**COP4710 – Theory and Structure of Databases**

**Summer 2016**

**Homework 3b**

Due Sunday Night, May 29, 2016

***►Eric Adams◄***

Refer to Read-Me 2040-SQLzoo in the *Hands-On* area at the bottom of the page.

We'll use the database servers at ***SQLzoo.net*** to practice with some SQL SELECT statements. There's a Read-Me telling you about SQLzoo – refer to this Read-Me and use SQLzoo.net to work the following problems:

Go to SQLzoo.net. Under the “Tutorial Section” heading click

***1) SELECT Name***

***Some pattern matching queries***

You're taken to a page with query entry boxes and results boxes.

***The table we're working with is not the one described on the page.*** (The entire database is available from every page, not just the table showcased). The table we use is from BBC-compiled statistics of many countries (The BBC Country Profiles database).

The The primary key column is **name**:

**bbc(name, region, area, population, gdp)**

**bbc**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **name** | **region** | **area** | **population** | **gdp** |
| Afghanistan | South Asia | 652225 | 26000000 | null |
| Albania | Europe | 28728 | 3200000 | 6656000000 |
| Algeria | Middle East | 2400000 | 32900000 | 75012000000 |
| Andorra | Europe | 468 | 64000 | null |
| . . . | . . . | . . . | . . . | . . . |

There are many other sets of tables, some are the data for entire applications.

**Homework Problems (Group A):**

Run these queries in the Query Entry Box in SQLzoo.net. See what they do. Paste your *query* (but ***not the results***) below the questions.

1. Write a SQL SELECT statement to list the regions in the bbc table.

SELECT region

FROM bbc

1. Modify the above statement to list the regions with no duplicates. (Use the keyword ***DISTINCT*** after the SELECT keyword. (SELECT DISTINCT regions … )

SELECT DISTINCT region

FROM bbc;

1. Write a SQL statement to display all countries with a population above 200,000,000 (don’t use commas in your numbers in the WHERE clause).

SELECT name

FROM bbc

WHERE population > 200000000;

1. List the countries and their populations with populations between 100,000,000 and 150,000,000 using BETWEEN.

SELECT name

FROM bbc

WHERE population BETWEEN 100000000 AND 150000000;

1. Modify the above query to sort the population results in descending order. (Use ***ORDER BY population DESC*** at the end of the statement. DESC means “in descending order”. ASC is the default).

SELECT name

FROM bbc

WHERE population BETWEEN 100000000 AND 150000000

ORDER BY population DESC;

1. a. Write a query to list all the countries whose names start with **“C”**.

b. Experiment to see if this search is case sensitive. Is it?

c. The default server is MySQL 5. Change the server to PostgreSQL and rerun you query. Is the search case sensitive now? How about with the other servers?

Note: PostgreSQL requires ***single*** quotes around the wildcard expression.

1. SELECT name FROM bbc

WHERE name LIKE 'C%';

1. It is not case sensitive.
2. PostgreSQL, Oracle, and DB2 are case sensitive.
3. Find the countries where the population attribute is NULL (has been left without a value). Use both **WHERE population IS NULL** and **WHERE population = NULL** One is correct, the other is not. Which is correct?

SELECT name

FROM bbc

WHERE population IS NULL;

**============================================================**

**Homework Problems (Group B):**

Now we use two different tables and join them; **ttms** and **country**, which are about the Olympic Games and medal counts:

**ttms(games, color, who, *country*)** <--- country is a foreign key pointing to country.id

**country(id, name)**

**ttms**

|  |  |  |  |
| --- | --- | --- | --- |
| **games** | **color** | **who** | **country** |
| 1992 | gold | Lu Lin | CHN |
| 1992 | silver | Wang Tao | CHN |
| 1996 | gold | Guoliang Liu | CHN |
| **. . .** | **. . .** | **. . .** | **. . .** |

**country**

|  |  |
| --- | --- |
| **id** | **name** |
| ALG | Algeria |
| ARG | Argentina |
| ARU | Aruba |
| **. . .** | **. . .** |

Run these queries in the Query Entry Box in SQLzoo.net. See what they do. Paste your *query* (but ***not the results***) below the questions.

1. Run these two queries and view the results:

SELECT \*

FROM ttms **LEFT** JOIN country

ON ttms.country = country.id;

SELECT \*

FROM ttms **RIGHT** JOIN country

ON ttms.country = country.id;

Explain the difference in the results.

The first displays all the rows of ttms and the corresponding rows in country (NULL if there is no match), and the second displays all the rows of country and the corresponding rows in ttms (NULL if there is no match)

1. Now run these two queries and view the results. (The only difference between these and those above are the addition of the WHERE clause):

SELECT \*

FROM ttms **LEFT** JOIN country

ON ttms.country = country.id

WHERE color = 'gold';

SELECT \*

FROM ttms **RIGHT** JOIN country

ON ttms.country = country.id

WHERE color = 'gold';

Why are the results of the two queries the same this time?

Only countries that have a match in ttms with a color of gold will be displayed in either of them.

1. Write a query to show the athlete’s name and the name of their country but only for names that contain the string 'wang' anywhere in the name.

SELECT who, name

FROM ttms LEFT JOIN country

ON ttms.country = country.id

WHERE who LIKE '%wang%';

1. Show the name of all the medal winners from 'Korea'.

SELECT who

FROM ttms LEFT JOIN country

ON ttms.country = country.id

WHERE name = 'Korea';

**============================================================**

**Homework Problems (Group C):**

One more time, go back to the homepage of SQLzoo.net, and click on

***7) More JOIN operations:***

***In which we join actors to movies in the Movie Database.***

In which we join three tables; movie, actor and casting.

movie(id, title, yr, score, votes, director)

actor(id, name)

casting(movieid, actorid, ord) <--- movieid & actorid are foreign keys to the other tables.

**movie**

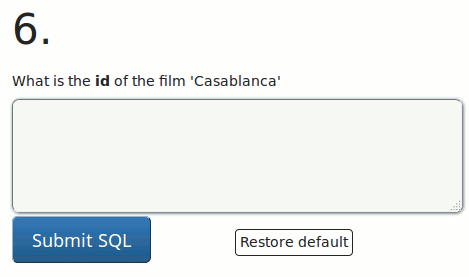
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **title** | **yr** | **director** | **budget** | **gross** |
| 10001 | $ | 1971 | 3 | null | null |
| 10002 | "Crocodile" Dundee | 1986 | 19 | null | 328203506 |
| 10003 | "Crocodile" Dundee II | 1988 | 36 | 15800000 | 239606210 |
| . . . | . . . | . . . | . . . | . . . | . . . |

**actor**

|  |  |
| --- | --- |
| **id** | **name** |
| 30509 | Tori Amos |
| 2 |  |
| 35329 | Barbara Leake |
| . . . | . . . |

**casting**

|  |  |  |
| --- | --- | --- |
| **movieid** | **actorid** | **ord** |
| 10001 | 4 | 1 |
| 10001 | 5 | 2 |
| 10001 | 6 | 3 |
| . . . | . . . | . . . |



Run these queries in the Query Entry Box in SQLzoo.net. See what they do. Paste your *query* (but ***not the results***) below the questions.

1. In this page of SQLzoo, do Problem 6. The site tells you when you get the right answer:

What is the id of the film 'Casablanca'?

SELECT id

FROM movie

WHERE title = 'Casablanca';

1. Now do problem 7. It uses output from 6.

Obtain the cast list for 'Casablanca'. Use the id value that you obtained in the previous question.

SELECT name

FROM actor LEFT JOIN casting

ON actor.id = casting.actorid

WHERE movieid = 11768;

1. Now do problem 8. It is just like 6 & 7 except you can do it all in a single query.

Obtain the cast list for the film 'Alien'.

SELECT name

FROM actor LEFT JOIN casting

ON actor.id = casting.actorid

WHERE movieid = (SELECT id FROM movie WHERE title = 'Alien');