Database Assignment

# Introduction

You have been asked to develop a database for a local leisure centre, as they are in the process of upgrading their existing paper base record keeping system building for their swimming pool. They would like to use the database to help them manage their swimming lessons. Below are the entities and their attributes which were extracted from their paper-based system by a database developer who never got to implement the database for the local leisure centre. Use the **entities** and their **attributes** provided to create the respective tables. In addition, the leisure centre has now decided that they want the database to be implemented using MySQL.

The entities (tables) for this assignment are Course, Members, and Lessons.

* Course (courseID, Level, Sessions, InstructorName, startDate, LessonTime)
  + Members (MemberID, Firstname, Surname, DOB, Address, City)
  + Lessons (LessonID, CourseID, MemberID)

The above highlighted in green are the primary keys.

Course: courseID - int,

level - varchar(30),

sessions - varchar(30),

instructorName - varchar(30),

startDate -date,

lesson time - time

Members: memberID – int,

firstName varchar(30),

surname varchar(30),

DOB date,

address varchar(50),

city varchar(20)

Lessons: lessonID – int,

courseID – int (foreign key – related with course id column in the course table)

memberID – Int (foreign key – related with member id column in the members table)

Once the tables are created, please insert some data [of your choice] into these tables.

EXERCISES:

1. **Use the SQL AND, OR and NOT Operators in your query (The WHERE clause can be combined with AND, OR, and NOT operators)**
2. Where courseID is equals to a number below 5 and the first name of any of the instructors
3. Where courseID is equals to a number above 5 and the lesson time is in the morning or afternoon.
4. **Order by the above results by:**
5. startDate in “*course*” table
6. MemberID in “*members*” table
7. **UPDATE the following:**
8. Members table, change the addresses of any three members.
9. Course table, change the startDate and lesson time for three of the sessions.
10. **Use the SQL MIN () and MAX () functions to return the smallest and largest value**
11. Of the LessonID column in the “*lesson*” table
12. Of the membersID column in the “*members*” table
13. **Use the SQL COUNT (), AVG () and SUM () functions for these:**
14. Count the total number of members in the “*members*” table
15. Count the total number of sessions in the “course” table
16. Find the average session time for all sessions in the “course” table
17. **WILDCARD queries (like operator)**
18. Find all the people from the “*members*” table whose last name starts with A.
19. Find all the people from the “*members*” table whose last name ends with A.
20. Find all the people from the “*members*” table that have "ab" in any position in the last name.
21. Find all the people from the “*members*” table that that have "b" in the second position in their first name.
22. Find all the people from the “*members*” table whose last name starts with "a" and are at least 3 characters in length:
23. Find all the people from the “*members*” table whose last name starts with "a" and ends with "y"
24. Find all the people from the “*members*” table whose last name **does not** starts with "a" and ends with "y"
25. **What do you understand by LEFT and RIGHT join? Explain with an example.**

create database Swimming\_lessons;

use Swimming\_lessons;

show databases;

/\* create tables :

The entities (tables) for this assignment are Course, Members, and Lessons.

Course (courseID, Level, Sessions, InstructorName, startDate, LessonTime)

Members (MemberID, Firstname, Surname, DOB, Address, City)

Lessons (LessonID, CourseID, MemberID)

The above highlighted in green are the primary keys.

Course: courseID - int,

level - varchar(30),

sessions - varchar(30),

instructorName - varchar(30),

startDate -date,

lesson time - time

Members: memberID – int,

firstName varchar(30),

surname varchar(30),

DOB date,

address varchar(50),

city varchar(20)

Lessons: lessonID – int,

courseID – int (foreign key – related with course id column in the course table)

memberID – Int (foreign key – related with member id column in the members table)

\*/

CREATE TABLE Course (

courseID INT PRIMARY KEY,

level VARCHAR(30),

sessions VARCHAR(30),

instructorName VARCHAR(30),

startDate DATE,

lessonTime TIME);

CREATE TABLE Members (

memberID INT PRIMARY KEY,

firstName VARCHAR(30),

surname VARCHAR(30),

DOB DATE,

address VARCHAR(50),

city VARCHAR(20));

CREATE TABLE Lessons (

lessonID INT PRIMARY KEY,

courseID INT,

memberID INT,

FOREIGN KEY (courseID) REFERENCES Course (courseID),

FOREIGN KEY (memberID) REFERENCES Members (memberID));

-- Insert random data into the tables --

INSERT INTO Course (courseID, level, sessions, instructorName, startDate, lessonTime)

VALUES (1, 'Beginner', '4', 'John Smith', '2020-04-01', '10:00:00');

INSERT INTO Course (courseID, level, sessions, instructorName, startDate, lessonTime)

VALUES (2, 'Intermediate', '6', 'Jane Doe', '2020-05-01', '14:00:00');

INSERT INTO Members (memberID, firstName, surname, DOB, address, city)

VALUES (1, 'John', 'Smith', '1990-01-01', '123 Main Street', 'New York');

INSERT INTO Members (memberID, firstName, surname, DOB, address, city)

VALUES (2, 'Jane', 'Doe', '1995-02-02', '456 Maple Street', 'Los Angeles');

INSERT INTO Lessons (lessonID, courseID, memberID)

VALUES (1, 1, 1);

INSERT INTO Lessons (lessonID, courseID, memberID)

VALUES (2, 1, 2), (3, 2, 1), (4, 2, 2);

/\* Use the SQL AND, OR and NOT Operators in your query (The WHERE clause can be combined with AND, OR, and NOT operators)

1.Where courseID is equals to a number below 5 and the first name of any of the instructors

2.Where courseID is equals to a number above 5 and the lesson time is in the morning or afternoon. \*/

SELECT \*

FROM Course

WHERE courseID < 5

AND instructorName = 'firstName';

SELECT \*

FROM Course

WHERE courseID > 5

AND (lessonTime = 'morning' OR lessonTime = 'afternoon');

/\* B.Order by the above results by:

1.startDate in “course” table

2.MemberID in “members” table \*/

SELECT \*

FROM Course

ORDER BY startDate ASC;

SELECT \*

FROM Members

ORDER BY memberID ASC;

/\*C. UPDATE the following:

1.Members table, change the addresses of any three members.

2.Course table, change the startDate and lesson time for three of the sessions. \*/

UPDATE Members

SET address = '111 Oak Avenue'

WHERE memberID = 1;

UPDATE Members

SET address = '123 Elm Street'

WHERE memberID = 2;

UPDATE Members

SET address = '456 Pine Street'

WHERE memberID = 3;

UPDATE Course

SET startDate = '2020-06-01', lessonTime = '12:00:00'

WHERE courseID = 1;

UPDATE Course

SET startDate = '2020-07-01', lessonTime = '16:00:00'

WHERE courseID = 2;

UPDATE Course

SET startDate = '2020-08-01', lessonTime = '18:00:00'

WHERE courseID = 3;

/\* D.Use the SQL MIN () and MAX () functions to return the smallest and largest value

1.Of the LessonID column in the “lesson” table

2.Of the membersID column in the “members” table \*/

SELECT MIN(lessonID) FROM Lessons;

SELECT MAX(memberID) FROM Members;

/\* E.Use the SQL COUNT (), AVG () and SUM () functions for these:

Count the total number of members in the “members” table

Count the total number of sessions in the “course” table

Find the average session time for all sessions in the “course” table \*/

SELECT COUNT(\*) FROM Members;

SELECT COUNT(\*) FROM Course;

SELECT AVG(lessonTime) FROM Course;

/\* WILDCARD queries (like operator)

a) Find all the people from the “members” table whose last name starts with A.

b) Find all the people from the “members” table whose last name ends with A.

c) Find all the people from the “members” table that have "ab" in any position in the last name.

d) Find all the people from the “members” table that that have "b" in the second position in their first name.

e) Find all the people from the “members” table whose last name starts with "a" and are at least 3 characters in length:

f) Find all the people from the “members” table whose last name starts with "a" and ends with "y"

g) Find all the people from the “members” table whose last name does not starts with "a" and ends with "y" \*/

SELECT \* FROM Members WHERE surname LIKE 'A%';

SELECT \* FROM Members WHERE surname LIKE '%A';

SELECT \* FROM Members WHERE surname LIKE '%ab%';

SELECT \* FROM Members WHERE firstName LIKE '\_b%';

SELECT \* FROM Members WHERE surname LIKE 'A\_%\_%';

SELECT \* FROM Members WHERE surname LIKE 'A%y';

SELECT \* FROM Members WHERE surname NOT LIKE 'A%y';

-- G.What do you understand by LEFT and RIGHT join? Explain with an example. --

/\* LEFT JOIN and RIGHT JOIN are types of outer join operations in SQL.

LEFT and RIGHT joins are types of join operations used in SQL to combine records from two or more tables. A LEFT join returns all of the records from the left table, even if there are no matches in the right table.

A RIGHT join returns all of the records from the right table, even if there are no matches in the left table.

An example of a LEFT JOIN would be: \*/

-- LEFT JOIN --

SELECT m.firstName, m.surname, c.level, c.instructorName

FROM Members m

LEFT JOIN Lessons l

ON m.memberID = l.memberID

INNER JOIN Course c

ON l.courseID = c.courseID;

-- RIGHT JOIN --

SELECT m.firstName, m.surname, c.level, c.instructorName

FROM Lessons l

RIGHT JOIN Members m

ON m.memberID = l.memberID

INNER JOIN Course c

ON l.courseID = c.courseID;

-- The end --