

Student : Agiopoulos S. Kapsikas (f3352208, akapsikas@aueb.gr)
Course : Large Scale Data Management
Description : Second Assignment, Report

M.Sc. in Data Science
School of Information Sciences and Technology
Athens University of Economics and Business
Full Time, Winter Trimester, 2023

This document serves as the report to be delivered along with the script files (SQL and Python files), which pertain to the implementation parts of the assignment. The purpose of this report, is to describe the adopted schema that has been used to load the data for the assignment. Its structure consists of two parts. In the first one, we describe the procedure followed for the implementation that utilized MonetDB. In the second one, we describe the same procedure, but for the case where PySpark is used instead.

I. Implementation using MonetDB (optionally, see APPENDIX A)

First, the already created by MonetDB database, called “demo,” has been used for the purposes of this assignment. In order to load the data into MonetDB, the loader functions that are provided were used. Such functions insert data into the database, using dictionaries, whose keys are the columns’ names and values are the columns’ values. In light of this particularity, we first had to construct a dictionary of such structure; we did that with the aid of the Python’s package, called Pandas. Specifically, we loaded all the available data into a data frame, and then column-by-column, extracted the columns’ names, stored as strings, and columns’ values, stored as lists. Then, while having these two pieces of information for each of the data frame’s columns, we constructed the dictionary of interest. Lastly, we created the loader function, using Python, and passed into it the dictionary. The proposed process handled the loading of data into the database flawlessly.

II. Implementation using PySpark (optionally, see APPENDIX B)

Firstly, Jupyter Notebook has been chosen for developing, using PySpark. After installing and importing the required Python packages, and initializing an Apache Spark session, we loaded the data. We did that, using the [DataFrameReader](#) class in PySpark, and particularly, one of its methods, called [csv](#), into which we then passed the CSV file’s destination path. The process executed without any problems.

APPENDIX A – MonetDB Installation Process & Embedding Python (Windows 10)

The first thing to do, was to download the latest release of MonetDB for the 64-bit (x86_64) version of Windows 10, which, at the time this report was being composed, was the “Sep2022-SP2.” In addition, both 32-bit and 64-bit ODBC drivers were also installed. While installing it, via the “Custom Setup,” the files required for embedding Python into MonetDB had also been included. This step is crucial and required, since it allows for constructing user defined functions (UDF), written in Python, which are required in order to perform analytical tasks on the data loaded and stored in MonetDB. Below, the configuration of the installation process is being shown:

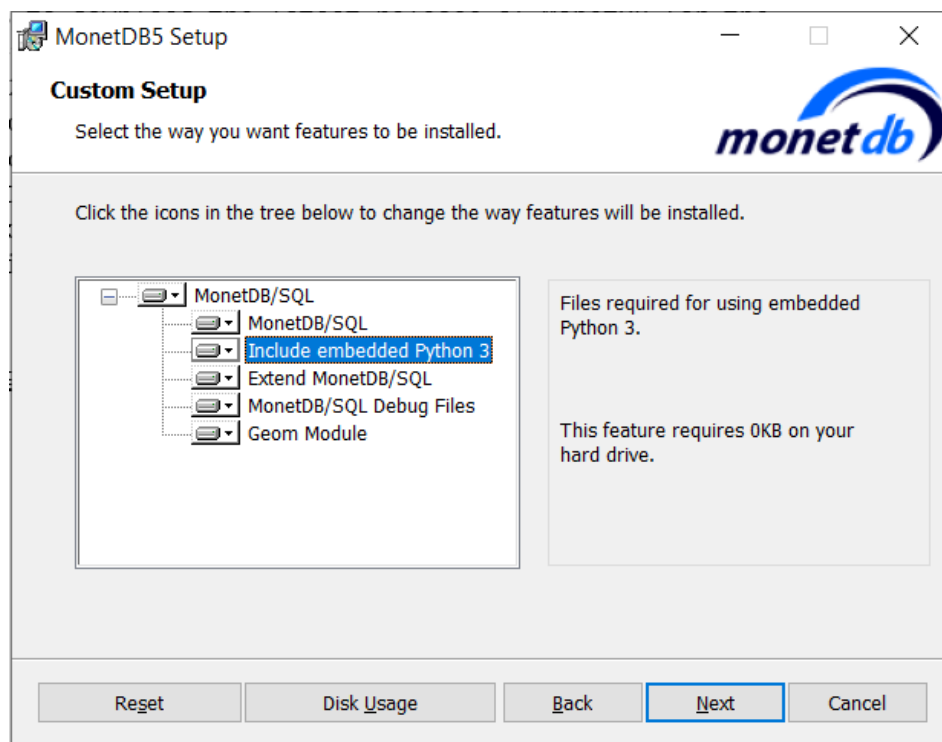


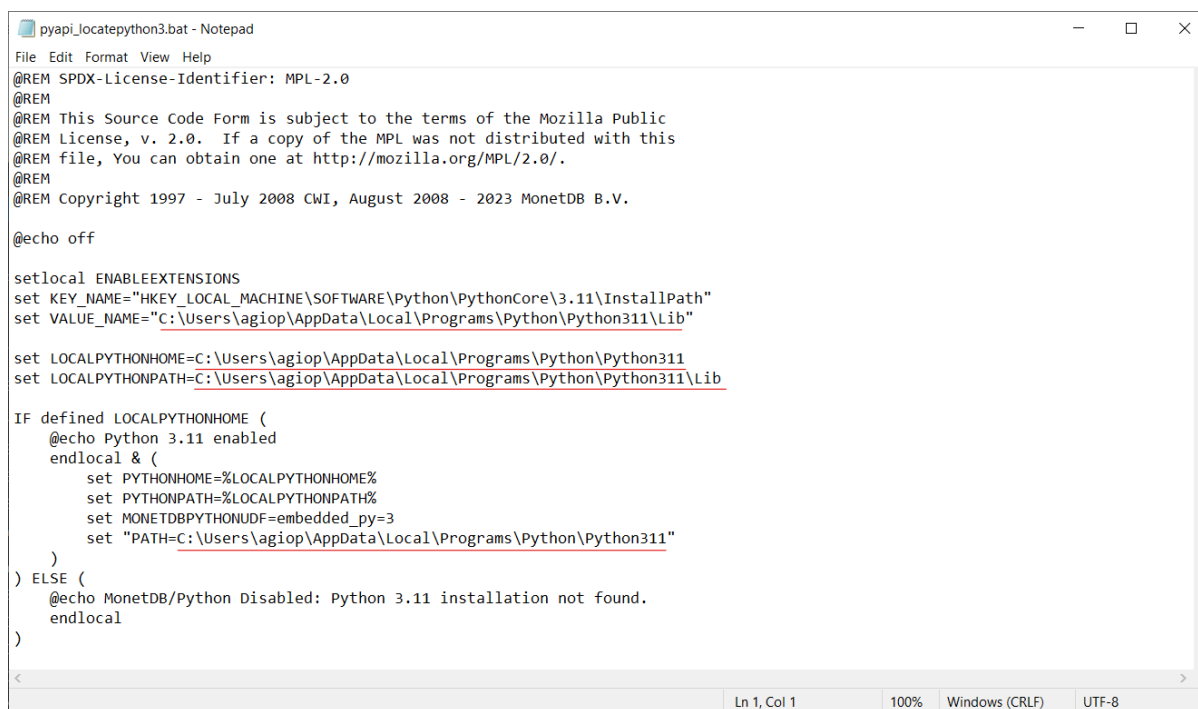
Figure 1 | MonetDB – Configuration of Installation Setup

After successfully downloading and installing MonetDB, with support for embedding Python, we need to connect Python with MonetDB, before we are able to successfully execute any of the required tasks. To that end, as we are also prompted by the assignment’s description, we first need to install the Python’s package that renders the use of Python through MonetDB possible. We can do that by simply typing into a command prompt, the line:

```
python -m pip install pymonetdb
```

One may find executing the above command more useful, while first having created a virtual environment. This will make dependency management easier.

Lastly, what we need to do, is to tell MonetDB, which Python Interpreter to use. We do that by editing the batch file “pyapi_locatepython3.bat” that is located into the folder: “C:\Program Files\MonetDB\MonetDB5”. The file should look something like the one shown below:



```

pyapi_locatepython3.bat - Notepad
File Edit Format View Help
@REM SPDX-License-Identifier: MPL-2.0
@REM
@REM This Source Code Form is subject to the terms of the Mozilla Public
@REM License, v. 2.0. If a copy of the MPL was not distributed with this
@REM file, You can obtain one at http://mozilla.org/MPL/2.0/.
@REM
@REM Copyright 1997 - July 2008 CWI, August 2008 - 2023 MonetDB B.V.

@echo off

setlocal ENABLEEXTENSIONS
set KEY_NAME="HKEY_LOCAL_MACHINE\SOFTWARE\Python\PythonCore\3.11\InstallPath"
set VALUE_NAME="C:\Users\agiop\AppData\Local\Programs\Python\Python311\Lib"

set LOCALPYTHONHOME=C:\Users\agiop\AppData\Local\Programs\Python\Python311
set LOCALPYTHONPATH=C:\Users\agiop\AppData\Local\Programs\Python\Python311\Lib

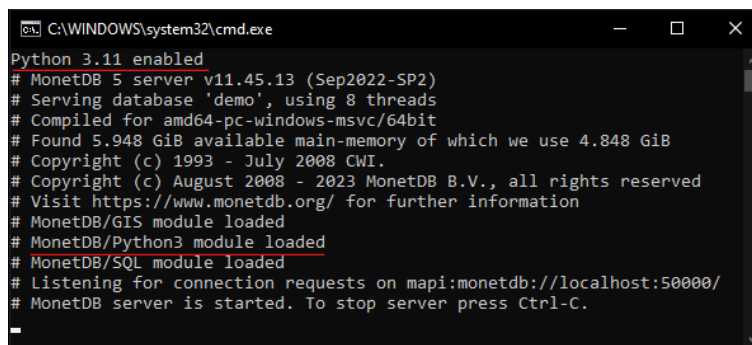
IF defined LOCALPYTHONHOME (
    @echo Python 3.11 enabled
    endlocal & (
        set PYTHONHOME=%LOCALPYTHONHOME%
        set PYTHONPATH=%LOCALPYTHONPATH%
        set MONETDBPYTHONUDF=embedded_py=3
        set "PATH=C:\Users\agiop\AppData\Local\Programs\Python\Python311"
    )
) ELSE (
    @echo MonetDB/Python Disabled: Python 3.11 installation not found.
    endlocal
)

```

Credits to Christos Stavropoulos, who figured out the way to embed Python into MonetDB.

Note that one simply needs to replace the underscored lines with the destination path, where Python's Interpreter is stored in the file system.

Once all of the above steps have been completed, one may start using MonetDB along with the Python programming language. In particular, upon initialization of MonetDB's server (M5server.bat), the below command line prompt should appear onto the screen:



```

C:\WINDOWS\system32\cmd.exe
Python 3.11 enabled
# MonetDB 5 server v11.45.13 (Sep2022-SP2)
# Serving database 'demo', using 8 threads
# Compiled for amd64-pc-windows-msvc/64bit
# Found 5.948 GiB available main-memory of which we use 4.848 GiB
# Copyright (c) 1993 - July 2008 CWI.
# Copyright (c) August 2008 - 2023 MonetDB B.V., all rights reserved
# Visit https://www.monetdb.org/ for further information
# MonetDB/GIS module loaded
# MonetDB/Python3 module loaded
# MonetDB/SQL module loaded
# Listening for connection requests on mapi:monetdb://localhost:50000/
# MonetDB server is started. To stop server press Ctrl-C.

```

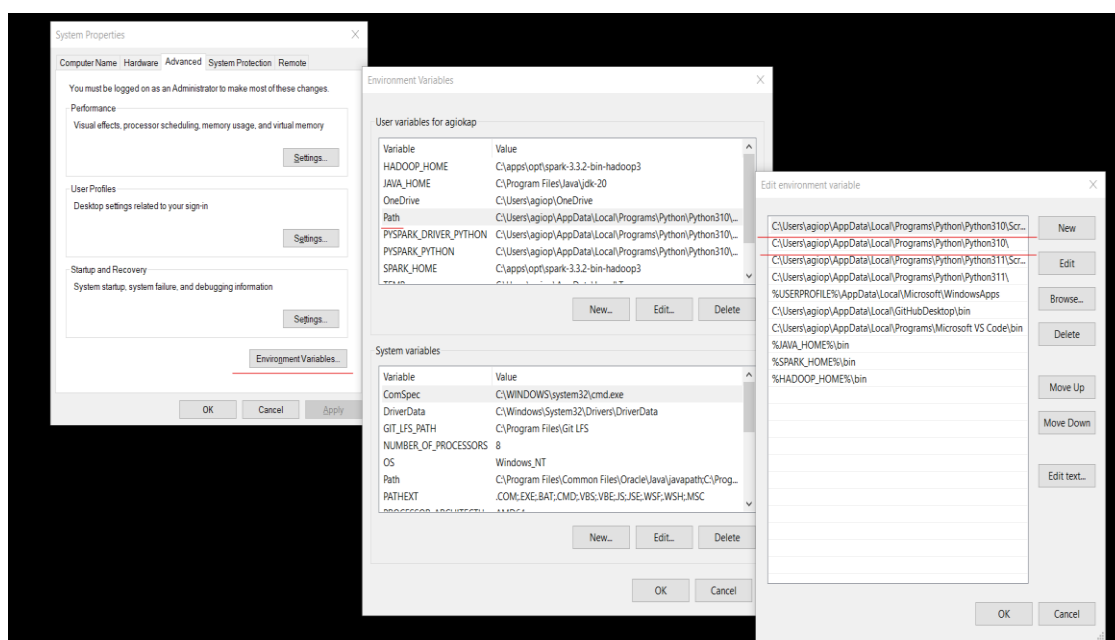
Figure 2 | MonetDB – Server when Python has been embedded

APPENDIX B – PySpark Setup Process

To run PySpark on a Windows, one needs to do the following steps:

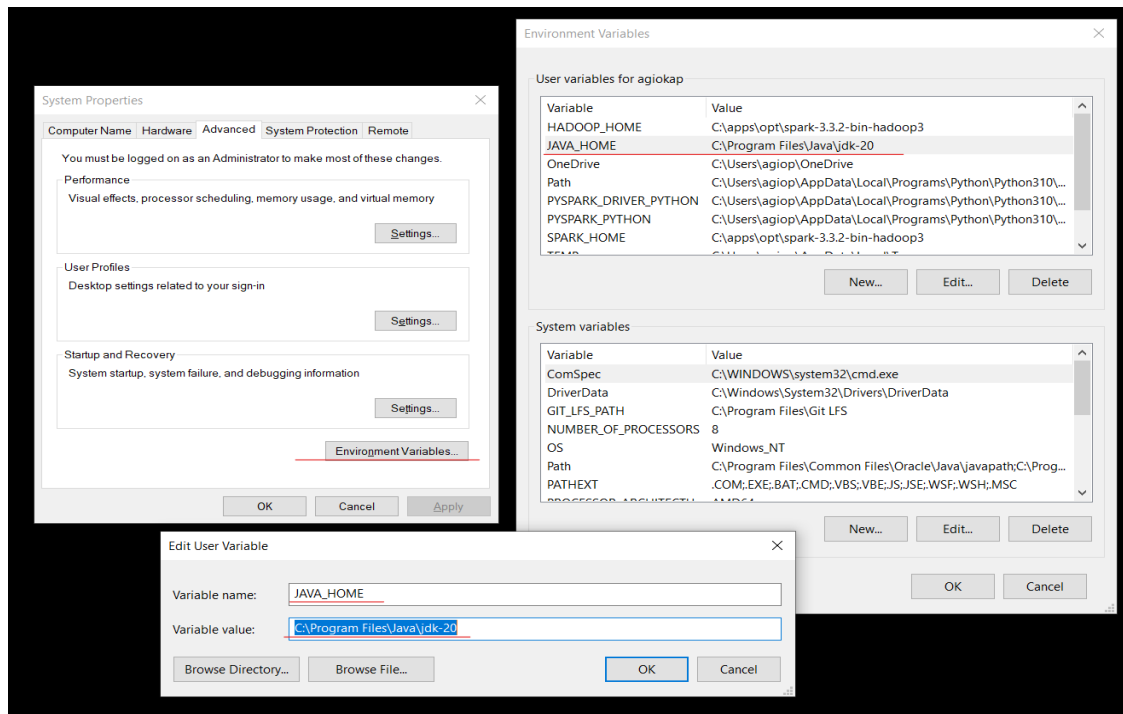
1. Python ([Download](#))

First, Python 3.10 (any), or earlier, should have been installed. That is because, currently, PySpark is able to properly function using only these version(s) of the Python programming language. It's important to note that the destination path of Python's interpreter must be included into "PATH" environment variable, as is shown in, below. To access these settings, search the start menu for "Edit the system environment variables." There, also, add two more user variables, called "PYSPARK_PYTHON" and "PYSPARK_DRIVER_PYTHON", which will both have the same interpreter's path, as before. These settings will let PySpark, which Python version to use. Should you need to run PySpark with another Python version, tweak the last two variables accordingly.



2. Java ([Download](#))

Second, on top of the Python programming language, you should have Java 8 (or newer) installed in your system. Once you have installed it, then follow its destination path, where it's stored in the file system. Locate destination path of Java of your choice. For example, if you had chosen to install Java 20, then (by default) the installation folder would be "C:\Program Files\Java\jdk-20". You should be able to find other versions' paths in similar fashion. Now, again, add another user variable, called "JAVA_HOME", whose value is that path. At the same time, also edit the "PATH" variable, by adding the new entry: "%JAVA_HOME\bin%". Upon their conclusion, these steps should have successfully configured Java for PySpark.



3. Apache Spark ([Download](#))

Third, Apache Spark should have also been installed, before being able to run PySpark on Windows. After downloading, and extracting it, you should have it moved into the destination path: “C:\apps\opt”. Then, you need to retrieve that destination path, where Apache Spark have been placed. For example, it would look something like this: “C:\apps\opt\spark-3.3.2-bin-hadoop3”, depending on which version you had previously chosen to download. Same as before, you need to include two other environment variables, called “SPARK_HOME” and “HADOOP_HOME”, the values of whose will both be the retrieved destination path of Apache Spark. Similarly, after doing that, you also need to edit the “PATH” variable by adding two more values: “%SPARK_HOME%\bin” and “%HADOOP_HOME%\bin”.

