pydm

The "pydm" is the Python Data Middleware that gives flexibility to interact with
databases, data analysis tool, data processing tool, algorithms library, and data visualization tool.
This middleware allows direct interaction with a database and perform SQL queries.

Package Structure & Installation:

```
datamidware/                    Top-level package
      __init__.py
      pyalgo/
          __init__.py
      pydp/
          __init__.py
      pydb/
          __init__.py
      pyviz/
          __init__.py
      pydm/
          __init__.py
          create_mysql_db.py
          file2db.py
          db2file.py
          csv2mysql.py
          json2mysql.py
          mysql2csv.py
          mysql_query.py
      settings/
            __init__.py
      tests/
          __init__.py
          test_data/
          test_result/
          test_pyviz.py
          test_search.py
          test_sort.py
          test_pydb.py
          test_pydp.py
          test_pydm
          test_mysql_query.py
```

Installation: ##

Modules:

----
1. file2db.file2db(host, user, password, filename, db_name, tb_name, file_type="file_type", db_type="db_type") :

   Imports raw structured/semi-structured data (csv, json) into database (MySQL, NoSQL).

   Parameters:
      host: host name
      user: user name
      password: password
      filename: filename to send to database
      file_type: file type (csv, json), str
      db_type: database type (mysql, nosql), str
      tb_name: name of the table where data will be stored

   Import statement: from datamidware.pydm import file2db

2. db2file.db2file(host, user, password, file_path, db_name, tb_name, file_type="file_type", db_type="db_type") :

    Exports table data as csv/json format from the database

    parameters:
        host: host name
        user: user name
        password: password
        file_path: file path to export data from database
        file_type: file type (csv, json)
        db_type: database type (mysql, nosql)
        db_name: database name from where data is exported
        tb_name: name of the table from where data will be exported

Import statement: from datamidware.pydm import db2file

3. create_mysql_db.create_mysql_db(host, user, password, db_name) :

    Creates a new MySQL database

    parameters:
        host: host name
        user: user name
        password: password
        db_name: database name to be created

Import statement: from datamidware.pydm import create_mysql_db

4. csv2mysql.csv2mysql(host, user, password, filename, db_name, tb_name) :

    Imports csv file into mysql database

    Parameters:
        host: host name
        user: user name
        password: password
        filename: filename to send to database
        db_name: name of the database -- if database already exists, import data
            in the existing database, if not exists, create new database and import
            data.
        tb_name: name of the table -- if table already exists, add data
            in the existing table, if not exists, create new table and import
            data.

Import statement: from datamidware.pydm import csv2mysql

5. json2mysql.json2mysql(host, user, password, filename, db_name, tb_name, key=None) :

    Imports json file and converts json file into pandas DataFrame.
    Sends DataFrame to mysql database table

    Parameters:
        host: host name
        user: user name
        password: password

filename: filename to send to database
db_name: name of the database -- if database already exists, import data
  in the existing database, if not exists, create new database and import
  data.
tb_name: name of the table -- if table already exists, add data
  in the existing table, if not exists, create new table and import
  data.
key: json key name to create mysql table

Import statement: from datamidware.pydm import json2mysql

6. mysql2csv.mysql2csv(host, user, password, file_path, db_name, tb_name) :

Exports csv file from mysql database table

Parameters:
host: host name
user: user name
password: password
file_path: file path to save csv file
db_name: name of the database from where data will be exported
tb_name: name of the table from where data will be exported

Import statement: from datamidware.pydm import mysql2csv

. class mysql_query.MySQLDatabase() :
Database connection class.
Performs mysql queries.
For example:

--> mysql_query.MySQLDatabase.select(tb_name, row_count="all")

Execute SQL query: SELECT * FROM table.
Selecting all(or one if row_count="one") rows from the table.

Parameters:
query: SQL query to select rows: SELECT * FROM <table>
row_count: "all" or "one" row. default "all".
return: list of rows selected.

--> mysql_query.MySQLDatabase.drop_column(tb_name, col_name)

Drop a column in a table.

Execute SQL query:

```
"ALTER TABLE <table name>
    DROP COLUMN <column name>"
```

Parameters:
tb_name: The name of the table to modify
col_name: The name of the column to delete from the table.
return: number of rows affected after modification

--> mysql_query.MySQLDatabase.rename_column(tb_name, old_name, new_name, col_def, col_pos=None)
Rename a column in a table.

Execute SQL query:

```
"ALTER TABLE <table name>
```

```
            CHANGE COLUMN <old name> <new name>
            column_definition
            [ FIRST | AFTER column_name ]"

        Parameters:
          tb_name: The name of the table to modify
          old_name: The column name to rename
          new_name: The new name for the column
          col_def: The data type and definition of the column (NULL or NOT NULL, etc).
                You must specify the column definition when renaming the column,
                even if it does not change.
          col_pos: Optional. It tells MySQL where in the table to position the column,
                if you wish to change its position.
          return: number of rows affected after modification
```

Configuration file & Unittest:

# To test all the modules, run following unittest command
from the top-level directory

```
python3 -m unittest tests/test_pydm.py
```

Note: To successfully run the test, config.ini file needs to be updated

To write config.ini file, follow the below steps:

-> go to settings
-> update write_config.py with database connection credentials, for e.g.,

(for MYSQL database connection)

```
    config_object["MYSQL"] = {
      "host": "hostname",
      "user": "username",
      "password": "password"
    }
```

===