

Name: Muhammad Rvail Naveed

Student No: 17321983

Database Design Report

This database is focused on the show “Breaking Bad” created by Vince Gilligan. The database consists of 7 tables. Characters, Cast, Episodes, Seasons, Writers, Locations and Groups, all meant to illustrate an important part of the tv show.

The Characters table consists of characters who each have fields to indicate their id, name, who they’re played by (entry in the cast table), their alias and their current status (DEAD or ALIVE).

Each cast member has an ssn, name and a foreign key to indicate which episode they first appeared in.

An episode has a unique episode\_id with the first 2 digits indicating the episode and the last digit indicating season e.g. 011 means episode 1 season 1. Along with this each entry also contains info about what season it belongs to, original air date and who wrote it.

Seasons have a unique season number, start date, end date and the number of episodes contained in that season.

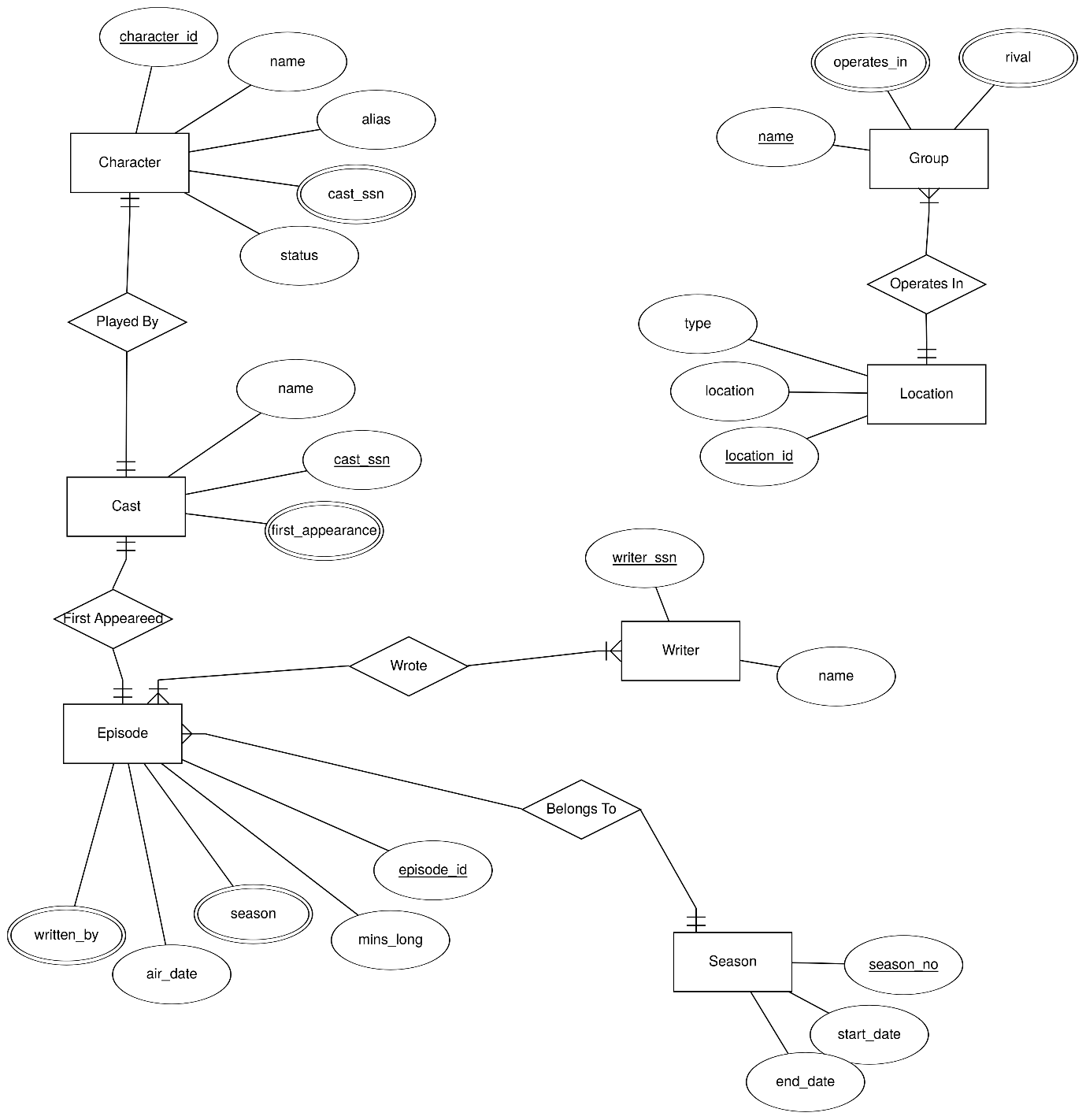
Every writer has a writer\_ssn and name.

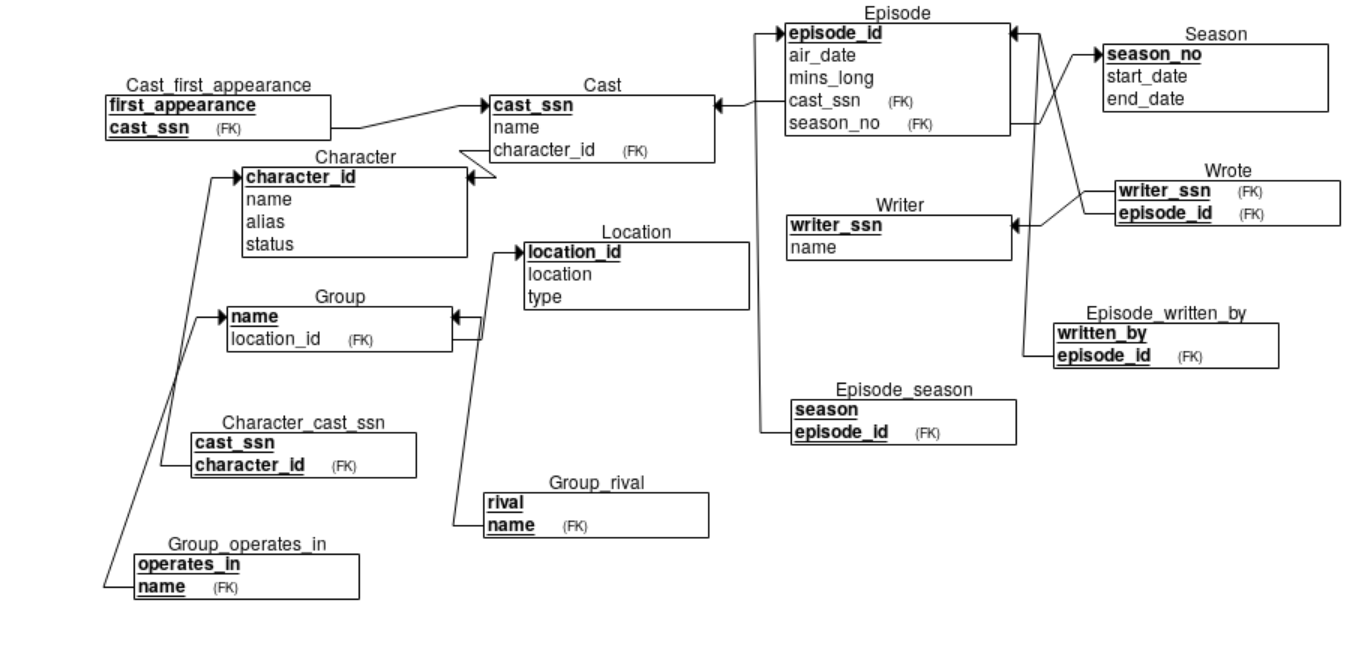
Locations have a six-digit id and name.

Groups have a unique name, what type of group they are and a reference to what location they operate in.

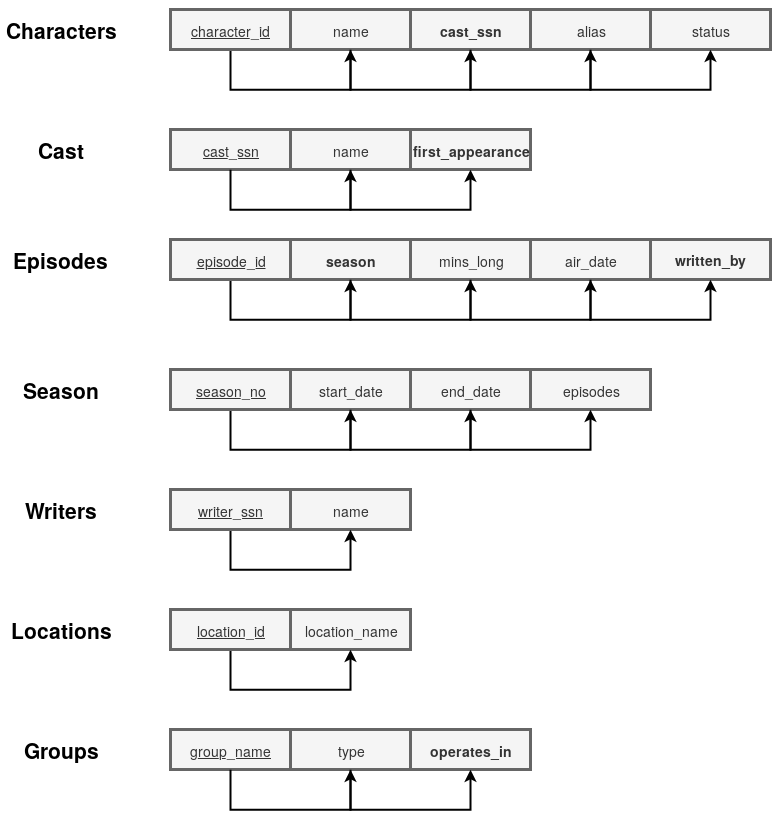
The following report demonstrates my methodologies, accompanied by Entity Relationship diagrams, Functional Dependency Diagram as well as a Relational Schema.

**Entity Relationship Diagram**



**Relational Schema**

**Functional Dependency Diagram**



* Attributes underlined indicate Primary Keys and attributes in **bold** are Foreign Keys.

I had to change my original design as I originally had the “Cast” table have a foreign key in the “Characters” table and then in turn the “Characters” table had a foreign key in the “Cast” table. This created a circular relationship which is not good database design as it can lead to updation/deletion anomalies. Changes in one table would need to be reflected in the other table. To combat this, I moved the “first\_appearance” attribute from the Characters table to the “Cast” table. This way I had Characters have a foreign key of Cast and Cast have a foreign key of Episode. The tables Groups and Locations were one tale before, however this introduced data redundancy as multiple groups could operate in one location. This violated normal form, so they were separated into two tables. Otherwise the database was fully normalized.

**Implicit Constraints**

**Characters**

Primary Key(AUTO\_INCREMENT=2130):

`character\_id` int(4) NOT NULL

Foreign Key(s):

FOREIGN KEY (`cast\_ssn`) REFERENCES `Cast` (`cast\_ssn`)

**Cast**

Primary Key(AUTO\_INCREMENT=3472):

`cast\_ssn` int(4) NOT NULL

Foreign Key(s): FOREIGN KEY (`first\_appearance`) REFERENCES `Episodes` (`episode\_id`)

**Episodes**

Primary Key:

`episode\_id` int(3) NOT NULL

Foreign Key(s):

FOREIGN KEY (`season`) REFERENCES `Seasons` (`season\_no`), FOREIGN KEY (`written\_by`) REFERENCES `Writers` (`writer\_ssn`)

**Seasons**

Primary Key:

`season\_no` int(1) NOT NULL

**Writers**

Primary Key(AUTO\_INCREMENT=4361):

`writer\_ssn` int(4) NOT NULL

**Locations**

Primary Key(AUTO\_INCREMENT=770421):

`location\_id` int(6) NOT NULL

**Groups**

Primary Key:

`group\_name` varchar(30)

Foreign Key(s):

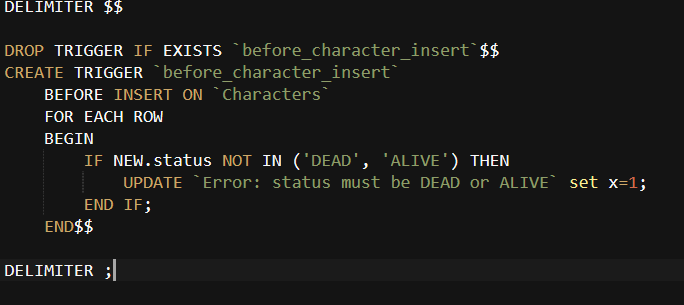
FOREIGN KEY (`operates\_in`) REFERENCES `Locations` (`location\_id`)

**Semantic Constraints**

For semantic constraints I implemented three triggers as follows:

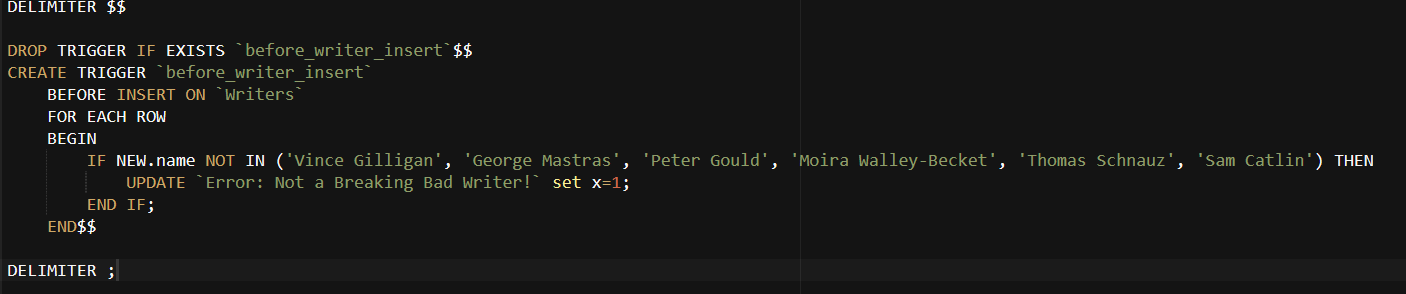
**Trigger: Before Character Insert**

Before a new character can be inserted, this trigger checks if their status is one of either “DEAD” or “ALIVE”.



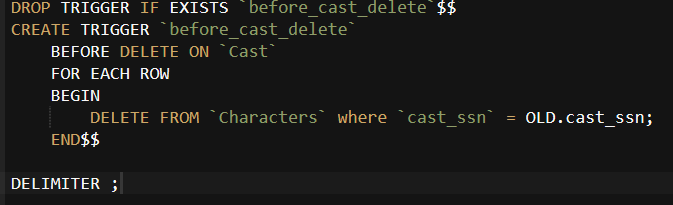
**Trigger: Before Writer Insert**

This trigger checks if the writer being inserted into the table is one of the breaking bad writers. If that is the case, they will be added successfully with an auto-incrementing primary key “writer\_ssn”, if not then an error will be thrown indicating that the writer is not a breaking bad writer.



**Trigger: Before Cast Delete**

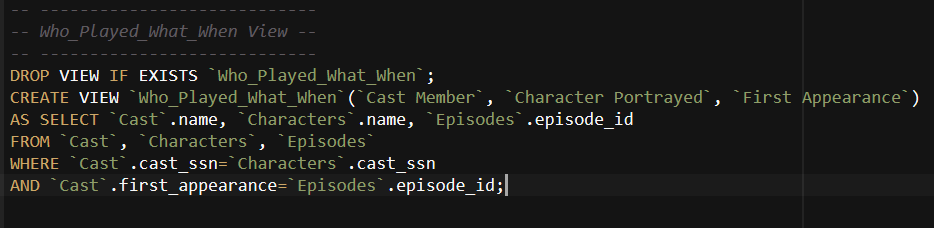
Before deletion of a cast member, their respective character is deleted from the Characters table. This is to avoid a character having a foreign key to an entry in the Cast table that doesn’t exist.



**Views**

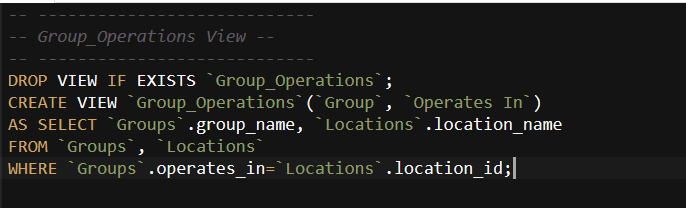
I described two views:

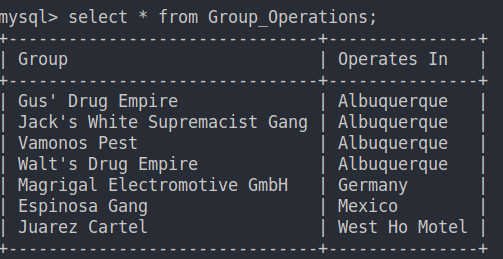
* One which would show the cast member who played a certain character and the date they first appeared on Breaking Bad.





* Another that described what group operated in what location.

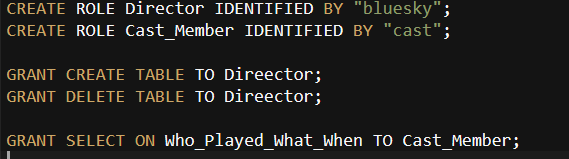




**Security**

Some simple security could be implemented by making roles and only allowing the show runner/director to make changes to the tables. This is obvious as cast members etc are not allowed to create new aspects of the show!

This could be done in MySQL like so:



**Appendix**

SET FOREIGN\_KEY\_CHECKS = 0;

*-- ---------------------*

*-- Characters Table --*

*-- ---------------------*

DROP TABLE IF EXISTS `Characters`;

CREATE TABLE `Characters`(

    `character\_id` int(4) NOT NULL PRIMARY KEY AUTO\_INCREMENT,

    `name` varchar(50) NOT NULL,

    `cast\_ssn` int(4) NOT NULL,

    `alias` varchar(20),

    `status` varchar(7) NOT NULL,

    CONSTRAINT FK\_cast\_ssn FOREIGN KEY (`cast\_ssn`) REFERENCES `Cast` (`cast\_ssn`)

);

*-- ---------------------*

*-- Init 'Characters' --*

*-- ---------------------*

BEGIN;

ALTER TABLE `Characters` AUTO\_INCREMENT=2130;

INSERT INTO `Characters` (`name`, `cast\_ssn`, `alias`, `status`)

VALUES

    ('Walter White', 3472, 'Heisenberg', 'DEAD'),

    ('Jesse Pinkman', 3473, "Cap'n Cook", 'ALIVE'),

    ('Skyler White', 3474, 'Sky', 'ALIVE'),

    ('Hank Schrader', 3475, NULL, 'DEAD'),

    ('Gustavo Fring', 3476, 'The Chicken Man', 'DEAD'),

    ('Mike Ehrmantraut', 3477, NULL, 'DEAD'),

    ('Walter White Jr.', 3478, 'Flynn', 'ALIVE'),

    ('Saul Goodman', 3479, "Slippin' Jimmy", 'ALIVE'),

    ('Marie Schrader', 3480, NULL, 'ALIVE'),

    ('Todd Alquist', 3481, NULL, 'ALIVE'),

    ('Lydia Rodart-Quayle', 3482, NULL, 'DEAD'),

    ('Tuco Salamanca', 3483, "Tuco", 'DEAD'),

    ('Kuby', 3484, NULL, 'ALIVE'),

    ('Jane Margolis', 3485, NULL, 'ALIVE');

COMMIT;

*-- ---------------------*

*-- Cast Table --*

*-- ---------------------*

DROP TABLE IF EXISTS `Cast`;

CREATE TABLE `Cast`(

    `cast\_ssn` int(4) NOT NULL PRIMARY KEY AUTO\_INCREMENT,

    `name` varchar(50) NOT NULL,

    `first\_appearance` int(3) NOT NULL,

    CONSTRAINT FK\_appearance\_id FOREIGN KEY (`first\_appearance`) REFERENCES `Episodes` (`episode\_id`)

);

BEGIN;

ALTER TABLE `Cast` AUTO\_INCREMENT=3472;

INSERT INTO `Cast` (`name`, `first\_appearance`)

VALUES

    ('Bryan Cranston', 011),

    ('Aaron Paul', 011),

    ('Anna Gunn', 011),

    ('Dean Norris', 011),

    ('Giancarlo Esposito', 112),

    ('Jonathon Banks', 132),

    ('RJ Mitte', 011),

    ('Bob Odenkirk', 082),

    ('Betsy Brandt', 011),

    ('Jesse Plemons', 035),

    ('Laura Fraser', 025),

    ('Raymond Cruz', 061),

    ('Bill Burr', 034),

    ('Krysten Ritter', 052);

COMMIT;

*-- ---------------------*

*-- Episodes Table --*

*-- ---------------------*

DROP TABLE IF EXISTS `Episodes`;

CREATE TABLE `Episodes`(

    `episode\_id` int(3) NOT NULL PRIMARY KEY,

    `season` int(1) NOT NULL,

    `mins\_long` int(2) NOT NULL,

    `air\_date` DATE NOT NULL,

    `written\_by` int(4) NOT NULL,

    CONSTRAINT FK\_season\_id FOREIGN KEY (`season`) REFERENCES `Seasons` (`season\_no`),

    CONSTRAINT FK\_writer\_id FOREIGN KEY (`written\_by`) REFERENCES `Writers` (`writer\_ssn`)

);

*-- ---------------------*

*-- Init 'Episodes' --*

*-- ---------------------*

BEGIN;

INSERT INTO `Episodes`

VALUES

    (011, 1, 58, '2008-01-20', 4361),

    (112, 2, 47, '2009-05-17', 4362),

    (082, 2, 47, '2009-04-26', 4363),

    (132, 2, 47, '2009-05-31', 4361),

    (035, 5, 48, '2012-07-29', 4363),

    (025, 5, 49, '2012-07-22', 4361),

    (061, 1, 46, '2008-03-02', 4362),

    (034, 4, 47, '2011-07-31', 4366),

    (052, 2, 49, '2009-04-05', 4364);

COMMIT;

*-- ---------------------*

*-- Seasons Table --*

*-- ---------------------*

DROP TABLE IF EXISTS `Seasons`;

CREATE TABLE `Seasons`(

    `season\_no` int(1) NOT NULL PRIMARY KEY AUTO\_INCREMENT,

    `start\_date` date NOT NULL,

    `end\_date` date NOT NULL,

    `episodes` int(2) NOT NULL

);

*-- ---------------------*

*-- Init 'Seasons' --*

*-- ---------------------*

BEGIN;

INSERT INTO `Seasons` (`start\_date`, `end\_date`, `episodes`)

VALUES

    ('2008-01-20', '2008-03-09', 7),

    ('2009-03-08', '2009-05-31', 13),

    ('2010-03-21', '2010-06-13', 13),

    ('2011-07-17', '2011-10-09', 13),

    ('2012-07-15', '2013-09-29', 16);

COMMIT;

*-- ---------------------*

*-- Writers Table --*

*-- ---------------------*

DROP TABLE IF EXISTS `Writers`;

CREATE TABLE `Writers`(

    `writer\_ssn` int(4) NOT NULL PRIMARY KEY AUTO\_INCREMENT,

    `name` varchar(20) NOT NULL

);

*-- ---------------------*

*-- Init 'Writers' --*

*-- ---------------------*

BEGIN;

ALTER TABLE `Writers` AUTO\_INCREMENT=4361;

INSERT INTO `Writers` (`name`)

VALUES

    ('Vince Gilligan'), *-- 4361*

    ('George Mastras'), *-- 4362*

    ('Peter Gould'), *-- 4363*

    ('Moira Walley-Becket'), *-- 4364*

    ('Thomas Schnauz'), *-- 4365*

    ('Sam Catlin'); *-- 4366*

COMMIT;

*-- ---------------------*

*-- Locations Table --*

*-- ---------------------*

DROP TABLE IF EXISTS `Locations`;

CREATE TABLE `Locations`(

    `location\_id` int(6) NOT NULL PRIMARY KEY AUTO\_INCREMENT,

    `location\_name` varchar(50) NOT NULL

);

*-- ---------------------*

*-- Init 'Locations' --*

*-- ---------------------*

BEGIN;

ALTER TABLE `Locations` AUTO\_INCREMENT=770421;

INSERT INTO `Locations` (`location\_name`)

VALUES

    ('Albuquerque'),

    ('Germany'),

    ('Mexico'),

    ('West Ho Motel'),

    ('Superlab'),

    ('Best Quality Vacuum'),

    ('White Residence'),

    ("Jesse's House"),

    ("Saul Goodman's Office");

COMMIT;

DROP TABLE IF EXISTS `Groups`;

CREATE TABLE `Groups`(

    `group\_name` varchar(30) NOT NULL PRIMARY KEY,

    `type` varchar(30) NOT NULL,

    `operates\_in` int(6) NOT NULL,

    CONSTRAINT FK\_location\_id FOREIGN KEY (`operates\_in`) REFERENCES `Locations` (`location\_id`)

);

BEGIN;

INSERT INTO `Groups`

VALUES

    ("Walt's Drug Empire", 'Drug Empire', 770421),

    ("Gus' Drug Empire", 'Drug Empire', 770421),

    ("Vamonos Pest", 'Pest Control Company', 770421),

    ("Magrigal Electromotive GmbH", 'Shipping Conglomerate', 770422),

    ("Jack's White Supremacist Gang", 'White power gang', 770421),

    ("Espinosa Gang", 'Drug Distributor', 770423),

    ("Juarez Cartel", 'Drug Cartel', 770424);

COMMIT;

*-- -------------------------------*

*-- Trigger for Character INSERT --*

*-- -------------------------------*

DELIMITER $$

DROP TRIGGER IF EXISTS `before\_character\_insert`$$

CREATE TRIGGER `before\_character\_insert`

    BEFORE INSERT ON `Characters`

    FOR EACH ROW

    BEGIN

        IF NEW.status NOT IN ('DEAD', 'ALIVE') THEN

            UPDATE `Error: status must be DEAD or ALIVE` set x=1;

        END IF;

    END$$

DELIMITER ;

*-- -------------------------------*

*-- Trigger for Writer INSERT --*

*-- -------------------------------*

DELIMITER $$

DROP TRIGGER IF EXISTS `before\_writer\_insert`$$

CREATE TRIGGER `before\_writer\_insert`

    BEFORE INSERT ON `Writers`

    FOR EACH ROW

    BEGIN

        IF NEW.name NOT IN ('Vince Gilligan', 'George Mastras', 'Peter Gould', 'Moira Walley-Becket', 'Thomas Schnauz', 'Sam Catlin') THEN

            UPDATE `Error: Not a Breaking Bad Writer!` set x=1;

        END IF;

    END$$

DELIMITER ;

*-- -------------------------------*

*-- Trigger for Cast DELETE --*

*-- -------------------------------*

DELIMITER $$

DROP TRIGGER IF EXISTS `before\_cast\_delete`$$

CREATE TRIGGER `before\_cast\_delete`

    BEFORE DELETE ON `Cast`

    FOR EACH ROW

    BEGIN

        DELETE FROM `Characters` where `cast\_ssn` = OLD.cast\_ssn;

    END$$

DELIMITER ;

*-- ----------------------------*

*-- Who\_Played\_What\_When View --*

*-- ----------------------------*

DROP VIEW IF EXISTS `Who\_Played\_What\_When`;

CREATE VIEW `Who\_Played\_What\_When`(`Cast Member`, `Character Portrayed`, `First Appearance`)

AS SELECT `Cast`.name, `Characters`.name, `Episodes`.episode\_id

FROM `Cast`, `Characters`, `Episodes`

WHERE `Cast`.cast\_ssn=`Characters`.cast\_ssn

AND `Cast`.first\_appearance=`Episodes`.episode\_id;

*-- ----------------------------*

*-- Group\_Operations View --*

*-- ----------------------------*

DROP VIEW IF EXISTS `Group\_Operations`;

CREATE VIEW `Group\_Operations`(`Group`, `Operates In`)

AS SELECT `Groups`.group\_name, `Locations`.location\_name

FROM `Groups`, `Locations`

WHERE `Groups`.operates\_in=`Locations`.location\_id;