CS360 Introduction to Database – Midterm (SQL Programming)

Please submit the SQL statement and the query result for each query. The *italic* word means a column name or a table name in DB. '[CS360] MySQL_additional_grammar.pdf' in KLMS can be helpful for you to solve problems. Read file '[CS360] Midterm-Guidelines.pdf' in KLMS for guidelines how to connect to MySQL and how to submit your queries and results.

Q1. (4pts) For each combination of category and actor, we want to find *category_id*, *actor_id*, and the number of the films which belong to the category, and at the same time, in which the actor appears. We want to find only the cases where the number of such films is more than five.

Q2. (5pts) For each store, we want to find *store_id* and the number of the films that are kept in inventory by the store, but have been rented less than five times. (Note: mySQL does not support the EXCEPT operation. So, you must write SQL without using EXCEPT.)

Q3. (6pts) We want to find *film_id*, *title*, and the number of rentals of the films that are rented in between 2005.06.01 and 2005.08.31. We want to find only the films which has been rented more than 26 times. We also want the films to be sorted by the number of rentals in descending order.

Q4. (5pts) We want to find *customer_id* and the number of overdues for every customer of the store of *store_id*=1. The condition of overdue is the gap between CURRENT_DATE() and *rental_date* is more than 30, and at the same time, *return_date* is NULL (i.e., not be returned yet). For a customer who has no overdue, the number of overdues in output should be NULL. Only print out the customers which *customer_id* is less than 100 as the query result.

Q5. (6pts) We want to find *customer_id*, *first_name*, and *last_name* of all the customers who have rented ALL the films which satisfy the following conditions: *film_id* is greater than or equal to 10, and at the same time, less than 20; *rental_rate* is greater than 4; at least one prolific actor appears. A prolific actor is the actor who has appeared in more than 30 films. (Note: mySQL does not support the DIVISION operation. So, you must write SQL using NOT EXISTS two times.)

Q6. (6pts) Each film has a different number of inventories. Let avgKeep be the average number of inventories for a film. Then, we want to find *film_id* and the number of rentals of the films that are kept in inventory more than avgKeep, and at the same time, have been rented greater than or equal to 30 times. We also want the films to be sorted by *film_id* in ascending order.

Q7. (5pts) For each country, we want to find country name, the number of stores, and the sum of incomes of all stores in the country. The income of a store can be calculated as the sum of *amount* attribute values in the *payment* tuples that are related to the store.

Q8. (5pts) For each category of each store, we want to find *store_id*, *name* of the category, and the number of rentals, if the rentals are done more than 500 times for the category in the store. We also want the result to be sorted by *store_id* in ascending order and sorted by the number of rentals in descending order.

Q9. (8pts) For each country and each category, we want to find country name, category name, and *actor_id* of the top actor in 2005. The top actor in a country and a category means that the films in which an actor has appeared have the highest income in the category and in the stores of the country. The income means the sum of *amount* values in the *payment* tuples. There exists a single top actor for each combination of country and category.