Department of Computer Engineering

University of Peradeniya

CO226-Database Systems

Lab Number : 02
Topic : Writing SQL Queries – Part II
Lab Date : 04th March 2020 from 2:00 PM to 4:00 PM
Due Date : 10th March 2020 before 11:55 PM
Submission : Submit the queries and results of task01 and task02 in a text file named E16XXXLab02.txt

Lab Task01: (20 marks)

Suppose that you have started a new movie-rating website and you have been collecting data on reviewer's rating of various movies.

Figure01 shows a certain instance of the populated database. Log into MySQL server and create a database named E16XXXLab02. Create necessary tables in the database considering the following:

- Decide suitable names and data types for each field,
- Define primary keys and foreign keys for each table,
- Choose referential integrity options that should be used on each of the following operations
    - ON UPDATE
    - ON DELETE

MOVIE

Movie ID	Title	Year	Director
101	Gone with the Wind	1939	Victor Fleming
102	Star Wars	1977	George Lucas
103	The Sound of Music	1965	Robert Wise
104	E.T.	1982	Steven Spielberg
105	Titanic	1997	James Cameron
106	Snow White	1937	NULL
107	Avatar	2009	James Cameron
108	Raiders of the Lost Ark	1981	Steven Spielberg

REVIEWER

Reviewer ID	Reviewer Name
201	Sarah Martinez
202	Daniel Lewis
203	Brittany Harris
204	Mike Anderson
205	Chris Jackson
206	Elizabeth Thomas
207	James Cameron
208	Ashley White

RATING

Reviewer ID	Movie ID	Stars	Rating Date
201	101	2	2011-01-22
201	101	4	2011-01-27
202	106	4	null
203	103	2	2011-01-20
203	108	4	2011-01-12
203	108	2	2011-01-30
204	101	3	2011-01-09
205	103	3	2011-01-27
205	104	2	2011-01-22
205	108	4	null
206	107	3	2011-01-15
206	106	5	2011-01-19
207	107	5	2011-01-20
208	104	3	2011-01-02

Figure 01: An instance of 'Movie Rating' database

Lab Task02: (75 marks = 3marks x 25)

Write the following SQL queries using MySQL, to retrieve the data from the database, you created in task01 above.

1. Find all the details about the movies presented in the populated **MOVIE** table.
2. Find all the details about the movies directed by '*James Cameron*'.
3. Find all the details about the movies directed by '*James Cameron*', on or after year 2000.
4. Find all the **stars** presented in the rating table.
5. Find the distinct **stars** presented in the table.
6. Find **movie id**s and each movie's **director**.
7. Find **movie id**s, **title**s, **year**s of the movies directed by '*Steven Spielberg*'.
8. Obtain the Cartesian product of the details presented in two tables **MOVIE** and **RATING**.
9. Obtain the Cartesian product of the **movie id** and **title** from **MOVIE** table with **movie id**, **reviewer id** and **stars** from **RATING** table.
10. Select **movie id**s of each movie with its **title**, **reviewer id** and **stars** received.
11. Select **movie id**s of each movie with its **title**, **reviewer id** and **stars** received, where number of **stars** are less than or equal to three.
12. Select **movie id**s of each movie with its **title**, **reviewer id** and **stars** received, where the number of **stars** is between two and four (two and four inclusive).
13. Select **reviewer id**s with the corresponding **movie id**s reviewed by each reviewer.
14. Select distinct tuples from the results produced by the execution of the above query (query number 14).
15. Select each **movie id** with its corresponding **title**, **reviewer id**, **reviewer name** and **stars** received.
16. Select each **movie id** with its corresponding **title**, **reviewer id**, **reviewer name** and **stars** received, where the number of **stars** received is equal to five.
17. Select movie **title** with its corresponding reviewer **name** and **stars**, where movie's **rating date** is missing.
18. Select all the movie **director** names and reviewer **name**s into one column. Do not include null values.
19. Select the details about the reviewers who have a last name called '*Martinez*'.
20. Select the details about the ratings which have been rated before the 10th day of the month. Use substring comparison.
21. Write the above query (query number 20) without using substring comparison.
22. Show the effect of giving one more star to the movies reviewed by '*Brittany Harris*'. Here select relevant details from the **RATING** table.
23. Select movie **title**s with its reviewer **name** and **stars** received. Order the result by movie **title** in the alphabetical order.
24. Select movie **title**s with its **stars** received and **rating date**. Order the result by **movie title** in the alphabetical order, then by **stars** and **rating date** both in descending order.
25. Write a nested query to retrieve the details of the movies directed by a director who is also a reviewer.