

## **Software Engineering + Data Wrangling with SQL**

*Autumn 2019 - Combined Assessment*

**Objective:** This assessment verifies your abilities to combine clean software engineering using Python as for implementation and Data Wrangling with SQL.

### **Grading:**

1. The assignment itself is graded out of 100%, pass grade at 50%.
2. 80% of your grade will be allocated to the Software Engineering module
3. 20% to the Data Wrangling with SQL module.
4. Half of these points of percentages go as bonus points to their respective modules.

### **How does it work?**

Let's say you get 80% for this assignment. It follows:

1.  $80 \times 0.8 = 64$
2.  $80 \times 0.2 = 16$

**Software Engineering** has two former assignments (*Classical Programming, Object-Oriented Concepts*).

Let say you got the following grades:

1. Classical Programming: 60%
2. Object-Oriented Concepts: 70%
3. Final result:  $[ 60 + 70 + (64/2) ] / 2 = 81 \rightarrow 85\%$  for the course

**Data Wrangling with SQL** has one evaluation.

Let say you got the following grade:

1. Mini-project: 60%
2. Final result:  $[ 60 + (16/2) ] = 68 \rightarrow 70\%$  for the course

### **How do I get graded for this assignment?**

1. 70 points are reserved for the goal: your code works (*it produces the expected output*) or it doesn't. This part is "honesty based". You will declare on Moodle whether you succeeded. *Don't lie, I will be looking at you code* 🤖
2. 15 points are also easy to achieve: cleanliness, structure and commenting of your code (*including expected input in you functions*) and results.
3. The final 15 points are reserved for engineering quality: are potential exceptions well-managed, is your code I/O-safe? Is your code properly structured in functions / classes (*if you choose to make it object-oriented*)? Are you isolating reusable code whenever applicable?

**Scope statement:**

In the Data Wrangling with SQL course, we have seen how we could write stored procedure/functions to build dynamic SQL pivot survey answers data in usable format for analysis in the toy database "SurveySample\_A19".

After a few iterations, we ended up with the following design:

1. A stored function `dbo.fn_GetAllSurveyDataSQL()` which generates and returns a dynamic SQL query string for extracting the pivoted survey answer data.
2. A trigger `dbo.trg_refreshSurveyView`
  - a. firing on INSERT, DELETE and UPDATE upon the table `dbo.SurveyStructure`
  - b. executing a CREATE OR ALTER VIEW `vw_AllSurveyData` AS + the string returned by `dbo.fn_GetAllSurveyDataSQL`

With this design, we have enforced an *"always fresh"* data policy in the view `vw_AllSurveyData`.

As discussed, this solution is "ideal" as it respects the principle of data locality. But it requires to have privileges for creating stored procedures/functions and triggers. If the former may be rare, the latter is often heavily restricted.

You are now in a scenario where the only databases operations allowed are:

1. to select data from tables.
2. to create/alter views.

You can use programmatic access to the database server via an ODBC library and you have to develop in Python 3.

**Your Python 3 application must accommodate the following requirements:**

1. Gracefully handle the connection to the database server.
2. Replicate the algorithm of the `dbo.fn_GetAllSurveyDataSQL` stored function.
3. Replicate the algorithm of the trigger `dbo.trg_refreshSurveyView` for creating/altering the view `vw_AllSurveyData` whenever applicable.
4. For achieving (3) above, a persistence component (*in any format you like: CSV, XML, JSON, etc.*), storing the last known surveys' structures should be in place. It is not acceptable to just recreate the view every time: the trigger behaviour must be replicated.
5. Of course, extract the *"always-fresh"* pivoted survey data, in a CSV file, adequately named.



In terms of allowed libraries and beyond the recommended `pyodbc` & `pandas`, you are free to use anything you like, **but** with this **mandatory requirement**: your Python application should not require the user to install packages before the run.

In order to do so, have a look here:  
<https://stackoverflow.com/questions/12332975/installing-python-module-within-code>

**Deadline: Sunday, 19<sup>th</sup> April 2020 at midnight.**

*(understanding the datetime representation of midnight with a computer is part of the assessment. No, this is not a joke.)*

**Moodle URL:** <https://a19.moodle.dsti.institute/mod/quiz/view.php?id=1319>

*A sample correction will appear as a feedback of your submission on Moodle once the deadline is passed.*

*Happy coding!  
Sébastien*

**Given on: 2<sup>nd</sup> March 2020**