Marc Tom Thorgersen, mthorg13
Søren Hvidberg Frandsen, sfrand12
Mathias Sass Michno, mmichn13
Troels Beck Krøgh, tkragh13

Self study 1 - Table Description for movie database

The database contains information of movies, the people who worked on the movies, users, and reviews made by these users. There is also information about awards, and which people and movies have won or been nominated for these awards.

Tables:

- **Movies**
  - This table has an ID for the movie, along with the title of the movie, its run-time and release date. The ID is used in other tables to relate the movies with other rows of other tables.
- **People**
  - A table that contains all persons that may be related to a role.
- **Crew**
  - Relates persons to movies and their role in the movies. Note that if a person has more than one role, additional rows are needed.
- **Role**
  - Describes the different roles, persons can have in a movie, e.g director, producer, actor, camera operator, assistants etc.
- **Award**
  - This table contains different awards with specification and year relevant to the given award.
- **MovieAward**
  - Relates movies awards and optionally a list ID that represents a group of persons. Moveover a bool indicates if the award is won or if it only is a nomination. The list ID is used for award such as “Best Cast”.
- **PersonList**
  - Lists persons for use in relation to awards. A list can contain several persons, and persons may be on several lists.
- **User**
  - Contains usernames and unique ID, and additional user information which we have not written here. It can be encrypted passwords, e-mails, birthdate etc.
- **Review**
  - This table relates users to movies and provides additional information about the specific review, such as rating, review title and review text

F-key = Foreign key
P-key = Primary key
Join table = Also know as junction tables, and defines many-to-many relations between entries in other tables.
OPT = Optional, can be NULL if not used.

Marc Tom Thorgersen, mthorg13
Søren Hvidberg Frandsen, sfrand12
Mathias Sass Michno, mmichn13
Troels Beck Krøgh, tkragh13

Movie:

Movie_ID (P-key)	Movie_Title	Movie_Runtime	Movie_Release_Date
1	Lord of The Rings: The Return of the King	201	2003
2	Deadpool	108	2016

People:

Person_ID (P-key)	Person_Name	Person_Birthdate	Person_PlaceOfBirth
1	Viggo Mortensen	1958-10-20	Denmark
2	Ryan Reynolds	1976-10-23	Canada

Crew (Join table):

Person_ID (F-key)	Movie_ID (F-Key)	Role_ID (F-Key)
1	1	1
1	2	1

Role:

Role_ID (P-key)	Role_Name
1	Actor

Award:

Award_ID (P-key)	4Award_Name	Award_Year
1	Oscar: Best Cast Ensemble	2003

MovieAward (Join table):

Movie_ID (F-key)	Award_ID (F-key)	List_ID (F-key, OPT)	Won? (bool)
1	1	1	1

PersonList (Join table):

List_ID (F-key)	Person_ID (F-key)
1	1

User:

User_ID (P-key)	User_Name	Additional User information
1	JensJensen	…

Review:

Review_ID (P-key)	User_ID (F-key)	Movie_ID (F-key)	Rating_Number	Review_Title	Review_Text
1	1	1	10	Best	Movie Ever
2	1	2	10	I like	Turtles

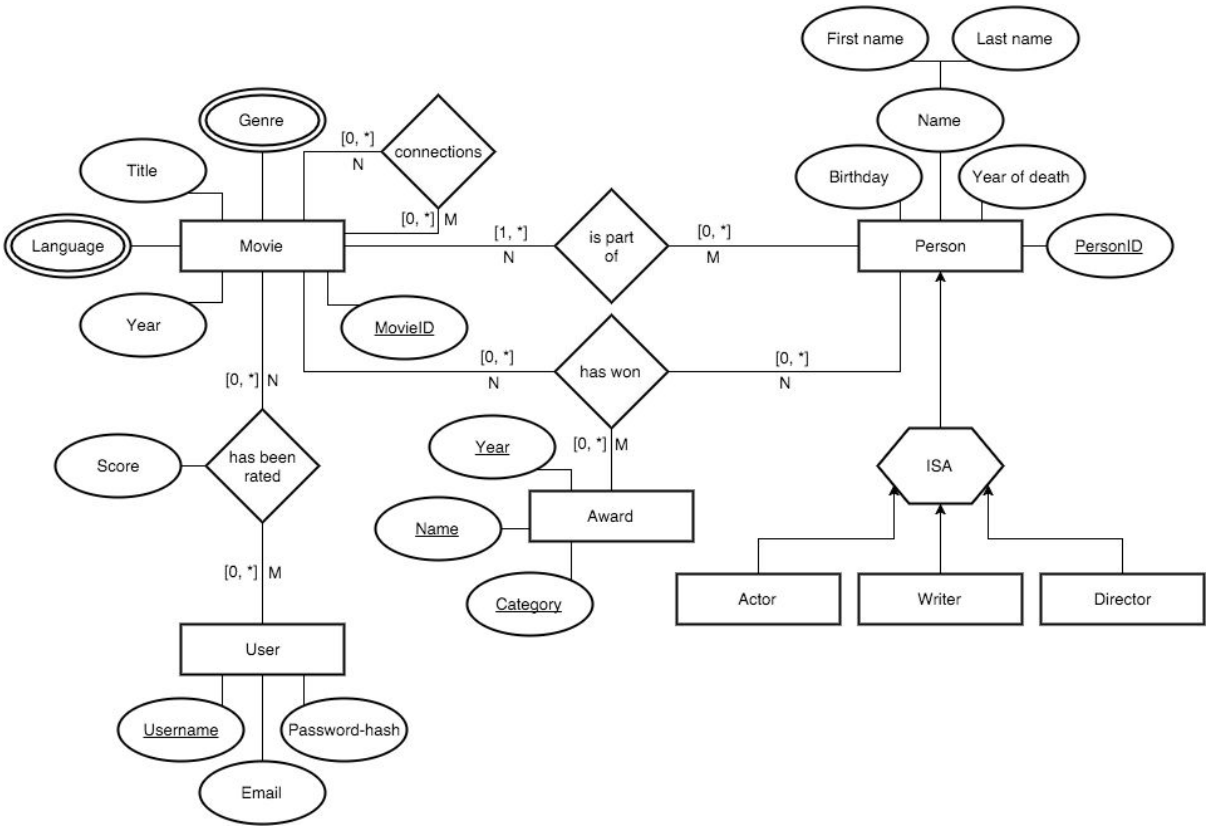
Marc Tom Thorgersen, mthorg13
Søren Hvidberg Frandsen, sfrand12
Mathias Sass Michno, mmichn13
Troels Beck Krøgh, tkragh13

Self study 2 - Entity Relationship Diagram

A) & B) - We did them in one go.

We have chosen to do an ISA pattern to use the inheritance since persons can have different kind of roles, i.e. writer or actor.

We have also used a compound attribute for Name on persons.
The name on awards is e.g. “Academy Award”, or “Golden Globe”.
On the entity “Movie”, “Genre” and “Language” are multi-value attributes.



C)

User: (Username, Email, Password-hash)
Has been rated: (MovieID, Username)
With foreign keys: {MovieID, Username} → {Movie.MovieID, User.Username}
Movie: (MovieID, Title, Language, Year, Genre}
Connections: (aMovieID, bMovieID)
With foreign keys: {aMovieID, bMovieID} → {Movie.MovieID, Movie.MovieID}
Person: (PersonID, First name, Last name, Birthday, Year of death)
Is part of: (MovieID, PersonID)
With foreign keys: {MovieID, PersonID} → {Movie.MovieID, Person.PersonID}
Award: (Year, Name, Category}
Has won: (Category, Year, Name, MovieID, PersonID}

With foreign keys {Category, Year, Name} → {Award.Category, Award.year, Award.Name}
(Partitioning is used here)
Actor: (PersonID)
With foreign key {PersonID} → {Person.PersonID}
Writer: (PersonID)
With foreign key {PersonID} → {Person.PersonID}
Director: (PersonID)
With foreign key {PersonID} → {Person.PersonID}

E)
We have removed the “roles” table, and instead used the ISA pattern.
We have also removed the AwardID, since we can use other attributes as keys.
We have removed the personList, and created the relationship type is part of.
We have also removed UserID, and use username as the key instead.
Reviews do not need a reviewID so that is gone as well, we have also removed some information on review title, but that is mostly because it seems the score is only relevant, so now that is just an attribute on the relationship type.

Marc Tom Thorgersen, mthorg13
Søren Hvidberg Frandsen, sfrand12
Mathias Sass Michno, mmichn13
Troels Beck Krøgh, tkragh13

Self study 3 - Refinement, normalization, and SQL-DDL

(a) Identify functional dependencies for each of the relations you created.

- User:

{ Username } → { Username, Email, Password-Hash }

- Has Been Rated:

{ MovieID, Username } → { MovieID, Username, Score }

- Movie:

{ MovieID } → { Title, Year, Genre, Language, MovieID }

- Connections:

{ aMovieID, bMovieID } → { aMovieID, bMovieID }

- Person:

{ PersonID } → { PersonID, First name, Last name, Birthday, Year of death }

- Is Part Of:

{ MovieID, PersonID } → { MovieID, PersonID}

- Award

{ Year, Name, Category } → { Year, Name, Category }

- Has Won:

{ Year, Name, Category } → { MovieID, PersonID, Year, Name, Category }

- Actor:

{ PersonID } → { PersonID }

- Writer:

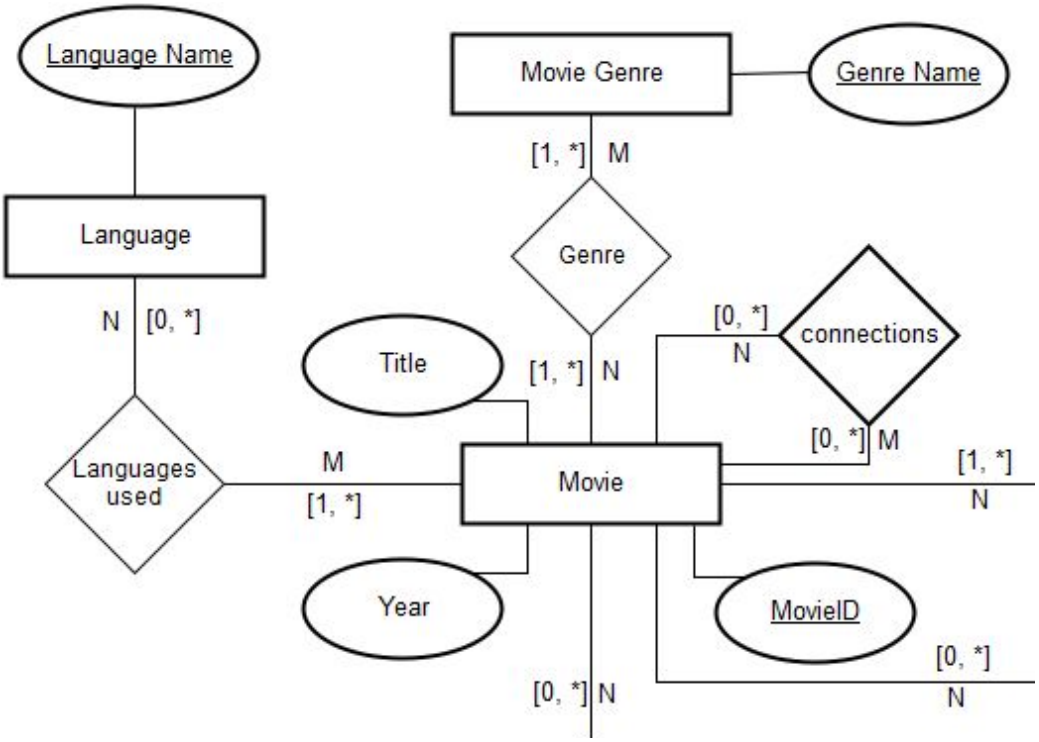
{ PersonID } → { PersonID }

- Director:

{ PersonID } → { PersonID }

(b) If necessary, transform your schema into 3NF and formally show that the result is actually in 3NF/BCNF.

Since we used multi-value attributes we need to change it in order to be in BCNF, the following ER diagram is constructed without them. Note that the picture is cropped unchanged information is omitted.



The following additions are made to the functional dependencies:

- Language

{ LanguageName } → { LanguageName }

- Languages used

{ MovieID } → { Language Name, MovieID }

- Movie Genre

{ GenreName } → { GenreName }

- Genre

{ MovieID } → { MovieID, Genre Name}

Moreover “Genre” and “Language” are removed from “Movie” in the functional dependencies listed in *(a)*

User, Movie and Person are all based on their primary key, therefore they follow the rule, “α is a super key of R”.

Award, Genre, Language Name, Actor, Writer and Director are all trivial.

The left side of the functional dependency of Has Been Rated is a super key of the relation. The same holds for the rest of the relationship types.

Thusly our schema is in BCNF and therefore also in 3NF

(c) Determine for each obtained relational schema the highest normal form it still supports, i.e., check if your relations are also in BCNF.

It is, see argument above.

(d) List the SQL statements that create all your tables properly and ensure the functional dependencies that you have identified.

```
CREATE TABLE Movie(
      MovieID INTEGER PRIMARY KEY NOT NULL,
      Title VARCHAR(250),
      /* For example this (existing) movie title is very long:
      'Night Of The Day Of The Dawn Of The Son Of The Bride Of The
      Return Of The Revenge Of The Terror Of The Attack Of The
      Evil, Mutant, Hellbound, Flesh-Eating, Crawling, Alien,
      Zombified, Subhumanoid Living Dead - Part 5'
      */
      vYear DATE);

CREATE TABLE Award (
      vYear DATE NOT NULL,
      vName VARCHAR(100) NOT NULL,
      Category VARCHAR(100) NOT NULL,
      PRIMARY KEY(vYear , vName, Category));

CREATE TABLE vUser (
      Username VARCHAR(32) PRIMARY KEY NOT NULL,
      Email VARCHAR(256),
      Password_Hash CHAR(64));

CREATE TYPE vNAME AS (
      First_Name VARCHAR(32),
      Last_Name VARCHAR(32));

CREATE TABLE Person (
      vName NAME,
```

```
	Birthday DATE,
	Year_Of_Death DATE,
	PersonID INTEGER PRIMARY KEY NOT NULL);


CREATE TABLE Actor (
	PersonID INTEGER PRIMARY KEY NOT NULL);

CREATE TABLE Director (
	PersonID INTEGER PRIMARY KEY NOT NULL);


CREATE TABLE Writer (
	PersonID INTEGER PRIMARY KEY NOT NULL);

CREATE TABLE Movie_Genre (
	Genre_Name VARCHAR(32) PRIMARY KEY NOT NULL);

CREATE TABLE Language (
	Language_Name VARCHAR(32) PRIMARY KEY NOT NULL);

CREATE TABLE Is_Part_Of (
	MovieID INTEGER REFERENCES Movie NOT NULL,
	PersonID INTEGER REFERENCES Person NOT NULL);

CREATE TABLE Has_Been_Rated (
	MovieID INTEGER REFERENCES Movie NOT NULL,
	Username VARCHAR(32) REFERENCES vUser NOT NULL,
	Score NUMERIC(3,1) CHECK (Score <= 10.0));

CREATE TABLE Has_Won (
	vYear DATE NOT NULL,
	vName VARCHAR(250) NOT NULL,
	Category VARCHAR(100) NOT NULL,
	MovieID INTEGER REFERENCES Movie NOT NULL,
	PersonID INTEGER REFERENCES Person NOT NULL,
	FOREIGN KEY (vYear, vName, Category) REFERENCES Award (vYear,
vName, Category));

CREATE TABLE Languages_Used (
	MovieID INTEGER REFERENCES Movie NOT NULL,
	Language_Name VARCHAR(32) REFERENCES Language NOT NULL);

CREATE TABLE Genre (
	MovieID INTEGER REFERENCES Movie NOT NULL,
	Genre_Name VARCHAR(32) REFERENCES Movie_Genre NOT NULL);
```

```
CREATE TABLE Connections (
      aMovieID INTEGER REFERENCES Movie NOT NULL,
      bMovieID INTEGER REFERENCES Movie NOT NULL);
```

Reflections

We have changed it from multi valued attributes into relationship types using many to many relations, in order to accommodate BCNF (and 3NF).

Self Study 5

1 | Setting up Postgres

Install PostgreSQL from `http://www.postgresql.org/download/`. Ensure that the `pqsl` is in the path of your preferred terminal. Ensure that PostgreSQL is running by browsing to it in the pgAdmin III tool. Ensure that PostgreSQL is running on port 5432 by looking in the pgAdmin III tool. It might be nesseary to set a enviroment varible with the password: `export PGPASSWORD=yourpasswordhere`. Then run the following commands:

```
psql -U postgres -h localhost -p 5432 -c "create database ss"
# OUTPUT:
CREATE DATABASE

psql -U postgres -h localhost -p 5432 -d ss -f schema.sql
# OUTPUT:
REATE TABLE
CREATE TABLE
/* ... */

psql -U postgres -h localhost -p 5432 -d ss -f insert.sql
# OUTPUT:
INSERT 0 251
INSERT 0 58179
/* ... */
```

Henceforth the database is ready to use, and should have the appropriate data. The database can be queried in the commandline by:

```
# Using a query directly
psql -U postgres -h localhost -p 5432 -d ss -c "select * from movie limit 10;"

# Using a file (query.sql)
psql -U postgres -h localhost -p 5432 -d ss -f query.sql

# Or interactive (NOT RECOMMENDED)
psql -U postgres -h localhost -p 5432 -d ss
```

Alternatively the GUI query tool can be used, this is opened though the pgAdmin III tool.

2 | Updates to the database

```
update person
  set gender='f'
  where id = 666;
```

```
update movie
  set title='Calculator Jokes'
  where id = 8008135;
```

3 | Queries

3.1 | How many Danish language movies are in the database?

```
select count(*)
  from movie as m
  where m.language = 'Danish';
```

```
 count
-------
   420
(1 row)
```

3.2 | For each year, what is the total number of reviews to movies from that year?

```
select year, count(*)
  from movie as m
  join ratings as r on r.movieId = m.id
  group by year
  order by year;
```

```
 year | count
------+-------
 1962 |     1
 1991 |     6
 1992 |     3
 1993 |     4
 1994 |    33
 1995 |    21
 1997 |    17
 1998 |    29
 1999 |    48
 2000 |    24
/* ... */
(21 rows)
```

3.3 | Which movies have John Travolta and Uma Thurman starred in together?

```
select m.title
  from movie as m
  join involved as i1 on i1.movieId = m.id
  join involved as i2 on i2.movieId = m.id
  join person as p1 on p1.id = i1.personId
  join person as p2 on p2.id = i2.personId
  where p1.name = 'John Travolta' and p2.name = 'Uma Thurman';
```

```
                        title
-------------------------------------------------------
 "Wetten, dass..?"
 "The Oprah Winfrey Show"
 "Entertainment Tonight"
 "Good Morning America"
 "Late Show with David Letterman"
 "The Tonight Show with Jay Leno"
 "E! True Hollywood Story"
 "Live with Regis and Kathie Lee"
 "The Charlie Rose Show"
 "The Rosie O'Donnell Show"
/* ... */
(21 rows)
```

3.4 | How many actors and directors have a first name starting with "Q"?

```
select i.role, count(*)
  from involved as i
  join person as p on i.personId = p.id
  where p.name like 'Q%'
  group by i.role;
```

```
   role    | count
-----------+-------
 actor     |  2100
 director  |    45
(2 rows)
```

3.5 | How many users rated at least 3 movies?

```
select count(*)
  from
  (
    select count(*)
    from ratings
    group by userId
    having count(*) >= 3
  ) as ratedby3ormore;
```

```
 count
-------
    34
(1 row)
```

3.6 | What is the name and birth year of all actors in "Pulp Fiction"?

```
select p.name, p.birthdate
  from person as p
  join involved as i on p.id = i.personId
  join movie as m on m.id = i.movieId
  where m.title = 'Pulp Fiction'
  order by p.birthdate;
```

```
        name          | birthdate
----------------------+------------
 Emil Sitka           | 1914-12-22
 Harvey Keitel        | 1939-05-13
 Rene Beard           | 1941-06-03
 Christopher Walken   | 1943-03-31
 Joseph Pilato        | 1949-03-16
 Brenda Hillhouse     | 1953-12-11
 John Travolta        | 1954-02-18
 Bruce Willis         | 1955-03-19
 Amanda Plummer       | 1957-03-23
 Lawrence Bender      | 1957-10-17
 /* ... */
(46 rows)
```

3.7 | What are the titles and years of all movies from the 1980s that John Travolta starred in?

```
select m.title, m.year
  from movie as m
  join involved as i on i.movieId = m.id
  join person as p on p.id = i.personId
  where p.name = 'John Travolta'
    and m.year >= 1980
    and m.year <= 1989
  order by year;
```

```
              title                | year
-----------------------------------+------
 Urban Cowboy                      | 1980
 "Wetten, dass..?"                 | 1981
 "Entertainment Tonight"           | 1981
 Blow Out                          | 1981
 Staying Alive                     | 1983
 Two of a Kind                     | 1983
 Perfect                           | 1985
 "Larry King Live"                 | 1985
 That's Dancing!                   | 1985
 "The Oprah Winfrey Show"          | 1986
/* ... */
(15 rows)
```

3.8 | What are the top-2 highest rated movies (average) from the 1990s according to the users?

```
select m.title, avg(r.rating)
  from movie as m
  join ratings as r on r.movieId = m.id
  where m.year >= 1990
    and m.year <= 1999
  group by m.title
  order by avg(r.rating) DESC
  limit 2;
```

```
          title           |        avg
--------------------------+-------------------
 The Usual Suspects       | 9.3333333333333333
```

```
 The Shawshank Redemption | 9.1250000000000000
(2 rows)
```

3.9 | What are the top-2 highest rated movies (average) from the 1990s according to at least 2 users?

```
select m.title, avg(r.rating)
  from movie as m
  join ratings as r on r.movieId = m.id
  where m.year >= 1990
    and m.year <= 1999
  group by m.title
  having count(r.rating) >= 2
  order by avg(r.rating) DESC
  limit 2;
```

```
          title            |        avg
---------------------------+-------------------
 The Usual Suspects        | 9.3333333333333333
 The Shawshank Redemption  | 9.1250000000000000
(2 rows)
```

3.10 | In 1994, what was the average rating of a movie for each language?

```
select m.language, avg(r.rating)
  from movie as m
  join ratings as r on r.movieId = m.id
  where m.year = 1994 and m.language != ''
  group by m.language
  order by avg(r.rating);
```

```
 language |        avg
----------+-------------------
 English  | 8.3043478260869565
 French   | 9.0000000000000000
(2 rows)
```

3.11 | Which actors in "Pulp Fiction" have never, before or after, starred in the same movie as one of the other actors in "Pulp Fiction"?

```
with pf as (
  select i1.personId
    from movie as m
    join involved as i1 on i1.movieId = m.id
    where m.title = 'Pulp Fiction'
),
act as (
  select distinct pf1.personid
    from pf as pf1
    join involved as i1 on i1.personid = pf1.personid
    join involved as i2 on i1.movieid = i2.movieid
    join pf as pf2 on i2.personid = pf2.personid
    where i1.personid != i2.personid
      and i1.movieid != (select id from movie where title = 'Pulp Fiction')
)
```

```
select p.name from pf
  join person as p on p.id = pf.personid
  where not exists (select NULL from act where act.personid = pf.personid);
```

```
      name
-----------------
 Kim A. Jakobsen
(1 row)
```

3.12 | Which movie starring John Travolta has the highest user ratings?

```
select m.title, avg(r.rating)
  from movie as m
  join ratings as r on r.movieId = m.id
  join involved as i1 on i1.movieId = m.id
  join person as p1 on p1.id = i1.personId
  where p1.name = 'John Travolta'
  group by m.id, m.title
  order by avg(rating) DESC
  limit 1;
```

```
    title     |        avg
--------------+-------------------
 Pulp Fiction | 9.0000000000000000
(1 row)
```

3.13 | How many actresses have not been alive at the same time as Charles Chaplin?

```
with cc as (select * from person where name = 'Charles Chaplin')

select count(*)
  from person as p, cc
  where p.gender = 'f'
  and (p.birthdate > cc.deathdate
    or cc.birthdate > p.deathdate);
```

```
 count
-------
  5405
(1 row)
```

3.14 | What is the average rating of movies from each genre?

```
select g.genre, avg(r.rating)
  from genre as g
  join ratings as r on r.movieId = g.movieId
  group by g.genre
  order by avg(r.rating) DESC;
```

```
   genre   |        avg
-----------+-------------------
 Crime     | 8.3949579831932773
 War       | 8.2692307692307692
```

```
 Biography | 8.1333333333333333
 History   | 8.1250000000000000
 Mystery   | 8.0895522388059701
 Drama     | 7.8423236514522822
 Thriller  | 7.7543859649122807
 Action    | 7.5347826086956522
 Sci-Fi    | 7.4710144927536232
 Adventure | 7.4479166666666667
/* ... */
(16 rows)
```

3.15 | What is the average rating of movies from each genre? List only genres with at least 2 ratings.

```
select g.genre, avg(r.rating)
  from genre as g
  join ratings as r on r.movieId = g.movieId
  group by g.genre
  having count(r.rating) >=2;
```

```
   genre    |        avg
------------+--------------------
 Drama      | 7.8423236514522822
 Fantasy    | 7.2685185185185185
 Comedy     | 7.2380952380952381
 Biography  | 8.1333333333333333
 Thriller   | 7.7543859649122807
 Crime      | 8.3949579831932773
 War        | 8.2692307692307692
 History    | 8.1250000000000000
 Adventure  | 7.4479166666666667
 Sci-Fi     | 7.4710144927536232
/* ... */
(15 rows)
```

3.16 | Which movie has the largest number of 2-link references?

```
select m.title, r.refs
        from movie as m
        join (select r1.fromid as id, count(r2.toid) as refs
                    from movieref as r1
                    join movieref as r2 on r1.toid = r2.fromid
                    group by r1.fromid)
        as r on m.id = r.id
            order by r.refs desc
            limit 1;
```

```
         title          | count
------------------------+-------
 "Saturday Night Live" | 38834
(1 row)
```

3.17 | How many actors have also been active as director of at least one movie?

```
select count(*)
        from person as p
        join (select distinct personid
                    from involved
                        where role = 'actor')
        as ia on p.id = ia.personid
        join (select distinct personid
                    from involved
                        where role = 'director')
        as ib on p.id = ib.personid;
```

```
 count
-------
  5930
(1 row)
```

3.18 | Which two genres are most often linked to the same movie? (Note that each movie has a set of genres.)

```
select g1.genre, g2.genre, count(*)
  from genre as g1
  join genre as g2 on g1.movieId = g2.movieId
  where g1.genre != g2.genre
  group by g1.genre, g2.genre
  having g1.genre < g2.genre /* To avoid symetric dublicates */
  order by count(*) DESC
  limit 1;
```

```
 genre |  genre  | count
-------+---------+-------
 Drama | Romance |  5837
(1 row)
```