CMPT 354
Database Systems

Simon Fraser University
Spring 2017

Instructor: Oliver Schulte

Assignment 3a: Application Development, Chapters 6 and 7.

Instructions: Check the instructions in the syllabus. The university policy on academic dishonesty and plagiarism (cheating) will be taken very seriously in this course. *Everything submitted should be your own writing or coding.* You must not let other students copy your work. On your assignment, put down your **name**, the number of the assignment and the number of the course. Spelling and grammar count.

Group Work: Discussions of the assignment is okay, for example to understand the concepts involved. If you work in a group, put down the name of all members of your group. There should be no group submissions. Each group member should write up their own solution to show their own understanding.

For the due date please see our course management server https://courses.cs.sfu.ca . Part of the assignment task is figuring out how to use the CSIL system. If your assignment is late because you did not figure this out soon enough, you will lose marks according to the syllabus policy.

Additional instructions for what to submit appear in a separate file on the course website.

There will be no email support for this assignment. Start early and ask your questions in class, office hours, or team up with classmates. You can use the discussion forum on courses.cs.sfu.ca .

Systems Issues

Systems Support. The purpose of this assignment is to give you experience with writing programs that interact with a database management system. You will learn hardly anything from typing in someone else's instructions; you will learn a lot from getting the system to work on your own. Therefore we provide minimal support for getting things to work on your own system. I suggest you get started on this assignment early to check your basic system setup.

1. *System Requirements*. Our basic requirement is that **we should be able to run your solution on the CSIL server.** This means that you should use one of the development setups available in CSIL. That's the only constraint. For a list of what's available in CSIL, see http://www.cs.sfu.ca/cc/CSILPC/software.html . General info about CSIL is posted at http://www.cs.sfu.ca/cc/Labs/. You can of course develop your solution on your home system and then test it on the CSIL set-up. Tip: Ask a friend to run your code on CSIL so you know someone else can run it with his or her settings. A few suggestions for planning your work.

2. The basic architecture required for the assignment is a client-server architecture where the client functions provide an interface for accepting user input, and the server is the SQL server. The program structures required are simple, so the main new challenges for you are a) the presentation layer and b) the interaction with the server. You may want to choose a development set-up that makes this as easy as possible. I list some of the common environments used by students, starting with the easiest first.
    a. *Python.* The CSIL Python installation comes with the database package you need (see slides). You just need to import it. This is by far the easiest for students. One student wrote that: "Time to do this assignment in Java – infinite. Time to do it in Python – 15 min".
    b. *Visual Studio* provides a graphical interface for creating forms, buttons etc. We also provide some support by way of example code and a flash video for it. If you are used to Java programming, but not to creating forms for user input or connecting to the database server, your best bet is probably to use Visual C#: writing code is easy to learn and Visual Studio simplifies the client-server tasks.
    c. Java.

3. In addition to examples and general principles covered in the class and the text, there is much documentation available with the common development tools, such as Visual Studio and in on-line discussion. You should not look for the specific solution, but feel free to look for general information (e.g., "what does this menu do?" or go through the Beginner's Development Tutorial in Visual Studio).

4. One of the most finicky and system-dependent parts is *establishing a connection* between your application and your SQL server. We will provide sample instructions for one system (full Visual Basic), but it's up to you to find out how

to establish a connection with your system (e.g. VB Express, JDBC.) Establishing a connection may be the part that takes the most time.

Part I: Database Connection

Create a new application. When the application is run, the following should happen.

1. Calculate how many customers there are in the **AdventureWorksLT** database via a SQL query to the database.
2. Write to the screen how many customers there are.

Grading Criteria:

Total Marks: 100 Marks
1. Code + Connection (40 Marks)
2. Query (30 Marks)
3. Output (30 Marks)

Requirements:

- Machine Environment: Your program should run on leto.csil.sfu.ca. Note: If you developed on your own system or via remote access, you may have to recompile your files on leto.
- Please include a **readme** file with instructions for running your code (e.g., open Visual Studio, open the project file "myproject", run it using Visual Studio…).
- Your application should connect to your own account on cypress.
- Your username and password (for the CSIL SQL server) should be in your code, so that the user doesn't have to enter them. To connect to the CSIL SQL server, currently Cypress, you will need to find out the exact hostname, your username on Cypress, and your password on Cypress. You can find this out using the CSIL instructions emailed earlier. One useful webpage is http://www.cs.sfu.ca/about/school-facilities/csil/windows/how-to-use-sql.html.
- Include supporting files if needed. (like any dll, jar or class files).

Part II: Stored Procedures/Functions

1. Write SQL code for a stored procedure (function) *AverageCost* that takes as input parameter a color and returns the average StandardCost of the products in the Product table that have that color (use the **AdventureWorksLT** database). Execute the SQL code to create the stored procedure.
2. Create a new application. When the application is run, the application should write to the screen what the average StandardCost of *red* products in the Product table is. The answer should be computed by calling the stored procedure that you wrote for part II.1 (Question 1 in Part II).

Grading Criteria:

Total Marks: 100 Marks
1. SQL code + creating stored procedure (50 Marks)
2. Application + output (50 Marks)

Requirements:

- Code design and documentation are part of the criteria. Remember that your TA may not be an expert in the development system you are using. The code required is so short that having an explanatory comment for each line is not overdoing it. In fact, it's a good habit to acquire.

- Your code should run in the CSIL environment so we can run it if necessary. However, if we have to check it out by running it, your documentation is probably insufficient.

What to Submit

Part I: Database Connection

Submit the following files:

1. Your source code sourceI.*, where * is the file extension required for your development setup. Please include supporting files if needed (like any dll, jar or class files). Your code should be self-contained. If your code comprises several files, please combine them into a single file sourceI.zip.
2. A readme file with instructions for how to run your code on leto.csil.sfu.ca . Call this file readmeI.rtf.
3. A screenshot of the output from running your application. Call this file outputI.pdf.

Put all these files together into a single archive called **solutionI.zip**.

Part II: Stored Procedures/Functions

Submit the following files:

1. For the SQL code a single sql script. This should execute without error on SQL Server. Call this file sqlII.sql .
2. Your source code sourceII.*, where * is the file extension required for your development setup. If your code comprises several files, please combine them into a single file sourceII.zip.
3. A readme file with instructions for how to run your code on leto.csil.sfu.ca . Call this file readmeII.rtf.
4. A screenshot of the output from running your application. Call this file outputII.pdf.

Put all these files together into a single archive called **solutionII.zip**.