Iraklis Bogiatziou
18329647

I have chosen to implement a database based on the show “Community” created by Dan Harmon. The database includes 10 tables. Episode, Character, Appears, Actor, Actor Gender, Production Staff, Staff role, Developed, Season, Rating.

The Episode table includes the episode_id which has 3 digits, the first of which indicates the current season, and the other two the current episode. The table also contains the title of the episode, the date it was aired, its duration as well as the season_id as a foreign key from the season table.

The Character has 4 attributes. Its unique ID, the first and last name of the character, and their occupation in the show.

The appears table takes the two IDs from Episode and Character and holds the characters of each episode.

The Actor table holds the ID, the first and last name of the actor, and the char_id foreign key from Character to link each actor to a specific character.

The Actor_Gender table is used to save the gender of each actor.

Production_Staff includes an ID and the first and last name of the staff member.

The Staff_Roles takes the ID from the Production_Staff table and saves the role of each staff which can be either “Director” or “Writer”.

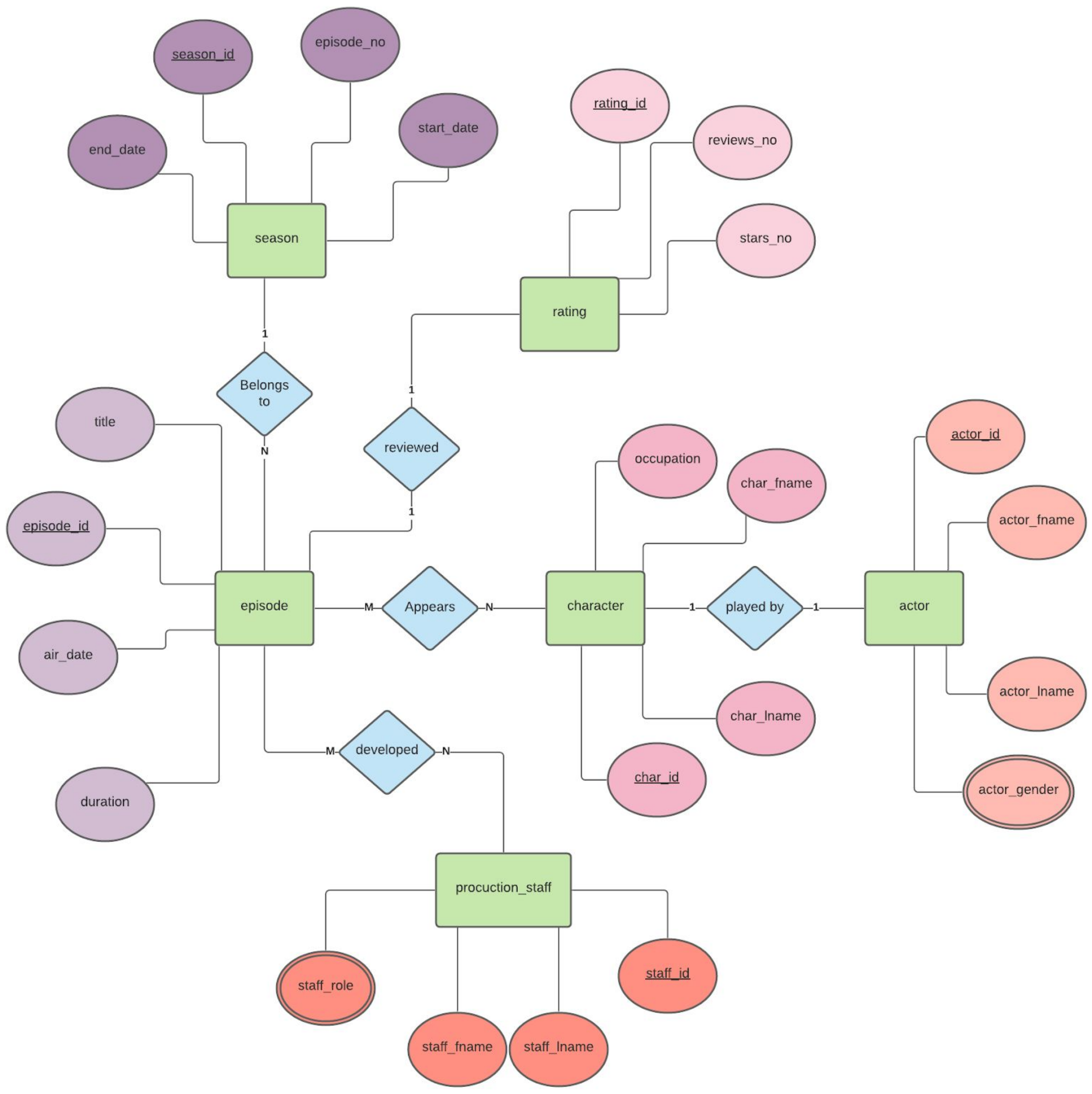
Developed shows Directors and Writers of each episode.

The Season table has a one-digit ID that represents the current season, it also includes the number of episodes in that season, as well as the start and end dates.

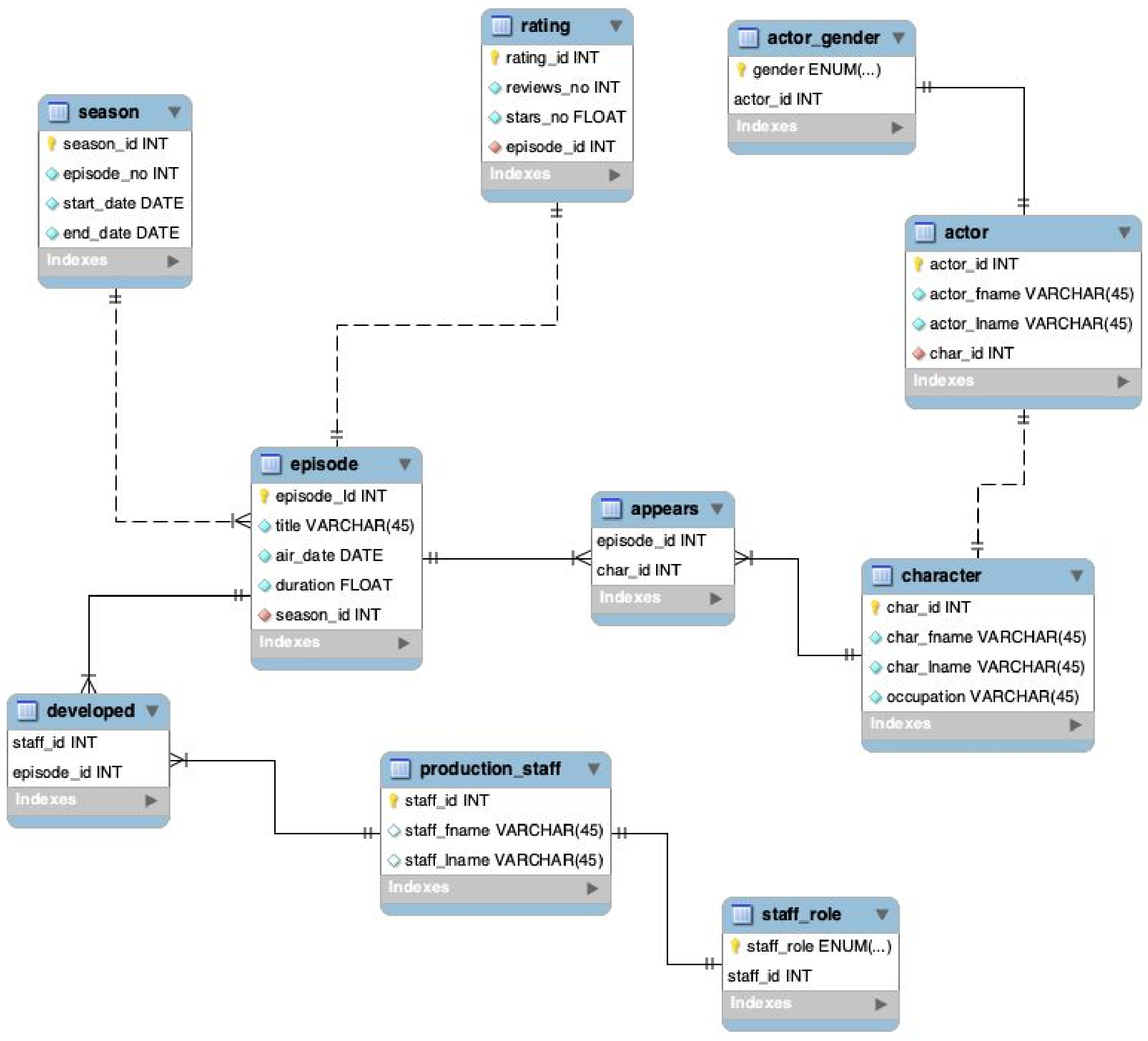
Lastly, the Rating table includes the ID of each rating, the number of reviews, and the star rating of each episode.

Below, I’ve included the Entity-Relationship Diagrams the Relational Schema and the Functional Dependency to accompany and better visualize my database

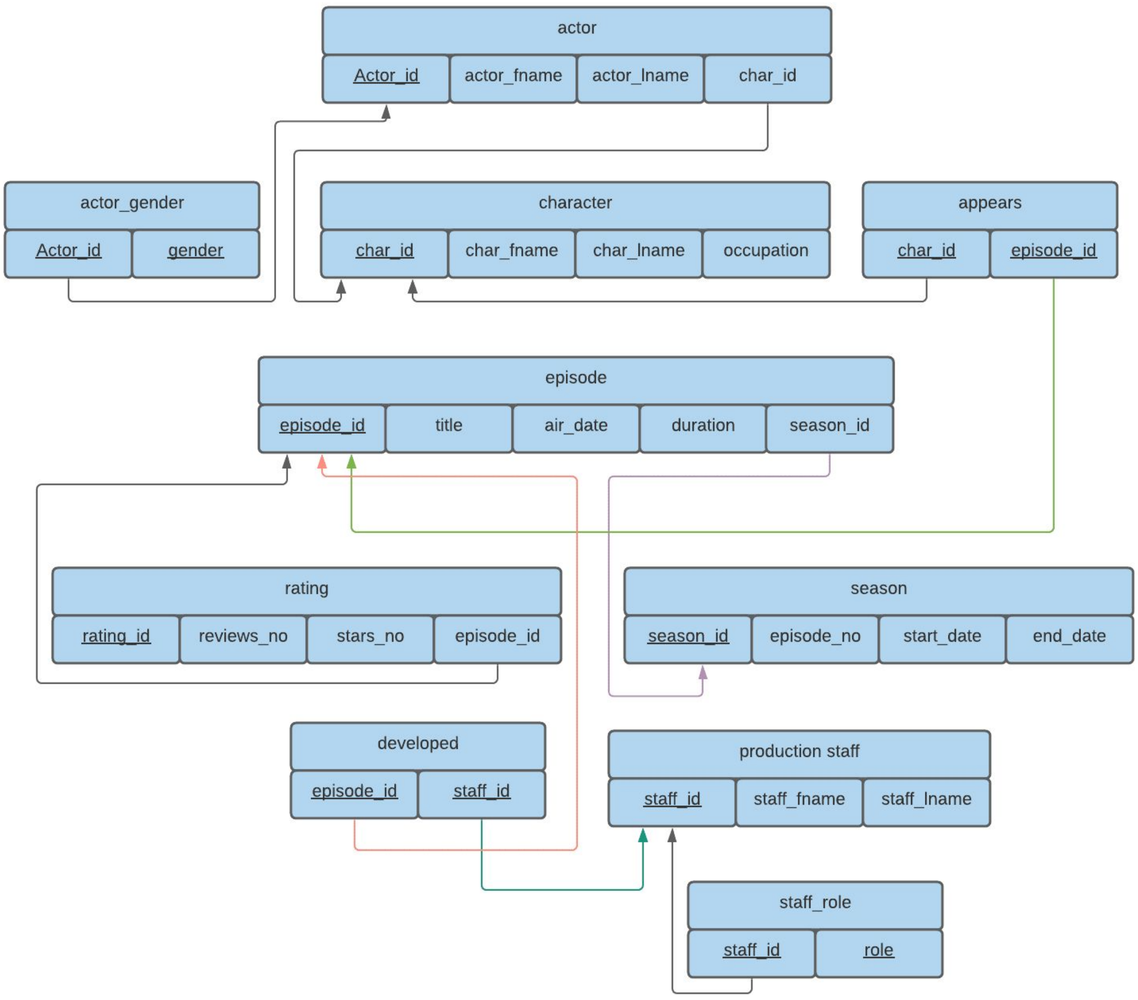
Entity Relationship Diagram



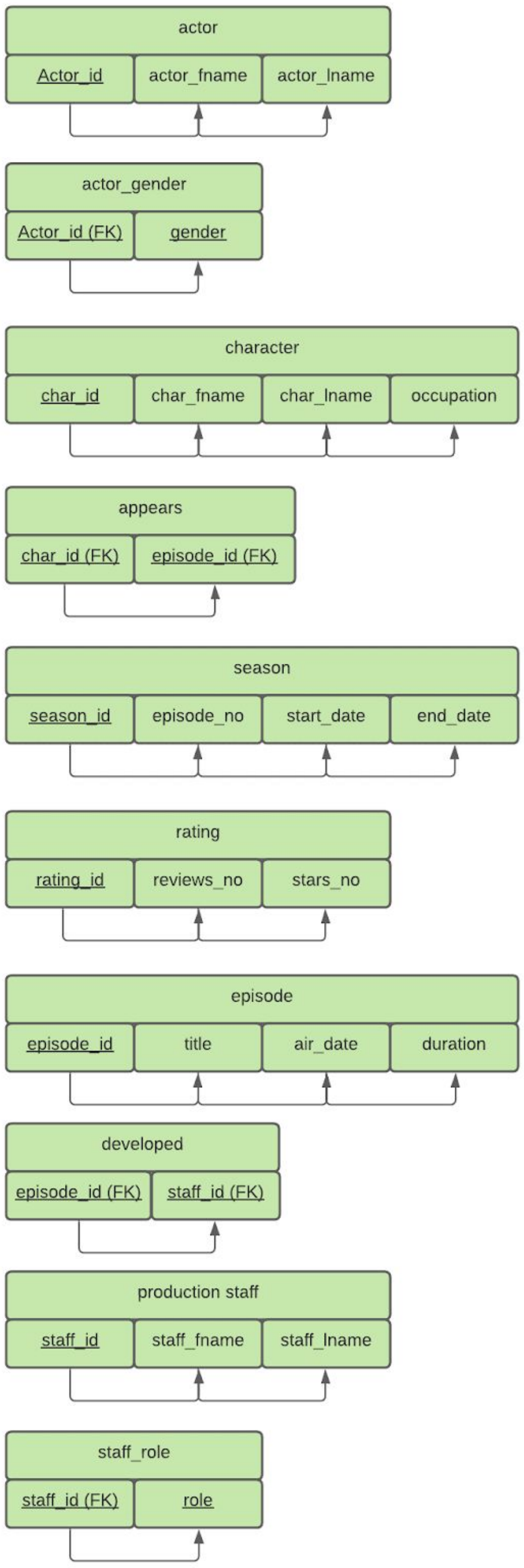
Entity Relationship Diagram on MySQL Workbench



Relational Schema

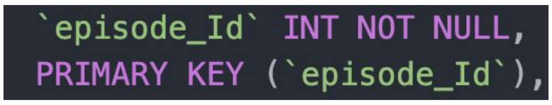


Functional Dependency

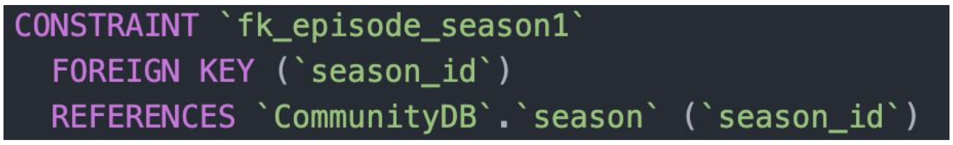


Implicit Constraints

Episode

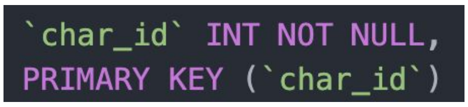


```
 `episode_Id` INT NOT NULL,
 PRIMARY KEY (`episode_Id`),
```



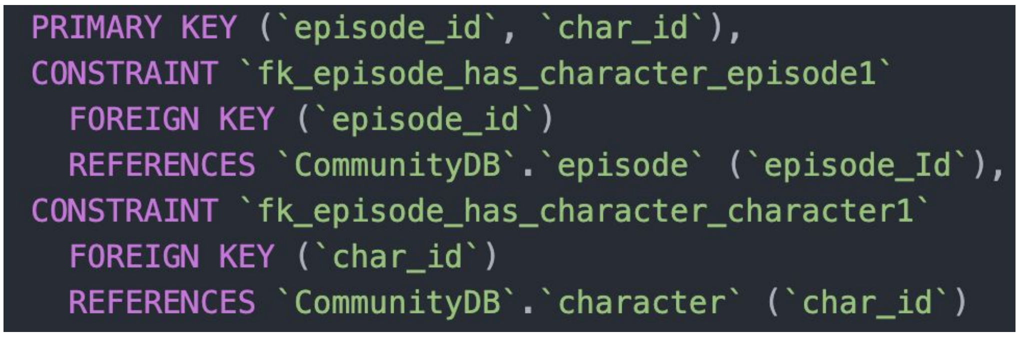
```
CONSTRAINT `fk_episode_season1`
  FOREIGN KEY (`season_id`)
  REFERENCES `CommunityDB`.`season` (`season_id`)
```

Character



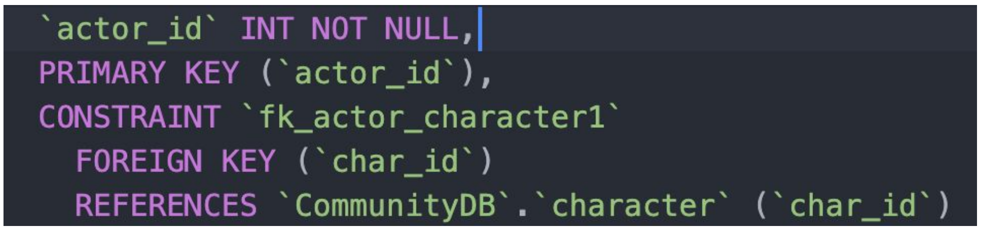
```
 `char_id` INT NOT NULL,
PRIMARY KEY (`char_id`)
```

Appears



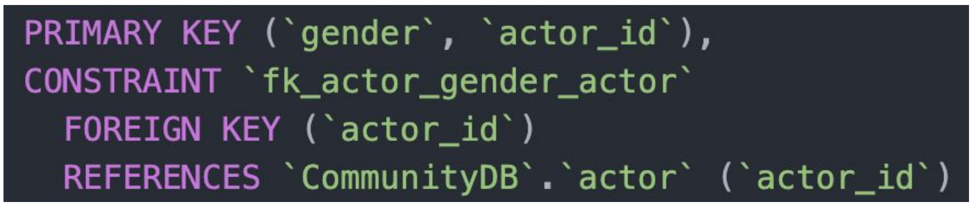
```
 PRIMARY KEY (`episode_id`, `char_id`),
 CONSTRAINT `fk_episode_has_character_episode1`
   FOREIGN KEY (`episode_id`)
   REFERENCES `CommunityDB`.`episode` (`episode_Id`),
 CONSTRAINT `fk_episode_has_character_character1`
   FOREIGN KEY (`char_id`)
   REFERENCES `CommunityDB`.`character` (`char_id`)
```

Actor



```
 `actor_id` INT NOT NULL,
 PRIMARY KEY (`actor_id`),
 CONSTRAINT `fk_actor_character1`
   FOREIGN KEY (`char_id`)
   REFERENCES `CommunityDB`.`character` (`char_id`)
```

Actor_gender



```
PRIMARY KEY (`gender`, `actor_id`),
CONSTRAINT `fk_actor_gender_actor`
  FOREIGN KEY (`actor_id`)
  REFERENCES `CommunityDB`.`actor` (`actor_id`)
```

Production_staff

```
`staff_id` INT NOT NULL,
PRIMARY KEY (`staff_id`)
```

Staff_role

```
`staff_role` ENUM('director', 'writer') NOT NULL,
`staff_id` INT NOT NULL,
PRIMARY KEY (`staff_role`, `staff_id`),
CONSTRAINT `fk_staff_role_production_staff1`
  FOREIGN KEY (`staff_id`)
  REFERENCES `CommunityDB`.`production_staff` (`staff_id`)
```

Developed

```
`staff_id` INT NOT NULL,
`episode_id` INT NOT NULL,
PRIMARY KEY (`staff_id`, `episode_id`),
CONSTRAINT `fk_production_staff_has_episode_production_staff1`
  FOREIGN KEY (`staff_id`)
  REFERENCES `CommunityDB`.`production_staff` (`staff_id`),
CONSTRAINT `fk_production_staff_has_episode_episode1`
  FOREIGN KEY (`episode_id`)
  REFERENCES `CommunityDB`.`episode` (`episode_Id`)
```

Season

```
`season_id` INT NOT NULL,
PRIMARY KEY (`season_id`)
```

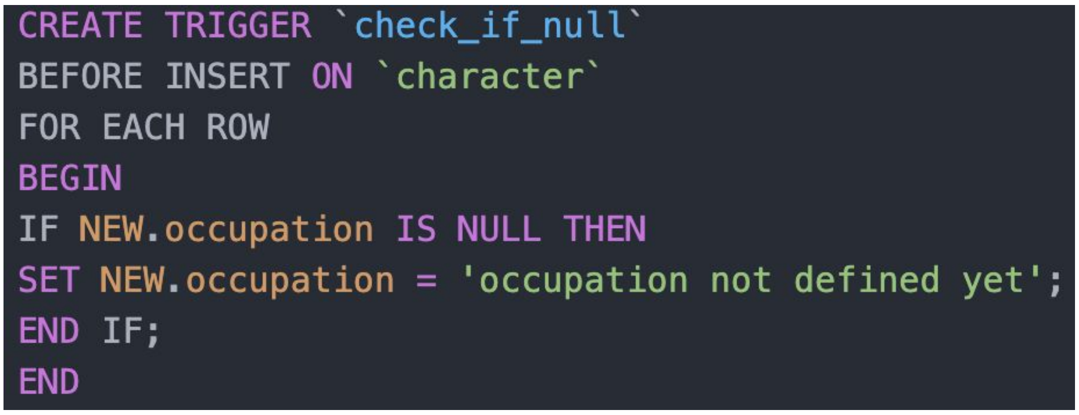
Rating

```
`rating_id` INT NOT NULL,
`episode_id` INT NOT NULL,
PRIMARY KEY (`rating_id`),
CONSTRAINT `fk_rating_episode1`
  FOREIGN KEY (`episode_id`)
  REFERENCES `CommunityDB`.`episode` (`episode_Id`)
```

Semantic Constraints

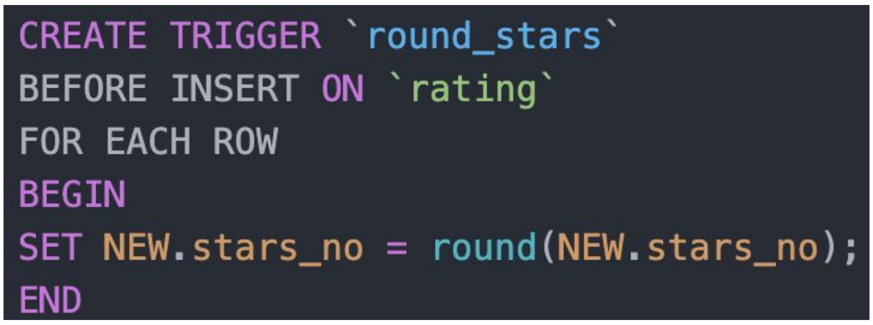
Triggers

Trigger to check if null before insert.
This trigger is used to check if the occupation of each character is not null. This is helpful to identify each character's role in the show.



```
CREATE TRIGGER `check_if_null`
BEFORE INSERT ON `character`
FOR EACH ROW
BEGIN
IF NEW.occupation IS NULL THEN
SET NEW.occupation = 'occupation not defined yet';
END IF;
END
```

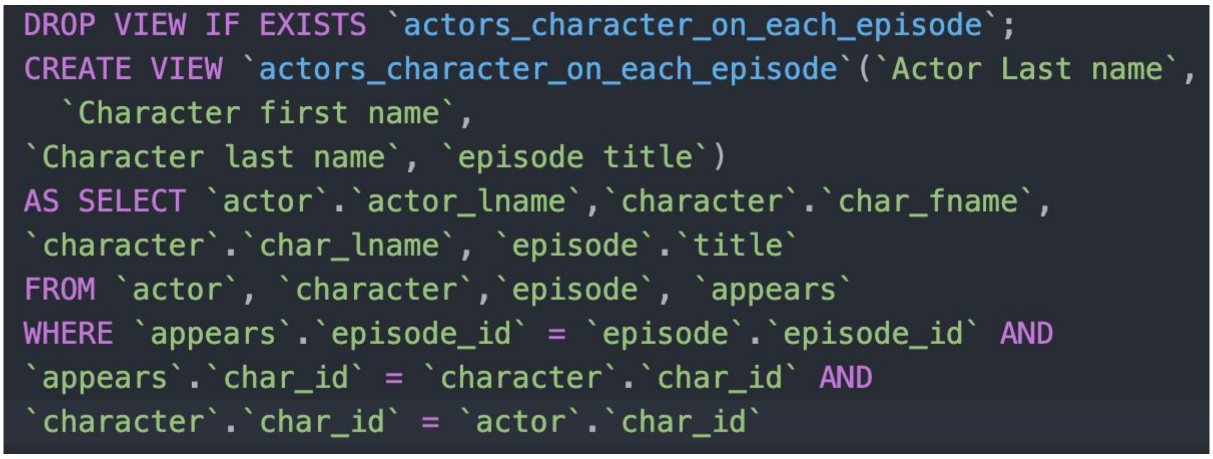
Trigger to check if null before insert.
This trigger uses the round function to round the star rating of each episode.



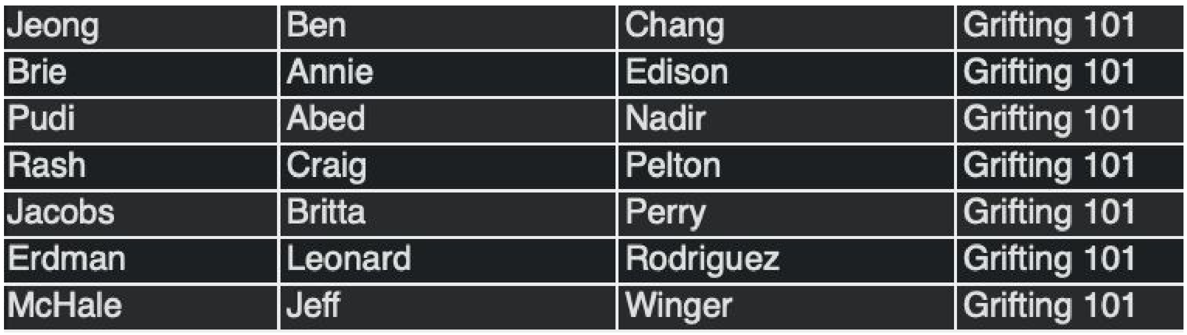
```
CREATE TRIGGER `round_stars`
BEFORE INSERT ON `rating`
FOR EACH ROW
BEGIN
SET NEW.stars_no = round(NEW.stars_no);
END
```

View

This view creates a table that includes the Actor's Last name, their Character's first and last name as well as the episode that they appear.



```sql
DROP VIEW IF EXISTS `actors_character_on_each_episode`;
CREATE VIEW `actors_character_on_each_episode`(`Actor Last name`,
  `Character first name`,
`Character last name`, `episode title`)
AS SELECT `actor`.`actor_lname`,`character`.`char_fname`,
`character`.`char_lname`, `episode`.`title`
FROM `actor`, `character`,`episode`, `appears`
WHERE `appears`.`episode_id` = `episode`.`episode_id` AND
`appears`.`char_id` = `character`.`char_id` AND
`character`.`char_id` = `actor`.`char_id`
```



Jeong	Ben	Chang	Grifting 101
Brie	Annie	Edison	Grifting 101
Pudi	Abed	Nadir	Grifting 101
Rash	Craig	Pelton	Grifting 101
Jacobs	Britta	Perry	Grifting 101
Erdman	Leonard	Rodriguez	Grifting 101
McHale	Jeff	Winger	Grifting 101

Appendix

```
-- -----------------------------------------------------
-- Schema CommunityDB
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `CommunityDB` DEFAULT
CHARACTER SET utf8 ;
USE `CommunityDB` ;

-- -----------------------------------------------------
-- Table `CommunityDB`.`character`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CommunityDB`.`character` (
  `char_id` INT NOT NULL,
  `char_fname` VARCHAR(45) NOT NULL,
  `char_lname` VARCHAR(45) NOT NULL,
  `occupation` VARCHAR(45),
  PRIMARY KEY (`char_id`)
);


INSERT INTO `character` (`char_id`, `char_fname`, `char_lname`,
`occupation`)
VALUES
(1001, 'Jeff', 'Winger', 'student'),
(1002, 'Britta', 'Perry', 'student'),
(1003, 'Abed', 'Nadir', 'student'),
(1004, 'Shirley', 'Bennet', 'student'),
(1005, 'Annie', 'Edison', 'student'),
(1006, 'Troy', 'Barnes', 'student'),
(1007, 'Piece', 'Hawthorne', 'student'),
(1008, 'Ian', 'Duncan', 'professor'),
(1009, 'Craig', 'Pelton', 'dean'),
(1010, 'Ben', 'Chang', 'everything'),
(1011, 'Andre', 'Bennet', 'husband'),
(1012, 'Buzz', 'Hickey', 'professor'),
(1013, 'Leonard', 'Rodriguez', 'student');
COMMIT;
```

```
-- -----------------------------------------------------
-- Table `CommunityDB`.`actor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CommunityDB`.`actor` (
  `actor_id` INT NOT NULL,
  `actor_fname` VARCHAR(45) NOT NULL,
  `actor_lname` VARCHAR(45) NOT NULL,
  `char_id` INT NOT NULL,
  PRIMARY KEY (`actor_id`),
  CONSTRAINT `fk_actor_character1`
    FOREIGN KEY (`char_id`)
    REFERENCES `CommunityDB`.`character` (`char_id`)
);

INSERT INTO `actor` (`actor_id`, `actor_fname`, `actor_lname`, `char_id`)
VALUES
(2001, 'Joel', 'McHale', 1001),
(2002, 'Gillian', 'Jacobs', 1002),
(2003, 'Danny', 'Pudi', 1003),
(2004, 'Yvette Nicole', 'Brown', 1004),
(2005, 'Alison', 'Brie', 1005),
(2006, 'Donald', 'Glover', 1006),
(2007, 'Chevy', 'Chase', 1007),
(2008, 'John', 'Oliver', 1008),
(2009, 'Jim', 'Rash', 1009),
(2010, 'Ken', 'Jeong', 1010),
(2011, 'Malcolm-Jamal', 'Warner', 1011),
(2012, 'Jonathan', 'Banks', 1012),
(2013, 'Richard', 'Erdman', 1013);
COMMIT;

-- -----------------------------------------------------
-- Table `CommunityDB`.`actor_gender`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CommunityDB`.`actor_gender` (
  `gender` ENUM('male', 'female') NOT NULL,
  `actor_id` INT NOT NULL,
  PRIMARY KEY (`gender`, `actor_id`),
  CONSTRAINT `fk_actor_gender_actor`
    FOREIGN KEY (`actor_id`)
    REFERENCES `CommunityDB`.`actor` (`actor_id`)
  );

  INSERT INTO `actor_gender` (`gender`, `actor_id`)
  VALUES
  ('male', 2001),
  ('female', 2002),
  ('male', 2003),
  ('female', 2004),
  ('female', 2005),
  ('male', 2006),
  ('male', 2007),
  ('male', 2008),
  ('male', 2009),
  ('male', 2010),
  ('male', 2011),
  ('male', 2012),
  ('male', 2013);
  COMMIT;

-- -----------------------------------------------------
-- Table `CommunityDB`.`season`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CommunityDB`.`season` (
  `season_id` INT NOT NULL,
  `episode_no` INT NOT NULL,
  `start_date` DATE NOT NULL,
  `end_date` DATE NOT NULL,
  PRIMARY KEY (`season_id`)
);

INSERT INTO `season` (`season_id`, `episode_no`, `start_date`, `end_date`)
VALUES
(001, 25, '2009-09-11', '2010-05-20'),
(002, 24, '2010-09-23', '2011-05-12'),
(003, 22, '2011-09-22', '2012-05-17'),
(004, 13, '2013-02-13', '2013-05-09'),
(005, 13, '2014-01-02', '2014-04-17'),
(006, 13, '2015-04-17', '2015-06-02');
COMMIT;
```

```
-- -----------------------------------------------------
-- Table `CommunityDB`.`episode`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CommunityDB`.`episode` (
  `episode_Id` INT NOT NULL,
  `title` VARCHAR(45) NOT NULL,
  `air_date` DATE NOT NULL,
  `duration` FLOAT NOT NULL,
  `season_id` INT NOT NULL,
  PRIMARY KEY (`episode_Id`),
  CONSTRAINT `fk_episode_season1`
    FOREIGN KEY (`season_id`)
    REFERENCES `CommunityDB`.`season` (`season_id`)
);

INSERT INTO `episode` (`episode_id`, `title`, `air_date`, `duration`,
`season_id`)
VALUES
(101, 'Pilot', '2009-09-17', 0.35, 001),
(212, 'Asion Population Studies', '2011-01-20', 0.35, 002),
(321, 'The First Chang Dynasty', '2012-05-17', 0.35, 003),
(408, 'Herstory of Dance', '2013-04-04', 0.35, 004),
(510, 'Advanced Advanced Dungeons & Dragons', '2014-03-20', 0.35, 005),
(609, 'Grifting 101', '2015-05-05', 0.467, 006);
COMMIT;


-- -----------------------------------------------------
-- Table `CommunityDB`.`appears`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CommunityDB`.`appears` (
  `episode_id` INT NOT NULL,
  `char_id` INT NOT NULL,
  PRIMARY KEY (`episode_id`, `char_id`),
  CONSTRAINT `fk_episode_has_character_episode1`
    FOREIGN KEY (`episode_id`)
    REFERENCES `CommunityDB`.`episode` (`episode_Id`),
  CONSTRAINT `fk_episode_has_character_character1`
```

```
    FOREIGN KEY (`char_id`)
    REFERENCES `CommunityDB`.`character` (`char_id`)
);


INSERT INTO `appears` (`episode_id`, `char_id`)
VALUES
(101, 1001),
(101, 1002),
(101, 1003),
(101, 1004),
(101, 1005),
(101, 1006),
(101, 1007),
(101, 1008),
(101, 1009),

(212, 1001),
(212, 1002),
(212, 1003),
(212, 1004),
(212, 1005),
(212, 1006),
(212, 1007),
(212, 1008),
(212, 1010),
(212, 1011),

(321, 1001),
(321, 1002),
(321, 1003),
(321, 1004),
(321, 1005),
(321, 1006),
(321, 1007),
(321, 1009),
(321, 1010),

(408, 1001),
(408, 1002),
```

```sql
(408, 1003),
(408, 1004),
(408, 1005),
(408, 1006),
(408, 1007),
(408, 1009),
(408, 1010),

(510, 1001),
(510, 1002),
(510, 1003),
(510, 1004),
(510, 1005),
(510, 1009),
(510, 1010),
(510, 1012),

(609, 1001),
(609, 1002),
(609, 1003),
(609, 1005),
(609, 1009),
(609, 1010),
(609, 1013);
COMMIT;

-- -----------------------------------------------------
-- Table `CommunityDB`.`rating`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CommunityDB`.`rating` (
  `rating_id` INT NOT NULL,
  `reviews_no` INT NOT NULL,
  `stars_no` FLOAT NOT NULL,
  `episode_id` INT NOT NULL,
  PRIMARY KEY (`rating_id`),
  CONSTRAINT `fk_rating_episode1`
    FOREIGN KEY (`episode_id`)
    REFERENCES `CommunityDB`.`episode` (`episode_Id`)
  );
```

```
 INSERT INTO `rating` (`rating_id`, `reviews_no`, `stars_no`, `episode_id`)
 VALUES
 (5101, 3926, 7.7, 101),
 (5212, 2549, 8.0, 212),
 (5321, 2977, 9.1, 321),
 (5408, 2410, 7.9, 408),
 (5510, 2326, 8.5, 510),
 (5609, 1878, 7.7, 609);
 COMMIT;

-- -----------------------------------------------------
-- Table `CommunityDB`.`production_staff`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CommunityDB`.`production_staff` (
 `staff_id` INT NOT NULL,
 `staff_fname` VARCHAR(45) NULL,
 `staff_lname` VARCHAR(45) NULL,
 PRIMARY KEY (`staff_id`)
);

INSERT INTO `production_staff` (`staff_id`, `staff_fname`, `staff_lname`)
VALUES
(3001, 'Anthony', 'Russo'),
(3002, 'Joe', 'Russo'),
(3003, 'Dan', 'Harmon'),
(3004, 'Emily', 'Cutler'),
(3005, 'Jay', 'Chandrasekhar'),
(3006, 'Matt', 'Fusfeld'),
(3007, 'Alex', 'Cuthbertson'),
(3008, 'Tristram', 'Shapeero'),
(3009, 'Jack', 'Kukoda'),
(3010, 'Tim', 'Saccardo'),
(3011, 'Matt', 'Roller'),
(3012, 'Rob', 'Schrab'),
(3013, 'Ryan', 'Ridley');
COMMIT;


-- -----------------------------------------------------
-- Table `CommunityDB`.`staff_role`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CommunityDB`.`staff_role` (
  `staff_role` ENUM('director', 'writer') NOT NULL,
  `staff_id` INT NOT NULL,
  PRIMARY KEY (`staff_role`, `staff_id`),
  CONSTRAINT `fk_staff_role_production_staff1`
    FOREIGN KEY (`staff_id`)
    REFERENCES `CommunityDB`.`production_staff` (`staff_id`)
  );

  INSERT INTO `staff_role` (`staff_role`, `staff_id`)
VALUES
('Director', '3001'),
('Director', '3002'),
('Writer', '3003'),
('Writer', '3004'),
('Director', '3005'),
('Writer', '3006'),
('Writer', '3007'),
('Director', '3008'),
('Writer', '3009'),
('Writer', '3010'),
('Writer', '3011'),
('Director', '3012'),
('Writer', '3013');
COMMIT;


-- -----------------------------------------------------
-- Table `CommunityDB`.`developed`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CommunityDB`.`developed` (
  `staff_id` INT NOT NULL,
  `episode_id` INT NOT NULL,
  PRIMARY KEY (`staff_id`, `episode_id`),
  CONSTRAINT `fk_production_staff_has_episode_production_staff1`
    FOREIGN KEY (`staff_id`)
    REFERENCES `CommunityDB`.`production_staff` (`staff_id`),
  CONSTRAINT `fk_production_staff_has_episode_episode1`
    FOREIGN KEY (`episode_id`)
    REFERENCES `CommunityDB`.`episode` (`episode_Id`)
  );

  INSERT INTO `developed` (`staff_id`, `episode_id`)
  VALUES
  (3001, 101),
  (3002, 101),
  (3003, 101),
  (3001, 212),
  (3003, 212),
  (3004, 212),
  (3005, 321),
  (3003, 321),
  (3006, 321),
  (3007, 321),
  (3008, 408),
  (3003, 408),
  (3009, 408),
  (3010, 408),
  (3002, 510),
  (3003, 510),
  (3011, 510),
  (3012, 609),
  (3003, 609),
  (3013, 609);
  COMMIT;

  DROP VIEW IF EXISTS `actors_character_on_each_episode`;
  CREATE VIEW `actors_character_on_each_episode`(`Actor Last name`,
`Character first name`,
  `Character last name`, `episode title`)
  AS SELECT `actor`.`actor_lname`,`character`.`char_fname`,
`character`.`char_lname`, `episode`.`title`
  FROM `actor`, `character`,`episode`, `appears`
  WHERE `appears`.`episode_id` = `episode`.`episode_id` AND
  `appears`.`char_id` = `character`.`char_id` AND `character`.`char_id` =
`actor`.`char_id`

CREATE TRIGGER `check_if_null`
BEFORE INSERT ON `character`
```

```
FOR EACH ROW
BEGIN
IF NEW.occupation IS NULL THEN
SET NEW.occupation = 'occupation not defined yet';
END IF;
END

CREATE TRIGGER `round_stars`
BEFORE INSERT ON `rating`
FOR EACH ROW
BEGIN
SET NEW.stars_no = round(NEW.stars_no);
END
```