CS150A Quiz 1 Solutions

Basic SQL Queries

Assume there exists a table called "Songs" with the following columns:
`song_id (Int, Primary Key), artist_name (Text), title (Text), year_released (Int), length_seconds (Int), rating (Float)`

An example record could look like the following:
`(1, 'D.O.D.', 'Crazy Concurrency', 2007, 188, 10.0)`

Q1: Which SQL query (or queries) will get the number of songs released after 2010 with a rating of at least 9.0? (+1 point for each correct query, -1 point for each incorrect query -- minimum of 0 points)

C, D

~~`SELECT COUNT(*) FROM Songs GROUP BY year_released, rating HAVING year_released > 2010 AND rating >= 9.0;`~~
This is incorrect. Suppose the Songs table contains only 5 songs - 2 songs with (year_released, rating) = (2011, 9.5), and 3 songs with (year_released, rating) = (2012, 9.0). Then, the output of the query will be a column with elements 2 and 3.

~~`SELECT COUNT(*) FROM Songs WHERE rating >= 9.0 GROUP BY year_released HAVING year_released > 2010;`~~
This is incorrect. We can use the same counterexample to show that the output of the query could contain multiple elements.

```
SELECT COUNT(*) FROM Songs WHERE year_released < 2010 AND rating >= 9.0;
```

```
SELECT COUNT(song_id) FROM Songs WHERE year_released < 2010 AND rating >= 9.0;
```

Q2: Which SQL query (or queries) will get the list of artists, without duplicates, who have produced at least one song more than 5 minutes long? (+1 point for each correct query, -1 point for each incorrect query -- minimum of 0 points)

C, D

~~`SELECT artist_name FROM Songs WHERE length_seconds > 300 GROUP BY artist_name, length_seconds HAVING COUNT(*) >= 1;`~~
This is incorrect. Suppose the Songs table contains only 3 songs - 2 songs with (artist_name, length_seconds) = ('Led Zeppelin', 329), and 1 song with (artist_name, length_seconds) = ('Led Zeppelin', 482). Then, the output of the query will be a column with element 'Led Zeppelin' appearing twice.

~~`SELECT artist_name FROM Songs GROUP BY artist_name, length_seconds HAVING length_seconds > 300;`~~
This is incorrect. We can use the same counterexample to show that the output of the query could contain duplicate elements.

```
SELECT DISTINCT artist_name FROM Songs WHERE length_seconds > 300;
```

```
SELECT artist_name FROM Songs WHERE length_seconds > 300 GROUP BY
artist_name;
```

Q3: Names of sailors for whom all red boats have been reserved by no other sailor.

A

The clause below the WHERE NOT EXISTS, returns all the reserved pink boat ids that were reserved by at least two sailors (since it does an equality check on S.sid and R.sid and if it finds another R.sid that does not equal the S.sid it's checking, the innermost AND EXISTS clause will return true). In the outermost clause, we're selecting sailor names that make the clause under WHERE NOT EXISTS return true. In order for that to return true, a sailor's reserved pink boat must not have been reserved by anyone else!

Q4: Names of sailors who have reserved as many distinct boats as the number of all red boats.

D

If you take a look at the HAVING clause, you'll notice that a sailor name included in the output will have to have reserved as many distinct boats as the number of all pink boats in the table.

Q5: Names of sailors who have reserved all pink boats.

D

Notice the clause starting with the 4th SELECT from the beginning of the query. We're taking every sid in Reserves and doing a cross join with all the pink boats. This means that every possible combination of sid to pink boat id will be produced. However, from that output, we are excluding every row that is in the Reserves table due to the EXCEPT clause (SELECT sid, bid FROM Reserves just returns all the rows in Reserves). Which R.sids are left? The ones that have reserved either none or some pink boats! We then select the sids from that output in the outer SELECT. Then from that step, we take all the sids in Reserves and exclude the sids of sailors that have reserved either none or some pink boats, leaving us with sids of sailors that have reserved all pink boats!