**1. pydm**

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
7. ***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