1.

University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: July 21, 2024

Group Number: 22

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Suhayl Patel	74030826	z8q5f	suhayl.patel@outlook.com
Keean Vidyarthi	80912504	k9x4u	keeanvid1@gmail.com
Weena Wibowo	74581323	r8q5n	weenawibowo@hotmail.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

2. Our project's database represents the structure of a streaming service, from the perspective of a streaming service provider (ex. Netflix). We've decided to model entities like Users, Subscriptions, Media, and additional aspects that would be beneficial for a streaming service provider to manage.

3. As mentioned in the Milestone 1 feedback we received, there were some issues in our ER diagram that needed refinement, specifically regarding the cardinalities and ternary relationship structure between entities. We've changed our E-R diagram to address each of the feedback comments provided, along with additional changes to simplify our database, while maintaining all necessary elements for the project. The key changes are outlined below:

- Our ISA hierarchy, separating Media into Movies and TV shows was missing constraints but has now been labeled as total and disjoint, as every media must be either a Movie or TV show, and cannot be both.

- We've further simplified the subscription plan and subscription entity sets, removing the subscription plan entity and combining its elements with the Subscription entity. This Subscription entity now has a one-to-one relationship with a user, as Each user can have one subscription and each subscription **must** be associated to a single user (total participation).

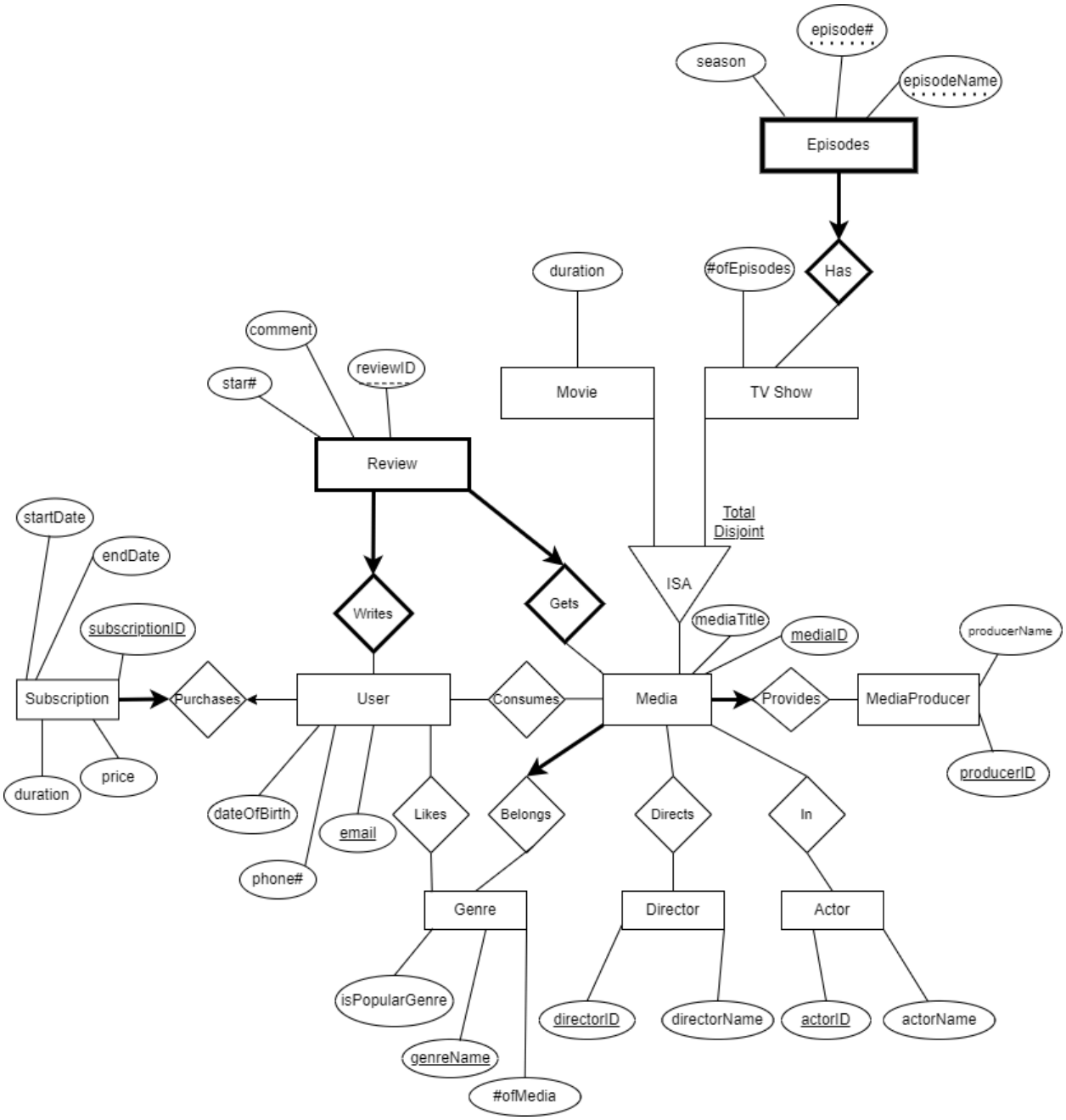
- The watch history, favorites list, and watch list entities have been removed as they were not key aspects of our database functionality – these can be added in the future as additional elements, as needed. This change eliminated our cardinality issue of each media being limited to only one user/watch history relationship.

- A similar cardinality issue existed in our ternary relationship between media, user, and review, limiting the ability for one media to be associated outside of a single user/review pair's relationship. To better represent this, we decided to separate the ternary relationship into two binary relationships, representing reviews as a weak entity attached to media and users.

- We've added additional entities Genre, Actor, and Director, to further focus on the entities participating in relationships with media entities.

- TV shows now have a weak entity, "episodes" connected to them, as each episode has its own attributes, and episodes cannot exist unless they belong to a TV show.

- Additional attribute changes have been made to ensure we were able to normalize and had sufficient elements of the streaming service process modeled within our database.



4. Primary keys are underlined, foreign keys are bolded, and candidate keys have been labeled (CK).

Subscription(subscriptionID: CHAR[50],
duration: CHAR[50] NOT NULL,
price: INT NOT NULL,
startDate: DATE NOT NULL,
endDate: DATE NOT NULL,
email: CHAR[50])

User(email: CHAR[50],
phone#: INT NOT NULL UNIQUE (CK),
subscriptionID: CHAR[50],
name: CHAR[50] NOT NULL,
dateOfBirth: DATE NOT NULL)

Genre(GenreName: CHAR[50],
#ofMedia: INT NOT NULL,
isPopularGenre: BOOLEAN)

Director(directorID: CHAR[50]
directorName: CHAR[50] NOT NULL)

Actor(actorID: CHAR[50],
actorName: CHAR[50] NOT NULL)

MediaProducer(producerID: CHAR[50]
producerName: CHAR[50] NOT NULL)

Movie(**mediaID**: CHAR[50],
duration: INT NOT NULL)

TVShow(**mediaID**: CHAR[50],
#ofEpisodes: INT NOT NULL)

Episodes_Has(**mediaID**: CHAR[50],
episodeName: CHAR[50],
episode#: INT NOT NULL,
season: INT NOT NULL)

Media(mediaID: CHAR[50],
mediaTitle: CHAR[50] NOT NULL,
producerID: CHAR[50],
genreName: CHAR[50])

Reviews_Gets_Writes(reviewID: CHAR[50],
comment: CHAR[250],
star#: INT NOT NULL,
email: CHAR[50],
mediaID: CHAR[50])

5. The functional dependencies in our relations, including the ones involving all candidate keys (including the primary key) have been listed below, without the trivial FDs:

Relation of Subscription
- subscriptionID -> duration
- subscriptionID -> price
- subscriptionID -> email (FK)
- subscriptionID -> startDate
- subscriptionID -> endDate
- duration -> price (violates BCNF)

Relation of User
- email -> phone
- email -> dob
- email -> subscriptionID (FK)
- phone (CK) -> dob
- phone (CK) -> subscriptionID (FK)

Relation of Media
- mediaID -> title
- mediaID -> producerID(FK)
- mediaID -> genreName(FK)

Relation of TV shows
- mediaID -> #episodes

Relation of Movie
- mediaID -> duration

Relation of Media Producer
- producerID -> producerName

Relation of Actor
- actorID -> actorName

Relation of Director
- directorID -> directorName

Relation of Genre
- genreName -> #ofMedia
- genreName -> isPopularGenre
- #ofMedia -> isPopularGenre (violates BCNF)

Relation of Review and its relationship *gets*

- reviewID, mediaID -> comment
- reviewID, mediaID -> star #
- reviewID, mediaID -> email

Relation of Review and its relationship *writes*
- reviewID, email -> comment
- reviewID , email -> star #
- reviewID , email -> mediaID

Relation of Episode and its relationship *has*
- mediaID, episode#, episodeName -> season

Relation of relationship Consumes
- No non-trivial FDs

Relation of relationship Likes
- No non-trivial FDs

Relation of relationship Belongs
- No non-trivial FDs

Relation of relationship Directs
- No non-trivial FDs

Relation of relationship In
- No non-trivial FDs

6. **The 2 relational tables that do not follow BCNF are in our Subscription and Genre entities, shown below:**

Subscription(subscriptionID: CHAR[50],
duration: CHAR[50] NOT NULL,
price: INT NOT NULL,
startDate: DATE NOT NULL,
endDate: DATE NOT NULL,
email: CHAR[50])

Relation of Subscription
- subscriptionID -> duration
- subscriptionID -> price
- subscriptionID -> email

- subscriptionID -> startDate
- subscriptionID -> endDate
- duration -> price (violates BCNF)

The functional dependency in bold violates BCNF as **duration** is not a super key, and is on the LHS of a FD within Subscription.

duration->price violates BCNF, so we need to decompose:

R1(subscriptionID, duration, email, startDate, endDate) and R2(duration, price)

Decomposing Subscription into R1 and R2 normalizes the relation as both R1 and R2 contain 2 attributes, ensuring that BCNF is followed.

Genre(GenreName: CHAR[50],
#ofMedia: INT NOT NULL,
isPopularGenre: BOOLEAN)

Relation of Genre
- genreName -> # of media in genre
- genreName -> isPopularGenre
- #ofMedia -> isPopularGenre (violates BCNF)

The functional dependency in bold violates BCNF as **#ofMedia** is not a super key, and is on the LHS of a FD within Genre.

Genre(genreName, #media, isPopular)

R1(#media, isPopular) R2(genreName,#media)

Decomposing Genre into R1 and R2 normalizes the relation as both R1 and R2 contain 2 attributes, ensuring that BCNF is followed.

7/8. Our tables and insertions (maintaining the minimum of 5 tuple insertions per table) have been listed below:

```
CREATE TABLE Subscription1(
subscriptionID CHAR[50],
duration CHAR[50] NOT NULL,
email CHAR[50],
startDate DATE NOT NULL,
endDate DATE NOT NULL,
```

```
PRIMARY KEY (subscriptionID)
FOREIGN KEY (email) REFERENCES User(email) ON DELETE CASCADE
);
```

INSERT
INTO Subscription1(subscriptionID, duration, email, startDate, endDate)
VALUES('sid1', 'ANNUALLY', SELECT email from User WHERE email='bob@gmail.com', 2021-11-27, 2022-11-27),
('sid2', 'MONTHLY', SELECT email from User WHERE email='jane@gmail.com', 2023-04-01, 2023-05-01),
('sid3', 'DAILY', SELECT email from User WHERE email='ruth@hotmail.com', 2023-11-03, 2023-12-03),
('sid4', 'WEEKLY', SELECT email from User WHERE email='michael@hotmail.com', 2023-02-15, 2024-02-22),
('sid5', 'BIWEEKLY', SELECT email from User WHERE email='john@gmail.com', 2024-01-01, 2024-01-14)

```
CREATE TABLE Subscription2(
duration CHAR[50],
price INT NOT NULL,
PRIMARY KEY (duration),
FOREIGN KEY (duration) REFERENCES Subscription1(duration) ON DELETE CASCADE
);
```

INSERT
INTO Subscription2(duration, price)
VALUES(SELECT duration from Subscription1 WHERE duration='ANNUALLY', 100),
(SELECT duration from Subscription1 WHERE duration='MONTHLY', 10),
(SELECT duration from Subscription1 WHERE duration='DAILY', 4),
(SELECT duration from Subscription1 WHERE duration='WEEKLY', 6),
(SELECT duration from Subscription1 WHERE duration='BIWEEKLY', 8)

```
CREATE TABLE User(
email CHAR[50],
phone# INT NOT NULL UNIQUE,
subscriptionID CHAR[50],
name CHAR[50] NOT NULL,
dateOfBirth DATE NOT NULL,
PRIMARY KEY (email),
```

```
FOREIGN KEY (subscriptionID) REFERENCES Subscription1(subscriptionID) ON
DELETE SET NULL
);
```

INSERT
INTO User(email, phone#, subscriptionID, name, dateOfBirth)
VALUES(‘bob@gmail.com’, 2041238888, SELECT subscriptionID from Subscription1 WHERE subscriptionID='sid1', 'BOB ROSS', 1995-09-12),
(‘jane@gmail.com’, 4038972345, SELECT subscriptionID from Subscription1 WHERE subscriptionID='sid2', 'JANE DOE', 1990-06-08),
(‘ruth@hotmail.com’, 6045892654, SELECT subscriptionID from Subscription1 WHERE subscriptionID='sid3', 'RUTH THEBOSS', 2000-01-10),
(‘michael@hotmail.com’, 5876231890, SELECT subscriptionID from Subscription1 WHERE subscriptionID='sid4', MICHAEL GODAD, 2002-05-09),
(‘john@gmail.com’, 7781459898, SELECT subscriptionID from Subscription1 WHERE subscriptionID='sid5', 'JOHN DONKINS', 2004-03-04)

```
CREATE TABLE Genre1(
genreName CHAR[50],
#ofMedia INT NOT NULL,
PRIMARY KEY (genreName)
);
```

INSERT
INTO Genre1(genreName, #ofMedia)
VALUES('Horror', 43),
(‘Romance Sci-Fi’, 2,),
(‘Thriller’, 121),
(‘Action’, 77),
(‘Comedy Adventure’, 5)

```
CREATE TABLE Genre2(
#ofMedia INT,
isPopularGenre BOOLEAN,
PRIMARY KEY (#ofMedia),
FOREIGN KEY (genreName) REFERENCES Genre1(genreName)
);
```

INSERT
INTO Genre2(#ofMedia, isPopularGenre)
VALUES(SELECT #ofMedia from Genre1 WHERE #ofMedia=43, TRUE),

(SELECT #ofMedia from Genre1 WHERE #ofMedia=2, FALSE)
(SELECT #ofMedia from Genre1 WHERE #ofMedia=121, TRUE)
(SELECT #ofMedia from Genre1 WHERE #ofMedia=77, TRUE)
(SELECT #ofMedia from Genre1 WHERE #ofMedia=5, FALSE)

```
CREATE TABLE Director(
directorID CHAR[50],
directorName CHAR[50] NOT NULL,
PRIMARY KEY (directorID)
);
```

INSERT
INTO Director(directorID, directorName)
VALUES('did1', 'Christopher Nolan'),
(‘did2’, ‘Steven Spielberg’),
(‘did3’, ‘Martin Scorsese’),
(‘did4’, ‘James Cameron’),
(‘did5’, ‘Michael Bay’)

```
CREATE TABLE Actor(
actorID CHAR[50],
actorName CHAR[50] NOT NULL,
PRIMARY KEY (actorID)
);
```

INSERT
INTO Actor(actorID, actorName)
VALUES('aid1', 'Tom Holland'),
('aid2', 'Weena Wibowo'),
('aid3', 'Suhayl Patel'),
('aid4', 'Keean Vidyarthi'),
('aid5', ‘John Pork')

```
CREATE TABLE MediaProducer(
producerID CHAR[50],
producerName CHAR[50] NOT NULL,
PRIMARY KEY (producerID)
);
```

INSERT
INTO MediaProducer(producerID, producerName)
VALUES('mpid1', 'Walt Disney'),
('mpid2', 'Pixar'),
('mpid3', 'Paramount'),
('mpid4', 'Netflix'),
('mpid5', 'Universal Studio')

```
CREATE TABLE Media(
mediaID CHAR[50],
mediaTitle CHAR[50] NOT NULL,
producerID CHAR[50],
genreName char[50],
PRIMARY KEY (mediaID),
FOREIGN KEY (producerID) REFERENCES MediaProducer(producerID) ON DELETE
CASCADE
FOREIGN KEY (genreName) REFERENCES Genre1(genreName) ON DELETE SET NULL
);
```

INSERT
INTO Media(mediaID, mediaTitle, producerID, genreName)
VALUES('mid1', 'Great Movie 1', SELECT producerID from MediaProducer WHERE producerID='pid1', SELECT genreName from Genre1 WHERE genreName='Horror'),
('mid2', 'Spectacular Movie 2', SELECT producerID from MediaProducer WHERE producerID='pid2', SELECT genreName from Genre1 WHERE genreName='Comedy Adventure'),
('mid3', 'OK Movie 3', SELECT producerID from MediaProducer WHERE producerID='pid3', SELECT genreName from Genre1 WHERE genreName='Action'),
('mid4', 'OK Movie 5', SELECT producerID from MediaProducer WHERE producerID='pid4', SELECT genreName from Genre1 WHERE genreName='Romance'),
('mid5', 'Decent Movie 3', SELECT producerID from MediaProducer WHERE producerID='pid5', SELECT genreName from Genre1 WHERE genreName='Thriller')
('mid6', 'Great TV show 1, SELECT producerID from MediaProducer WHERE producerID='pid1', SELECT genreName from Genre1 WHERE genreName='Horror'),
('mid7', 'Spectacular TV show', SELECT producerID from MediaProducer WHERE producerID='pid2', SELECT genreName from Genre1 WHERE genreName='Romantic Comedy'),
('mid8', Mid TV show', SELECT producerID from MediaProducer WHERE producerID='pid3', SELECT genreName from Genre1 WHERE genreName='Fantasy'),
('mid9', 'OK TV show 1', SELECT producerID from MediaProducer WHERE producerID='pid4', SELECT genreName from Genre1 WHERE genreName='Fiction'),
('mid10', 'Decent TV show 2', SELECT producerID from MediaProducer WHERE producerID='pid5', SELECT genreName from Genre1 WHERE genreName='Western')

```
CREATE TABLE Movie(
mediaID CHAR[50],
duration INT NOT NULL,
PRIMARY KEY (mediaID),
FOREIGN KEY (mediaID) REFERENCES Media(mediaID) ON DELETE CASCADE
);
```

INSERT
INTO Movie(mediaID, duration)
VALUES(SELECT mediaID from Media WHERE mediaID='mid1', 124),
(SELECT mediaID from Media WHERE mediaID='mid2', 189),
(SELECT mediaID from Media WHERE mediaID='mid3', 143),
(SELECT mediaID from Media WHERE mediaID='mid4', 93),
(SELECT mediaID from Media WHERE mediaID='mid5', 67)

```
CREATE TABLE TVShow(
mediaID CHAR[50],
#ofEpisodes INT NOT NULL,
PRIMARY KEY (mediaID),
FOREIGN KEY (mediaID) REFERENCES Media(mediaID) ON DELETE CASCADE
);
```

INSERT
INTO TVShow(mediaID, #ofEpisodes)
VALUES(SELECT mediaID from Media WHERE mediaID='mid6', 14),
(SELECT mediaID from Media WHERE mediaID='mid7', 20),
(SELECT mediaID from Media WHERE mediaID='mid8', 57),
(SELECT mediaID from Media WHERE mediaID='mid9', 4),
(SELECT mediaID from Media WHERE mediaID='mid10', 25)

```
CREATE TABLE Consumes(
mediaID CHAR[50],
email CHAR[50],
PRIMARY KEY (mediaID, email),
FOREIGN KEY (mediaID) REFERENCES Media(mediaID),
FOREIGN KEY (email) REFERENCES User(email)
);
```

INSERT
INTO Consumes(mediaID, email)

VALUES(SELECT mediaID from Media WHERE mediaID=’mid1’, SELECT email from User WHERE email=’bob@gmail.com'),
(SELECT mediaID from Media WHERE mediaID=’mid2’, SELECT email from User WHERE email=’jane@gmail.com’),
(SELECT mediaID from Media WHERE mediaID=’mid3’, SELECT email from User WHERE email=’ruth@hotmail.com’),
(SELECT mediaID from Media WHERE mediaID=’mid4’, SELECT email from User WHERE email=’michael@hotmail.com’),
(SELECT mediaID from Media WHERE mediaID=’mid5’, SELECT email from User WHERE email=‘john@gmail.com’)

```
CREATE TABLE Directs(
mediaID CHAR[50],
directorID CHAR[50],
PRIMARY KEY (mediaID, directorID),
FOREIGN KEY (mediaID) REFERENCES Media(mediaID),
FOREIGN KEY (directorID) REFERENCES Director(directorID)
);
```

INSERT
INTO Directs(mediaID, directorID)
VALUES(SELECT mediaID from Media WHERE mediaID=’mid1’, SELECT directorID from Director WHERE directorID='did1'),
(SELECT mediaID from Media WHERE mediaID=’mid2’, SELECT directorID from Director WHERE directorID='did2'),
(SELECT mediaID from Media WHERE mediaID=’mid3’, SELECT directorID from Director WHERE directorID='did3'),
(SELECT mediaID from Media WHERE mediaID=’mid4’, SELECT directorID from Director WHERE directorID='did4'),
(SELECT mediaID from Media WHERE mediaID=’mid5’, SELECT directorID from Director WHERE directorID='did5')

```
CREATE TABLE Likes(
email CHAR[50],
genreName CHAR[50],
PRIMARY KEY (email, genreName),
FOREIGN KEY (email) REFERENCES User(email),
FOREIGN KEY (genreName) REFERENCES Genre1(genreName)
)
```

INSERT

INTO Likes(email, genreName)
VALUES(SELECT email from User WHERE email=’bob@gmail.com', SELECT genreName from Genre1 WHERE genreName=’Horror’),
(SELECT email from User WHERE email=’jane@gmail.com’, SELECT genreName from Genre1 WHERE genreName=’Thriller’),
(SELECT email from User WHERE email=’ruth@hotmail.com’, SELECT genreName from Genre1 WHERE genreName=’Romance’),
(SELECT email from User WHERE email=’michael@hotmail.com’, SELECT genreName from Genre1 WHERE genreName='Action’),
(SELECT email from User WHERE email=’john@gmail.com’, SELECT genreName from Genre1 WHERE genreName='Comedy Adventure’)

```
CREATE TABLE In(
mediaID CHAR[50],
actorID CHAR[50],
FOREIGN KEY (mediaID) REFERENCES Media(mediaID),
FOREIGN KEY (actorID) REFERENCES Actor(actorID)
);
```

INSERT
INTO Likes(mediaID, actorID)
VALUES(SELECT mediaID from Media WHERE mediaID=’mid1’, SELECT directorID from Director WHERE directorID='did1'),
(SELECT mediaID from Media WHERE mediaID=’mid2’, SELECT directorID from Director WHERE directorID='did2'),
(SELECT mediaID from Media WHERE mediaID=’mid3’, SELECT directorID from Director WHERE directorID='did3'),
(SELECT mediaID from Media WHERE mediaID=’mid4’, SELECT directorID from Director WHERE directorID='did4'),
(SELECT mediaID from Media WHERE mediaID=’mid5’, SELECT directorID from Director WHERE directorID='did5')

```
CREATE TABLE Review_Gets_Writes(
reviewID CHAR[50],
comment CHAR[250],
star# INT NOT NULL,
email CHAR[50],
mediaID CHAR[50],
PRIMARY KEY (reviewID, email, mediaID),
FOREIGN KEY (email) REFERENCES User(email) ON DELETE CASCADE,
FOREIGN KEY (mediaID) REFERENCES Media(mediaID) ON DELETE CASCADE
);
```

INSERT
INTO Review_Gets_Writes(reviewID, comment, star#, email, mediaID)
VALUES('rid1', 'This movie sucked!', 1, SELECT email from User WHERE email=’bob@gmail.com’, SELECT mediaID from Media WHERE mediaID=’mid1’),
('rid2', 'I am a new person after watching this. RECOMMEND!!!!', 5, SELECT email from User WHERE email=’jane@gmail.com', SELECT mediaID from Media WHERE mediaID=’mid2’),
('rid3', 'This movie was great!', 3, SELECT email from User WHERE email=’ruth@hotmail.com’, SELECT mediaID from Media WHERE mediaID=’mid3’),
('rid4', 'This show was so epic!', 4, SELECT email from User WHERE email= michael@hotmail.com', SELECT mediaID from Media WHERE mediaID=’mid6’),
('rid5', 'Mid TV show to be honest', 2, SELECT email from User WHERE email=’’john@gmail.com’, SELECT mediaID from Media WHERE mediaID=’mid7’)

```
CREATE TABLE Episode_Has(
mediaID CHAR[50],
episode# INT NOT NULL,
episodeName CHAR[50] NOT NULL,
season INT NOT NULL,
PRIMARY KEY (mediaID, episode#, episodeName),
FOREIGN KEY (mediaID) REFERENCES Media(mediaID) ON DELETE CASCADE
);
```

INSERT
INTO Episode_Has(mediaID, episode#, episodeName, season)
VALUES(SELECT mediaID from Media WHERE mediaID=’mid6’, 12, 'Trifling tree tops', 1),
(SELECT mediaID from Media WHERE mediaID=’mid7’, 1, 'Pilot', 3),
(SELECT mediaID from Media WHERE mediaID=’mid8’, 6, 'Finale', 7),
(SELECT mediaID from Media WHERE mediaID=’mid9’, 2, 'Roundness', 2),
(SELECT mediaID from Media WHERE mediaID=’mid10’, 11, 'Uh Oh!', 3)