**SQL**

You have been asked to help a school get all their data on their computers, and to help draw some reports. A previous developer has already started with the database, and added a class’s marks. You have been tasked with completing the project.

Open the database SchoolApp. It contains 3 tables already. Students, subjects and subjectMarks.

1. A) Write a new stored procedure that will bring back all the students, the subjects they are taking, and the mark achieved in the last exam. The principal wants to see only the child’s name, surname, subject, and mark.
2. If part of a child’s name and/or surname is passed into the stored procedure, only the relevant results should come back. Tip: There should only be 1 input.
3. Write another stored procedure that will bring back the students name, surname, the amount of subjects taken, and the average mark to 2 decimals. Only show results for the students who are taking 5 or more subjects.

The school also wants to see who is doing sports, and what sports they are doing. The school currently does the following sports:

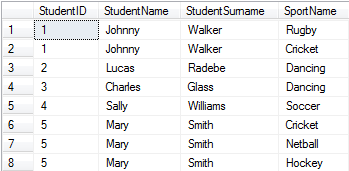
Rugby;Soccer;Cricket;Netball;Hockey

1. Create the necessary tables to the database to meet the requirements.
2. After you complete those changes, the principal wants you to capture the following details into the database:

Johnny Walker plays Rugby and Cricket  
Lucas Radebe plays Soccer  
Mary Smith plays Cricket, Netball and Hockey  
Daniel Steele plays Rugby, Soccer and Cricket.

1. Create a stored procedure that will display all the students. We want to see their name, surname, and age. If they play a sport, show the sport name. If they play more than one sport, list the other sports below. If they play no sports, the principal wants them to participate in the school’s big annual dance. Thus he wants the word ‘Dancing’ to display under SportName if they don’t do any sports.

It should look more or less like this, although in our example we want the age, not the StudentID:



**Coding C# test**

Part 1:

1. Create a new console application which can read input values, and then run the relevant command. If one command is finished, it can be executed again, or other commands can be executed. Only close the command line on ‘Exit’ prompt.
2. Parameters “ChildMarkReport” and “Charles Glass” should call the stored procedure that brings back a specific child’s name, surname, subjects taken and the marks. For example “ChildMarkReport, Charles Glass”.
3. Parameter “ClassAveragesReport” should call the stored procedure that brings back all the children in the class, how many subjects taken and their average marks.
4. Parameter “SportsTaken” should call the stored procedure that brings back all the children in the class and the sports they play.
5. All results should be saved as separate text files to a newly created path on your computer “C:\Test\CodingTest\C#\Applicant\SchoolReports\”. It should be readable.

Part 2:

1. Use the below rules to validate an account number being used as a reference. This routine will be used to validate an account number. Google ‘What is a cdv check’ for reference.

Write a console app that accepts a reference, validates the reference, and then returns a message stating either “success”, or “Failure”.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Position | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|  | C | F | 0 | 0 | 0 | 0 | 0 | 2 | 7 | 9 |
| Weighting |  |  | 1 | 3 | 7 | 1 | 3 | 7 | 1 |  |
| Product |  |  | 0 | 0 | 0 | 0 | 0 | 14 | 7 |  |
| Sum of Product | 21 |  |  |  |  |  |  |  |  |  |
| Mod10 | 9 |  |  |  |  |  |  |  |  |  |

1. The reference should be 10 Alphanumeric characters
2. The first 2 characters should always be “CF”.
3. Characters 3 to 10 will be Numeric
4. Characters 3 – 9 carries a weighting to calculate the 10th Character.

The following logic is applied:

For each number in the reference

{

checkTotal = checkTotal + (number \* numberWeight)

}

Then divide the final checkTotal by 10. The 10th character will be 10 minus the remainder. Check example above.

Input = CF00000279

Number = 0000027

Weight = 1371371

2 \* 7 (weight) = 14

7 \* 1 (weight) = 7

Added together = 21

Divided by 10 the remainder is 1

10 minus 1 (remainder) = 9 (10th character)

Valid reference = CF00000279

**Test Data:**Valid  
CF00000019         
CF00000176         
CF00000242         
CF00000279       
CF00000345           
CF00000381         
CF00000439         
CF00000532         
CF00000550         
CF00000587

Invalid  
CF00000017        
CF00000178         
CF00000240         
CF00000277       
CF00000343           
CF00000383         
CF00000437  
CF00000538