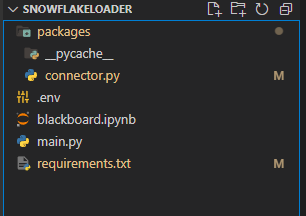
# Loader to Snowflake

Remember to open the terminal and execute: **pip install -r requirements.txt** to install all the dependencies that the project requires.

In each script you will find the code full of comments and each function’s docstrings but I will walk you through the project structure:

In the root directory you will find:

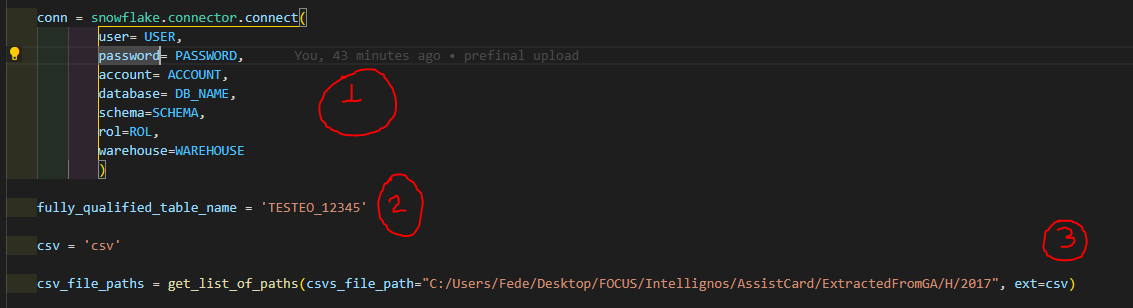


Packages:

*You Will find the explanation in the code*

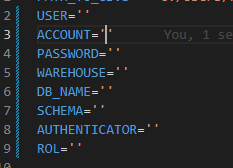
* Connector.py: Set of functions to execute what was asked.
  + **get\_list\_of\_paths()**
  + **load\_df\_to\_snowflake\_tb()**
  + **load\_csv\_to\_snowflake\_table()**
  + **load\_csvs\_to\_snowflake\_table()**
  + **run()**: wrap up the function’s program

In **run()** you must define the next variables:



1. The Snowflake connector with the required variables (which you will previously define in .env)
2. The name of the table in snowflake to upload the csv files
3. The list of csv paths (I called an auxiliary function named get\_list\_of\_paths to create a list of paths and test the function, but regarding with what the client asked to be developed, **he must pass a list of it’s own paths**)

.env:



In this file you’ll have to define what is required to connect to Snowflake via Python.

Blackboard.ipynb:

If you want to test some code before executing main.py

Main.py:

This script will run the program once you defined. env and what is needed in run()

**Modifications if deployed in Production:**

It’s most likely that our client wants to have a service deployed, when I say service, I mean an application(product) that executes several connected instructions and produces some useful output. In order to deliver this kind of service, I’d list some possible upgrades:

* **Full development in OOP**: We could leverage this development with OOP, packetize these functions and convert them into a scalable product. This will serve another project or demands from our client or ANY other client that has the same necessity. And we will do it just once, honoring DRY Concepts from software development (Don’t Repeat Yourself). Moreover, when products are developed in OOP, it’s easier to debug and build in functionalities.
* **Preprocessing, cleaning & normalizing**: Talking about specifically from our application, we should deep dive into data, it’s almost certain that we must do some preprocessing over data. Normalizing or cleaning: filling zeros, null/nan values or empty rows, standardizing values if needed, parsing datetime, casting columns as integers or strings, renaming columns, delete duplicated rows or columns or any another tasks in terms of preprocessing.
* **Keep our product safe**: We could develop a pipeline with solid logic and flow control (i.e.: our product could evaluate file’s modification date or check if preprocessing it’s OK) before deploying to avoid any possible further issues or bugs in production.
* **Multiprocessing**: We could test implementing *Multiprocessing* and try to upload multiple files at the same time. Nevertheless, we have to be really careful when doing multiprocessing, most of the libraries available executes instructions on the iterables in an unordered way, I assume that won’t be any trouble to coexist with the Pool of threads that is already working, but anyway, we have to be careful before upgrading to production this functionality. I’ll attach the first lines of code:

