**READEME**

*12191632 Jinyong Yun*

*After creating the tables and inserting the initial data, I added a trigger function to automatically update the avgRate of ‘movie’ whenever the ‘customerRate’ table is updated. Like Output Example 1 of the PPT, I accessed the ID by searching clause, but like in Output Example 2, I accessed it by directly passing the ID to the function, and used both methods interchangeably.*

**Brief explanation**

2.1 : I added ‘Best Actor in a Leading Role’ to the ‘award’ table, and added the corresponding award to ‘actorObtain’. I used a select statement to find the awardID and actorID, which were used to add the information to ‘actorObtain’

2.2: Added 'Best Supporting Actor' to the award table and added the corresponding information to actorObtain table.

2.3: Added 'Best Direction' to the award table and added the corresponding information to directorObtain table.

2.4: Added 'Best Director' twice with different years and 'Best Writing' to the award table, and added the corresponding information to directorObtain table.

2.5: Added 'Best Picture' to the award table and added the corresponding information to movieObtain table.

2.6: Added 'Best Sound' to the award table and added the corresponding information to movieObtain table.

2.7: Added 'Best Performance by an Actor in a Leading Role' to the award table and printed the corresponding information to actorObtain table.

3.1: Since the trigger function has already been written, adding the information to customerRate automatically updates the avgRate of the movie.

3.2: Added the ids of the movies 'Titanic' and 'Avatar' directed by James Cameron to the customerRate table.

3.3: Retrieved the movieIDs of Kate Winslet and Zoe Saldana and added them to the customerRate table.

3.4: Retrieved the movieIDs that belong to the 'fantasy' genre from movieGenre table and added them to the customerRate table.

3.5: Retrieved the movieIDs that have 'paramount pictures' as the publisher from the movie table and added them to the customerRate table.

4. Perform an inner join between casting and actor tables, group by casting.movieID (using actorID) and filter out any rows where dateOfDeath is not null (meaning the actor is still alive). This will select movies featuring actors who have already passed away. Then perform another inner join with the movie table to retrieve the movieName.

5. Join the make and casting tables using movieID, group by directorID and actorID. Use the having clause to select only those records where the count of movieID is greater than 1 (meaning the director and actor worked together on more than one movie). Then join this result with the director table and retrieve the directorName.

6. Perform a natural join between the movie and movieGenre tables. This will add the genreName column to the movie table. Then group the table by genreName (using movieID) and select only those records where the count of movieID is greater than 1 (meaning the movies have at least one genre in common).

7. Delete all records from the movie table where releaseYear is less than 2001. The cascade option was set when declaring the table, so any associated records in other tables will also be deleted automatically.

8. Keep the table but use a select statement to find all customerIDs and delete them. The cascade option was set when declaring the table, so any associated records in other tables will also be deleted automatically.

9. Delete the tables in the reverse order of their reference to avoid any foreign key constraints.

movieGenre,movieObtain,actorObtain,casting,make,customerRate,directorObtain

director,actor,movie,award,genre,customer

**HOW TO COMPILE AND RUN & RESULT DISPLAY**

**COMPILE ENVIRONMENT: intellij IDEA 2023.1**

**"C:\Program Files\Microsoft\jdk-11.0.16.101-hotspot\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.1\lib\idea\_rt.jar=58439:C:\Program Files\JetBrains\IntelliJ IDEA 2023.1\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\jinyo\IdeaProjects\project\_movie\out\production\project\_movie;C:\Users\jinyo\Downloads\postgresql-42.6.0.jar Main**

**PostgreSQL JDBC Driver Registered!**

**org.postgresql.jdbc.PgConnection@4b5d6a01**

**You made it, take control your database now!**

**Table created!**

**Initial data inserted!**

**2.insert the proper data from the following statements**

**problem:2.1**

**Statement: Marlon Brando won the "Best Actor in a Leading Role" award in 1973**

**Translated SQL: insert into award(awardName) values ('Best Actor in a Leading Role')**

**Translated SQL: select awardID from award where awardName = 'Best Actor in a Leading Role')**

**Translated SQL: select actorID from actor where actorName = 'Marlon Brando'**

**Translated SQL: insert into actorObtain values (" + marlonid + "," + balrawardid + ",1973)**

**Updated tables: award, actorObtain**

**----------------< award >----------------**

**awardID awardName**

**1 Best Actor in a Leading Role**

**----------------< actorObtain >----------------**

**actorID awardID year**

**1 1 1973**

**problem:2.2**

**Statement:Robert Duvall won the "Best Supporting Actor" award in 1980**

**Translated SQL: insert into award values (2, 'Best Supporting Actor')**

**Translated SQL: insert into actorObtain values (2,2,1980)**

**Updated tables: award, actorObtain**

**----------------< award >----------------**

**awardID awardName**

**2 Best Supporting Actor**

**----------------< actorObtain >----------------**

**actorID awardID year**

**2 2 1980**

**problem:2.3**

**Statement:The movie Francis Ford Coppola won the "Best Direction" award in 1980**

**Translated SQL: insert into award values (3, 'Best Direction')**

**insert into directorObtain values (1,3,1980)**

**Updated tables: award, directorObtain**

**----------------< award >----------------**

**awardID awardName**

**3 Best Direction**

**----------------< directorObtain >----------------**

**directorID awardID year**

**1 3 1980**

**problem:2.4**

**Statement:James Cameron won the "Best Director" award in 1998 and 2010, "Best Writing" award in 2010**

**Translated SQL: insert into award values (4, 'Best Director')**

**Translated SQL: insert into award values (5, 'Best Writing')**

**Translated SQL: insert into directorObtain values (2,4,1998)**

**Translated SQL: insert into directorObtain values (2,4,2010)**

**Translated SQL: insert into directorObtain values (2,5,2010)**

**Updated tables: award, directorObtain**

**----------------< directorObtain >----------------**

**directorID awardID year**

**2 4 1998**

**directorID awardID year**

**2 4 2010**

**directorID awardID year**

**2 5 2010**

**problem:2.5**

**The movie Titanic won the "Best Picture" award in 1998**

**Translated SQL:insert into award values (6,'Best Picture')**

**insert into movieObtain values (4,6,1998)**

**Updated tables: award, movieObtain**

**----------------< award >----------------**

**awardID awardName**

**6 Best Picture**

**----------------< movieObtain >----------------**

**movieID awardID year**

**4 6 1998**

**problem:2.6**

**Statement:The movie Dunkirk won the "Best Sound" award in 2018**

**Translated SQL: insert into award values (7,'Best Sound')**

**Translated SQL: insert into movieObtain values (5,7,2018)**

**Updated tables: actor, award, actorObtain**

**----------------< award >----------------**

**awardID awardName**

**7 Best Sound**

**----------------< movieObtain >----------------**

**movieID awardID year**

**5 7 2018**

**problem:2.7**

**Statement:Leonardo DiCaprio won the "Best Performance by an Actor in a Leading Role" award in 2016**

**Translated SQL: insert into award values (8, 'Best Performance by an Actor in a Leading Role')**

**Translated SQL: insert into actorObtain values (6,8,2016)')**

**Updated tables: award, actorObtain**

**----------------< award >----------------**

**awardID awardName**

**8 Best Performance by an Actor in a Leading Role**

**----------------< actorObtain >----------------**

**actorID awardID year**

**6 8 2016**

**3. insert date to the proper tables based on the following statements and update avgRate if necessary**

**problem:3.1**

**Statement:Jill rates 5 to "Apocalypse Now"**

**Translated SQL: insert into customerRate values (4,1,5)**

**Updated tables: customerRate, movie(Trigger used)**

**----------------< customerRate >----------------**

**customerID movieID rate**

**4 1 5**

**----------------< movie >----------------**

**movieID movieName avgRate**

**1 Apocalypse Now 5**

**problem:3.2**

**Statement:Bill rates 5 to the movies whose director is "James Cameron"**

**Translated SQL: insert into customerRate values (5,3,5)**

**Translated SQL: insert into customerRate values (5,4,5)**

**Updated tables: customerRate, movie(Trigger used)**

**----------------< customerRate >----------------**

**customerID movieID rate**

**5 3 5**

**customerID movieID rate**

**5 4 5**

**----------------< movie >----------------**

**movieID movieName avgRate**

**3 Avatar 5**

**movieID movieName avgRate**

**4 Titanic 5**

**problem:3.3**

**Statement:Bob rates 4 to the movies whose main actor is female**

**Translated SQL: insert into customerRate values (1,3,4)**

**Translated SQL: insert into customerRate values (1,4,4)**

**Updated tables: customerRate, movie(Trigger used)**

**----------------< customerRate >----------------**

**customerID movieID rate**

**5 3 5**

**customerID movieID rate**

**5 4 5**

**customerID movieID rate**

**1 3 4**

**customerID movieID rate**

**1 4 4**

**----------------< movie >----------------**

**movieID movieName avgRate**

**3 Avatar 4**

**movieID movieName avgRate**

**4 Titanic 4**

**3.4**

**Statement:Jack rates 4 to the Action movies**

**Translated SQL: select distinct moveid from movieGenre where genreName = 'Action'**

**Translated SQL: insert into customerRate values(3, actionresults.getInt(movieID),4)**

**Updated tables: customerRate, movie(Trigger used)**

**----------------< customerRate >----------------**

**customerID movieID rate**

**3 5 4**

**customerID movieID rate**

**3 6 4**

**----------------< movie >----------------**

**movieID movieName avgRate**

**5 Dunkirk 4**

**movieID movieName avgRate**

**6 Inception 4**

**problem:3.5**

**Statement:John rates 5 to the movies produced by Paramount Picture**

**Translated SQL: select distinct moveid from movie where publisherName = 'Paramount Pictures'**

**Translated SQL: insert into customerRate values(2,pararesults.getInt(movieID),5)**

**Updated tables: customerRate, movie(Trigger used)**

**----------------< customerRate >----------------**

**customerID movieID rate**

**5 4 5**

**customerID movieID rate**

**1 4 4**

**customerID movieID rate**

**2 2 5**

**customerID movieID rate**

**2 4 5**

**----------------< movie >----------------**

**movieID movieName avgRate**

**2 The Godfather 5**

**movieID movieName avgRate**

**4 Titanic 4**

**problem:4**

**Statement:Select the names of the movies whose actor is dead**

**Translated SQL: select distinct moveid from movie where publisherName = 'Paramount Pictures'**

**Translated SQL: insert into customerRate values(2, pararesults.getInt(movieID) ,5)**

**----------------< movie >----------------**

**movieName**

**Apocalypse Now**

**The Godfather**

**problem:5**

**Statement:Select the names of the directors who cast the same actor more than once**

**Translated SQL: SELECT DISTINCT D.directorName**

**FROM (**

**SELECT M.directorID, C.actorID**

**FROM make M**

**JOIN casting C ON M.movieID = C.movieID**

**GROUP BY M.directorID, C.actorID**

**HAVING COUNT(DISTINCT C.movieID) > 1**

**) MandC**

**JOIN director D ON D.directorID = MANDC.directorID**

**----------------< director >----------------**

**directorName**

**Francis Ford Coppola**

**problem:6**

**Statement:Select the name of the movies and the genres, where movies have the common genre**

**Translated SQL: select genreName, movieName**

**from movie natural join movieGenre**

**where movieGenre.genreName in (select genreName**

**from movieGenre**

**group by genreName**

**having count(movieID) > 1)**

**----------------< genreName and movieName >----------------**

**genreName movieName**

**Drama Apocalypse Now**

**War Apocalypse Now**

**Drama The Godfather**

**Action Avatar**

**Adventure Avatar**

**Drama Titanic**

**Action Dunkirk**

**Drama Dunkirk**

**War Dunkirk**

**Action Inception**

**Adventure Inception**

**problem:7**

**Statement:Delete the movies which were released before 2000**

**Translated SQL: select movieID from movie where movie.realseYear < 2001**

**----------------< movie >----------------**

**genreName movieName ReleaseDate**

**3 Avatar 2009**

**5 Dunkirk 2017**

**6 Inception 2010**

**----------------< make >----------------**

**movieID directorID**

**3 2**

**5 3**

**6 3**

**problem:8**

**Statement:Delete all customers and delete data from related tables**

**Translated SQL: delete from customer where customerID =customerid[0]**

**Translated SQL: delete from customer where customerID =customerid[1]**

**Translated SQL: delete from customer where customerID =customerid[2]**

**Translated SQL: delete from customer where customerID =customerid[3]**

**Translated SQL: delete from customer where customerID =customerid[4]**

**----------------< customer >----------------**

**customerID customerName gender**

**----------------< customerRate >----------------**

**customerID movieID rate**

**Statement:Delete all tables and data**

**Translated SQL: DROP TABLE movieGenre,movieObtain,actorObtain,casting,make,customerRate,directorObtain**

**Translated SQL: DROP TABLE director,actor,movie,award,genre,customer**

**Drop all tables : complete**

**Process finished with exit code 0**

**TALK ABOUT MY EXPERIENCE OF DOING THIS PROJECT**

I couldn't be happier when I finished this assignment. I regretted not doing this assignment before the midterm exam because I think I would have become more proficient in using PostgreSQL. I was relieved that the assignment was completed quickly, contrary to my concerns. It was physically and mentally exhausting to struggle with various queries to get the desired results, but on the other hand, I was grateful and happy that I was able to accomplish this much on my own. I heard that the final project will be even more difficult, but I'm already looking forward to it to see how challenging it will be

**CONTACT**

**E-mail:** [**dbswlsdyd730@naver.com**](mailto:dbswlsdyd730@naver.com)

**Phone: 010-2510-6462**