Portfolio Project:

Johnson Video Store Database

Cecil Kitch

Colorado State University – Global Campus

**Preface**

Given a brief summary of the Johnson Video Store business operation, a task was assigned to develop a PostgreSQL database to automate record keeping of inventory and business transactions. The final structure of the operational database consists of fourteen entity tables and four association tables required to achieve many-to-many relationships. All tables were then successfully populated with an adequate quantity of sample data required to meet testing requirements, ensuring that all business operations were supported.

The database life cycle describes the stages involved in development of an implementation of a database. The process begins by gathering business requirements by examining the summary provided in Module 8. Entities, attributes, and relationships were discovered, and relationship sentence pairs were developed.



Figure 1 – Entity Relationship Diagram of Johnson Video Store Database.

The field of conceptual modeling began nearly forty-years ago and continues to evolve, but Chen’s seminal paper delivered the entity relationship model that has become virtually synonymous with database design (Storey, Trujillo, & Liddle, 2015). Conceptual Data Modeling can be performed to produce an Entity Relationship Diagram (ERD). Depicted in Figure 1, the Johnson Video Store Database ERD describes all the entities (rectangles) and relationships (lines) discovered while gathering business requirements. Attributes or fields are also enumerated with an appropriate chosen datatype; primary and foreign keys are also shown. Entities housing foreign keys of child entities may be referred to as the owning entity. Dotted lines were used to indicated non-mandatory relationships, conversely solid lines represent mandatory or defining relationships. For example, the lines between movie\_has\_category, indicates in order for the relationship to exist, foreign keys are required to be present from both the movie and category entities.  
 The next phase in the design process is the conversion of the ERD to a logical design by constructing tables, columns, primary keys, foreign keys and their constraints. Normalization also occurs during this phase, removing any redundancies producing. During this phase, a decision was made to factor out address details from the customer, staff, and distributor tables into an additional address table. This prevented repetition of attributes in aforenoted entities and promoted flexibility for any evolution that might be required by unanticipated future business requirements.

The third phase, the distributed design phase, is virtually non-existent with regards to this project as it is small in scope and not distributed. However, this phase would have been more apparent had Johnson Video Store had multiple stores. The design is normalized to the extent that a future implementation supporting a store entity could be related to the staff table with the inserting a foreign key store id. There would be no effect upon any of the other data structures.

The final phase, the physical database design, also does not have much to elaborate upon. Additional indexes were not warranted beyond the default generated by PostGRES for primary keys. With regards to concern for performance, in terms of response time, while hard to measure, it is unlikely that multiple users will be performing operations on the system concurrently. In terms of conservation of disk space, there is very little redundancy. Because the data was already well normalized during the conceptual modeling phase, few adjustments were made. However, a missed requirement was discovered during data population, and required the addition of the account number attribute in the customer entity.

**Data Definition Language (DDL) – Create Schema Script**

DROP DATABASE IF EXISTS jvdb;

CREATE DATABASE jvdb

    WITH OWNER = postgres

    ENCODING = 'UTF8'

    CONNECTION LIMIT = -1;

\connect jvdb

CREATE TABLE public.language (

  id SERIAL,

  name VARCHAR(25) NOT NULL,

  PRIMARY KEY (id));

CREATE TYPE rating AS ENUM ('G', 'PG', 'PG-13', 'R', 'NC-17', 'Not Rated');

CREATE TYPE mediatype AS ENUM ('DVD', 'VIDEO');  
CREATE TYPE mediacondition

AS ENUM ('NEW', 'GOOD', 'FAIR', 'REFURB', 'DAMAGED');

CREATE TABLE public.address (

  id SERIAL,

  street VARCHAR(50) NOT NULL,

  city VARCHAR(50) NOT NULL,

  state\_province VARCHAR(50) NOT NULL,

  postal\_code VARCHAR(10) NULL,

  phone VARCHAR(20) NOT NULL,

  PRIMARY KEY (id));

CREATE TABLE public.movie (

  id SERIAL,

  title VARCHAR(255) NOT NULL,

  description TEXT NULL,

  release\_year INT NULL,

  length SMALLINT NULL,

  rating rating NULL,

  language\_id INT NOT NULL,

  PRIMARY KEY (id),

  CONSTRAINT fk\_movie\_language1

    FOREIGN KEY (language\_id)

    REFERENCES public.language (id)

    ON DELETE RESTRICT

    ON UPDATE NO ACTION);

CREATE TABLE public.director (

  id SERIAL,

  first\_name VARCHAR(45) NOT NULL,

  last\_name VARCHAR(45) NOT NULL,

  PRIMARY KEY (id));

CREATE TABLE public.actor (

  id SERIAL,

  first\_name VARCHAR(45) NOT NULL,

  last\_name VARCHAR(45) NOT NULL,

  PRIMARY KEY (id));

CREATE TABLE public.movie\_has\_actor (

  movie\_id INT NOT NULL,

  actor\_id INT NOT NULL,

  PRIMARY KEY (movie\_id, actor\_id),

  CONSTRAINT fk\_movie\_has\_actor\_movie

    FOREIGN KEY (movie\_id)

    REFERENCES public.movie (id)

    ON DELETE CASCADE

    ON UPDATE NO ACTION,

  CONSTRAINT fk\_movie\_has\_actor\_actor1

    FOREIGN KEY (actor\_id)

    REFERENCES public.actor (id)

    ON DELETE CASCADE

    ON UPDATE NO ACTION);

CREATE TABLE public.movie\_has\_director (

  movie\_id INT NOT NULL,

  director\_id INT NOT NULL,

  PRIMARY KEY (movie\_id, director\_id),

  CONSTRAINT fk\_movie\_has\_director\_movie1

    FOREIGN KEY (movie\_id)

    REFERENCES public.movie (id)

    ON DELETE CASCADE

    ON UPDATE NO ACTION,

  CONSTRAINT fk\_movie\_has\_director\_director1

    FOREIGN KEY (director\_id)

    REFERENCES public.director (id)

    ON DELETE RESTRICT

    ON UPDATE NO ACTION);

CREATE TABLE public.category (

  id SERIAL,

  name VARCHAR(25) NULL,

  PRIMARY KEY (id));

CREATE TABLE public.movie\_has\_category (

  movie\_id INT NOT NULL,

  category\_id INT NOT NULL,

  PRIMARY KEY (movie\_id, category\_id),

  CONSTRAINT fk\_movie\_has\_category\_movie1

    FOREIGN KEY (movie\_id)

    REFERENCES public.movie (id)

    ON DELETE CASCADE

    ON UPDATE NO ACTION,

  CONSTRAINT fk\_movie\_has\_category\_category1

    FOREIGN KEY (category\_id)

    REFERENCES public.category (id)

    ON DELETE CASCADE

    ON UPDATE NO ACTION);

CREATE TABLE public.distributor (

  id SERIAL,

  name VARCHAR(255) NOT NULL,

  established\_date TIMESTAMP NOT NULL,

  address\_id INT NULL,

  PRIMARY KEY (id),

  CONSTRAINT fk\_distributor\_address1

    FOREIGN KEY (address\_id)

    REFERENCES public.address (id)

    ON DELETE SET NULL

    ON UPDATE NO ACTION);

CREATE TABLE public.catalog\_item (

  id SERIAL,

  distributor\_id INT NOT NULL,

  title VARCHAR(255) NOT NULL,

  serial\_number INT NOT NULL,

  wholesale\_cost NUMERIC(5,2) NOT NULL,

  movie\_id INT NULL,

  create\_date TIMESTAMP NOT NULL,

  media\_type mediatype NOT NULL,

  update\_date TIMESTAMP NULL,

  PRIMARY KEY (id),

  CONSTRAINT fk\_distributor\_item\_distributor1

    FOREIGN KEY (distributor\_id)

    REFERENCES public.distributor (id)

    ON DELETE CASCADE

    ON UPDATE NO ACTION,

  CONSTRAINT fk\_distributor\_item\_movie1

    FOREIGN KEY (movie\_id)

    REFERENCES public.movie (id)

    ON DELETE SET NULL

    ON UPDATE NO ACTION);

CREATE TABLE public.inventory\_item (

  id SERIAL,

  movie\_id INT NULL,

  rental\_rate NUMERIC(5,2) NOT NULL,

  rental\_duration VARCHAR(45) NOT NULL,

  replacement\_cost NUMERIC(4,2) NOT NULL,

  media\_type mediatype NOT NULL,

  media\_condition mediacondition NULL,

  create\_date TIMESTAMP NOT NULL,

  update\_date TIMESTAMP NULL,

  distributor\_catalog\_id INT NOT NULL,

  PRIMARY KEY (id),

  CONSTRAINT fk\_inventory\_item\_movie1

    FOREIGN KEY (movie\_id)

    REFERENCES public.movie (id)

    ON DELETE SET NULL

    ON UPDATE NO ACTION,

  CONSTRAINT fk\_inventory\_item\_distributor\_item1

    FOREIGN KEY (distributor\_catalog\_id)

    REFERENCES public.catalog\_item (id)

    ON DELETE SET NULL

    ON UPDATE NO ACTION);

CREATE TABLE public.customer (

  id SERIAL,

  account\_number CHAR(7),

  first\_name VARCHAR(45) NOT NULL,

  last\_name VARCHAR(45) NOT NULL,

  email VARCHAR(50) NULL,

  active SMALLINT NULL,

  create\_date TIMESTAMP NOT NULL,

  update\_date TIMESTAMP NULL,

  address\_id INT NOT NULL,

  PRIMARY KEY (id),

  CONSTRAINT fk\_customer\_address1

    FOREIGN KEY (address\_id)

    REFERENCES public.address (id)

    ON DELETE NO ACTION

    ON UPDATE NO ACTION);

CREATE TABLE public.staff (

  id SERIAL,

  first\_name VARCHAR(45) NOT NULL,

  last\_name VARCHAR(45) NOT NULL,

  email VARCHAR(50) NULL,

  active SMALLINT NOT NULL,

  username VARCHAR(16) NOT NULL,

  password VARCHAR(40) NULL,

  address\_id INT NOT NULL,

  PRIMARY KEY (id),

  CONSTRAINT fk\_staff\_address1

    FOREIGN KEY (address\_id)

    REFERENCES public.address (id)

    ON DELETE NO ACTION

    ON UPDATE NO ACTION);

CREATE TABLE public.academy\_award (

  id SERIAL,

  award\_category VARCHAR(50) NOT NULL,

  PRIMARY KEY (id));

CREATE TABLE public.rental (

  id SERIAL,

  inventory\_id INT NULL,

  rental\_date TIMESTAMP NOT NULL,

  rental\_tax NUMERIC(4,2) NOT NULL,

  customer\_id INT NULL,

  return\_date TIMESTAMP NULL,

  staff\_id INT NULL,

  late\_fee NUMERIC(5,2) NULL,

  damage\_fee NUMERIC(4,2) NULL,

  rewind\_fee NUMERIC(4,2) NULL,

  PRIMARY KEY (id),

  CONSTRAINT fk\_rental\_inventory\_item1

    FOREIGN KEY (inventory\_id)

    REFERENCES public.inventory\_item (id)

    ON DELETE SET NULL

    ON UPDATE NO ACTION,

  CONSTRAINT fk\_rental\_customer1

    FOREIGN KEY (customer\_id)

    REFERENCES public.customer (id)

    ON DELETE SET NULL

    ON UPDATE NO ACTION,

  CONSTRAINT fk\_rental\_staff1

    FOREIGN KEY (staff\_id)

    REFERENCES public.staff (id)

    ON DELETE SET NULL

    ON UPDATE NO ACTION);

CREATE TABLE public.payment (

  id SERIAL,

  rental\_id INT NULL,

  amount NUMERIC(5,2) NOT NULL,

  payment\_date TIMESTAMP NOT NULL,

  PRIMARY KEY (id),

  CONSTRAINT fk\_payment\_rental1

    FOREIGN KEY (rental\_id)

    REFERENCES public.rental (id)

    ON DELETE RESTRICT

    ON UPDATE NO ACTION);

CREATE TABLE public.movie\_has\_academy\_award (

  movie\_id INT NOT NULL,

  academy\_award\_id INT NOT NULL,

  PRIMARY KEY (movie\_id, academy\_award\_id),

  CONSTRAINT fk\_movie\_has\_academy\_award\_movie1

    FOREIGN KEY (movie\_id)

    REFERENCES public.movie (id)

    ON DELETE CASCADE

    ON UPDATE NO ACTION,

  CONSTRAINT fk\_movie\_has\_academy\_award\_academy\_award1

    FOREIGN KEY (academy\_award\_id)

    REFERENCES public.academy\_award (id)

    ON DELETE NO ACTION

    ON UPDATE NO ACTION);

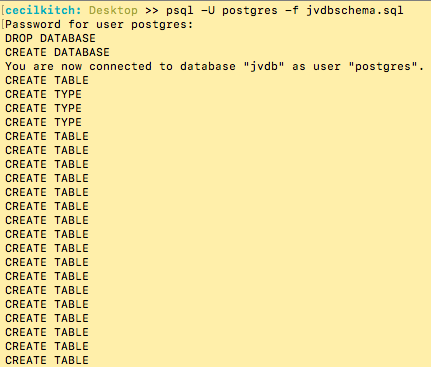


Figure 2 – Output of executing the schema creation script from file jvdbschema.sql.

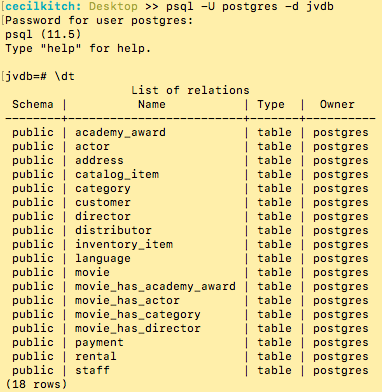
****

Figure 3 – List of all relations in jvdb.

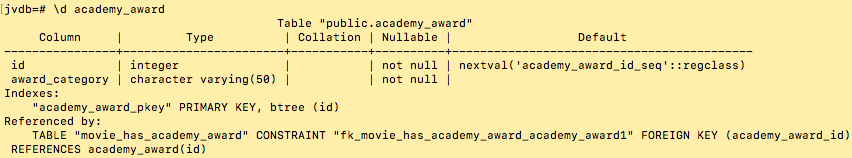


Figure 4 - Description of academy award table.: records academy award categories.

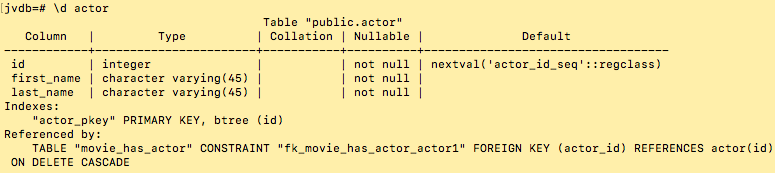


Figure 5 – Description of actor table: records actors/actress names referenced by a movie(s).

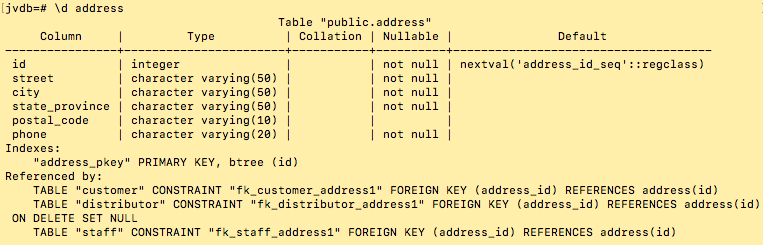


Figure 5 – Description of address table: records addresses of customers, distributors, and staff.

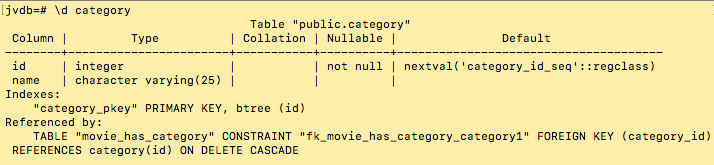


Figure 6 – Description of category table: records movie genres.

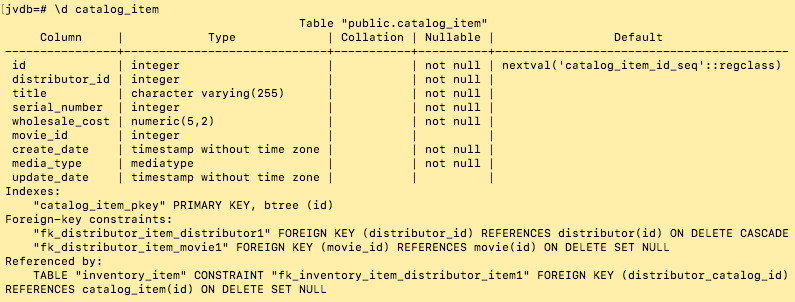


Figure 7 – Description of catalog item table, records distributor catalog data.



Figure 8 – Description of customer table, records customer account data.

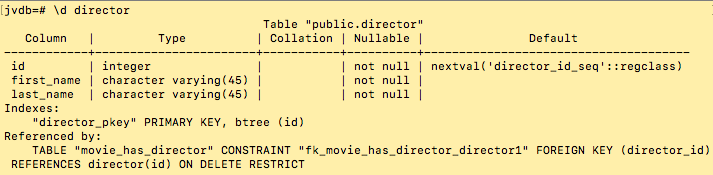


Figure 9 – Description of director table, records the name of movie directors.

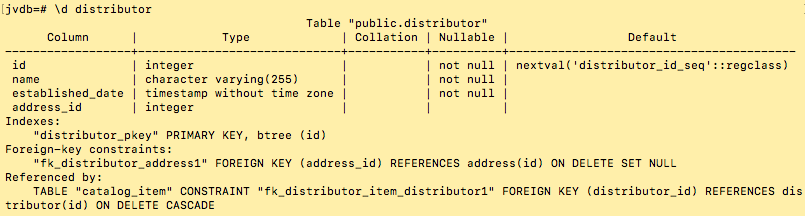
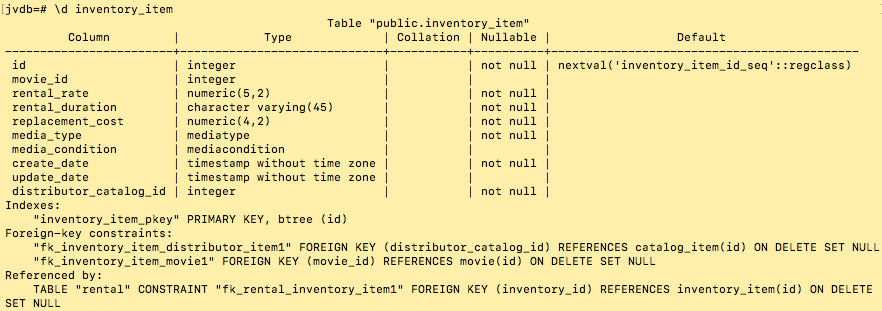


Figure 10 – Description of distributor table, records movie distributor relationship.

Figure 11 – Description of inventory item, inventory items relate a movie copy to catalog item.

****

Figure 12- Description of language table, records contain the language of a film.

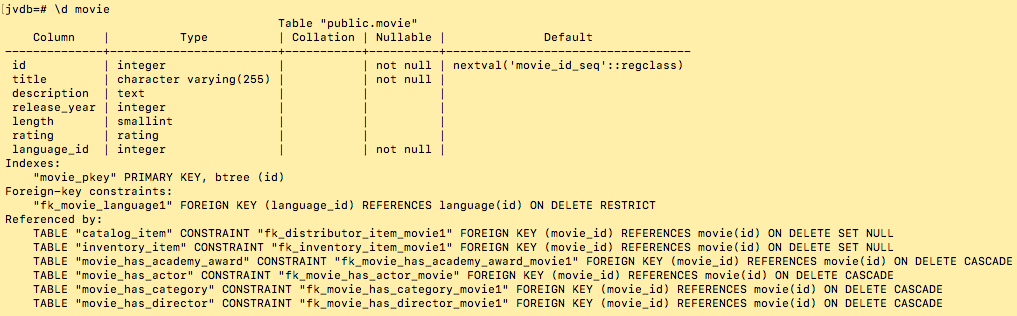
****

Figure 13 – Description of movie table, records contain details about a movie.

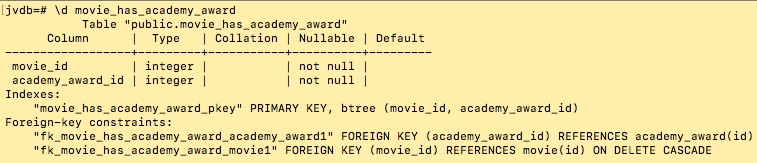


Figure 14- Description of movie\_has\_academy\_award, records contain only foreign keys, a movie id and a academy award id. Required by the M:M relationship.

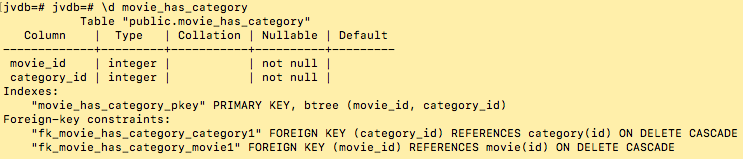


Figure 15- Description of movie\_has\_category, records contain only foreign keys, a movie id and a category id. Required by the M:M relationship.

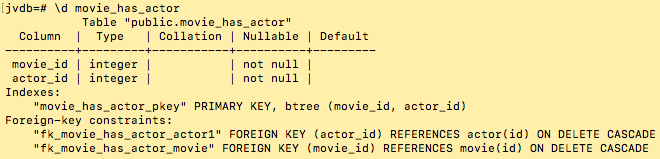


Figure 16- Description of movie\_has\_actor, records contain only foreign keys, a movie id and a actor id. Required by the M:M relationship.

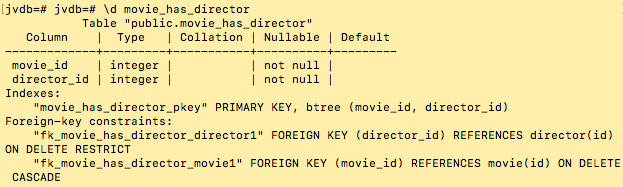


Figure 16- Description of movie\_has\_director, records contain only foreign keys, a movie id and a director id. Required by the M:M relationship.

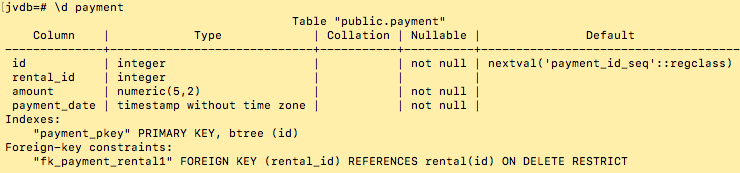


Figure 17- Description of payment table, records the data about payments related to a rental.

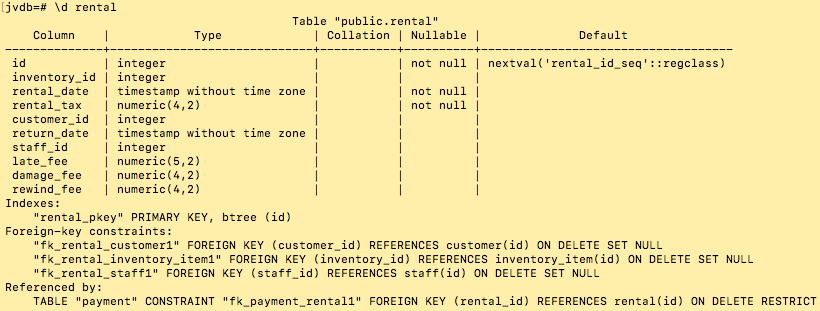


Figure 18 – Description of rental table, records the data of a rental transaction.

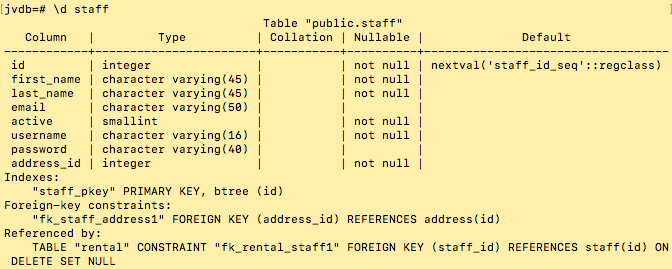


Figure 19 – Description of staff table, records data about video store employees.

**Data Manipulation Language(DML): Insert Statements Script**

INSERT INTO public.language VALUES (1, 'English');

INSERT INTO public.language VALUES (2, 'Japanese');

INSERT INTO public.language VALUES (3, 'French');

INSERT INTO public.language VALUES (4, 'Italian');

INSERT INTO public.language VALUES (5, 'German');

INSERT INTO public.movie VALUES (1, 'ANTITRUST TEEN', 'A Brilliant Epistle of a Composer And a Database Administrator', 1988, 150, 'PG', 1);

INSERT INTO public.movie VALUES (2, 'APOLLO TOMATOES', 'A Intrepid Documentary of a Boat And a Crocodile', 2005, 50, 'G', 1);

INSERT INTO public.movie VALUES (3, 'BARABELLA LIBERTY', 'A Amazing Documentary of a Secret Agent And a Astronaut', 2003, 86, 'NC-17', 1);

INSERT INTO public.movie VALUES (4, 'BIRD WHISPERER', 'A Fateful Yarn of a Mad Cow And a Waitress', 2018, 150, 'PG-13', 1);

INSERT INTO public.movie VALUES (5, 'CARIBBEAN INDEPENDENCE', 'A Unbelieveable Epistle of a Robot And a Husband', 1995, 94, 'R', 3);

INSERT INTO public.director VALUES (1, 'Sandra', 'Streep');

INSERT INTO public.director VALUES (2, 'Elvis', 'Davis');

INSERT INTO public.director VALUES (3, 'Lucille', 'Fawcett');

INSERT INTO public.director VALUES (4, 'Joseph', 'Paltrow');

INSERT INTO public.director VALUES (5, 'Jennifer', 'Marx');

INSERT INTO public.actor VALUES (1, 'Woody', 'Costner');

INSERT INTO public.actor VALUES (2, 'Scarlett', 'Hudson');

INSERT INTO public.actor VALUES (3, 'Cate', 'Streep');

INSERT INTO public.actor VALUES (4, 'Morgan', 'Bridges');

INSERT INTO public.actor VALUES (5, 'Angela', 'Keitel');

INSERT INTO public.movie\_has\_actor VALUES (1, 1);

INSERT INTO public.movie\_has\_actor VALUES (1, 2);

INSERT INTO public.movie\_has\_actor VALUES (1, 3);

INSERT INTO public.movie\_has\_actor VALUES (2, 1);

INSERT INTO public.movie\_has\_actor VALUES (2, 2);

INSERT INTO public.movie\_has\_actor VALUES (2, 3);

INSERT INTO public.movie\_has\_actor VALUES (2, 4);

INSERT INTO public.movie\_has\_actor VALUES (3, 4);

INSERT INTO public.movie\_has\_actor VALUES (3, 5);

INSERT INTO public.movie\_has\_actor VALUES (3, 3);

INSERT INTO public.movie\_has\_actor VALUES (4, 5);

INSERT INTO public.movie\_has\_actor VALUES (4, 2);

INSERT INTO public.movie\_has\_actor VALUES (4, 1);

INSERT INTO public.movie\_has\_actor VALUES (4, 4);

INSERT INTO public.movie\_has\_actor VALUES (5, 4);

INSERT INTO public.movie\_has\_actor VALUES (5, 1);

INSERT INTO public.movie\_has\_director VALUES (1, 2);

INSERT INTO public.movie\_has\_director VALUES (1, 4);

INSERT INTO public.movie\_has\_director VALUES (2, 5);

INSERT INTO public.movie\_has\_director VALUES (3, 1);

INSERT INTO public.movie\_has\_director VALUES (4, 2);

INSERT INTO public.movie\_has\_director VALUES (5, 3);

INSERT INTO public.category VALUES (1, 'Action');

INSERT INTO public.category VALUES (2, 'Sci-Fi');

INSERT INTO public.category VALUES (3, 'Drama');

INSERT INTO public.category VALUES (4, 'Documentary');

INSERT INTO public.category VALUES (5, 'Horror');

INSERT INTO public.movie\_has\_category VALUES (1, 3);

INSERT INTO public.movie\_has\_category VALUES (1, 1);

INSERT INTO public.movie\_has\_category VALUES (1, 2);

INSERT INTO public.movie\_has\_category VALUES (2, 1);

INSERT INTO public.movie\_has\_category VALUES (2, 4);

INSERT INTO public.movie\_has\_category VALUES (3, 5);

INSERT INTO public.movie\_has\_category VALUES (4, 1);

INSERT INTO public.movie\_has\_category VALUES (5, 3);

INSERT INTO public.address VALUES (1, '1111 Loja Avenue', 'Shawnee', 'Kansas', '79388', '440512153169');

INSERT INTO public.address VALUES (2, '28 MySQL Boulevard', 'Mission', 'Kansas', '72878', '665356572025');

INSERT INTO public.address VALUES (3, '1795 Santiago de Compostela Way', 'Merriam', 'Kansas', '94399', '648856936185');

INSERT INTO public.address VALUES (4, '33 Gorontalo Way', 'Overland Park', 'Missouri', '76022', '680768868518');

INSERT INTO public.address VALUES (5, '7710 Bydgosz Avenue', 'Kansas City', 'Kansas', '23616', '177727722820');

INSERT INTO public.address VALUES (6, '1718 Valencia Street', 'Shawnee Mission', 'Kansas', '53628', '378318851631');

INSERT INTO public.address VALUES (7, '1997 Halifax Drive', 'Turner', 'Missouri', '18746', '912257250465');

INSERT INTO public.address VALUES (8, '1717 Guadalajara Lane', 'North Kansas City', 'Kansas', '18743', '705800322606');

INSERT INTO public.address VALUES (9, '1150 Kimchon Manor', 'Garden City', 'Kansas', '17886', '262076994845');

INSERT INTO public.address VALUES (10, '1668 Saint Louis Place', 'Olathe', 'Kansas', '83579', '171822533480');

INSERT INTO public.address VALUES (11, '1213 Ranchi Parkway', 'Hamilton', 'Missouri', '45844', '347487831378');

INSERT INTO public.address VALUES (12, '1557 Ktahya Boulevard', 'Parkville', 'Missouri', '42399', '697760867968');

INSERT INTO public.address VALUES (13, '1926 El Alto Avenue', 'Grandview', 'Missouri', '83579', '333489324603');

INSERT INTO public.address VALUES (14, '733 Mandaluyong Place', 'North Kansas City', 'Missouri', '17886', '107137400143');

INSERT INTO public.address VALUES (15, '1386 Yangor Avenue', 'Parkville', 'Missouri', '65599', '255800440636');

INSERT INTO public.distributor VALUES (1, 'El Camino Young Films', '2015-08-29 00:00:00', 1);

INSERT INTO public.distributor VALUES (2, 'North Milwuake Movie Company', '2007-09-15 00:00:00', 6);

INSERT INTO public.distributor VALUES (3, 'Bingham Young Pictures', '2012-08-16 00:00:00', 7);

INSERT INTO public.distributor VALUES (4, 'Sunset Studios', '2003-08-16 00:00:00', 8);

INSERT INTO public.distributor VALUES (5, 'Annoying Productions', '2014-07-30 00:00:00', 9);

INSERT INTO public.catalog\_item VALUES (1, 1, 'ANTITRUST TEEN', 123456, 13.99, 1, '2018-10-25 14:48:20', 'DVD', '2019-11-18 14:48:20');

INSERT INTO public.catalog\_item VALUES (2, 2, 'APOLLO TOMATOES', 456123, 16.99, 2, '2017-12-15 14:48:20', 'DVD', '2018-10-25 14:48:20');

INSERT INTO public.catalog\_item VALUES (3, 3, 'BARABELLA LIBERTY', 789456, 17.50, 3, '2019-11-18 14:48:20', 'DVD', '2019-11-18 14:48:20');

INSERT INTO public.catalog\_item VALUES (4, 4, 'BIRD WHISPERER', 987654, 19.20, 4, '2016-09-22 14:48:20', 'VIDEO', '2017-08-16 14:48:20');

INSERT INTO public.catalog\_item VALUES (5, 5, 'CARIBBEAN LIBERTY', 321654, 15.25, 5, '2017-08-16 14:48:20', 'VIDEO', '2016-09-22 14:48:20');

INSERT INTO public.inventory\_item VALUES (1, 5, 3.99, '5', 21.99, 'VIDEO', 'NEW', '2015-10-29 14:48:20', NULL, 4);

INSERT INTO public.inventory\_item VALUES (2, 4, 4.99, '5', 18.99, 'VIDEO', 'GOOD', '2015-10-29 14:48:20', NULL, 5);

INSERT INTO public.inventory\_item VALUES (3, 3, 5.99, '3', 17.99, 'DVD', 'FAIR', '2015-10-29 14:48:20', NULL, 3);

INSERT INTO public.inventory\_item VALUES (4, 2, 7.99, '3', 22.99, 'DVD', 'REFURB', '2015-10-29 14:48:20', NULL, 2);

INSERT INTO public.inventory\_item VALUES (5, 1, 1.99, '3', 39.99, 'DVD', 'DAMAGED', '2015-10-29 14:48:20', NULL, 1);

INSERT INTO public.customer VALUES (1, 'HW12451', 'Helen', 'Willson', 'helen.willson@jdvidcustomer.org', 1, '2019-05-27 17:07:36', '2019-10-27 17:07:36', 7);

INSERT INTO public.customer VALUES (2, 'BB34521', 'Betty', 'Brown', 'betty.brown@jdvidcustomer.org', 1, '2019-10-25 08:30:16', '2019-10-29 14:30:16', 9);

INSERT INTO public.customer VALUES (3, 'NJ34512', 'Nancy', 'Jones', 'nancy.jones@jdvidcustomer.org', 1, '2019-08-25 10:30:16', '2019-09-25 10:34:45', 12);

INSERT INTO public.customer VALUES (4, 'KM87152', 'Karen', 'Moore', 'karen.moore@jdvidcustomer.org', 1, '2019-06-18 20:51:07', '2019-07-06 18:35:25', 3);

INSERT INTO public.customer VALUES (5, 'LT45381', 'Lisa', 'Taylor', 'lisa.taylor@jdvidcustomer.org', 1, '2019-06-21 13:51:07', '2019-08-21 11:25:40', 1);

INSERT INTO public.staff VALUES (1, 'Susan', 'Reik', 'sreik@jdvidstaff.org', 1, 'sreik', 'encryptedtext', 6);

INSERT INTO public.staff VALUES (2, 'Michael', 'Christensen', 'mchristensen@jdvidstaff.org', 1, 'mchristensen', 'encryptedtext', 4);

INSERT INTO public.staff VALUES (3, 'Brenda', 'Turner', 'bturner@jdvidstaff.org', 1, 'bturner', 'encryptedtext', 2);

INSERT INTO public.staff VALUES (4, 'Greg', 'Welch', 'gwelch@jdvidstaff.org', 1, 'gwelch', 'encryptedtext', 12);

INSERT INTO public.staff VALUES (5, 'Diane', 'Oswatamie', 'doswatamie@jdvidstaff.org', 1, 'doswatamie', 'encryptedtext', 11);

INSERT INTO public.rental VALUES (1, 5, '2019-10-12 14:45:20', 1.25, 5, '2019-10-16 10:45:20', 2, 1.75, NULL, NULL);

INSERT INTO public.rental VALUES (2, 4, '2019-10-30 13:44:10', 1.25, 3, NULL, 2, NULL, NULL, NULL);

INSERT INTO public.rental VALUES (3, 3, '2019-10-09 12:55:15', 1.25, 4, '2019-10-15 10:45:20', 2, 2.50, NULL, NULL);

INSERT INTO public.rental VALUES (4, 2, '2019-10-18 11:22:44', 1.25, 1, '2019-10-23 20:31:44', 4, NULL, 5.00, NULL);

INSERT INTO public.rental VALUES (5, 3, '2019-10-13 09:48:22', 1.25, 2, '2019-10-16 19:17:36', 4, NULL, NULL, NULL);

INSERT INTO public.rental VALUES (6, 2, '2019-10-07 11:48:17', 1.25, 3, '2019-10-16 11:25:18', 4, 5.50, NULL, 2.50);

INSERT INTO public.rental VALUES (7, 3, '2019-10-21 15:33:17', 1.25, 1, '2019-10-24 14:15:01', 4, NULL, NULL, NULL);

INSERT INTO public.rental VALUES (8, 1, '2019-10-20 17:45:59', 1.25, 3, '2019-10-25 14:35:19', 4, NULL, NULL, 2.50);

INSERT INTO public.rental VALUES (9, 2, '2019-10-15 19:38:20', 1.25, 5, '2019-10-20 18:45:21', 4, NULL, 5.00, NULL);

INSERT INTO public.payment VALUES (1, 5, 2.99, '2015-10-29 20:48:20');

INSERT INTO public.payment VALUES (2, 4, 5.99, '2015-10-29 11:22:20');

INSERT INTO public.payment VALUES (3, 3, 0.99, '2015-10-29 12:55:20');

INSERT INTO public.payment VALUES (4, 2, 4.99, '2015-10-29 13:35:20');

INSERT INTO public.payment VALUES (5, 1, 7.99, '2015-10-29 14:45:20');

INSERT INTO public.academy\_award VALUES (1, 'BEST ACTOR');

INSERT INTO public.academy\_award VALUES (2, 'BEST PICTURE');

INSERT INTO public.academy\_award VALUES (3, 'BEST ACTRESS');

INSERT INTO public.academy\_award VALUES (4, 'BEST DIRECTOR');

INSERT INTO public.academy\_award VALUES (5, 'BEST STORY');

INSERT INTO public.movie\_has\_academy\_award VALUES (1, 5);

INSERT INTO public.movie\_has\_academy\_award VALUES (1, 4);

INSERT INTO public.movie\_has\_academy\_award VALUES (2, 4);

INSERT INTO public.movie\_has\_academy\_award VALUES (2, 2);

INSERT INTO public.movie\_has\_academy\_award VALUES (3, 3);

INSERT INTO public.movie\_has\_academy\_award VALUES (4, 2);

INSERT INTO public.movie\_has\_academy\_award VALUES (5, 1);

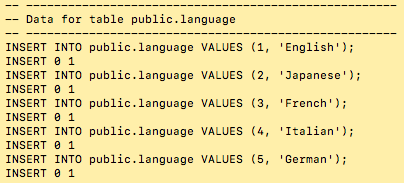
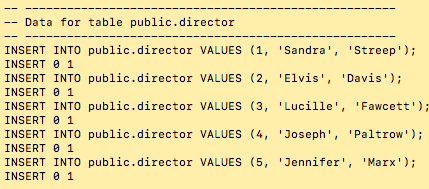
****

Figure 20 – Output for insert statements for language table.



Figure 21- Output for insert statements for movie table.

  
Figure 22 – Output for insert statements for director table.

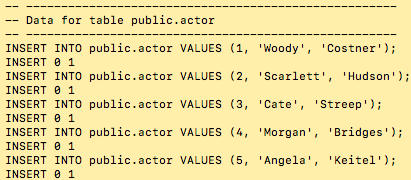


Figure 23 – Output for insert statements for actor table.

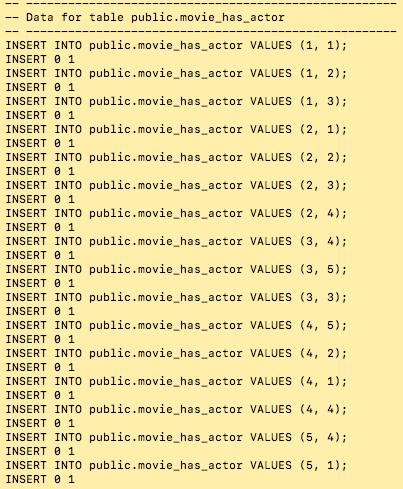


Figure 24- Output for insert statements for movie\_has\_actor table.

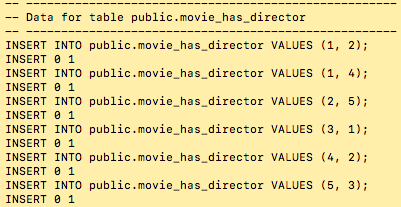


Figure 25 – Output for insert statements for movie\_has\_director table.

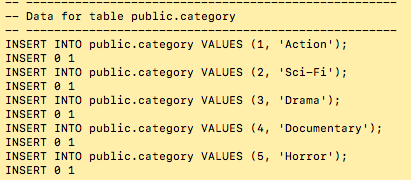


Figure 25 – Output for insert statements for category table.

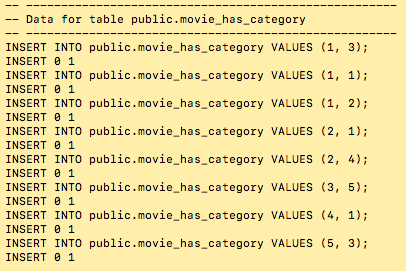


Figure 26 – Output for insert statements for movie\_has\_category table.



Figure 27 – Output for insert statements for address table.

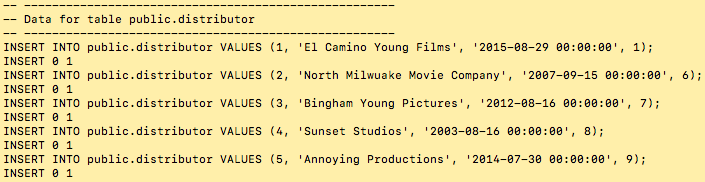


Figure 28 – Output for insert statements for distributor table.

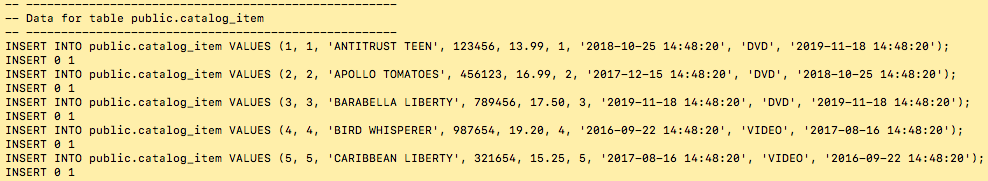


Figure 29 – Output for insert statements for catalog item table.

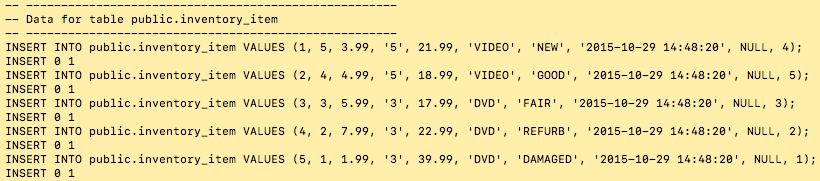


Figure 30 – Output for insert statements for inventory item table.

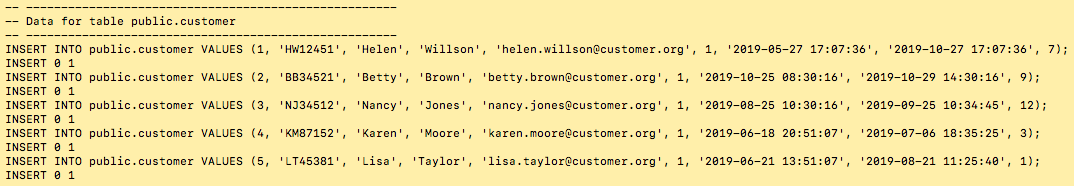
  
Figure 31- Output for insert statements for customer table.



Figure 32 – Output for insert statements for staff table.



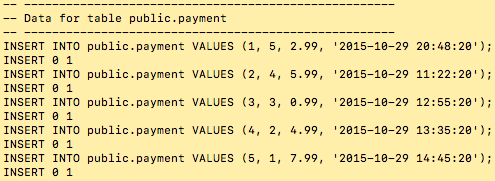
Figure 33 – Output for insert statements for rental table.  
  


Figure 34 – Output for insert statements for payment table.

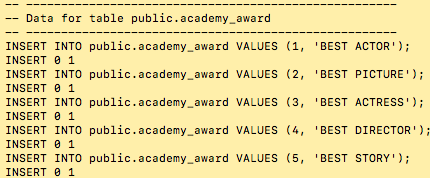


Figure 35- Output for insert statements for academy award table.

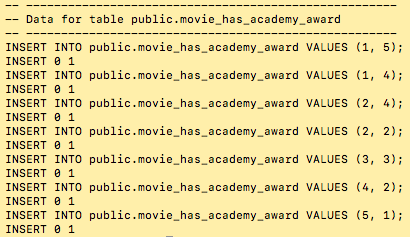
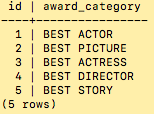


Figure 36 – Output for insert statements for movie has academy award table.

**DML: Queries and CRUD Operations**

1. Show the contents of all tables.

SELECT \* FROM academy\_award;

  
Figure 37 – Contents of academy award.

SELECT \* FROM actor;

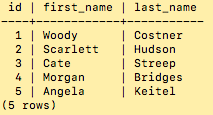


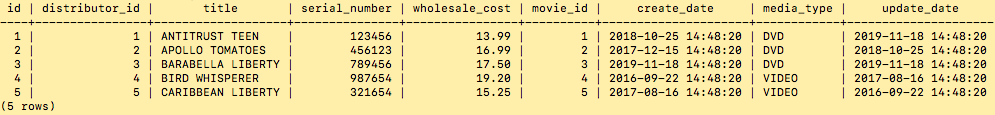
Figure 38 – Contents of actor

SELECT \* FROM address;



Figure 39 – Contents of address.

SELECT \* FROM catalog\_item;

  
Figure 40 – Contents of catalog item.

SELECT \* FROM category;

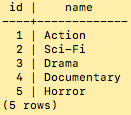


Figure 41- Contents of category.

SELECT \* FROM director;

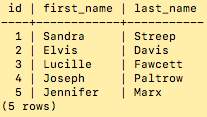


Figure 42- Contents of director

SELECT \* FROM language;

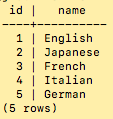


Figure 43- Contents of language.

SELECT \* FROM customer;

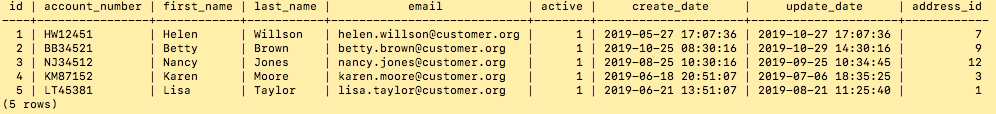


Figure 44 – Contents of customer.

SELECT \* FROM distributor;

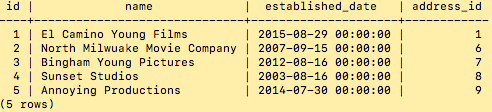


Figure 45 – Contents of distributor.

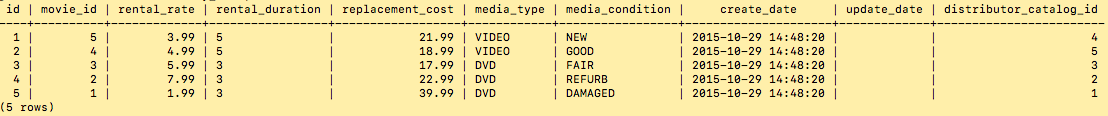
SELECT \* FROM inventory\_item;  


Figure 45 – Contents of inventory item.

SELECT \* FROM movie;

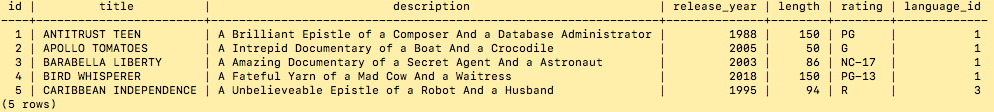


Figure 46 – Contents of movie.

SELECT \* FROM payment;

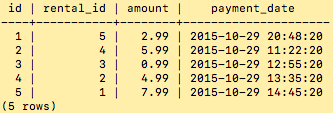


Figure 47 – Contents of payment.

SELECT \* FROM rental;

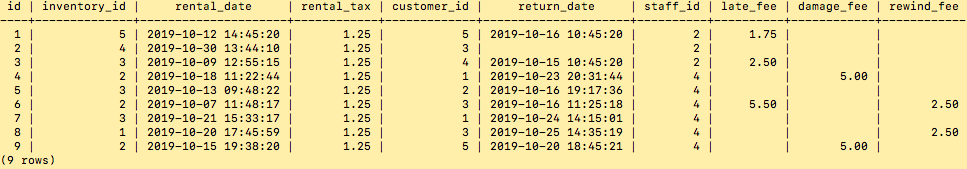


Figure 48 – Contents of rental.

SELECT \* FROM staff;

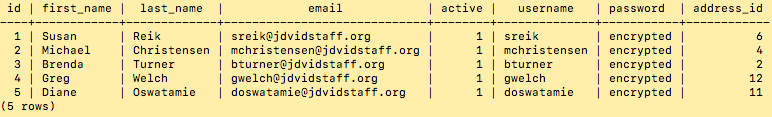
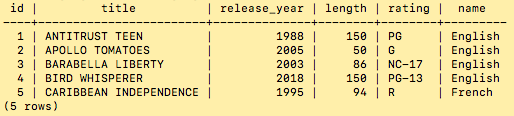


Figure 49 – Contents of staff.

SELECT m.id, m.title, m.release\_year, m.length, m.rating, l.name   
FROM movie m

JOIN language l on m.language\_id=l.id  
ORDER BY m.id ASC;

  
Figure 49 – Contents of movie joined with language.

SELECT \* FROM movie\_has\_academy\_award;

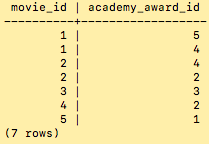


Figure 50– Contents of movie has academy award.

SELECT \* FROM movie\_has\_director;

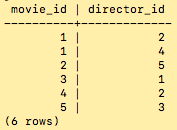


Figure 51– Contents of movie has director.

SELECT \* FROM movie\_has\_actor;

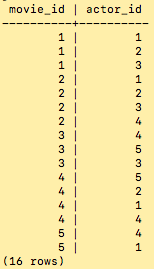


Figure 52 – Contents of movie has actor.

SELECT \* FROM movie\_has\_category;

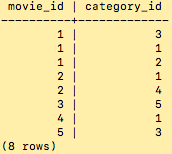
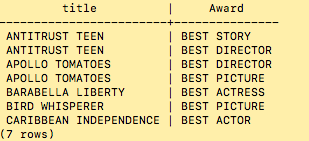


Figure 53 – Contents of movie has category.

SELECT m.title, a.award\_category AS "Award"   
FROM movie\_has\_academy\_award ah

JOIN movie m ON m.id=ah.movie\_id

JOIN academy\_award a ON ah.academy\_award\_id=a.id;

  
Figure 54 – Contents of movie has award table using joins.

SELECT m.title, **CONCAT**(d.first\_name, ' ', d.last\_name) AS "Director"

FROM movie\_has\_director md

JOIN movie m ON md.movie\_id=m.id

JOIN director d ON md.director\_id= d.id;

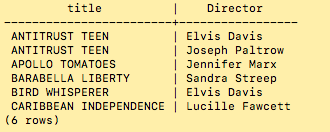


Figure 55 – Contents of movie has director using joins.

SELECT m.title, **CONCAT**(a.first\_name, ' ', a.last\_name) AS "Cast"

FROM movie\_has\_actor ma  
JOIN movie m on ma.movie\_id=m.id

JOIN actor a on ma.actor\_id=a.id;

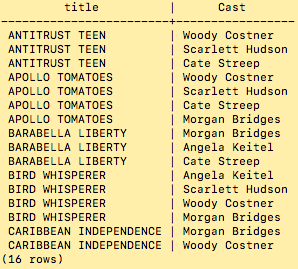


Figure 56– Contents of movie has actor using joins.

SELECT m.title, c.name AS "Category"

FROM movie\_has\_category mc

JOIN movie m ON  mc.movie\_id= m.id

JOIN category c ON mc.category\_id=c.id;

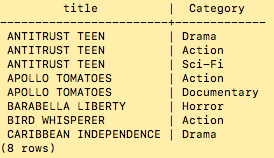


Figure 57– Contents of movie has category using joins.

1. Retrieve all the customers' names, account numbers, and addresses (street and zip code only), sorted by account number.

SELECT c.account\_number, c.first\_name, c.last\_name, a.street, a.postal\_code

FROM customer c

JOIN address a ON c.address\_id = a.id

ORDER BY c.account\_number ASC;

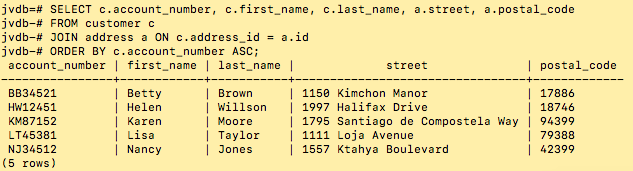


Figure 58– Customers' names, account numbers, and addresses.

1. Retrieve all of the DVDs rented in the last 30 days and sort in chronological rental date order.

SELECT c.first\_name, c.last\_name, r.rental\_date, m.title, i.media\_type FROM rental r

JOIN customer c ON r.customer\_id=c.id

JOIN inventory\_item i ON r.inventory\_id=i.id

JOIN movie m ON i.movie\_id=m.id

WHERE i.media\_type='DVD'

AND r.rental\_date > (current\_date - interval '30 day') ORDER BY r.rental\_date ASC;

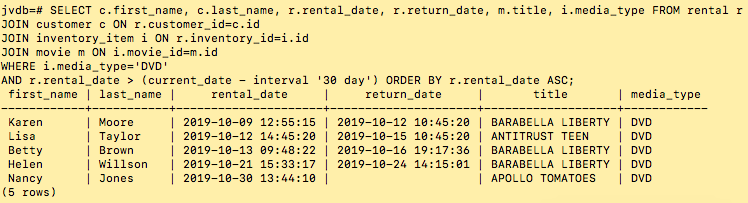


Figure 59–Query showing all DVD’s rented in the last 30 days sorted chronologically.

1. Update a customer name to change their maiden names to married name.

UPDATE customer SET last\_name='Mirren' WHERE id=1;

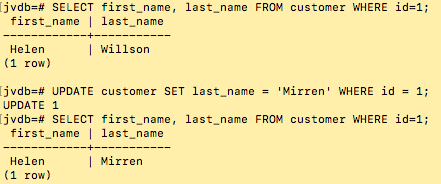


Figure 60– Output of updating a customer last name.

1. Delete a specific customer from the database.

DELETE FROM customer WHERE id=1;

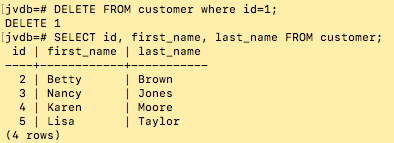


Figure 61– Output of deleting a customer.

**Lessons Learned**

At the beginning of this process, research was performed to learn about the best practices and procedures for database design and management. It was determined that overall practices are highly varied between individuals and across organizations. The only consensuses that did seem to emerge involving best practices were in areas deemed most crucial such as security and resiliency (Holt, Ramage, Kear, & Heap, 2015, pp. 168). Practices in the areas are relatively standardized in implementation, spanning many relational database management systems (RDMS). The conclusion is that there is no true consensus regarding best practices for design and maintenance of an operational database, although various methodologies exist such as Agile and Waterfall.

There was the additionally the realization that a major challenge is ensuring the data your SQL returns is in fact the correct data. There are many instances where the output appears correct but are incorrect. Perhaps you have queried the wrong table yet received a similar result. Throughout the duration of my project, this memorably occurred once: when querying for the contents of the movie\_has\_academy\_award table. Initially the movie table had been queried, joined movie\_has\_academy\_award, also joined with the award table. The output appeared identical.

The first major challenge concerned productivity: How to develop the schema, populate sample data, and test the results using the least amount of labor (hours). The integration of graphical user interfaces and the command line interface greatly increased overall productivity. According to Codd, relational database management systems that provide both of these tools greatly increase the productivity of both programmers and database administrators. This was reinforced for me particularly throughout this project by utilizing the pgAdmin interface to construct tables, insertions, and perform a data-dump. This tool significantly decreased the amount of time required to generate the DDL SQL. The process would have been much more laborious if I did not have the graphical tool to aid and automate the process. However, that does not imply that using the data dump in a final script was enough. A significant portion of the SQL produced by the dump performed operations that could simply be allowed happen by default. A substantial minimization was achieved by editing and reviewing the output from pgdump.

Preferring the command line interface psql, it was required to learn new commands in order to execute scripts locate inside SQL script files, as well learning to produce an output describing tables and their structures. The primary source of reference was obtained by accessing the documentation section on the PostgreSQL website. Overall, the result was a real strengthening that of my relational algebra skills, being necessary to perform multiple joins in order to achieve complex queries to display the data shown in figures 54-57.

Additionally, a newfound appreciation was attained for the benefit of having conducted a thorough conceptual modeling phase. According to research done by Yang, there are two major approaches to this phase. The first involves construction of a single large relational table, and then decomposing it higher normal forms. The other method, popular in the software design industry, involves the creation of several relational tables, that are further normalized to reduce or eliminate redundancy (Yang, 2003). I am thankful for having been exposed to the latter methodology, although both demonstrably may have merit, the former appears historic.

The relevance of Yang’s research to my own experience, is that multiple errors would have persisted in my model had I blindly followed the initial model. For example, an error was discovered in the first version that depicted staff as being related to a single rental. The address relation was also depicted as a one-to-one relationship, which is inaccurate because customers, distributors, and staff may in-fact have the same address. The model was revised accordingly.

References

The PostgreSQL Global Development Group. (n.d.). PostgreSQL: Documentation: 11: psql. Retrieved October 31, 2019, from <https://www.postgresql.org/docs/11/app-psql.html>

Codd, E. (1982). Relational database: a practical foundation for productivity. *Communications of the ACM*, *25*(2), 109–117. <https://doi.org/10.1145/358396.358400>

Holt, V., Ramage, M., Kear, K., & Heap, N. (2015). The usage of best practices and procedures in the database community. *Information Systems*, *49*(C), 163–181. <https://doi.org/10.1016/j.is.2014.12.004>

Juba, S., & Volkov, A. (2017). *Learning PostgreSQL 10: A beginner’s guide to building high-performance PostgreSQL database solutions* [Redshelf]. Retrieved from <https://platform.virdocs.com/r/s/0/doc/592940/>

Storey, V. C., Trujillo, J. C., & Liddle, S. W. (2015). Research on conceptual modeling: Themes, topics, and introduction to the special issue. *Data & Knowledge Engineering*, *98*, 1–7. <https://doi.org/10.1016/j.datak.2015.07.002>

V. Mannino, M. V. (2018). *Database Design, Application Development, and Administration, 7e* [Redshelf]. Retrieved from <https://platform.virdocs.com/r/s/0/doc/592088/>

Yang, H. (2003). Comparing relational database designing approaches: some managerial implications for database training. *Industrial Management & Data Systems*, *103*(3), 150–166. <https://doi.org/10.1108/02635570310465634>