applicable, thus in BCNF

    User_3(<u>birthdate</u>, **<u>age (ref. User_1)</u>**)
    - automatically in BCNF

3. Content(contentID, *title, releaseDate*, ageRating, ageRestricted)
    - contentID-> title, ageRating, releaseDate, ageRestricted
    - title, releaseDate -> contentID, ageRating, ageRestricted
    - ageRating -> ageRestricted

contentID$^+$ = {contentID, title, ageRating, releaseDate, ageRating, ageRestricted}

title, releaseDate$^+$ = {title, releaseDate, contentID, ageRating, ageRestricted}

ageRating$^+$ = {ageRating, ageRestricted}

Notes:

- contentID and {title, releaseDate} are superkeys.
- ageRating is not a superkey, and so it violates BCNF.

Decomposition:

ageRating - ageRestricted

Content_1(ageRating, ageRestricted)

- automatically in BCNF

Content_2(contentID, **ageRating (ref. Content_1)**, *title, releaseDate*)

- no FD's are applicable, thus in BCNF

4. Movie(**contentID (ref. Content)**, duration, lengthType)
    - contentID -> duration, lengthType
    - duration -> lengthType

    contentID$^+$ = {contentID, duration, lengthType}

    duration$^+$ = {duration, lengthType}

    Notes:
    - contentID is a superkey, therefore it does not violate BCNF
    - duration is not a superkey, therefore it does violate BCNF

    Decomposition:

    contentID - duration - lengthType

    Movie_1(duration, lengthType)

    Movie_2(**contentID (ref. Content_2)**, **duration (ref. Movie_1)**)

    - Both relations are in BCNF.

5. TVShow(**contentID (ref. Content)**, numSeasons, seriesType)
    - contentID -> numSeasons, seriesType
    - numSeasons -> seriesType

    contentID$^+$ = {contentID, numSeasons, seriesType)

    numSeasons$^+$ = {numSeasons, seriesType}

    Notes:
    - contentID is a superkey, which does not violate BCNF
    - numSeasons is not a superkey, which does violate BCNF

    Decomposition:

    contentID - numSeasons - seriesType

    TVShow_1(**contentID (ref. Content_2)**, **numSeasons (ref. TVShow_2)**)

    TVShow_2(numSeasons, seriesType)

    - Both relations are in BCNF.

Tables Post Normalization

Legend:

Primary Key, **Foreign Key**, *Candidate Key* - implies UNIQUE, Not Null

Tables:

Entities

1. Review
    1. Review_1(score: int (score >=1 AND score <= 10), category: varchar(100))
        - score -> category
    2. Review_2(reviewID: int, **score (ref. Review_1)**: int (score >=1 AND score <= 10), reviewText: varchar(100))
        - reviewID -> score, reviewText
2. User
    1. User_1(age: int, ageLock: bool)
        - age -> ageLock
    2. User_2(userID: int, **birthdate (ref. User_3)**: date, *email*: varchar, userPassword: varchar, *username*: char[16])
        - userID -> birthdate, email, userPassword, username
        - email -> userID, birthdate, userPassword, username
        - username -> userID, birthdate, email, userPassword
    3. User_3(birthdate: date, **age (ref. User_1)**: int)
        - birthdate -> age
3. Content
    1. Content_1(ageRating: char[10], ageRestricted: bool)
        - ageRating -> ageRestricted
    2. Content_2(contentID: int, **ageRating (ref. Content_1)**: char[10], *title*: varchar, *releaseDate*: date)
        - contentID -> ageRating, title, releaseDate

- title, releaseDate -> contentID, ageRating

4. Movie
    1. Movie_1(duration: int, lengthType: varchar)
        - duration -> lengthType

    2. Movie_2(**contentID (ref. Content_2)**: int, **duration (ref. Movie_1):** int)
        - contentID -> duration

5. TVShow
    1. TVShow_1(**contentID (ref. Content_2)**: int, **numSeasons (ref. TVShow_2)**: int)
        - contentID -> numSeasons

    2. TVShow_2(numSeasons: int, seriesType: varchar)
        - numSeasons -> seriesType

6. Watchlist(watchlistID: int, name: varchar)
    - watchlistID -> Name

7. Language(languageName: varchar)

8. Genre(genreName: varchar)

9. Country(countryName: varchar)

10. StreamingService(streamingServiceName: varchar)

11. Episode(**contentID (ref. TVShow)**: int, season#: int, episode#: int, duration: int, title: varchar)
    - contentID, season, episode# -> duration, title

Relationships

1. TranslatedAs(**languageName (ref. Language)**: varchar, **contentID (ref. Content)**: int, audio: bool, subtitles: bool)
    - languageName, contentID -> audio, subtitles

2. Writes(**userID (ref. User)**: int**,** **reviewID (ref. Review)**: int)

3. Creates(**watchlistID (ref. Watchlist)**: int, **userID (ref. User)**: int)

4. Collects(**watchlistID (ref. Watchlist)**: int**,** **contentID (ref. Content)**: int)

5. CategorizedAs(**genreName (ref. Genre)**: **varchar**, **contentID (ref. Content)**: int)

6. Has(**streamingServiceName (ref StreamingService)**: varchar, **contentID (ref. Content)**: int)

7. AvailableIn(**countryName (ref. Country)**: varchar, **streamingServiceName (ref. StreamingService)**: varchar)

8. LeftOn(**contentID (ref. Content)**: int, **reviewID (ref. Review)**: int)

SQL DDL Statements

Entities

```
CREATE TABLE Review_1 (
  score INTEGER PRIMARY KEY,
  category VARCHAR(255)
);

CREATE TABLE Review_2 (
  reviewID INTEGER PRIMARY KEY,
  score INTEGER,
  reviewText VARCHAR(255),
  FOREIGN KEY (score) REFERENCES Review_1(score)
);

CREATE TABLE User_1 (
  age INTEGER PRIMARY KEY,
  ageLock BOOLEAN
);

CREATE TABLE User_3 (
  birthDate DATE PRIMARY KEY,
  age INTEGER,
  FOREIGN KEY (age) REFERENCES User_1(age)
);

CREATE TABLE User_2 (
  userID INTEGER PRIMARY KEY,
  birthDate DATE,
  email VARCHAR(255) UNIQUE,
  userPassword VARCHAR(255),
  username VARCHAR(255) UNIQUE,
  FOREIGN KEY (birthDate) REFERENCES User_3(birthDate)
);

CREATE TABLE Content_1 (
  ageRating CHAR(10) PRIMARY KEY,
  ageRestricted BOOLEAN
);

CREATE TABLE Content_2 (
  contentID INTEGER PRIMARY KEY,
  ageRating CHAR(10),
  title VARCHAR(255) NOT NULL,
  releaseDate DATE NOT NULL,
  UNIQUE (title, releaseDate),
  FOREIGN KEY (ageRating) REFERENCES Content_1(ageRating)
);
```

```sql
CREATE TABLE Movie_1 (
  duration INTEGER PRIMARY KEY,
  lengthType VARCHAR(255)
);

CREATE TABLE Movie_2 (
  contentID INTEGER PRIMARY KEY,
  duration INTEGER,
  FOREIGN KEY (duration) REFERENCES Movie_1(duration),
  FOREIGN KEY (contentID) REFERENCES Content_2(contentID)
);

CREATE TABLE TVShow_2 (
  numSeasons INTEGER PRIMARY KEY,
  seriesType VARCHAR(255)
);

CREATE TABLE TVShow_1 (
  contentID INTEGER PRIMARY KEY,
  numSeasons INTEGER,
  FOREIGN KEY (numSeasons) REFERENCES TVShow_2(numSeasons),
  FOREIGN KEY (contentID) REFERENCES Content_2(contentID)
);

CREATE TABLE Watchlist (
  watchlistID INTEGER PRIMARY KEY,
  name VARCHAR(255) NOT NULL
);

CREATE TABLE Language (
  languageName VARCHAR(255) PRIMARY KEY
);

CREATE TABLE Genre (
  genreName VARCHAR(255) PRIMARY KEY
);

CREATE TABLE Country (
  countryName VARCHAR(255) PRIMARY KEY
);

CREATE TABLE StreamingService (
  streamingServiceName VARCHAR(255) PRIMARY KEY
);

CREATE TABLE Episode (
  contentID INTEGER,
  season INTEGER,
```

```
  episode INTEGER,
  duration INTEGER,
  title VARCHAR(255),
  PRIMARY KEY (contentID, season, episode),
  FOREIGN KEY (contentID) REFERENCES TVShow_1(contentID)
);
```

Relationships

```
CREATE TABLE TranslatedAs (
  languageName VARCHAR(255),
  contentID INTEGER,
  audio BOOLEAN,
  subtitles BOOLEAN,
  PRIMARY KEY (languageName, contentID),
  FOREIGN KEY (languageName) REFERENCES Language(languageName),
  FOREIGN KEY (contentID) REFERENCES Content_2(contentID)
);

CREATE TABLE Writes (
  userID INTEGER,
  reviewID INTEGER PRIMARY KEY,
  FOREIGN KEY (userID) REFERENCES User_2(userID),
  FOREIGN KEY (reviewID) REFERENCES Review_2(reviewID)
);

CREATE TABLE Creates (
  watchlistID INTEGER,
  userID INTEGER,
  PRIMARY KEY (watchlistID, userID),
  FOREIGN KEY (watchlistID) REFERENCES Watchlist(watchlistID),
  FOREIGN KEY (userID) REFERENCES User_2(userID)
);

CREATE TABLE Collects (
  watchlistID INTEGER,
  contentID INTEGER,
  PRIMARY KEY (watchlistID, contentID),
  FOREIGN KEY (watchlistID) REFERENCES Watchlist(watchlistID),
  FOREIGN KEY (contentID) REFERENCES Content_2(contentID)
);

CREATE TABLE CategorizedAs (
  genreName VARCHAR(255),
  contentID INTEGER,
  PRIMARY KEY (genreName, contentID),
  FOREIGN KEY (genreName) REFERENCES Genre(genreName),
  FOREIGN KEY (contentID) REFERENCES Content_2(contentID)
```

```
);

CREATE TABLE Has (
  streamingServiceName VARCHAR(255),
  contentID INTEGER,
  PRIMARY KEY (streamingServiceName, contentID),
  FOREIGN KEY (streamingServiceName) REFERENCES StreamingService(streamingServiceName),
  FOREIGN KEY (contentID) REFERENCES Content_2(contentID)
);

CREATE TABLE AvailableIn (
  countryName VARCHAR(255),
  streamingServiceName VARCHAR(255),
  PRIMARY KEY (countryName, streamingServiceName),
  FOREIGN KEY (countryName) REFERENCES Country(countryName),
  FOREIGN KEY (streamingServiceName) REFERENCES StreamingService(streamingServiceName)
);

CREATE TABLE LeftOn (
  contentID INTEGER,
  reviewID INTEGER PRIMARY KEY,
  FOREIGN KEY (contentID) REFERENCES Content_2(contentID),
  FOREIGN KEY (reviewID) REFERENCES Review_2(reviewID)
);
```

INSERT Statements

Entities

```
INSERT INTO Review_1(score, category) VALUES
(1, "Terrible"),
(2, "Poor"),
(3, "Bad"),
(4, "Average"),
(5, "Good");

INSERT INTO Review_2(reviewID, score, reviewText) VALUES
(1, 1, "Did not like this at all."),
(2, 2, "Could be better."),
(3, 3, "It was bad but had some redeeming qualities."),
(4, 4, "Not great, but not bad either"),
(5, 5, "Enjoyed it!");

INSERT INTO User_1(age, ageLock) VALUES
(13, TRUE),
(30, FALSE),
(10, TRUE),
(40, FALSE),
(25, FALSE);

INSERT INTO User_3(birthdate, age) VALUES
('1998-01-01', 25),
('1993-02-10', 30),
('1988-03-20', 35),
('1983-04-15', 40),
('1978-05-25', 45);

INSERT INTO User_2(userID, birthdate, email, userPassword, username) VALUES
(1, '1998-01-01', 'user1@example.com', 'password1', 'user1'),
(2, '1993-02-10', 'user2@example.com', 'password2', 'user2'),
(3, '1988-03-20', 'user3@example.com', 'password3', 'user3'),
(4, '1983-04-15', 'user4@example.com', 'password4', 'user4'),
(5, '1978-05-25', 'user5@example.com', 'password5', 'user5');

INSERT INTO Content_1(ageRating, ageRestricted) VALUES
('PG', FALSE),
('PG-13', TRUE),
('R', TRUE),
('G', FALSE),
('NR', FALSE);

INSERT INTO Content_2(contentID, ageRating, title, releaseDate) VALUES
```

```
(1, 'PG', 'Movie 1', '2022-01-01'),
(2, 'PG-13', 'Movie 2', '2022-02-01'),
(3, 'R', 'Movie 3', '2022-03-01'),
(4, 'G', 'Movie 4', '2022-04-01'),
(5, 'NR', 'Movie 5', '2022-05-01');

INSERT INTO Movie_1(duration, lengthType) VALUES
(120, 'Medium'),
(90, 'Short'),
(150, 'Long'),
(110, 'Medium'),
(95, 'Short');

INSERT INTO Movie_2(contentID, duration) VALUES
(1, 120),
(2, 90),
(3, 150),
(4, 110),
(5, 95);

INSERT INTO TVShow_2(numSeasons, seriesType) VALUES
(1, 'Mini-Series'),
(3, 'Longform'),
(5, 'Longform'),
(2, 'Longform'),
(4, 'Longform');

INSERT INTO TVShow_1(contentID, numSeasons) VALUES
(6, 1),
(7, 3),
(8, 5),
(9, 2),
(10, 4);

INSERT INTO Watchlist(watchlistID, name) VALUES
(1, 'Summer Binge List'),
(2, 'Drama Favorites'),
(3, 'Weekend Chill'),
(4, 'Documentary Picks'),
(5, 'Family Night');

INSERT INTO Language(languageName) VALUES
('English'),
('Spanish'),
('French'),
('German'),
('Chinese');

INSERT INTO Genre(genreName) VALUES
('Drama'),
```

```
('Comedy'),
('Action'),
('Thriller'),
('Romance');

INSERT INTO Country(countryName) VALUES
('USA'),
('UK'),
('Canada'),
('Australia'),
('India');

INSERT INTO StreamingService(streamingServiceName) VALUES
('Netflix'),
('Amazon Prime'),
('Disney+'),
('Hulu'),
('HBO Max');

INSERT INTO Episode(contentID, season, episode, duration, title) VALUES
(6, 1, 1, 40, 'Pilot Episode'),
(7, 1, 2, 45, 'The Reunion'),
(8, 1, 3, 42, 'Twist and Turns'),
(9, 2, 1, 50, 'A New Beginning'),
(10, 2, 2, 48, 'Unexpected Events');
```

Relationships

```
INSERT INTO TranslatedAs(languageName, contentID, audio, subtitles) VALUES
('English', 1, TRUE, TRUE),
('Spanish', 1, FALSE, TRUE),
('French', 2, FALSE, TRUE),
('German', 3, TRUE, FALSE),
('Chinese', 4, FALSE, TRUE);

INSERT INTO Writes(userID, reviewID) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5);

INSERT INTO Creates(watchlistID, userID) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5);
```

```sql
INSERT INTO Collects(watchlistID, contentID) VALUES
(1, 1),
(1, 2),
(2, 3),
(3, 4),
(4, 5);

INSERT INTO CategorizedAs(genreName, contentID) VALUES
('Drama', 1),
('Comedy', 2),
('Action', 3),
('Thriller', 4),
('Romance', 5);

INSERT INTO Has(streamingServiceName, contentID) VALUES
('Netflix', 1),
('Amazon Prime', 2),
('Disney+', 3),
('Hulu', 4),
('HBO Max', 5);

INSERT INTO AvailableIn(countryName, streamingServiceName) VALUES
('USA', 'Netflix'),
('UK', 'Amazon Prime'),
('Canada', 'Disney+'),
('Australia', 'Hulu'),
('India', 'HBO Max');

INSERT INTO LeftOn(contentID, reviewID) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5);
```