CS585 Homework2
You Wu

Our database have the following 5 tables:

- users (**id** , name, date_of_birth).
- movies (**id** , name, genre, release_date)
- reviews (**user_id, movie_id** , rating, comment)
- actors (**id** , name, gender, date_of_birth)
- lead (**actor_id, movie_id**)

To finish the assignment, first we need to create a database witch contains these 5 tables, the creation statement are shown below:

```
CREATE DATABASE hw2;
USE hw2;
CREATE TABLE users(
        id VARCHAR(10) NOT NULL UNIQUE,
        name VARCHAR(30) NOT NULL,
        date_of_birth DATE NOT NULL,
        PRIMARY KEY (id)
);

CREATE TABLE movies(
        id VARCHAR(10) NOT NULL UNIQUE,
        name VARCHAR(30) NOT NULL,
        genre VARCHAR(15) NOT NULL,
        release_date DATE NOT NULL,
        PRIMARY KEY (id)
);

CREATE TABLE reviews(
        user_id VARCHAR(10)  NOT NULL,
        movie_id VARCHAR(10) NOT NULL,
        rating INTEGER(10) NOT NULL,
        comment  VARCHAR(5000),
        PRIMARY KEY(user_id, movie_id),
        FOREIGN KEY(user_id) REFERENCES users(id),
        FOREIGN KEY(movie_id) REFERENCES movies(id)
);

CREATE TABLE actors(
        id VARCHAR(10) NOT NULL UNIQUE,
        name VARCHAR(30) NOT NULL,
        gender VARCHAR(15) NOT NULL,
        date_of_birth DATE NOT NULL,
        PRIMARY KEY (id)
);


CREATE TABLE lead(
        actor_id VARCHAR(10) NOT NULL,
        movie_id VARCHAR(10) NOT NULL,
        PRIMARY KEY(actor_id, movie_id),
        FOREIGN KEY(actor_id) REFERENCES actors(id),
        FOREIGN KEY(movie_id) REFERENCES movies(id)
);
```

Note:

1. We assume that users can only give a integer rating score, where 0<=rating<=10.
2.

Insertion for table users:

```
insert into users(id,name,date_of_birth) values ("008","Kidd",'2006-04-06');

insert into users(id,name,date_of_birth) values ("002","Oliver",'1974-04-
28');

insert into users(id,name,date_of_birth) values ("035","You",'1994-11-15');

insert into users(id,name,date_of_birth) values ("003","John Doe",'1988-10-
03');
```

Insertion for table movies:

```
insert into movies(id,name,genre,release_date) values
("001","Notebook",”Romantic drama”,'2012-11-11');

insert into movies(id,name,genre,release_date) values ("002","The
Avengers",”Action”,'2012-05-25');
```

Insertion for table reviews:

```
insert into reviews (user_id,movie_id,rating,comment) values
("035","001",7,"good enough!");

insert into reviews (user_id,movie_id,rating,comment) values
("008","001",3,"REAL BAD...");

insert into reviews (user_id,movie_id,rating,comment) values
("002","001",5,"so so");
```

```
insert into reviews (user_id,movie_id,rating,comment) values
("002","002",10,"100!!!");

insert into reviews (user_id,movie_id,rating,comment) values
("003","001",8,"good");

insert into reviews (user_id,movie_id,rating,comment) values
("003","002",3,"good");
```

Q1:

Subquery showing the movie id witch name is Notebook

```
Select id from movies where name=”Notebook”;
```

Subquery showing the user id who rated the movie “001” with rating less or equal tro 8:

```
Select user_id from reviews where rating<=8 AND movie_id=”001”;
```

Join the table users,reviews and movies to a same table, and choose the entries with corresponding requests:

```
Select users.name FROM users,reviews,movies WHERE users.id=reviews.user_id AND
reviews.movie_id=movies.id AND movies.name="Notebook" AND
MONTH(users.date_of_birth)=4 ORDER BY name DESC;
```

Q2:

Check in the review table whether there are users whose named is John Doe :

```
EXISTS( Select id from user where users.id=reviews.user_id AND name=”John
Doe” )
```

Join the table movies and reviews first, then group it by genre and select it:

```
Select genre FROM movies,reviews where reviews.movie_id=movies.id GROUP BY
genre
```

Join the table movies and reviews and group it by genre, then select the average rating for each genre category:

```
Select genre,AVG(rating) AS avg_rating FROM movies,reviews where
reviews.movie_id=movies.id GROUP BY genre
```

Join the table movies, reviews and users and group it by genre, then select the average rating for each genre category where the user name is John Doe:

```
Select genre,AVG(rating) AS avg_rating FROM movies,reviews,users where
reviews.movie_id=movies.id AND reviews.user_id=users.id AND users.name="John
Doe" GROUP BY genre;
```

Select the Maximum number of the average rating table obtained above:

```
Select MAX(subq.avg_rating) FROM(Select genre,AVG(rating) AS avg_rating FROM
movies,reviews,users where reviews.movie_id=movies.id AND
reviews.user_id=users.id AND users.name="John Doe" GROUP BY genre) as subq
```

The answer of this question:

```
Select genre,name from movies
Where EXISTS(
Select subq1.genre FROM (Select genre,AVG(rating) AS avg_rating FROM
movies,reviews,users where reviews.movie_id=movies.id AND
reviews.user_id=users.id AND users.name="John Doe" GROUP BY genre ORDER BY
genre ) as subq1 WHERE subq1.avg_rating=( Select MAX(subq2.avg_rating)
FROM(Select genre,AVG(rating) AS avg_rating FROM movies,reviews,users where
reviews.movie_id=movies.id AND reviews.user_id=users.id AND users.name="John
Doe" GROUP BY genre) as subq2) AND subq1.genre=movies.genre
)order by genre,name ASC;
```

3.

The number of male actors for a specific movie_id:

```
SELECT COUNT(lead.actor_id) FROM lead,actors WHERE lead.actor_id=actors.id
AND actors.gender=″Male″ AND lead.movie_id=movies.id
```

The number of female actors for a specific movie_id:

```
SELECT COUNT(lead.actor_id) FROM lead,actors WHERE lead.actor_id=actors.id
AND actors.gender=″Female″ AND lead.movie_id=movies.id
```

List all the movie id where the number of male actors is greater than the number of female actress:

```
SELECT movies.id FROM movies WHERE (SELECT COUNT(lead.actor_id) FROM
lead,actors WHERE lead.actor_id=actors.id AND actors.gender=″Male″ AND
lead.movie_id=movies.id
)>( SELECT COUNT(lead.actor_id) FROM lead,actors WHERE
lead.actor_id=actors.id AND actors.gender=″Female″ AND
lead.movie_id=movies.id) ORDER BY movies.id DESC;
```

4.

Average rating for each movie_id:

```
SELECT AVG(rating) FROM reviews GROUP BY movie_id;
```

The average of movie average ratings:

```
SELECT AVG(avg_rating) FROM (SELECT AVG(rating) as avg_rating FROM reviews
GROUP BY movie_id) as avg_table;
```

SELECT movie_id,AVG(rating) FROM reviews GROUP BY movie_id;

```
SELECT movies.name FROM movies WHERE EXISTS(
SELECT reviews.movie_id,AVG(rating) as avg FROM reviews GROUP BY
reviews.movie_id HAVING reviews.movie_id=movies.id AND avg>( SELECT
AVG(avg_rating) FROM (SELECT AVG(rating) as avg_rating FROM reviews GROUP BY
movie_id) as avg_table )
) AND movies.genre= "comedy" AND YEAR(release_date)<2006 ORDER BY
movies.name;
```

5.

List all the lead information that related to actor Mark Clarkson:

```
SELECT * FROM lead,actors WHERE lead.actor_id=actors.id AND actors.name="Mark
Clarkson";
```

To check whether the movie (movie_id) is related with actor Mark Clarkson

```
EXISTS(SELECT * FROM lead,actors WHERE lead.actor_id=actors.id AND
lead.movie_id= NOW AND actors.name="Mark Clarkson");
```

Answer of this question:

```
SELECT movie_id, AVG(rating) as avg FROM reviews GROUP BY movie_id HAVING
avg>=9 AND EXISTS(SELECT * FROM lead,actors WHERE lead.actor_id=actors.id AND
lead.movie_id= reviews.movie_id AND actors.name="Mark Clarkson");
```

6.

Need to self join the table lead(A) and lead(B), also self join the actors table as (a_1) and (a_2):

```
SELECT a_1.name,a_2.name,(SELECT COUNT(A.actor_id) FROM lead as A,lead as B
WHERE A.actor_id<>B.actor_id AND A.movie_id=B.movie_id AND A.actor_id=a_1.id
AND B.actor_id=a_2.id) AS cnts FROM actors AS a_1,actors AS a_2 WHERE
a_1.id<>a_2.id
```