DATA607: Week 2 Assignment

DH Kim

September 5, 2020

Overview

This assignment exercises database as a method to store data and do data analysis. The week 1 assignment dealt with data with a single text file with extension csv. It is one way widely used for data analysis. Another is, as will be shown, relational database and sql which I in this assignment will explore with an example of movie recommendation. I think that, through the assignment, it is important to think about how to store and handle data. Each would have strength and weakness.

Movie Database

As will be seen, the survey results are stored in a database. Before going into that, I think it is worthy to note that. The table below shows data on movie ratings of six movies from five friends (A to E), which is a good way to display data. The format is exactly what was in my mind as a way to store the survey results in a csv file with comma delimited. From data perspective, it is called *wide format*. Note that (1) there are multiple columns for movie ratings and (2) it can be converted to *long format* which has just one column for the rating.

	A	B	C	D	E
Kingsman	5	4	4	4	5
GoneGirl	4	3	4	4	2
Parasite	5	5	4	5	3
Robot&Frank	4	5	4	5	5
CashBack	3	4	3	3	5
ARainyDayInNewYork	2	3	3	2	2

The database used is Oracle database 11g Express Edition with SQL developer which can be freely downloaded from their websites. I think that it should be done with other databases, MySQL and PostgreSQL, and also R markdown which I am working with. The R markdown with DBI connection would be great but it remains as future work. Here, we will work with Oracle SQL developer to create the movie database. The three steps do the work: (1) Create a user named *moviedb*, (2) Create three tables named *moving_ratings*, *friends*, and *movies* and (3) insert data into relevant tables. It can be done by executing two sql files (*setup_moviedb.sql* and *moviedb.sql*) attached. The more detailed procedure is as follows:

Step 1: To create a user, connect to the 'system' in Oracle SQL developer. It can be done with the green "+" icon. Then in the worksheet, execute *setup_moviedb.sql* file. Or type the "CREATE USER" command and press the green forward arrow to run it.

Step 2: Connect to the *moviedb* user created.

Step 3: To create tables and insert records, run *moviedb.sql* using "CREATE TABLE" and "INSERT INTO" command.

Movie Recommendation

We now have data on movie recommendation. SQL provides very powerful in data analysis (refer to *prac.sql* file for some sql queries). The *movies* and *movie_ratings* data are shown in the Appendix. Note that, among other things, the rating data is in long format.

Product recommendation is an interesting field. Below shows a simple movie recommendation by just averaging movie ratings from friends. Another would be just adding the ratings of five friends. However, average is more robust than sum in the cast of missing values. The query is as follows:

```
SELECT movie_title, AVG(movie_rating) AS RecommendScore
FROM movie_ratings
  JOIN movies
    ON movie_ratings.movie_id = movies.movie_id
GROUP BY movie_title
ORDER BY RecommendScore desc;
```

Robot & Frank (2013, Ford) is the highest.

	RecommendScore
Robot&Frank	4.6
Kingsman	4.4
Parasite	4.4
CashBack	3.6
GoneGirl	3.4
ARainyDayInNewYork	2.4

Findings and Conclusions

During the database exercise with movie recommendation, I realized that it is interesting and important to understand various aspects of data storage and handling. It is not simple and straightforward to store data. So it needs to think about efficient ways of managing data. Relational database with SQL is an efficient way and has strength in security. The use of csv file and R-SQL connection are future works.

Appendix

The *movie_ratings* table is as follows:

```
knitr::include_graphics("pre_result2.pdf")
```

RATING_ID	FRIEND_ID	MOVIE_ID	MOVIE_RATING
1	1	1	5
2	1	2	4
3	1	3	5
4	1	4	4
5	1	5	3
6	1	6	2
7	2	1	4
8	2	2	3
9	2	3	5
10	2	4	5
11	2	5	4
12	2	6	3
13	3	1	4
14	3	2	4
15	3	3	4
16	3	4	4
17	3	5	3
18	3	6	3
19	4	1	4
20	4	2	4
21	4	3	5
22	4	4	5
23	4	5	3
24	4	6	2
25	5	1	5
26	5	2	2
27	5	3	3
28	5	4	5
29	5	5	5
30	5	6	2

The *movie* table is as follows:

```
knitr::include_graphics("pre_result1.pdf")
```

MOVIE_ID	MOVIE_TITLE	MOVIE_YEAR	MOVIE_DIRECTOR
1	KingsmanOne	2015	Vaughn
2	Gone Girl	2014	Fincher
3	Parasite	2019	Bong
4	RobotAndFrank	2013	Ford
5	Cashback	2006	Ellis
6	A Rainy Day in New York	2019	Allen

```
knitr::include_graphics("result1.pdf")
```

MOVIE_TITLE	MOVIE_YEAR	FRIEND_NAME	MOVIE_RATING
KingsmanOne	2015	E	5
KingsmanOne	2015	D	4
KingsmanOne	2015	C	4
KingsmanOne	2015	B	4
KingsmanOne	2015	A	5
Gone Girl	2014	E	2
Gone Girl	2014	D	4
Gone Girl	2014	C	4
Gone Girl	2014	B	3
Gone Girl	2014	A	4
RobotAndFrank	2013	E	5
RobotAndFrank	2013	D	5
RobotAndFrank	2013	C	4
RobotAndFrank	2013	B	5
RobotAndFrank	2013	A	4
Cashback	2006	E	5
Cashback	2006	D	3
Cashback	2006	C	3
Cashback	2006	B	4
Cashback	2006	A	3
A Rainy Day in New York	2019	E	2
A Rainy Day in New York	2019	D	2
A Rainy Day in New York	2019	C	3
A Rainy Day in New York	2019	B	3
A Rainy Day in New York	2019	A	2
Parasite	2019	E	3
Parasite	2019	D	5
Parasite	2019	C	4
Parasite	2019	B	5
Parasite	2019	A	5

```
knitr::include_graphics("result2.pdf")
```

MOVIE_TITLE	RECOMMENDSCORE
RobotAndFrank	4.6
KingsmanOne	4.4
Parasite	4.4
Cashback	3.6
Gone Girl	3.4
A Rainy Day in New York	2.4