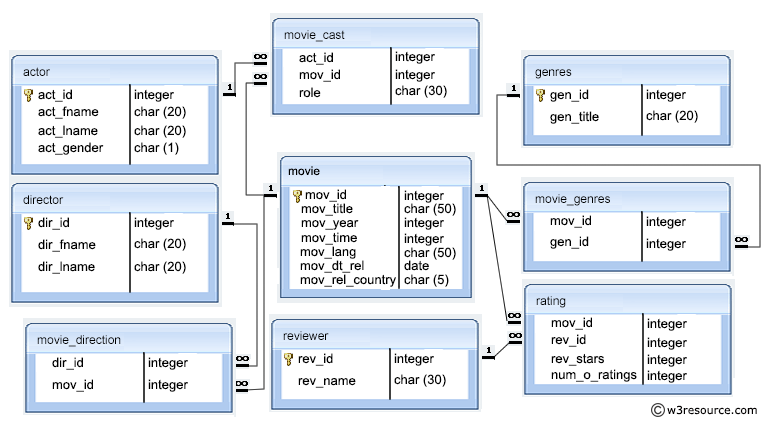
**SQL USE CASE:**

A XYZ corporation is a organization where they are designing database as per the requirement below cases should be satisfied.

**ER Diagram:**



Actor Director

Table

Description automatically generated Table

Description automatically generated

Genres Movie\_cast

A picture containing table

Description automatically generated Table

Description automatically generated

Movie

Table

Description automatically generated

Movie\_direction Movie\_genres

Table

Description automatically generated with medium confidence Table

Description automatically generated with medium confidence

Rating Reviewer

Table

Description automatically generated Table

Description automatically generated with medium confidence

**Section A**

1. Create Database project\_movie\_data and deploy table data according to ER diagram.

**Code:**

* **create database project\_movie\_data ;**
* **use project\_movie\_data ;**
* **create table actor (act\_id int not null , act\_fname varchar (20) , act\_lname varchar (20) , act\_gender varchar(4) , primary key ( act\_id ));**
* **create table director ( dir\_id int not null, dir\_fname varchar(20), dir\_lname varchar(20), primary key ( dir\_id ) );**
* **create table movie ( mov\_id int not null, mov\_title varchar(50), mov\_year int not null, mov\_time int not null, mov\_lang varchar(50), mov\_dt\_rel date, mov\_rel\_country varchar(5), primary key ( mov\_id) );**
* **create table reviewer (rev\_id int not null, rev\_name varchar(30), primary key ( rev\_id ));**
* **create table genres ( gen\_id int not null, gen\_title varchar(20), primary key (gen\_id) );**
* **create table movie\_cast ( act\_id int not null, mov\_id int not null, role varchar(20), foreign key ( act\_id ) references actor (act\_id), foreign key ( mov\_id ) references movie ( mov\_id) );**
* **create table movie\_direction ( dir\_id int not null, mov\_id int not null, foreign key ( dir\_id ) references director ( dir\_id ), foreign key ( mov\_id) references movie (mov\_id) );**
* **create table movie\_genres ( mov\_id int not null, gen\_id int not null, foreign key (mov\_id) references movie (mov\_id), foreign key (gen\_id) references genres (gen\_id) );**
* **create table rating ( mov\_id int not null, rev\_id int not null, rev\_stars int not null, num\_o\_rateing int not null, foreign key (mov\_id) references movie ( mov\_id ), foreign key (rev\_id) references reviewer (rev\_id));**

1. Create 9 tables which are presented in ER model and load the data with their foreign key and primary key values.

**Code:**

* **insert into actor (act\_id, act\_fname, act\_lname, act\_gender) values**

**(101, ' James ', ' Stewart ', 'M'),**

**(102, ' Deborah ', ' Kerr ', 'F'),**

**(103, ' Peter ', ' OToole ', 'M'),**

**(104, ' Robert ', ' De Niro ', 'M'),**

**(105, ' F. Murray', ' Abraham ', 'M'),**

**(106, ' Harrison ', ' Ford ', 'M'),**

**(107, ' Nicole ', ' Kidman ', 'F'),**

**(108, ' Stephen ', ' Baldwin ', 'M'),**

**(109, ' Jack ', ' Nicholson', 'M'),**

**(110, ' Mark ', ' Wahlberg ', 'M'),**

**(111, ' Woody ', ' Allen ', 'M'),**

**(112, ' Claire ', ' Danes ', 'F'),**

**(113, ' Tim ', ' Robbins ', 'M'),**

**(114, ' Kevin ', ' Spacey ', 'M'),**

**(115, ' Kate ', ' Winslet ', 'F'),**

**(116, ' Robin ', ' Williams ', 'M'),**

**(117, ' Jon ', ' Voight ', 'M'),**

**(118, ' Ewan ', ' McGregor ', 'M'),**

**(119, ' Christian', ' Bale ', 'M'),**

**(120, ' Maggie ', ' Gyllenhaa', 'F'),**

**(121, ' Dev ', ' Patel ', 'M'),**

**(122, ' Sigourney', ' Weaver ', 'F'),**

**(123, ' David ', ' Aston ', 'M'),**

**(124, ' Ali ', ' Astin ', 'F');**

* **select \* from actor;**
* **insert into director (dir\_id, dir\_fname, dir\_lname) values**

**(201, ' Alfred ', ' Hitchcock'),**

**(202, ' Jack ', ' Clayton'),**

**(203, ' David ', ' Lean'),**

**(204, ' Michael ', ' Cimino'),**

**(205, ' Milos ', ' Forman'),**

**(206, ' Ridley ', ' Scott'),**

**(207, ' Stanley ', ' Kubrick'),**

**(208, ' Bryan ', ' Singer'),**

**(209, ' Roman ', ' Polanski'),**

**(210, ' Paul ', ' Thomas An'),**

**(211, ' Woody ', ' Allen'),**

**(212, ' Hayao ', ' Miyazaki'),**

**(213, ' Frank ', ' Darabont'),**

**(214, ' Sam ', ' Mendes'),**

**(215, ' James ', ' Cameron'),**

**(216, ' Gus ', ' Van Sant'),**

**(217, ' John ', ' Boorman'),**

**(218, ' Danny ', ' Boyle'),**

**(219, ' Christoph', ' Nolan'),**

**(220, ' Richard ', ' Kelly'),**

**(221, ' Kevin ', ' Spacey'),**

**(222, ' Andrei ', ' Tarkovsky'),**

**(223, ' Peter ', ' Jackson');**

* **select \* from director;**
* **insert into genres (gen\_id, gen\_title) values**

**(1001, ' Action'),**

**(1002, ' Adventure'),**

**(1003, ' Animation'),**

**(1004, ' Biography'),**

**(1005, ' Comedy'),**

**(1006, ' Crime'),**

**(1007, ' Drama'),**

**(1008, ' Horror'),**

**(1009, ' Music'),**

**(1010, ' Mystery'),**

**(1011, ' Romance'),**

**(1012, ' Thriller'),**

**(1013, ' War');**

* **select \* from genres;**
* **insert into Movie (mov\_id, mov\_title, mov\_year, mov\_time, mov\_lang, mov\_dt\_rel, mov\_rel\_country) values**

**(901, ' Vertigo ', 1958, 128, ' English ', '1958-08-24', ' UK'),**

**(902, ' The Innocents ', 1961, 100, ' English ', '1962-02-19', ' SW'),**

**(903, ' Lawrence of Arabia ', 1962, 216, ' English ', '1962-12-11', ' UK'),**

**(904, ' The Deer Hunter ', 1978, 183, ' English ', '1979-03-08', ' UK'),**

**(905, ' Amadeus ', 1984, 160, ' English ', '1985-01-07', ' UK'),**

**(906, ' Blade Runner ', 1982, 117, ' English ', '1982-09-09', ' UK'),**

**(907, ' Eyes Wide Shut ', 1999, 159, ' English ', '0000-00-00', ' UK'),**

**(908, ' The Usual Suspects ', 1995, 106, ' English ', '1995-08-25', ' UK'),**

**(909, ' Chinatown ', 1974, 130, ' English ', '1974-08-09', ' UK'),**

**(910, ' Boogie Nights ', 1997, 155, ' English ', '1998-02-16', ' UK'),**

**(911, ' Annie Hall ', 1977, 93, ' English ', '1977-04-20', ' USA'),**

**(912, ' Princess Mononoke ', 1997, 134, ' Japanese ', '2001-10-19', ' UK'),**

**(913, ' The Shawshank Redem', 1994, 142, ' English ', '1995-02-17', ' UK'),**

**(914, ' American Beauty ', 1999, 122, ' English ', '0000-00-00', ' UK'),**

**(915, ' Titanic ', 1997, 194, ' English ', '1998-01-23', ' UK'),**

**(916, ' Good Will Hunting ', 1997, 126, ' English ', '1998-06-03', ' UK'),**

**(917, ' Deliverance ', 1972, 109, ' English ', '1982-10-05', ' UK'),**

**(918, ' Trainspotting ', 1996, 94, ' English ', '1996-02-23', ' UK'),**

**(919, ' The Prestige ', 2006, 130, ' English ', '2006-11-10', ' UK'),**

**(920, ' Donnie Darko ', 2001, 113, ' English ', '0000-00-00', ' UK'),**

**(921, ' Slumdog Millionaire', 2008, 120, ' English ', '2009-01-09', ' UK'),**

**(922, ' Aliens ', 1986, 137, ' English ', '1986-08-29', ' UK'),**

**(923, ' Beyond the Sea ', 2004, 118, ' English ', '2004-11-26', ' UK'),**

**(924, ' Avatar ', 2009, 162, ' English ', '2009-12-17', ' UK'),**

**(925, ' Braveheart ', 1995, 178, ' English ', '1995-09-08', ' UK'),**

**(926, ' Seven Samurai ', 1954, 207, ' Japanese ', '1954-04-26', ' JP'),**

**(927, ' Spirited Away ', 2001, 125, ' Japanese ', '2003-09-12', ' UK'),**

**(928, ' Back to the Future ', 1985, 116, ' English ', '1985-12-04', ' UK');**

* **select \* from Movie**
* **insert into movie\_cast (act\_id, mov\_id, role) values**

**(101, 901, ' John Scottie Fergus'),**

**(102, 902, ' Miss Giddens'),**

**(103, 903, ' T.E. Lawrence'),**

**(104, 904, ' Michael'),**

**(105, 905, ' Antonio Salieri'),**

**(106, 906, ' Rick Deckard'),**

**(107, 907, ' Alice Harford'),**

**(108, 908, ' McManus'),**

**(110, 910, ' Eddie Adams'),**

**(111, 911, ' Alvy Singer'),**

**(112, 912, ' San'),**

**(113, 913, ' Andy Dufresne'),**

**(114, 914, ' Lester Burnham'),**

**(115, 915, ' Rose DeWitt Bukater'),**

**(116, 916, ' Sean Maguire'),**

**(117, 917, ' Ed'),**

**(118, 918, ' Renton'),**

**(120, 920, ' Elizabeth Darko'),**

**(121, 921, ' Older Jamal'),**

**(122, 922, ' Ripley'),**

**(114, 923, ' Bobby Darin'),**

**(109, 909, ' J.J. Gittes'),**

**(119, 919, ' Alfred Borden');**

* **select \* from movie\_cast;**
* **insert into movie\_direction (dir\_id, mov\_id) values**

**(201, 901),**

**(202, 902),**

**(203, 903),**

**(204, 904),**

**(205, 905),**

**(206, 906),**

**(207, 907),**

**(208, 908),**

**(209, 909),**

**(210, 910),**

**(211, 911),**

**(212, 912),**

**(213, 913),**

**(214, 914),**

**(215, 915),**

**(216, 916),**

**(217, 917),**

**(218, 918),**

**(219, 919),**

**(220, 920),**

**(218, 921),**

**(215, 922),**

**(221, 923);**

* **select \* from movie\_direction;**
* **insert into movie\_genres (mov\_id, gen\_id) values**

**(922, 1001),**

**(917, 1002),**

**(903, 1002),**

**(912, 1003),**

**(911, 1005),**

**(908, 1006),**

**(913, 1006),**

**(926, 1007),**

**(928, 1007),**

**(918, 1007),**

**(921, 1007),**

**(902, 1008),**

**(923, 1009),**

**(907, 1010),**

**(927, 1010),**

**(901, 1010),**

**(914, 1010),**

**(906, 1012),**

**(904, 1013);**

* **select \* from movie\_genres;**
* **insert into reviewer ( rev\_id , rev\_name ) values**

**(9001,'Righty Sock'),**

**(9002,'Jack Malvern'),**

**(9003,'Flagrant Baronessa'),**

**(9004,'Alec Shaw'),**

**(9005,''),**

**(9006,'Victor Woeltjen'),**

**(9007,'Simon Wright'),**

**(9008,'Neal Wruck'),**

**(9009,'Paul Monka'),**

**(9010,'Mike Salvati'),**

**(9011,' '),**

**(9012,'Wesley S.Walker'),**

**(9013,'Sasha Goldshtein'),**

**(9014,'Josh Cates'),**

**(9015,'Krug Stillo'),**

**(9016,'Scott LeBrun'),**

**(9017,'Hannah Steele'),**

**(9018,'Vincent Cadena'),**

**(9019,'Brandt Sponseller'),**

**(9020,'Richard Adams');**

* **select \* from reviewer;**
* **insert into rating (mov\_id, rev\_id, rev\_stars, num\_o\_rateing) values**

**(901, 9001, '8', 263575),**

**(902, 9002, '8', 20207),**

**(903, 9003, '8', 202778),**

**(906, 9005, '8', 484746),**

**(924, 9006, '7', 0),**

**(908, 9007, '9', 779489),**

**(909, 9008, '0', 227235),**

**(910, 9009, '3', 195961),**

**(911, 9010, '8', 203875),**

**(912, 9011, '8', 0),**

**(914, 9013, '8', 862618),**

**(916, 9014, '4', 642132),**

**(925, 9015, '8', 81328),**

**(918, 9016, '0', 580301),**

**(920, 9017, '8', 609451),**

**(921, 9018, '8', 667758),**

**(922, 9019, '8', 511613),**

**(923, 9020, '7', 13091);**

1. Write a query in SQL to list the Horror movies?

Code:

* **select mov\_title**

**from Movie**

**where mov\_id in(**

**select mov\_id**

**from movie\_genres**

**where gen\_id in(**

**select gen\_id**

**from genres**

**where gen\_title=' Horror'));**

or

* **select mov\_title**

**from movie where**

**mov\_id=(select mov\_id from**

**movie\_genres**

**where gen\_id in (select**

**gen\_id from genres where**

**gen\_title=' Horror'));**

1. Write a query in SQL to find the name of all reviewers who have rated 8 or more stars?

Code:

* **SELECT reviewer.rev\_name**

**FROM reviewer, rating**

**WHERE rating.rev\_id = reviewer.rev\_id**

**AND rating.rev\_stars>=8**

**AND reviewer.rev\_name IS NOT NULL;**

Or

* **select rev\_name**

**from reviewer R, rating RR**

**where R.rev\_id = RR.rev\_id**

**and rev\_stars>=8;**

1. Write a query in SQL to list all the information of the actors who played a role in the movie ‘Deliverance’

Code:

* **SELECT \***

**FROM actor**

**WHERE act\_id IN(**

**SELECT act\_id**

**FROM movie\_cast**

**WHERE mov\_id IN (**

**SELECT mov\_id**

**FROM movie**

**WHERE mov\_title=' Deliverance '**

**));**

1. Write a query in SQL to find the name of the director (first and last names) who directed a movie that casted a role for ‘Eyes Wide Shut’. (using subquery)

Code:

* **SELECT dir\_fname, dir\_lname**

**FROM director**

**WHERE dir\_id in (**

**SELECT dir\_id**

**FROM movie\_direction**

**WHERE mov\_id in(**

**SELECT mov\_id**

**FROM movie\_cast WHERE role = ANY (**

**SELECT role**

**FROM movie\_cast**

**WHERE mov\_id IN (**

**SELECT mov\_id**

**FROM movie**

**WHERE mov\_title=' Eyes Wide Shut '))));**