Prime Video DBMS

Submitted by:

Dhruv Atreja

2019UMC4513

Submitted to:

Professor Swati Aggarwal

Index:

Problem and objectives

Problem:

There is a need for a database to be developed to manage and store information for a video streaming platform containing information about/for:

- TV Shows
- Movies
- Admin team
- Accounting
- Individual Users

Objective:

To develop a database that

- Satisfies rules of normalisation
- Captures all required information for a video streaming platform

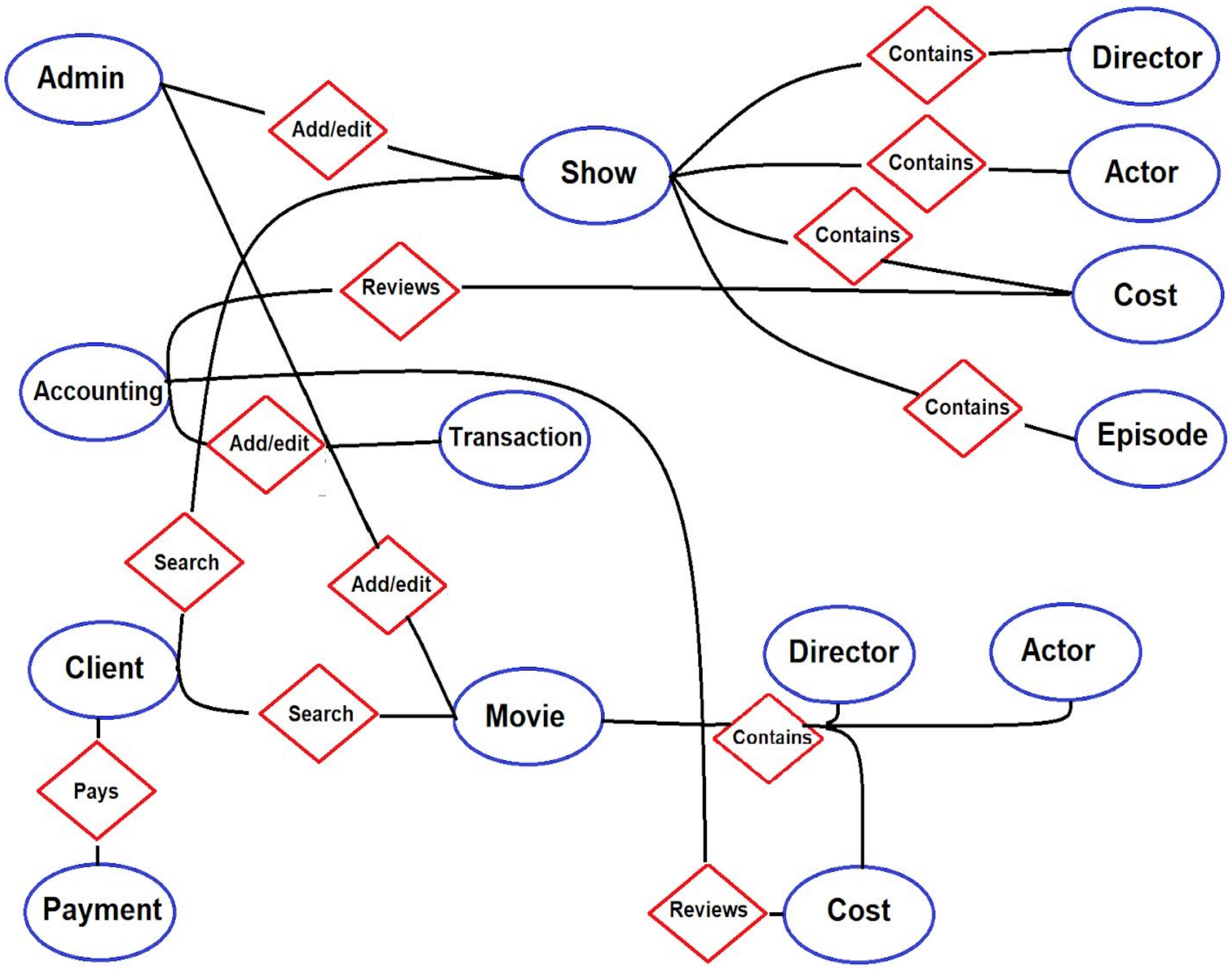
Meditations

- There are two primary advantages of having a highly normalized data schema:
    1. Increased consistency. Information is stored in one place and one place only, reducing the possibility of inconsistent data.
    2. Easier object-to-data mapping. Highly-normalized data schemas in general are closer conceptually to object-oriented schemas because the object-oriented goals of promoting high cohesion and loose coupling between classes results in similar solutions (at least from a data point of view).
- How will the data be used? Is it mostly transactional (OLTP) or mostly for generating reporting views (OLAP)?
- Is there any duplication which can (reasonably) be avoided, or anything which needs taking into account to prevent this in the future?
- Are there any potentially large or complex relationships which will need special treatment?
- What is likely to change in the next 3–6 months, 1–2 years, etc and how can I best allow for it now?

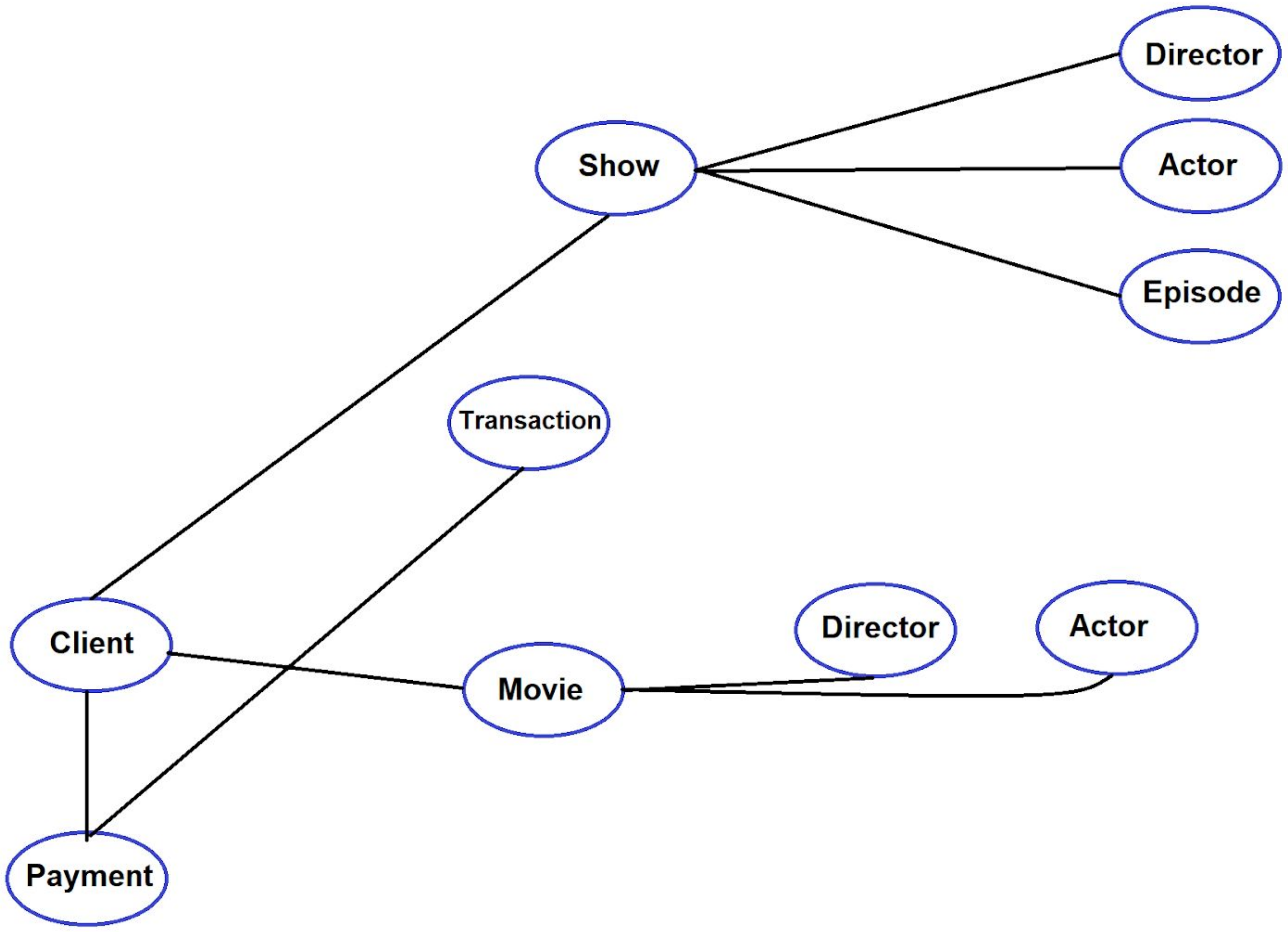
Design

1. Semantic design
2. Client view
3. ER diagram
4. Tables

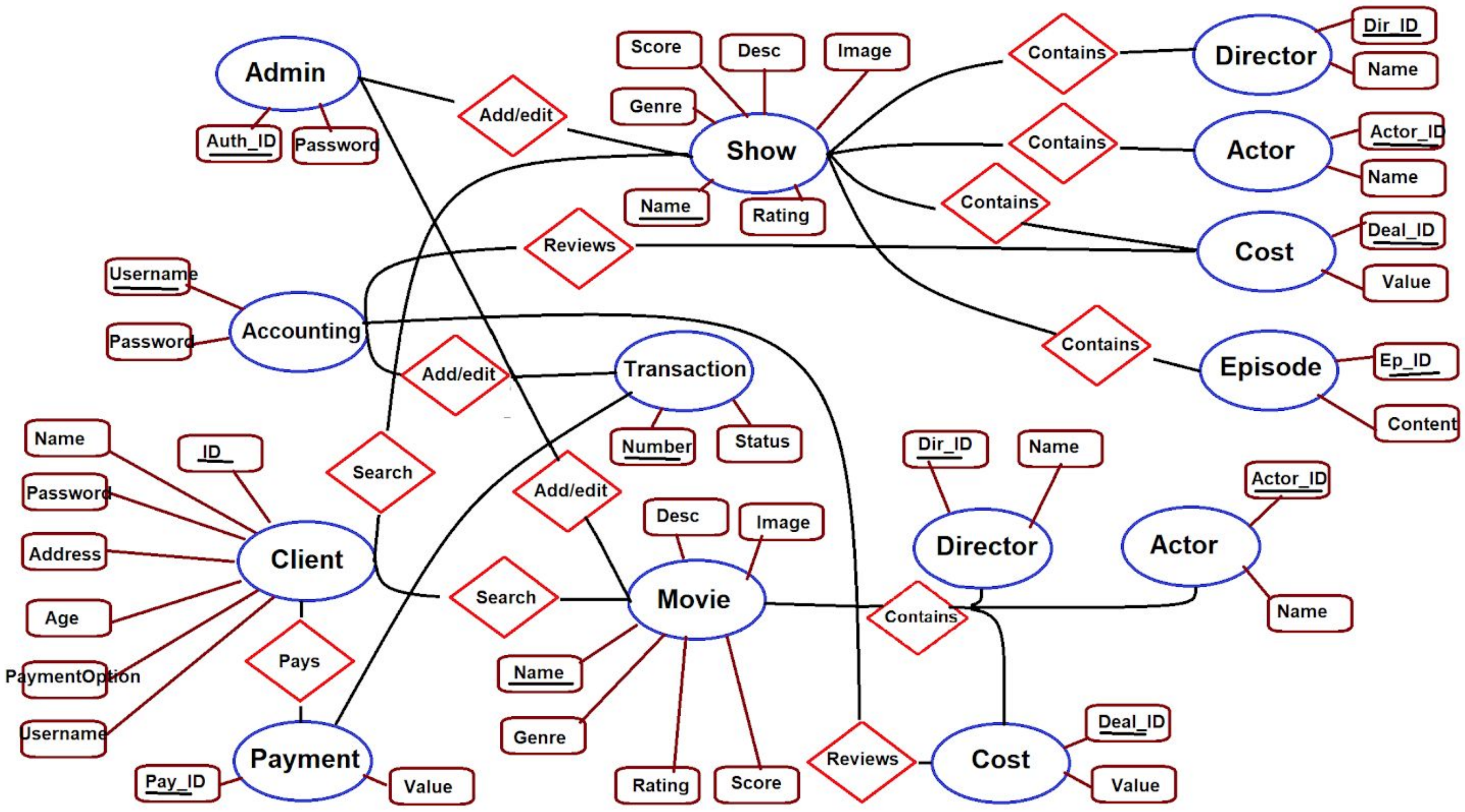
Semantic Design



Client View


Director
Show
Actor
Episode
Transaction
Client
Director
Actor
Movie
Payment

E-R diagram



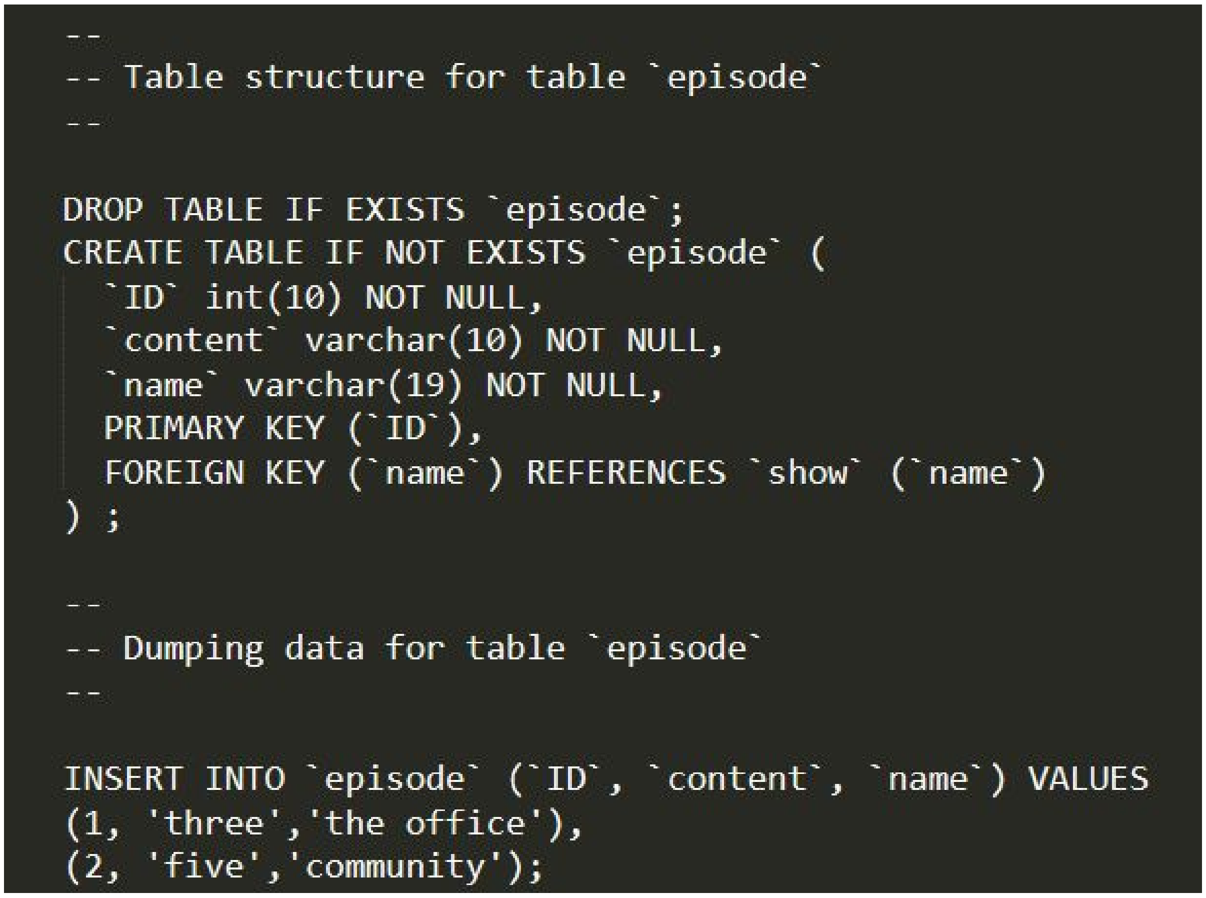
Implementation

The database has been implemented in MYSQL
Github link: https://github.com/DhruvAtreja/DBMS_project

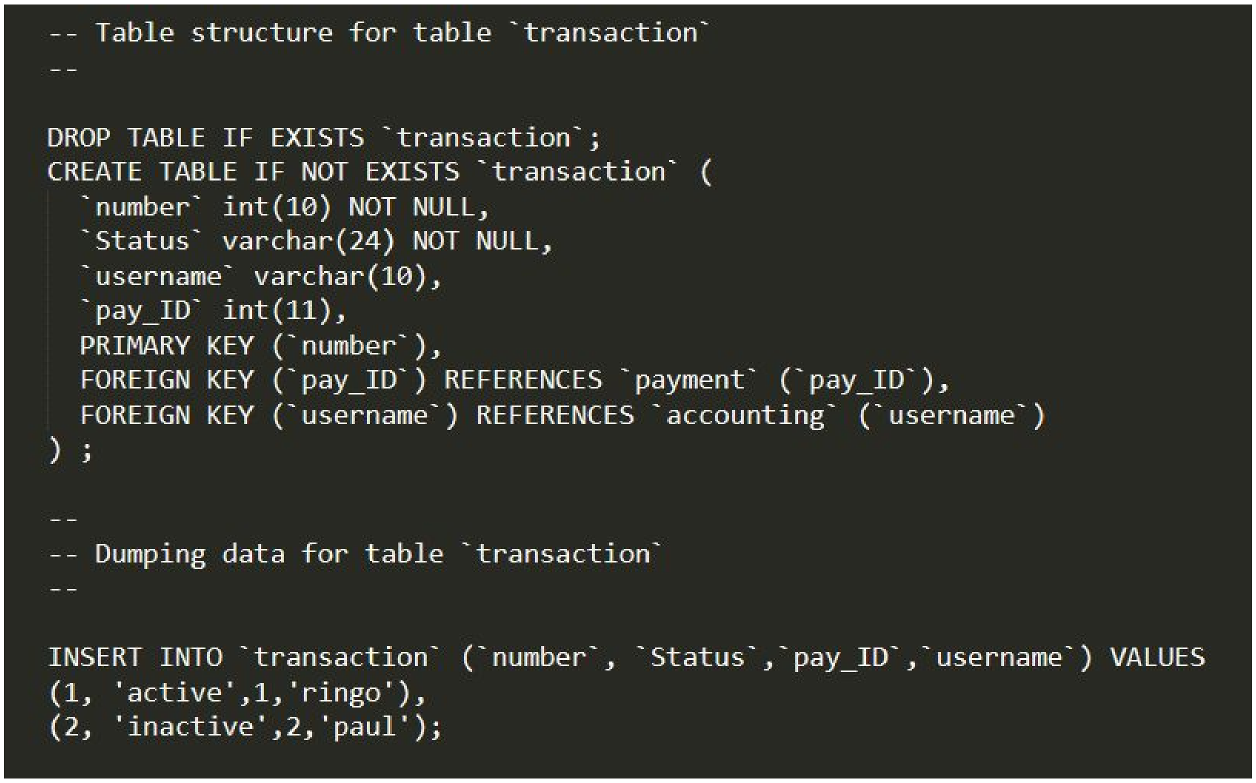
Each table has one PK on which all other attributes are functionally dependent(My implementation has 13 tables, hence I

haven't shown their individual normalisation). Hence, our dbms is in **BCNF normal form.**
Also, many tables have **super keys,** as i will show in the subsequent section of tables. The table transaction has **2 super keys since it relates to clients transactions as well as Amazon’s transactions.**

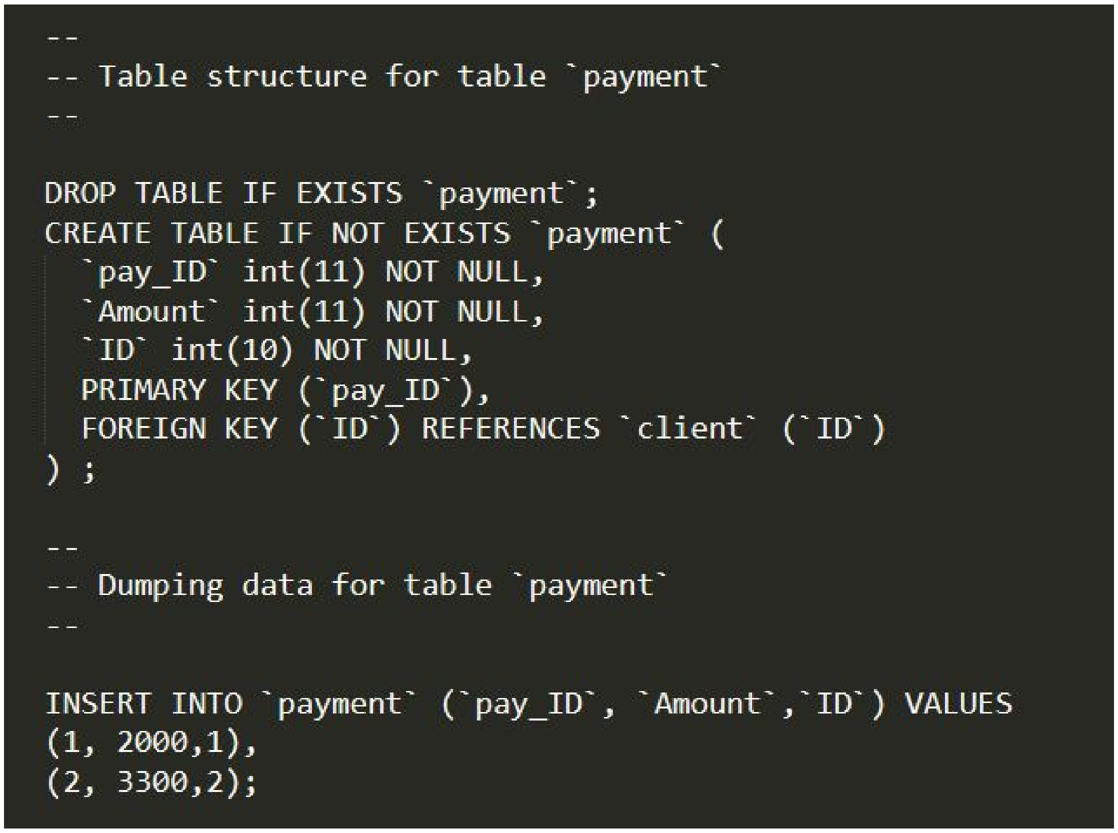
Tables:



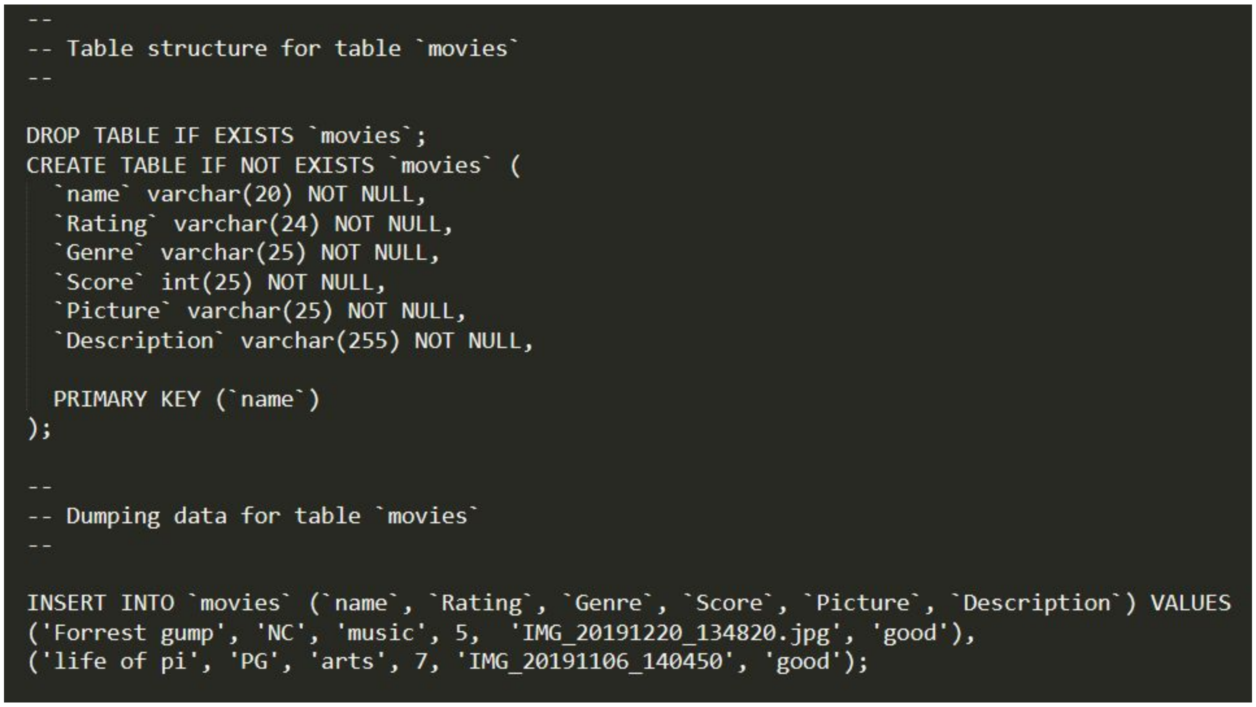
```
--
-- Table structure for table `episode`
--

DROP TABLE IF EXISTS `episode`;
CREATE TABLE IF NOT EXISTS `episode` (
  `ID` int(10) NOT NULL,
  `content` varchar(10) NOT NULL,
  `name` varchar(19) NOT NULL,
  PRIMARY KEY (`ID`),
  FOREIGN KEY (`name`) REFERENCES `show` (`name`)
) ;

--
-- Dumping data for table `episode`
--

INSERT INTO `episode` (`ID`, `content`, `name`) VALUES
(1, 'three','the office'),
(2, 'five','community');
```



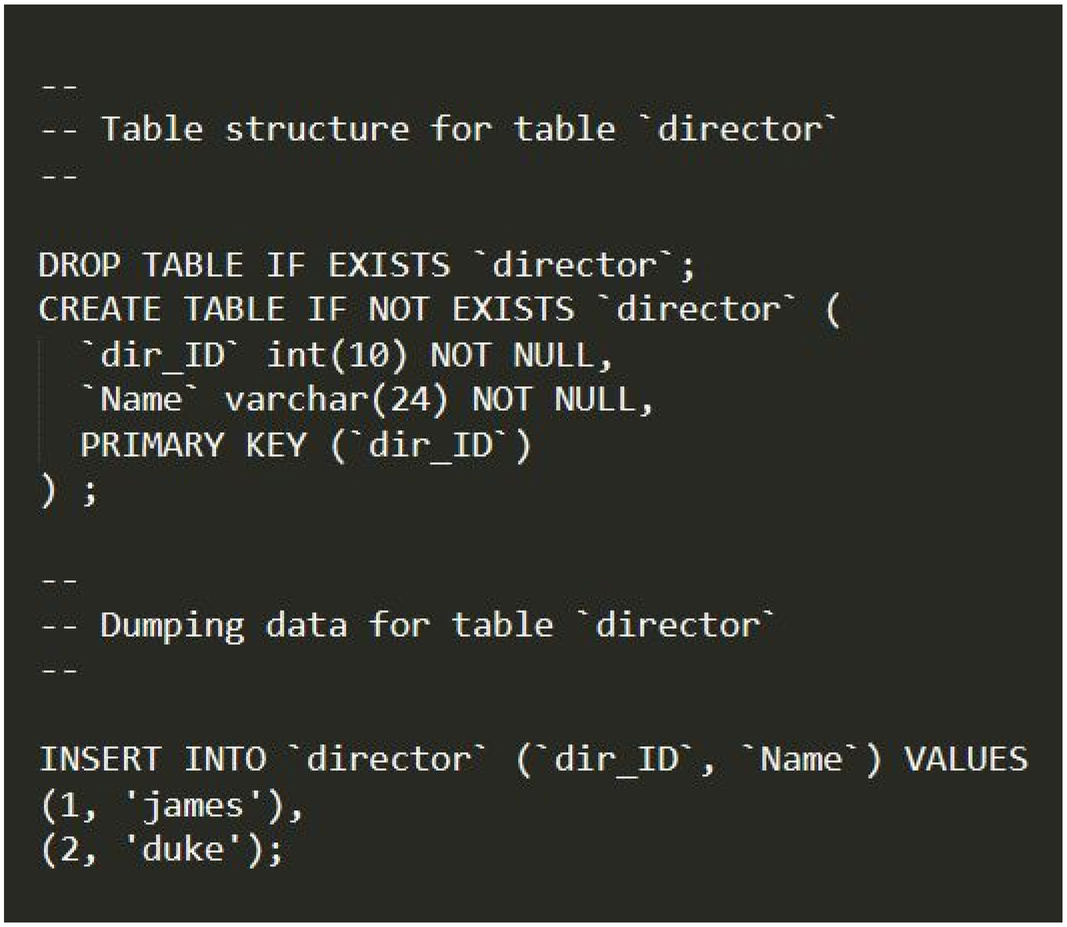
```
-- Table structure for table `transaction`
--

DROP TABLE IF EXISTS `transaction`;
CREATE TABLE IF NOT EXISTS `transaction` (
  `number` int(10) NOT NULL,
  `Status` varchar(24) NOT NULL,
  `username` varchar(10),
  `pay_ID` int(11),
  PRIMARY KEY (`number`),
  FOREIGN KEY (`pay_ID`) REFERENCES `payment` (`pay_ID`),
  FOREIGN KEY (`username`) REFERENCES `accounting` (`username`)
) ;

--
-- Dumping data for table `transaction`
--

INSERT INTO `transaction` (`number`, `Status`,`pay_ID`,`username`) VALUES
(1, 'active',1,'ringo'),
(2, 'inactive',2,'paul');
```



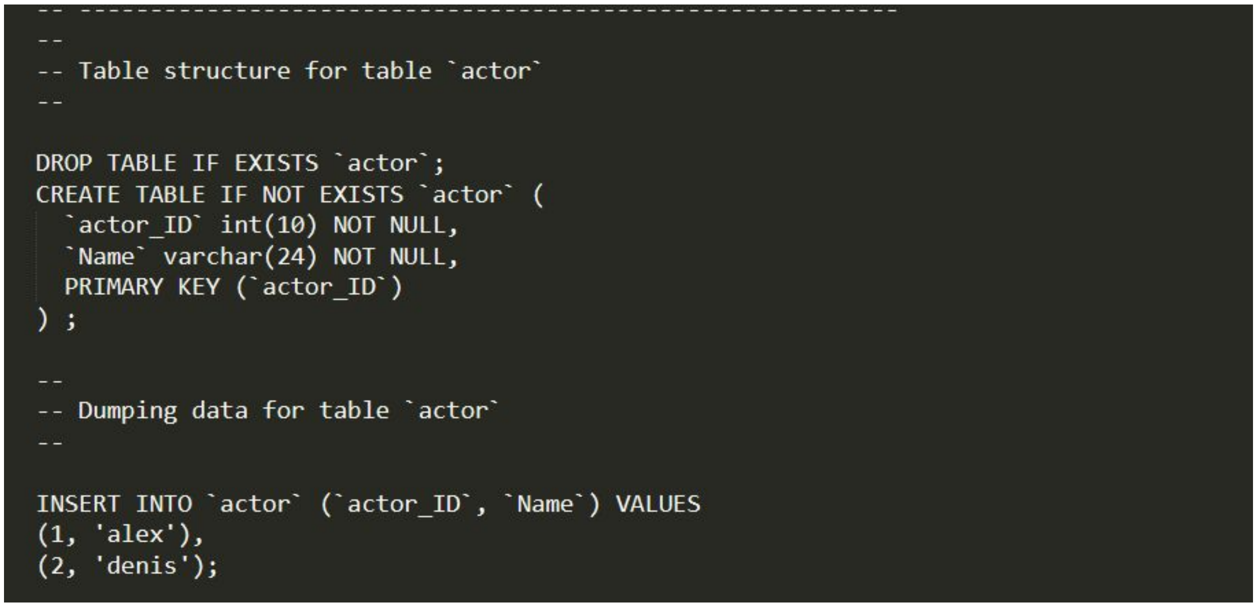
```
--
-- Table structure for table `payment`
--

DROP TABLE IF EXISTS `payment`;
CREATE TABLE IF NOT EXISTS `payment` (
  `pay_ID` int(11) NOT NULL,
  `Amount` int(11) NOT NULL,
  `ID` int(10) NOT NULL,
  PRIMARY KEY (`pay_ID`),
  FOREIGN KEY (`ID`) REFERENCES `client` (`ID`)
) ;

--
-- Dumping data for table `payment`
--

INSERT INTO `payment` (`pay_ID`, `Amount`,`ID`) VALUES
(1, 2000,1),
(2, 3300,2);
```



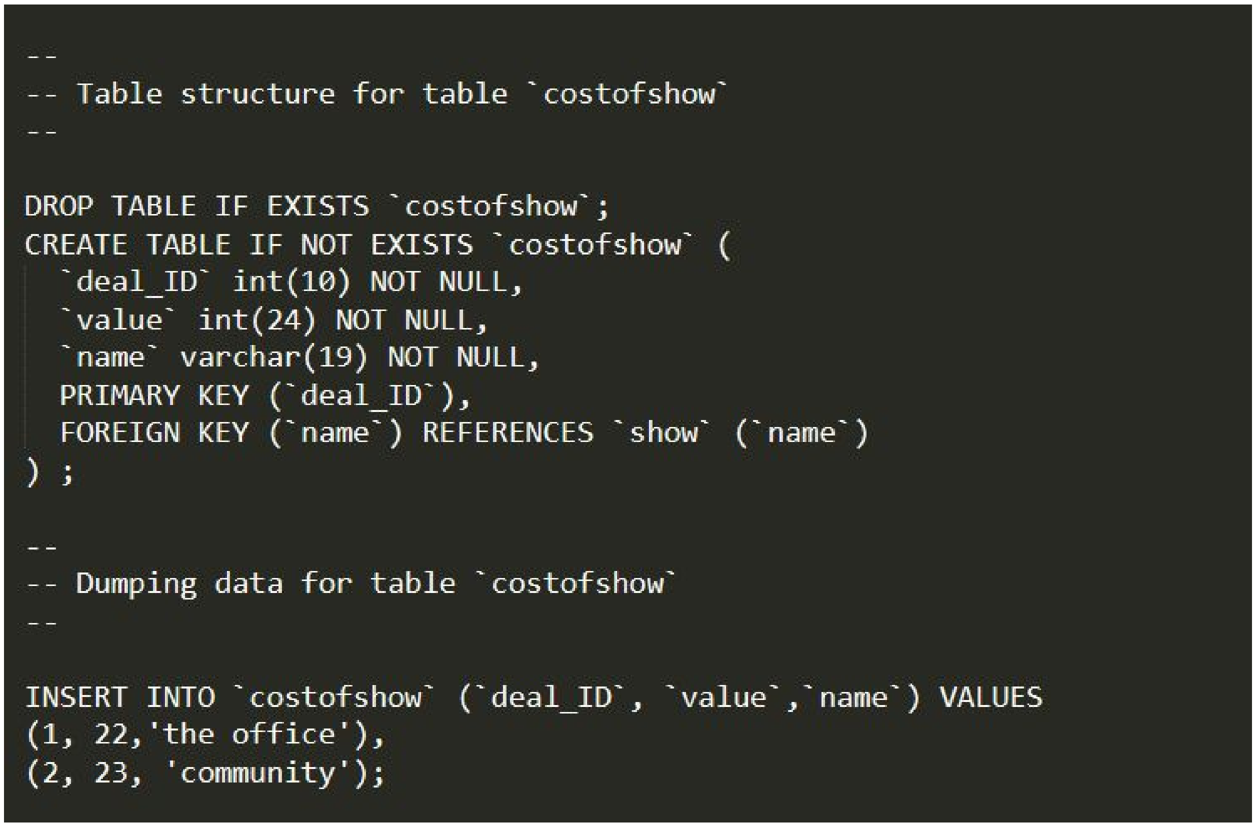
```
--
-- Table structure for table `movies`
--

DROP TABLE IF EXISTS `movies`;
CREATE TABLE IF NOT EXISTS `movies` (
  `name` varchar(20) NOT NULL,
  `Rating` varchar(24) NOT NULL,
  `Genre` varchar(25) NOT NULL,
  `Score` int(25) NOT NULL,
  `Picture` varchar(25) NOT NULL,
  `Description` varchar(255) NOT NULL,

  PRIMARY KEY (`name`)
);

--
-- Dumping data for table `movies`
--

INSERT INTO `movies` (`name`, `Rating`, `Genre`, `Score`, `Picture`, `Description`) VALUES
('Forrest gump', 'NC', 'music', 5,  'IMG_20191220_134820.jpg', 'good'),
('life of pi', 'PG', 'arts', 7, 'IMG_20191106_140450', 'good');
```



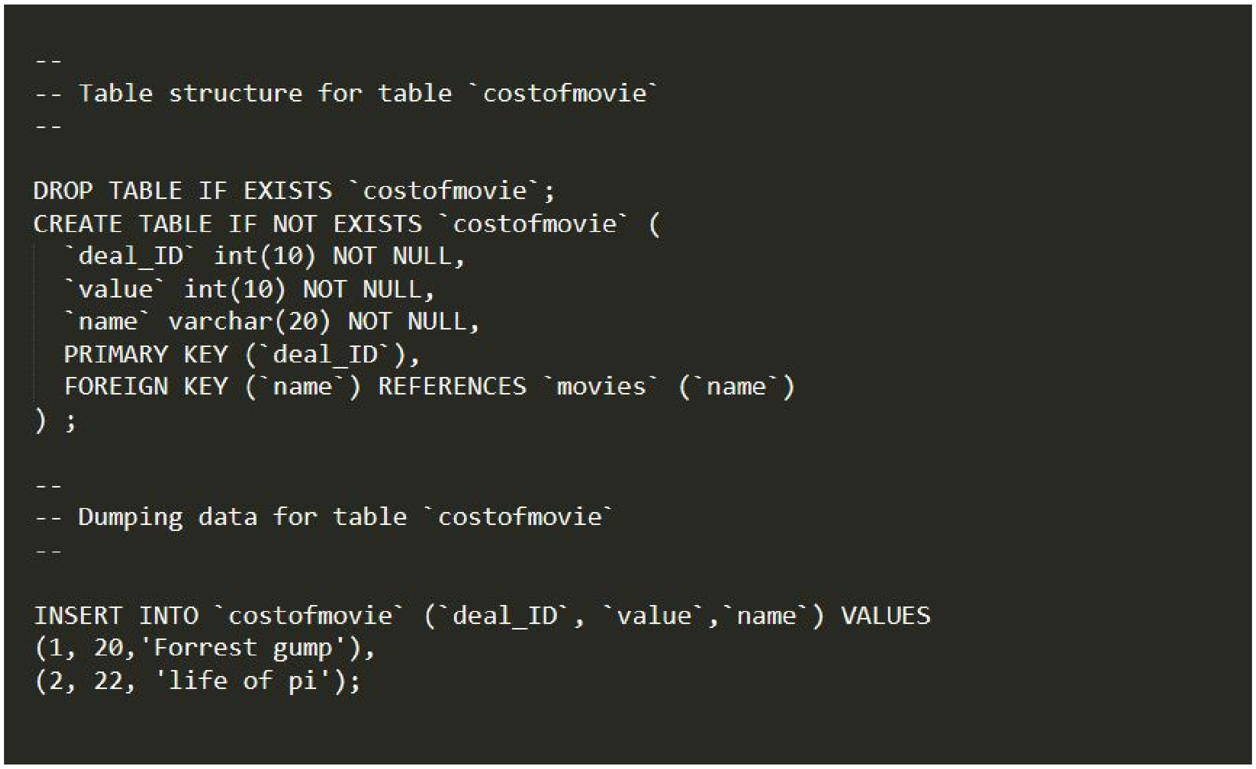
```
--
-- Table structure for table `director`
--

DROP TABLE IF EXISTS `director`;
CREATE TABLE IF NOT EXISTS `director` (
  `dir_ID` int(10) NOT NULL,
  `Name` varchar(24) NOT NULL,
  PRIMARY KEY (`dir_ID`)
) ;

--
-- Dumping data for table `director`
--

INSERT INTO `director` (`dir_ID`, `Name`) VALUES
(1, 'james'),
(2, 'duke');
```



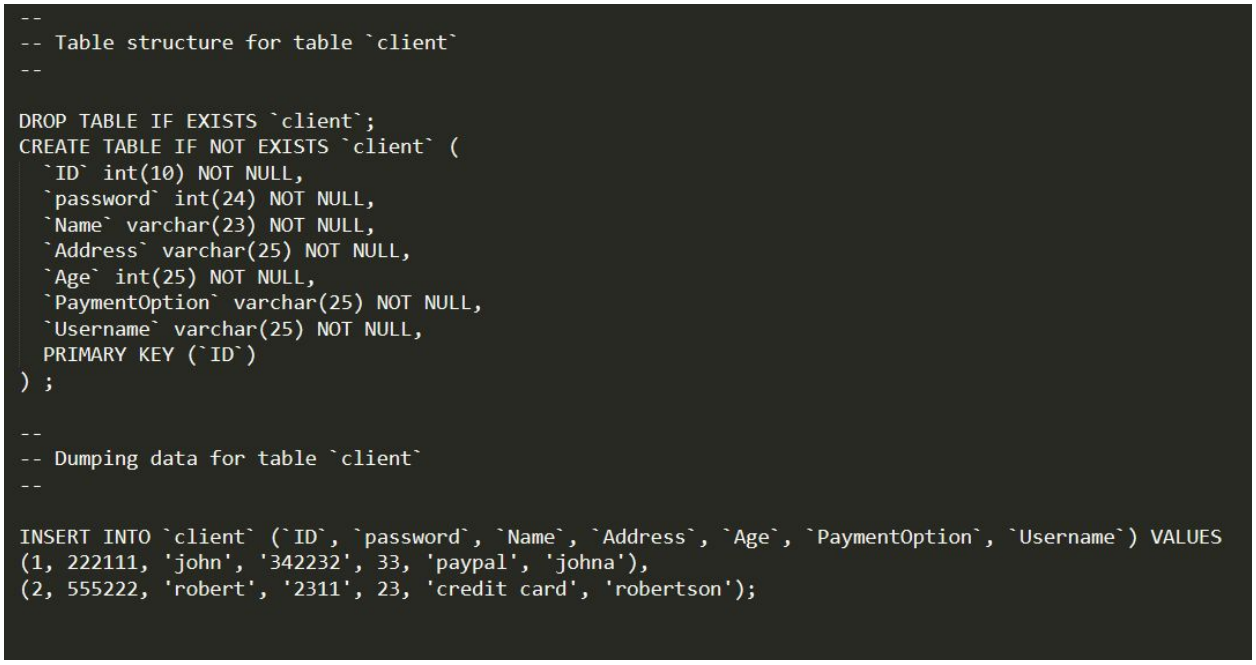
```
-- --------------------------------------------------------
--
-- Table structure for table `actor`
--

DROP TABLE IF EXISTS `actor`;
CREATE TABLE IF NOT EXISTS `actor` (
  `actor_ID` int(10) NOT NULL,
  `Name` varchar(24) NOT NULL,
  PRIMARY KEY (`actor_ID`)
) ;

--
-- Dumping data for table `actor`
--

INSERT INTO `actor` (`actor_ID`, `Name`) VALUES
(1, 'alex'),
(2, 'denis');
```



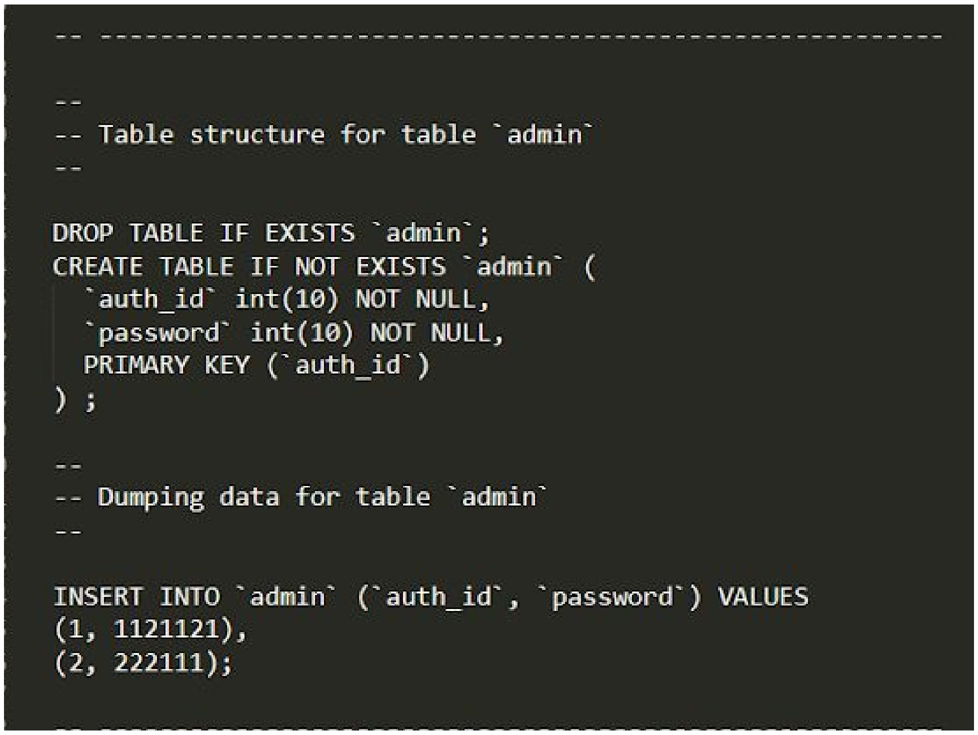
```
--
-- Table structure for table `costofshow`
--

DROP TABLE IF EXISTS `costofshow`;
CREATE TABLE IF NOT EXISTS `costofshow` (
  `deal_ID` int(10) NOT NULL,
  `value` int(24) NOT NULL,
  `name` varchar(19) NOT NULL,
  PRIMARY KEY (`deal_ID`),
  FOREIGN KEY (`name`) REFERENCES `show` (`name`)
) ;

--
-- Dumping data for table `costofshow`
--

INSERT INTO `costofshow` (`deal_ID`, `value`,`name`) VALUES
(1, 22,'the office'),
(2, 23, 'community');
```



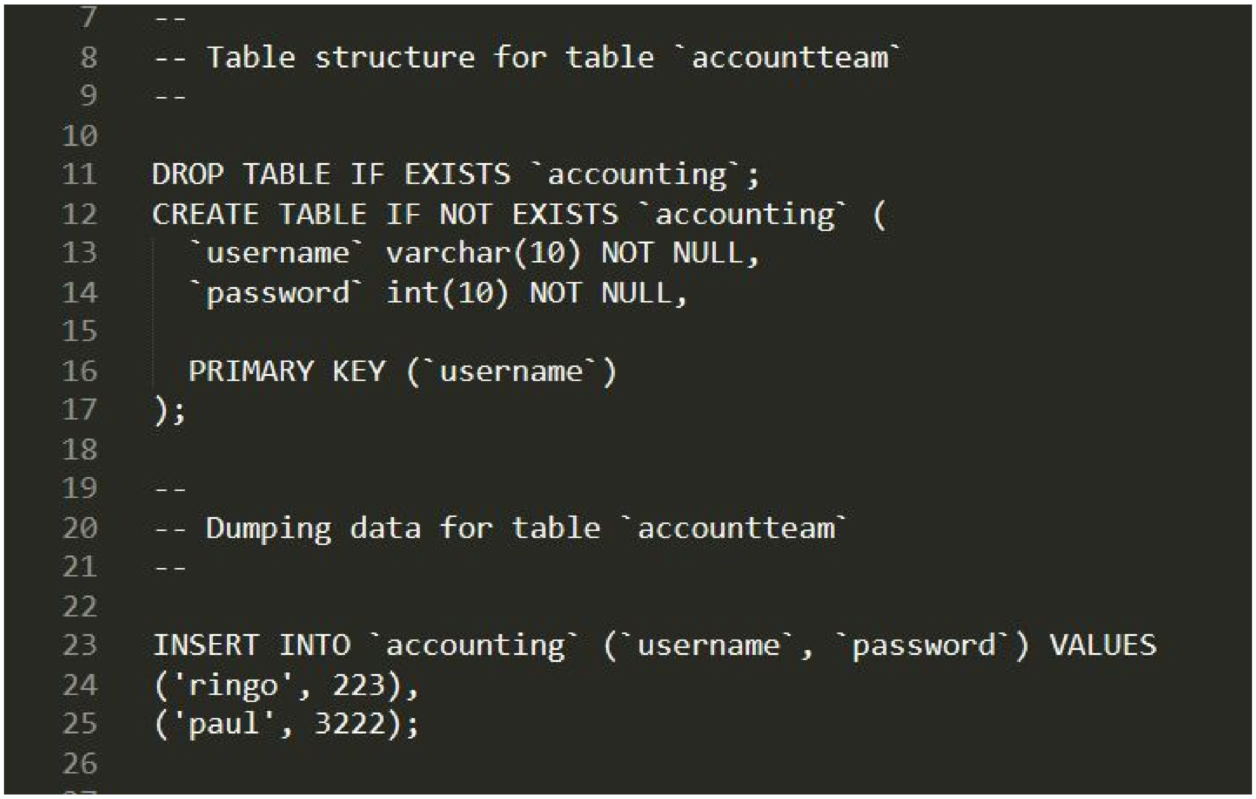
```
--
-- Table structure for table `costofmovie`
--

DROP TABLE IF EXISTS `costofmovie`;
CREATE TABLE IF NOT EXISTS `costofmovie` (
  `deal_ID` int(10) NOT NULL,
  `value` int(10) NOT NULL,
  `name` varchar(20) NOT NULL,
  PRIMARY KEY (`deal_ID`),
  FOREIGN KEY (`name`) REFERENCES `movies` (`name`)
) ;

--
-- Dumping data for table `costofmovie`
--

INSERT INTO `costofmovie` (`deal_ID`, `value`,`name`) VALUES
(1, 20,'Forrest gump'),
(2, 22, 'life of pi');
```



```
--
-- Table structure for table `client`
--

DROP TABLE IF EXISTS `client`;
CREATE TABLE IF NOT EXISTS `client` (
  `ID` int(10) NOT NULL,
  `password` int(24) NOT NULL,
  `Name` varchar(23) NOT NULL,
  `Address` varchar(25) NOT NULL,
  `Age` int(25) NOT NULL,
  `PaymentOption` varchar(25) NOT NULL,
  `Username` varchar(25) NOT NULL,
  PRIMARY KEY (`ID`)
) ;

--
-- Dumping data for table `client`
--

INSERT INTO `client` (`ID`, `password`, `Name`, `Address`, `Age`, `PaymentOption`, `Username`) VALUES
(1, 222111, 'john', '342232', 33, 'paypal', 'johna'),
(2, 555222, 'robert', '2311', 23, 'credit card', 'robertson');
```



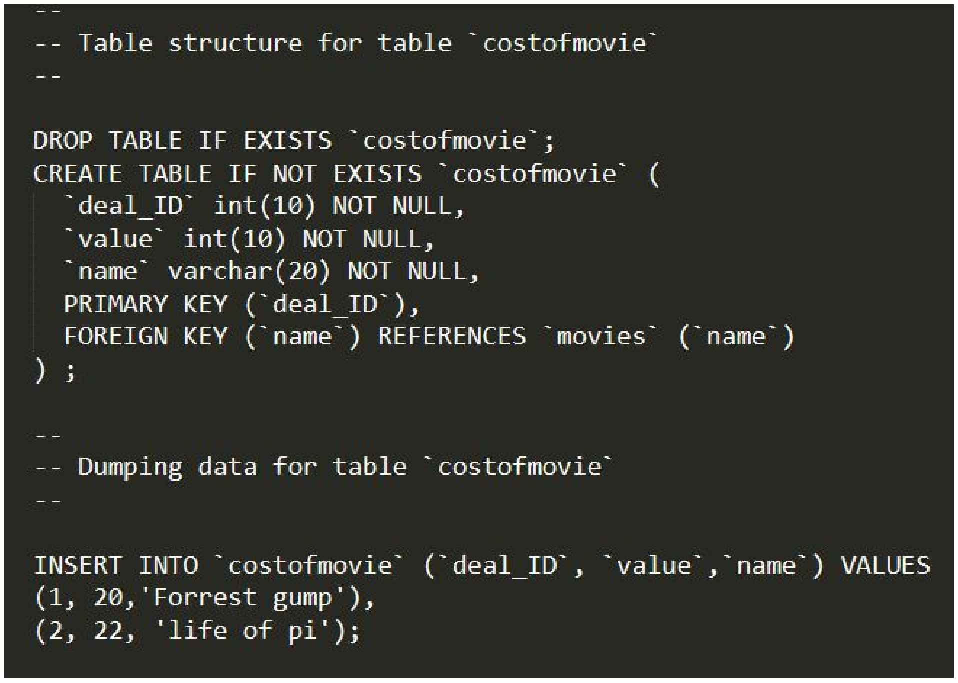
```sql
-- --------------------------------------------------------

--
-- Table structure for table `admin`
--

DROP TABLE IF EXISTS `admin`;
CREATE TABLE IF NOT EXISTS `admin` (
  `auth_id` int(10) NOT NULL,
  `password` int(10) NOT NULL,
  PRIMARY KEY (`auth_id`)
) ;

--
-- Dumping data for table `admin`
--

INSERT INTO `admin` (`auth_id`, `password`) VALUES
(1, 1121121),
(2, 222111);
```



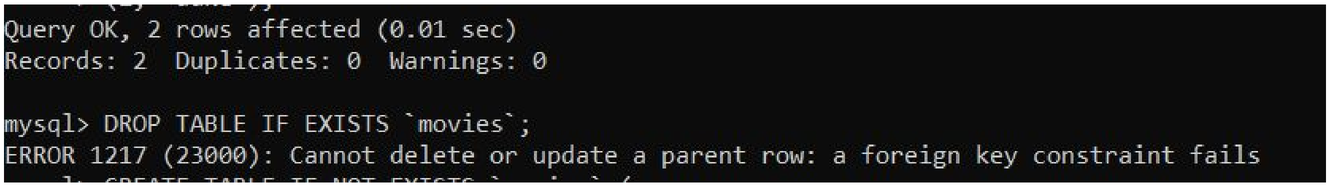
```sql
--
-- Table structure for table `accountteam`
--

DROP TABLE IF EXISTS `accounting`;
CREATE TABLE IF NOT EXISTS `accounting` (
  `username` varchar(10) NOT NULL,
  `password` int(10) NOT NULL,

  PRIMARY KEY (`username`)
);

--
-- Dumping data for table `accountteam`
--

INSERT INTO `accounting` (`username`, `password`) VALUES
('ringo', 223),
('paul', 3222);
```

Queries

1. Checking foreign key constraints

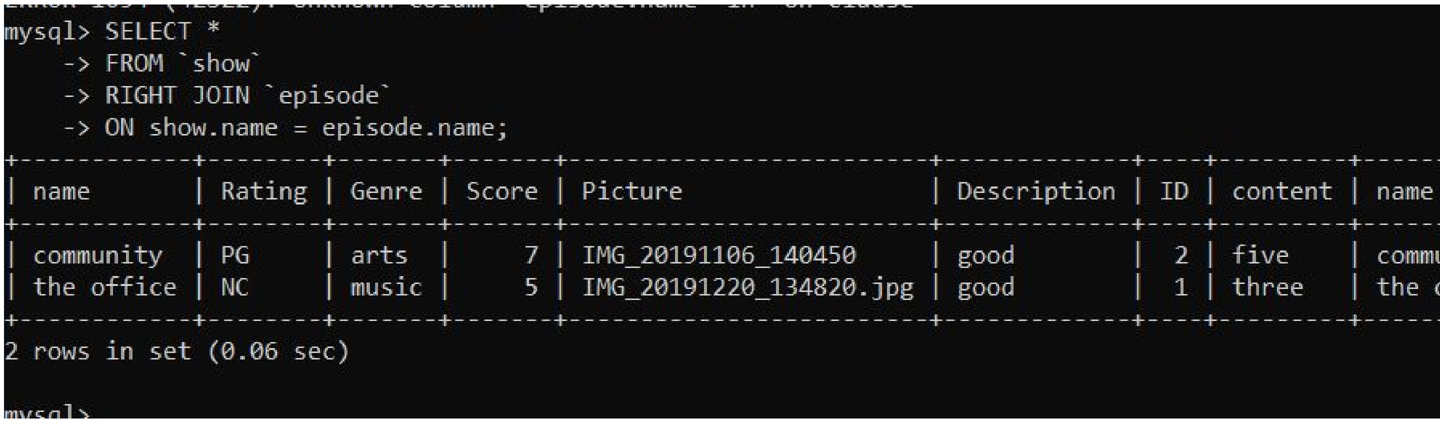


```
--
-- Table structure for table `costofmovie`
--

DROP TABLE IF EXISTS `costofmovie`;
CREATE TABLE IF NOT EXISTS `costofmovie` (
  `deal_ID` int(10) NOT NULL,
  `value` int(10) NOT NULL,
  `name` varchar(20) NOT NULL,
  PRIMARY KEY (`deal_ID`),
  FOREIGN KEY (`name`) REFERENCES `movies` (`name`)
) ;

--
-- Dumping data for table `costofmovie`
--

INSERT INTO `costofmovie` (`deal_ID`, `value`,`name`) VALUES
(1, 20,'Forrest gump'),
(2, 22, 'life of pi');
```



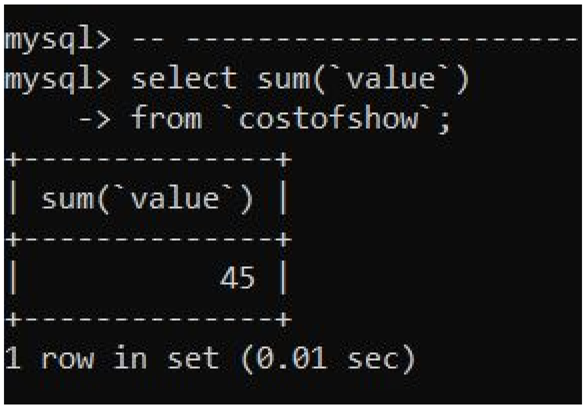
```
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> DROP TABLE IF EXISTS `movies`;
ERROR 1217 (23000): Cannot delete or update a parent row: a foreign key constraint fails
```

2. Seeing all episodes of all shows



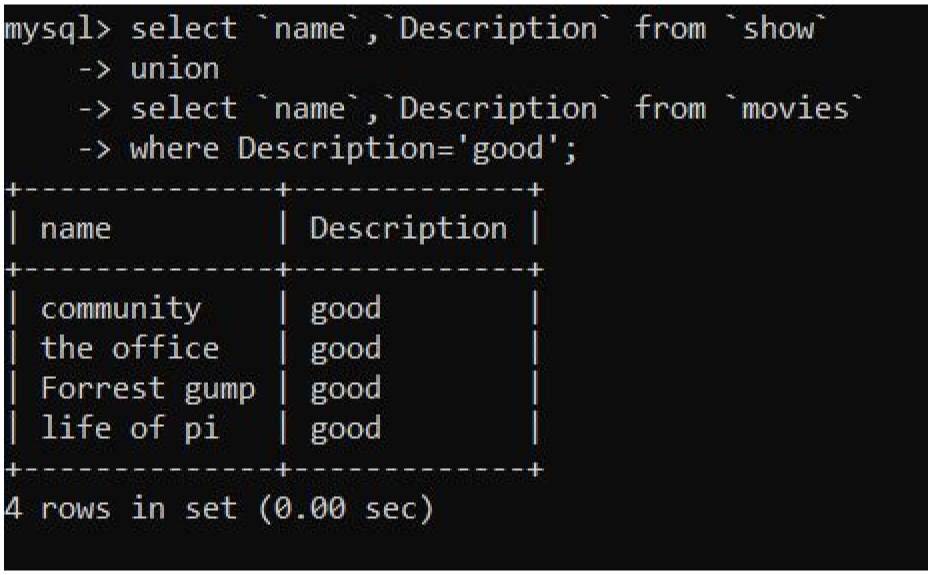
```
mysql> SELECT *
    -> FROM `show`
    -> RIGHT JOIN `episode`
    -> ON show.name = episode.name;
+------------+--------+-------+-------+-------------------------+-------------+----+---------+-----
| name       | Rating | Genre | Score | Picture                 | Description | ID | content | name
+------------+--------+-------+-------+-------------------------+-------------+----+---------+-----
| community  | PG     | arts  |     7 | IMG_20191106_140450     | good        |  2 | five    | commu
| the office | NC     | music |     5 | IMG_20191220_134820.jpg | good        |  1 | three   | the o
+------------+--------+-------+-------+-------------------------+-------------+----+---------+-----
2 rows in set (0.06 sec)

mysql>
```

3. Total cost to company of all shows



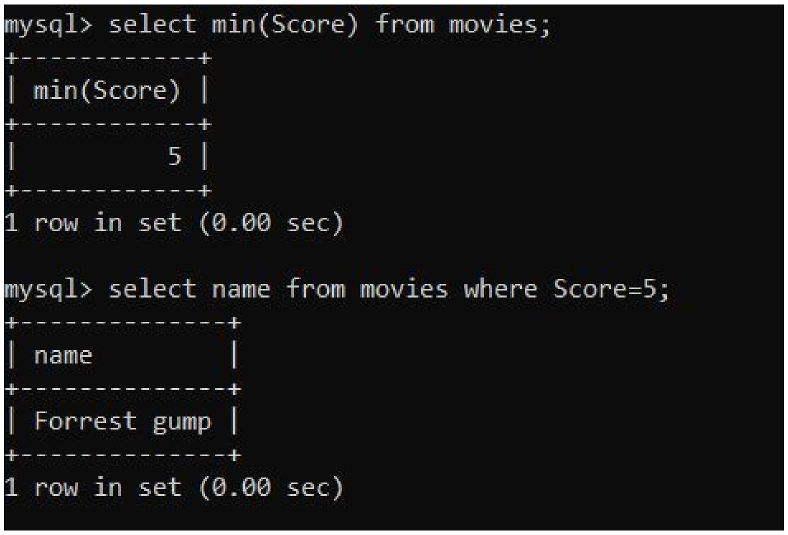
```
mysql> -- ----------------------
mysql> select sum(`value`)
    -> from `costofshow`;
+--------------+
| sum(`value`) |
+--------------+
|           45 |
+--------------+
1 row in set (0.01 sec)
```

4. All the good shows Union All the good movies



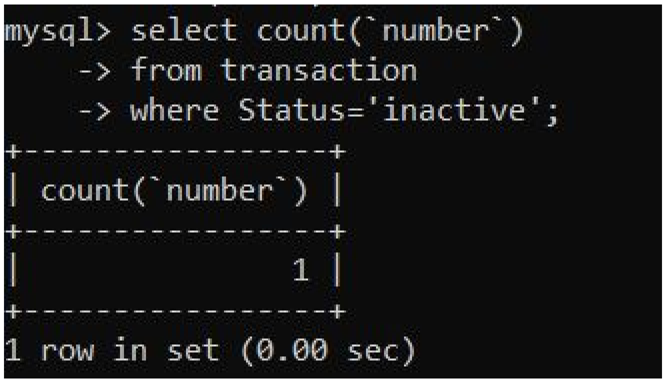
```
mysql> select `name`,`Description` from `show`
    -> union
    -> select `name`,`Description` from `movies`
    -> where Description='good';
+--------------+-------------+
| name         | Description |
+--------------+-------------+
| community    | good        |
| the office   | good        |
| Forrest gump | good        |
| life of pi   | good        |
+--------------+-------------+
4 rows in set (0.00 sec)
```

5. Movie with the minimum score



```
mysql> select min(Score) from movies;
+------------+
| min(Score) |
+------------+
|          5 |
+------------+
1 row in set (0.00 sec)

mysql> select name from movies where Score=5;
+--------------+
| name         |
+--------------+
| Forrest gump |
+--------------+
1 row in set (0.00 sec)
```

6. How many users have not paid?



```
mysql> select count(`number`)
    -> from transaction
    -> where Status='inactive';
+-----------------+
| count(`number`) |
+-----------------+
|               1 |
+-----------------+
1 row in set (0.00 sec)
```

Proposed changes

- The database should be able to give discounts for most frequent customers
- The database should other entertainment mediums and sports
- The database should allow rollbacks on the database

Conclusion

Web clients will be able to search movies/shows in the database.

Beside web clients the database will also contain admin users that are able to maintain the movie/show database. These admin users will login through another web application where they'll be able to search movies/shows by ID and edit these. Of course, they'll also be able to insert and delete movies/shows. For audit purposes the admin users will be able to view who last modified/created/deleted a movie/show and when.

Accounting team will use another web application in order to view deposits and costs associated with the Video Streaming platform. People from the accounting team will be identified based on their login name and password. These users will be able to generate and view profit and loss reports generated each month (total distribution cost for the shows added in the given month and total client's payments that month).