**A Task to implement clean up, database update, validations, and modularize script, and documentation**

Adel Abdallah, Feb 7, 2017

1. **Quick cleanup and updates to the previous package:** [this one is from the previous task called: “Description of the programming logic to import Data Values from the spreadsheet to the WaMDaM database, Adel Abdallah, Jan 18, 2017”]
   1. Add new empty folder called: “Queries”. Add an empty ReadME text file inside it.

I'd like later to put text files of generic queries there

* 1. Change the reading row number from 3 to 1 in the class “CrossTabulatedTask1():” See here:

def get\_sheet\_rows(self):

return [row for row in self.sheet.get\_rows()][3:]

* 1. Separate the ExcelDataShaper.py into two files, one for each task. We will add more classes or tasks later so it’s a good practice to keep them separate from each other for future updates or changes.
  2. Update, adapt, and then move the Errores.py function into the utils folder. I’d like to export a log of the error messages into a text file called “logging”. The text file location should be in the main directory next to wamdam.py. Delete the author info in this file and update the variables and comments to English. Delete the Tools folder.
  3. Delete the velantur text file. Delete the test.py (if you don’t use it). Rename the whole package with a data stamp that reflects the day you will finish the package. Delete the date stamp \_Jan5 from the .py files in the utils folder. Is there any use to the “load\_step\_5.py”, if not, delete it. Also, delete the old spreadsheet called WaMDaM\_InputData\_Jan4.xlsx
  4. Change the name of the python file “setup.py” to ReadSpreadsheets.py and change the “step\_variables.py” to ReadSpreadsheets\_variables.py

1. **Database update**

The updated ER diagram is here

http://amabdallah.github.io/test2/diagrams/03\_WaMDaM\_wCVs.html

* 1. I deleted the ObjectCategoryCV and the AttributeCategoryCV tables. Instead of their foreign keys in the ObjectTypeCV and AttributesCV tables, now they're fields with text values.
  2. All the CV table names are preceded by CV\_
  3. The second field in each "CV\_" table is identical to the first one but it has the addition "Term". This second field will contain values with no spaces in them, it has a special use later.
  4. Added a new table called InstanceCategory (I need to add a field called InstanceCategoryName)
  5. All the CV tables are connected/related directly through the "Name" column. So users dont need to join over the CV table to get the Name.
  6. There are a few minor changes in the names of some tables or fields in the database. I forgot which one they are for now.
  7. The workbook template needs to be updated

Changes to reflect to the script:

* The binary tables are now called: BooleanValues and CV\_BooleanValueMeaning. See the changes in the columns
* The File-based table is now called: Files. Delete the column called: File Location on Desk. This column is only provided in the spreadsheet to tell the script where to look for the folder. In the database, store the file in the column called: File which has blob data type
* The Cv\_seasonName has no Start and End dates. delete both of them from the script. I added a column called SeasonName, its different from the existing SeasonNameCV
* In the Time SeriesValues, change the column called "CoresspondingValue" to just "Value"
* Spatial Reference Table: I delete the attribute called SRCode
* I added a table in the spreadsheet and a CV table called CV\_ObjectTopology
* One significant change in reading and loading all the DataValues tables: I added the ObjectType column in the spreadsheet. You will need to use both the provided attribute and ObjectType as a combination to look up the primary key of the right attribute. Before the AttributeCode was unique but now, the attribute is not unique so you need to use the ObjectType with it
* The table called AttributeTypeCV is now called CV\_AttributeDataType and so its columns
* The AttributeTypeCV Table in the database seems to read the row that has "optional" and "required" info. Make sure to exclude this line from being read into the database

1. **Implement basic validation checks for the Networks and DataValues** [From previous tasks]
   1. Check for correct foreign keys in all the tables similar to the validations made in the previous tasks.
   2. Validations for the links. Each “link instance” must have different a start and end “node instances” so a link must not reference other link instances in the “Connection” table.
   3. Validations for importing the correct “data type” as selected in the Attributes Table (AttributeDataType). So if the user selects a “Parameter” AttributeDataType, then only the Parameters table can load data to this attribute for all the node or link instances. If the user tries to load Time Series to this attribute, an error message will them that there are trying to load data values from a type that is different than the selected one as defined in the attributes table.
2. **Modularize the script around smaller data loading steps as follows**

The current script calls multiple data loading steps together. I guess each sub step should have a separate function or class name to be called by-itself. Then the major step calls all the sub-steps together in the end. Users in the future should be able to call one sub-step by itself. The code should be organized into modules as listed below for each step. Let's leave the controlled vocabulary (CVs) tables a side for now. But each table should be called on its own. There is a chance to update the whole script to load all the CVs tables together at once at the beginning. Will come back to them later.

**Under Step 1:**

Organizations

People

Sources

Methods

**Under Step 2:**

Dataset

ObjectTypes

Network Attributes

Attributes

**Under Step 3:**

MasterNetworks

Scenarios

Nodes

Links

**Under Step 4:**

Binarys

Parameters

SeasonalParameters

TextControlled

TextFree

FileBased

TimeSeriesSites

MultiColumnArrays

1. **documentations for files, functions, and classes**

I would like to follow a very consistent and extensive documentation throughout the Wizard scripts. The documentation's target is for someone who is not a computer scientist and they want to get an inside high level understanding of how the Wizard script works.

Below I provide a template for how each python file should include. This template and descriptions could be updated or adjusted for a more professional looking style. It should include 1. File General Info, 2. Imported Python Libraries used in this file, 3., Imported Python modules from within the Wizard used in this file, 4., Script begins here (the body of the script).

The last section below "Script begins here" follows similar documentation you used here for all the classes and functions in the wizard. Please make sure to first describe the logic of the class or function in high level plain English and then specify how the logic work by referring the variables used in it.

Throughout the documentation, delete any unused imported libraries or script pieces. For example, the "Load\_step.py" files still have the name of the spreadsheet input file hard coded like this below. So delete them if not used anymore.

**if**\_\_name\_\_ == '\_\_main\_\_':  
    instance = Load\_Networks\_Data('./WaMDaM\_InputData.xlsx')  
    instance.load\_data()

**# File General Info**

#e.g., this file is the main WaMDaM Wizard launcher and it calls the main GUI form called "frm\_Home".

or something more detailed like this:

'''  
    The load\_step\_4.py is a python script that is used

    to load some CV's of Data Values to the sqlite  
    database.The file has a class called "Load\_Data\_Values()" that   
     uses xlrd library to handle the xlsx files, load, read  
    data and workbook sheets from the file.  
'''

#-------------------------------------------------------------------------------------------------------------------

**# Imported Python Libraries used in this file (examples below)**

import xlrd

#This xlrd package is for reading data and formatting information from older Excel files (ie: .xls)

from openpyxl import load\_workbook

# this package does xyz

Import  datetime

#The datetime library supplies classes for manipulating dates and times in both simple and complex ways

#-------------------------------------------------------------------------------------------------------------------

**# Imported Python modules within the Wizard used in this file** **(examples below)**

from .step\_variables import Network\_sheets\_ordered, row\_start\_network

# this module is imported to do xyz in this place

#-------------------------------------------------------------------------------------------------------------------

**# Script begins here**

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*#  
#                                                                                  #  
#                      This is the class to load CV data                           #  
#                                                                                  #  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* #

for simpler classes, just add one comment line on top of the class

**class Load\_CV\_To\_DB**(Parse\_Excel\_File, DB\_Setup):

*"""  
        This class "Load\_CV\_To\_DB" inherits methods from Parse\_Excel\_Files  
        and DB\_Setup classes so as to parse excel file and load  
        subsequent data received from parser  
    """*

for simpler functions, just add one comment line on top of the function

**def load\_data**(self, sheet\_names):  
    *"""  
        This function "load\_data()" takes a list of sheet names and using  
        the dictionary of rows return from the excel parser  
        it load the data in the appropriate table* ***:param****sheet\_names:* ***:return****: None  
    """*

**Add a comment to explain the purpose of key variables like this one in the Load\_steps\_2.py file**

#this foreing\_key variable does x,y and z

foreing\_key = -1