



COMBINED ASSESSMENT

Software Engineering & Data Wrangling with SQL

Done by: Mohamed Abid

Date: January 6, 2021

Contents

1	Introduction	5
2	Install and import packages	6
3	Initialize the parameters	8
4	Connect to the database	8
5	Extract the table “SurveyStructure” from the database and update the view “vw_AllSurveyData”.	12
6	Tools	16
6.1	Data manipulation in Python	16
6.2	Connection to the database	18
6.3	About the function “install_and_import”	18
7	Conclusion	19

List of Figures

1	Illustration of successful import of packages.	6
2	Illustration of successful download and installation.	6
3	Illustration of impossible download of packages.	7
4	Automatic connection to the database.	8
5	Database connection menu.	9
6	Manual connection menu (parameters entered one by one).	10
7	Manual connection menu (parameters entered in the same text). . .	10
8	Generation of the view since the CSV file does not exist.	13
9	Nothing is done since the two table "SurveyStructure" are the same.	14
10	Generation of the view since the two table "SurveyStructure" are different.	15
11	The constructor of the class "Table".	16
12	Segment of the function "get_all_survey_data" illustrating the creation of the Dataframe "df_currentQuestion".	17
13	The function "refresh_survey_view".	18

List of diagrams

1	The general diagram of the program.	5
2	Install and import packages.	7
3	Connect to the database.	11
4	Compare and update “SurveyStructure”.	12

1 Introduction

The aim of this program is to communicate with a data base server in order to generate and possibly extract the always fresh view (pivoted survey data). For that, it inspects the table “SurveyStructure” in the database and compare it with its last copy in the hard drive. The generation of the view is performed only if a modification has been noted. The general diagram of the program is described in Diagram 1.

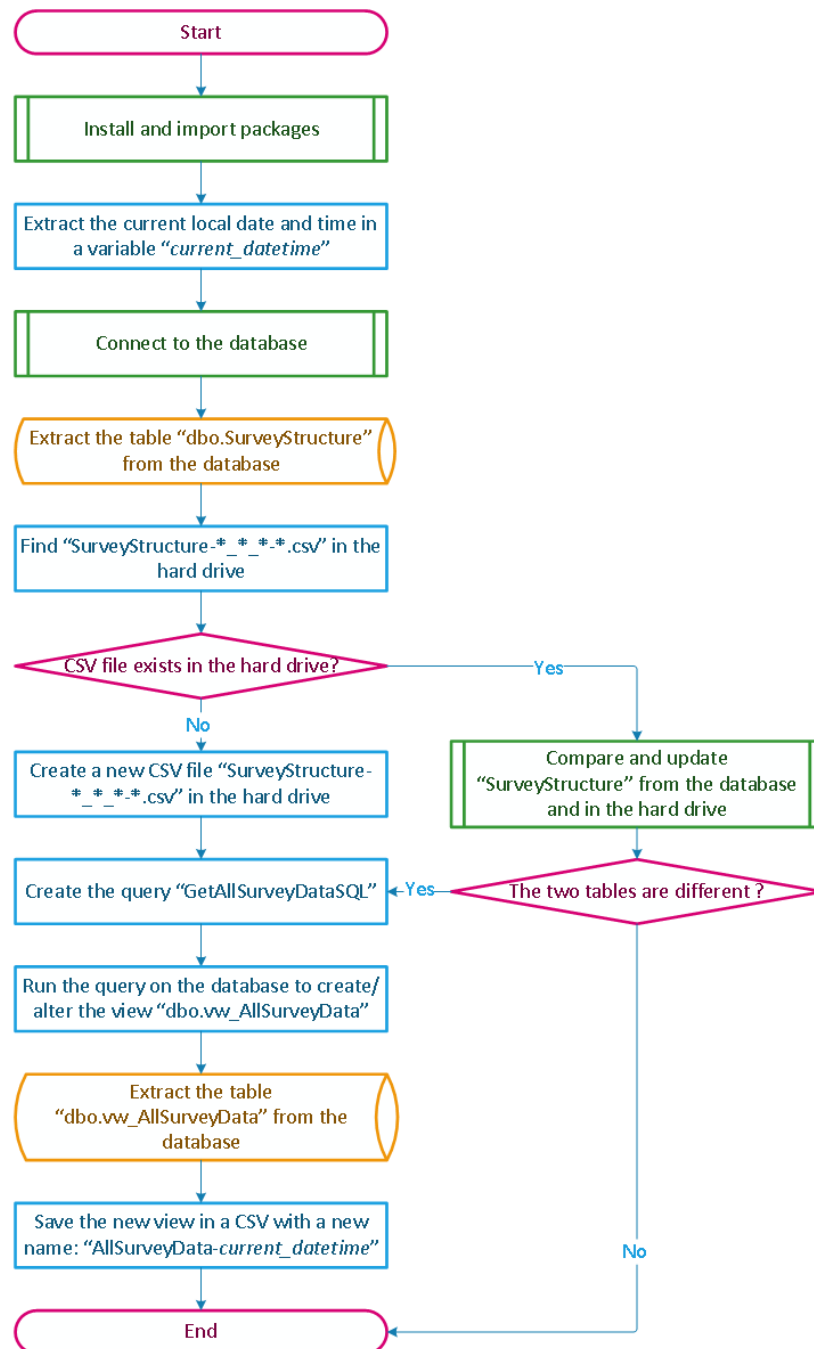


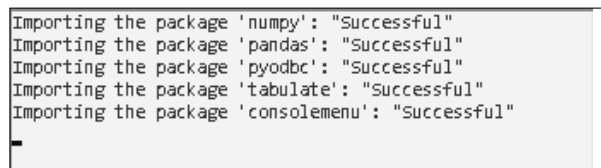
Diagram 1: The general diagram of the program.

The code is composed of three python files:

1. “PivotSurveyAnswers.py”: containing the main function, set as startup file.
2. “CompTools.py”: gathering all the functions we made and which are called by the main function.
3. “base.py”: containing
 - The commands we used to import the required packages
 - The function “install_and_import” used to try to import the packages which do not exist by default in Python and download and install them if they are missing.
 - The class “Table” inheriting from “pandas.DataFrame” to represent data tables in the form of Dataframes. This allows to facilitate manipulation of these tables.

2 Install and import packages

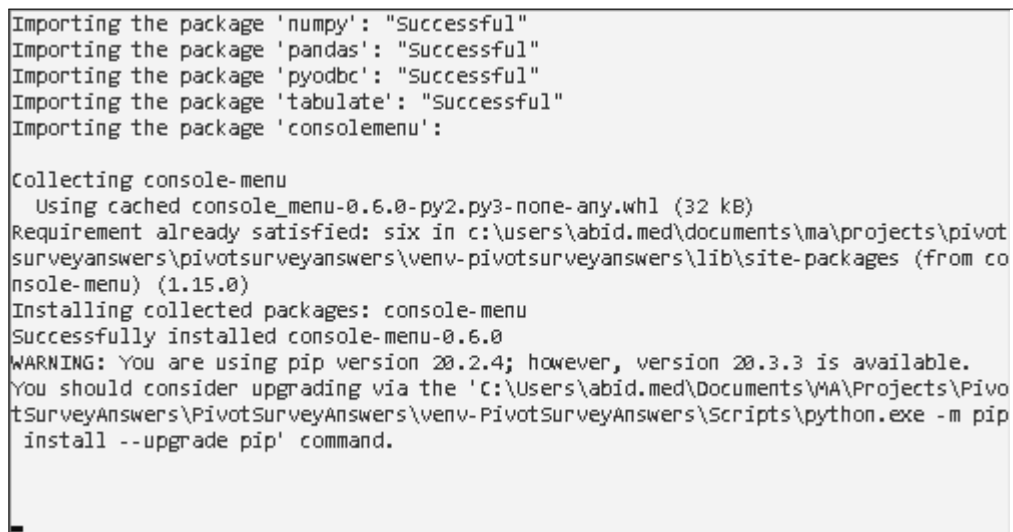
Based on Python 3, the program starts by importing the needed packages, some of the them namely : “numpy”, “pandas”, “pyodbc”, “tabulate”, “consolemenu” may not be available. The program tries to import them (see: Fig. 1).



```
Importing the package 'numpy': "Successful"
Importing the package 'pandas': "Successful"
Importing the package 'pyodbc': "Successful"
Importing the package 'tabulate': "Successful"
Importing the package 'consolemenu': "Successful"
```

Figure 1: Illustration of successful import of packages.

If they are missing, it downloads and installs them automatically (see: Fig. 2).



```
Importing the package 'numpy': "Successful"
Importing the package 'pandas': "Successful"
Importing the package 'pyodbc': "Successful"
Importing the package 'tabulate': "Successful"
Importing the package 'consolemenu':

Collecting console-menu
  Using cached console_menu-0.6.0-py2.py3-none-any.whl (32 kB)
Requirement already satisfied: six in c:\users\abid.med\documents\ma\projects\pivot
surveyanswers\pivotsurveyanswers\venv-pivotsurveyanswers\lib\site-packages (from co
nsole-menu) (1.15.0)
Installing collected packages: console-menu
Successfully installed console-menu-0.6.0
WARNING: You are using pip version 20.2.4; however, version 20.3.3 is available.
You should consider upgrading via the 'C:\Users\abid.med\Documents\MA\Projects\Pivo
tSurveyAnswers\PivotSurveyAnswers\venv-PivotSurveyAnswers\Scripts\python.exe -m pip
install --upgrade pip' command.
```

Figure 2: Illustration of successful download and installation.

In case there is no internet connection the program notifies the user before exiting (see: Fig. 3).

```
Importing the package 'numpy': "Successful"  
Importing the package 'pandas': "Successful"  
Importing the package 'pyodbc': "Successful"  
Importing the package 'tabulate': "Successful"  
Importing the package 'consolemenu':  
  
> Enable to establish internet connection to download the package 'console-menu'.  
Press any key to continue . . . ■
```

Figure 3: Illustration of impossible download of packages.

The general diagram of this step is described in Diagram 2.

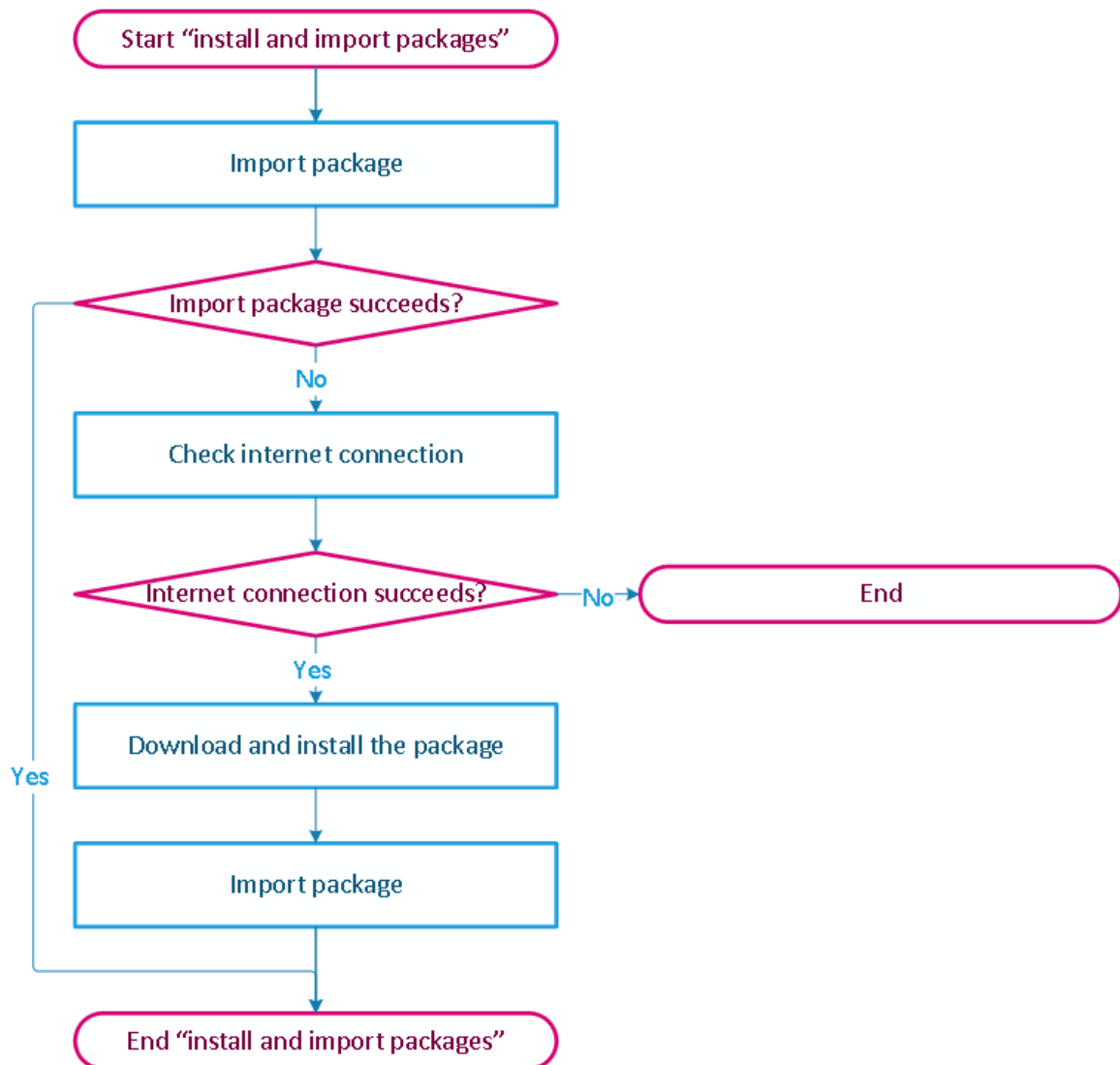


Diagram 2: Install and import packages.

3 Initialize the parameters

In the first step, we start by initializing the important variables in the program namely:

- The name of the database: “Survey_Sample_A19”
- The name of the view in the database: “vw_AllSurveyData”
- The name of the folder in which the data will be stored: “Data_PivotSurveyAnswers”
- The name of the CSV files corresponding to the view and the table in the database “SurveyStructure”, appended with the current date and time.

In the second step we create the folder “Data_PivotSurveyAnswers” if it does not exist in which data will be stored.

4 Connect to the database

The connection to the database has been carried out using the package “pyodbc”. By default the program attempts to automatically connect to the database by looking for the required parameters without asking the user (see: Fig. 4).

```
Importing the package 'numpy': "Successful"
Importing the package 'pandas': "Successful"
Importing the package 'pyodbc': "Successful"
Importing the package 'tabulate': "Successful"
Importing the package 'consolemenu': "Successful"

The folder in which data are saved is 'Data_PivotSurveyAnswers'

Connecting to the database 'Survey_Sample_A19' with the following parameters:

    The name of the driver: ODBC Driver 17 for SQL Server
    The name of the server: DESKTOP-GHJ1950
    Authentication mode : "Windows authentication"

Successfully connected.
```

Figure 4: Automatic connection to the database.

If the connection fails it gives the handle to the user to attempt manual connection (see: Fig. 5).

```
Connecting to the database 'Survey_Sample_000' with the following parameters:

    The name of the driver: ODBC Driver 17 for SQL Server
    The name of the server: DESKTOP-GHJ1950
    Authentification mode : "Windows authentication"

Connection error to the database. Error Code: 28000

Message:
[28000] [Microsoft][ODBC Driver 17 for SQL Server][SQL Server]Login failed for user 'DESKTOP-GHJ1950\abid.med'. (18456) (SQLDriverConnect); [28000] [Microsoft][ODBC Driver 17 for SQL Server][SQL Server]Cannot open database "Survey_Sample_000" requested by the login. The login failed. (4060); [28000] [Microsoft][ODBC Driver 17 for SQL Server][SQL Server]Login failed for user 'DESKTOP-GHJ1950\abid.med'. (18456); [28000] [Microsoft][ODBC Driver 17 for SQL Server][SQL Server]Cannot open database "Survey_Sample_000" requested by the login. The login failed. (4060)

Impossible to establish an automatic connection to the database "Survey_Sample_000".

DataBase connection menu

These are the possible choices:

    1 - Enter manually the parameters of the connection
    2 - Enter all the connection parameters in the same text
    3 - Exit Application

Please select your entry

>> -
```

Figure 5: Database connection menu.

Two manual connections are supported:
 In the first type of connection, the user should enter the parameters one by one (see: Fig. 6).

```

DataBase connection menu

These are the possible choices:

1 - Enter manually the parameters of the connection
2 - Enter all the connection parameters in the same text
3 - Exit Application

Please select your entry

>> 1
Enter the name of the database: Survey_Sample_A19
Enter the name of the driver : ODBC Driver 17 for SQL Server
Enter the name of the server :
Enter the the user ID : sa
Enter the password : SQL19
Connecting to the database 'Survey_Sample_A19' with the following parameters:
The name of the driver: ODBC Driver 17 for SQL Server
The name of the server: DESKTOP-GHJ1950
Authentication mode : User ID : sa
                        Password: SQL19
Successfully connected.

```

Figure 6: Manual connection menu (parameters entered one by one).

In the second type of connection, all parameters are entered in the same text (see: Fig. 7).

```

>> 2
Enter the text of the connection:
e.g.: DRIVER={ODBC Driver 17 for SQL Server};SERVER=DESKTOP-GHJ1950;DATABASE=Survey_Sample_A19;Trusted_Connection=yes
>> DRIVER={ODBC Driver 17 for SQL Server};SERVER=DESKTOP-GHJ1950;DATABASE=Survey_Sample_A19;Trusted_Connection=yes
Connecting to the database with the following parameters:
DRIVER={ODBC Driver 17 for SQL Server};SERVER=DESKTOP-GHJ1950;DATABASE=Survey_Sample_A19;Trusted_Connection=yes
Successfully connected.

```

Figure 7: Manual connection menu (parameters entered in the same text).

The general diagram of this step is described in Diagram 3.

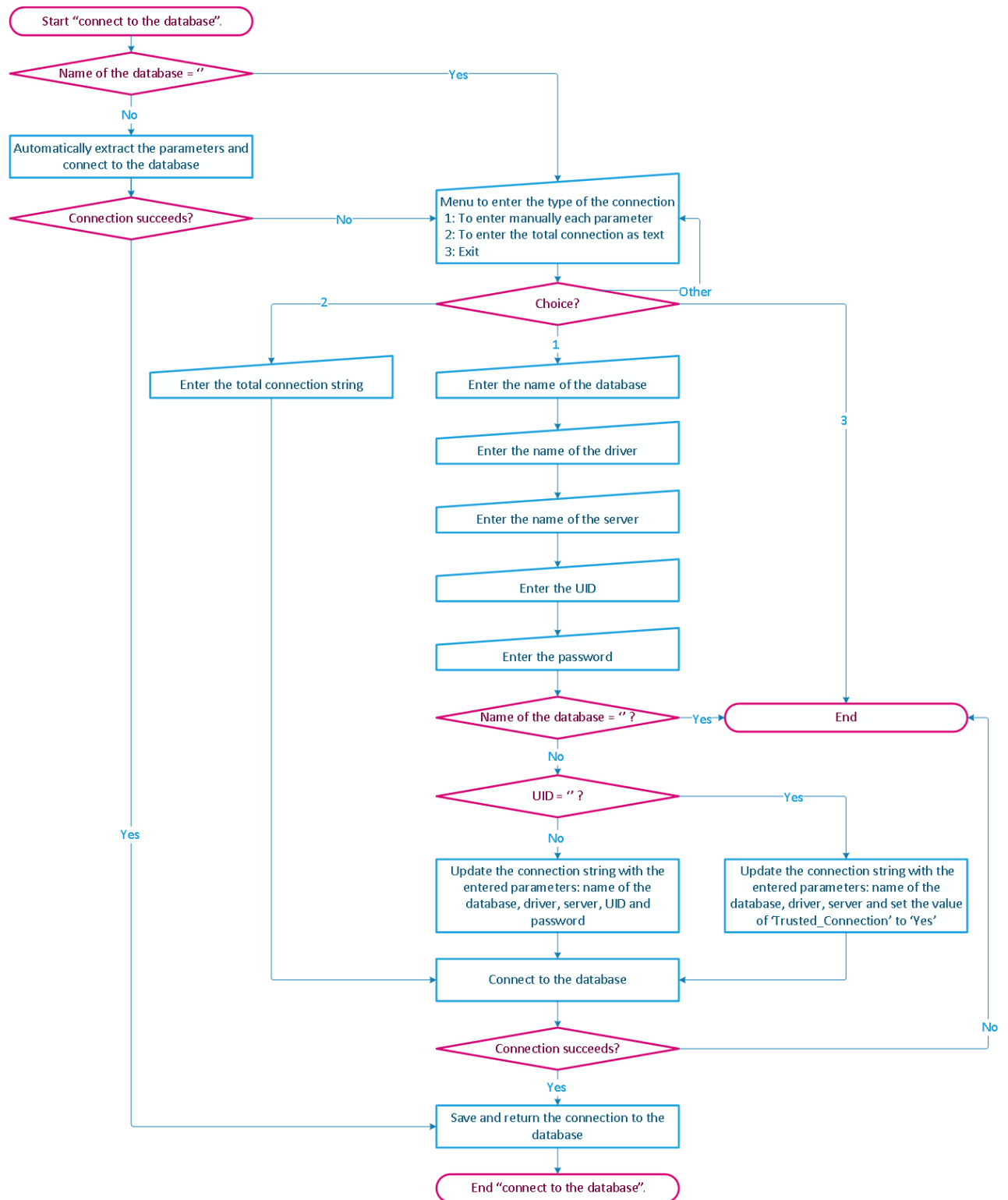


Diagram 3: Connect to the database.

5 Extract the table “SurveyStructure” from the database and update the view “vw_AllSurveyData”.

To replicate the behaviour of the trigger, the view (pivoted survey data) in the database is created or updated only if the table “SurveyStructure” has been modified. To achieve this goal the program compares between the table “SurveyStructure” in the database and the last copy of it in the hard drive (CSV file of “SurveyStructure”). The general diagram of this step is described in Diagram 4.

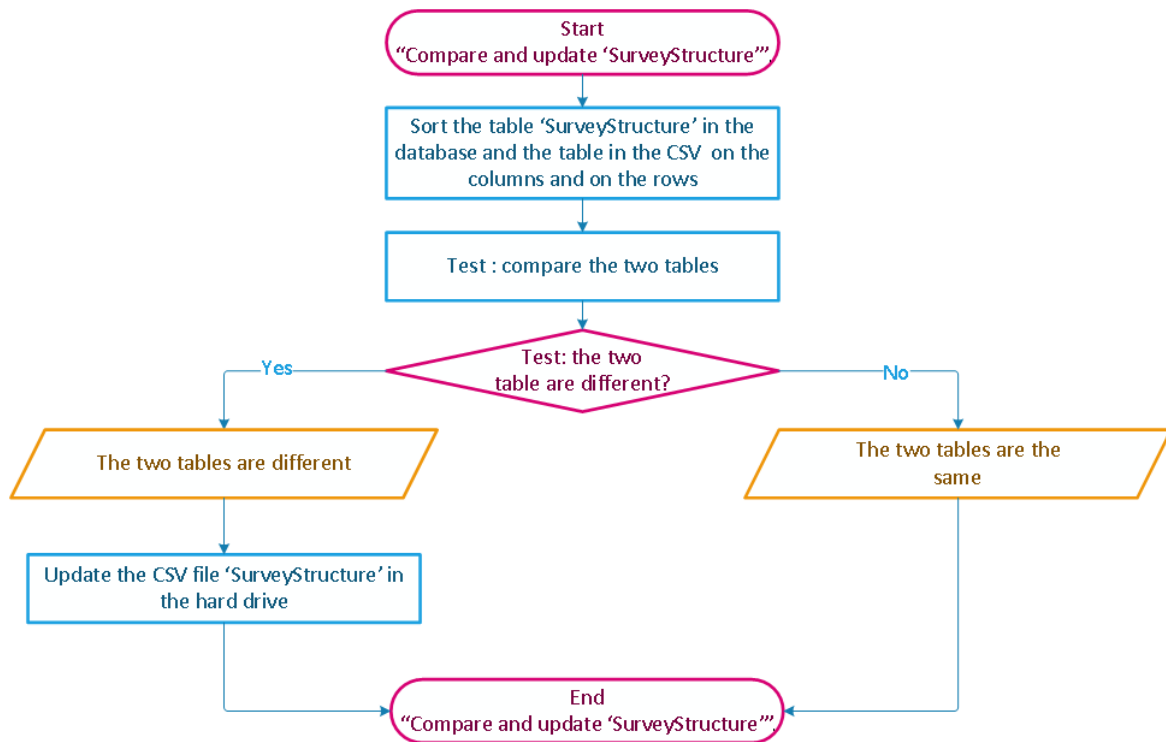


Diagram 4: Compare and update “SurveyStructure”.

This comparison may result in three different scenarios.

Scenario 1: The CSV file of “SurveyStructure” does not exist in the hard drive
In this case (see: Fig. 8):

- * The CSV file of “SurveyStructure” is created in hard drive.
- * The program creates a string query which is then executed on the database to create the view.
- * The program extracts a copy of the view and stores a CSV copy of it in the hard drive.

5 EXTRACT THE TABLE "SURVEYSTRUCTURE" FROM THE DATABASE AND UPDATE THE VIEW "VW_ALLSURVEYDATA".

```

Successfully connected.

    The table "SurveyStructure" in the database:
    =====
+-----+-----+-----+
| SurveyId | QuestionId | OrdinalValue |
+-----+-----+-----+
| 1 | 1 | 1 |
| 1 | 2 | 2 |
| 2 | 2 | 1 |
| 2 | 3 | 2 |
+-----+-----+-----+
The CSV file SurveyStructure does not exist in the had drive
Creation of a new CSV file.
> The CSV file 'SurveyStructure-2021_01_04-174201.csv' has been correctly saved in the hard drive.
> The view vw_AllSurveyData has been successfully created or updated in the database.

    The view "vw_AllSurveyData" in the database (a sample of 10 rows):
    =====
+-----+-----+-----+-----+-----+-----+
| UserId | SurveyId | ANS_Q1 | ANS_Q2 | ANS_Q3 | ANS_Q4 |
+-----+-----+-----+-----+-----+-----+
| 1387 | 3 | nan | nan | nan | nan |
| 138292 | 2 | nan | 8 | -1 | nan |
| 219152 | 1 | 9 | -1 | nan | nan |
| 260974 | 2 | nan | 8 | -1 | nan |
| 392332 | 2 | nan | -1 | 7 | nan |
| 538255 | 2 | nan | 1 | -1 | nan |
| 798860 | 1 | -1 | 2 | nan | nan |
| 819242 | 2 | nan | -1 | 9 | nan |
| 997918 | 3 | nan | nan | nan | nan |
| 1014241 | 1 | -1 | 2 | nan | nan |
+-----+-----+-----+-----+-----+-----+

> The CSV file 'vw_AllSurveyData-2021_01_04-174201.csv' has been correctly saved in the hard drive.

The CSV files are located in :
C:\Users\abid.med\Documents\MA\Projects\PivotSurveyAnswers\PivotSurveyAnswers\Data_PivotSurveyAnswers

Thank you.

Press any key to continue . . .

```

Figure 8: Generation of the view since the CSV file does not exist.

5 EXTRACT THE TABLE "SURVEYSTRUCTURE" FROM THE DATABASE AND UPDATE THE VIEW "VW_ALLSURVEYDATA".

Scenario 2: The CSV file of "SurveyStructure" exists in the hard drive and it is identical to the table "SurveyStructure" in the database:

There is no need to generate in this case the view, a procedure that conforms with the behaviour of the trigger (see: Fig. 9).

```
Successfully connected.

    The table "SurveyStructure" in the database:
    =====
+-----+-----+-----+
| SurveyId | QuestionId | OrdinalValue |
+-----+-----+-----+
| 1         | 1           | 1             |
| 1         | 2           | 2             |
| 2         | 2           | 1             |
| 2         | 3           | 2             |
+-----+-----+-----+

    The table "SurveyStructure" in the hard drive:
    =====
+-----+-----+-----+
| SurveyId | QuestionId | OrdinalValue |
+-----+-----+-----+
| 1         | 1           | 1             |
| 1         | 2           | 2             |
| 2         | 2           | 1             |
| 2         | 3           | 2             |
+-----+-----+-----+

> The two tables of "SurveyStructure" in the database and in the hard drive are the same.
> The view will not be generated.

Thank you.

Press any key to continue . . . █
```

Figure 9: Nothing is done since the two table "SurveyStructure" are the same.

5 EXTRACT THE TABLE "SURVEYSTRUCTURE" FROM THE DATABASE AND UPDATE THE VIEW "VW_ALLSURVEYDATA".

Scenario 3: The CSV file of "SurveyStructure" exists in the hard drive and it is different from the table "SurveyStructure" in the database (see: Fig. 10):

- * The CSV file of "SurveyStructure" is created in hard drive.
- * The program creates a string query which is then executed on the database to create the view.
- * The program extracts a copy of the view and stores a CSV copy of it in the hard drive.

```
Successfully connected.

The table "SurveyStructure" in the database:
=====
+-----+-----+-----+
| SurveyId | QuestionId | OrdinalValue |
+-----+-----+-----+
| 1 | 1 | 1 |
| 1 | 2 | 2 |
| 2 | 2 | 1 |
| 2 | 3 | 2 |
| 3 | 1 | 1 |
| 3 | 4 | 2 |
+-----+-----+-----+

The table "SurveyStructure" in the hard drive:
=====
+-----+-----+-----+
| SurveyId | QuestionId | OrdinalValue |
+-----+-----+-----+
| 1 | 1 | 1 |
| 1 | 2 | 2 |
| 2 | 2 | 1 |
| 2 | 3 | 2 |
+-----+-----+-----+
> The two tables of "SurveyStructure" in the database and in the hard drive are NOT the same.

> The CSV file 'SurveyStructure-2021_01_04-201432.csv' has been correctly saved in the hard drive.

> The view vw_AllSurveyData has been successfully created or updated in the database.

The view "vw_AllSurveyData" in the database (a sample of 10 rows):
=====
+-----+-----+-----+-----+-----+-----+
| UserId | SurveyId | ANS_Q1 | ANS_Q2 | ANS_Q3 | ANS_Q4 |
+-----+-----+-----+-----+-----+-----+
| 63159 | 1 | 1 | -1 | nan | nan |
| 104550 | 3 | 3 | nan | nan | -1 |
| 141604 | 1 | 6 | -1 | nan | nan |
| 244551 | 3 | 4 | nan | nan | -1 |
| 279445 | 3 | 6 | nan | nan | -1 |
| 545933 | 2 | nan | -1 | 7 | nan |
| 683500 | 3 | 0 | nan | nan | -1 |
| 792143 | 2 | nan | 3 | -1 | nan |
| 792547 | 1 | -1 | 6 | nan | nan |
| 804804 | 3 | 3 | nan | nan | -1 |
+-----+-----+-----+-----+-----+-----+

> The CSV file 'vw_AllSurveyData-2021_01_04-201432.csv' has been correctly saved in the hard drive.

The CSV files are located in :
C:\Users\abid.med\Documents\MA\Projects\PivotSurveyAnswers\PivotSurveyAnswers\Data_PivotSurveyAnswers

Thank you.

Press any key to continue . . .
```

Figure 10: Generation of the view since the two table "SurveyStructure" are different.

6 Tools

6.1 Data manipulation in Python

This program involves manipulation of data tables to display, store and compare between them. We choose to represent these data tables as instances of a subclass of “pandas.DataFrame” which we call “Table”. This allows us to leverage all the existing methods of “pandas.DataFrame”.

The main features of the subclass are that it can generate, depending on the scenario, a DataFrame from either a string containing a query or a CSV file (see: Fig. 11).

```
class Table(pd.DataFrame):
    """
    This class inherits from pandas.DataFrame.
    The instances of this class can be created from a dataframe, an SQL query or a CSV file
    """
    # Add the attribute 'title' to the class which will be defined for each instance.
    _metadata = ['title']

    # Constructor
    @property
    def _constructor(self):
        return Table

    def __init__(self, *args, **kwargs):
        if args != ():
            super(Table, self).__init__(*args, **kwargs)

        elif kwargs != {}:
            # Extract the first key of the dictionary 'kwargs'
            key0 = next(iter(kwargs.keys()))
            # Construct a dataframe, inheriting from pd.DataFrame, from an SQL query.
            if key0 == 'query':
                try:
                    super(Table, self).__init__(pd.read_sql_query(kwargs[key0],
                                                                    kwargs['connec'],
                                                                    params = kwargs['param']))
                except Exception as exc:
                    print('Some occurred error to execute the query:\n {}'.format(exc))
                    os._exit(1)

            # Construct a dataframe, inheriting from pd.DataFrame, from a CSV file.
            elif key0 == 'file_path':
                super(Table, self).__init__(pd.read_csv(kwargs[key0], sep=';'))

        # initialization of the title of the table
        self.title = kwargs['title']
```

Figure 11: The constructor of the class “Table”.

This is used to :

- Create a DataFrame from the CSV file “SurveyStructure” in the hard drive.
- Create a DataFrame from the table “SurveyStructure” by sending an SQL query to download it from the database.
- Create a DataFrame from the view “vw_AllSurveyData” by sending an SQL query to download it from the database.
- Create a DataFrame Which is called “df_currentQuestion” that contains the number of questions and whether they exist in each survey by sending an SQL query.

In this case we create a dynamic query in which the parameters are passed as named parameters. This approach is favoured in practice for its robustness to SQL injection attacks (see: Fig. 12).

```
# main loop, over all the surveys.
for ind_SurveyID, currentSurveyId in enumerate(df_SurveyID.SurveyId):
    # Construct the answer column queries for each survey
    question_query = '''SELECT *
                        FROM
                        (
                            SELECT
                                SurveyId,
                                QuestionId,
                                1 as InSurvey
                            FROM
                                SurveyStructure
                            WHERE
                                SurveyId = ?
                            UNION
                            SELECT
                                ? as SurveyId,
                                Q.QuestionId,
                                0 as InSurvey
                            FROM
                                Question as Q
                            WHERE NOT EXISTS
                                (
                                    SELECT *
                                    FROM SurveyStructure as S
                                    WHERE S.SurveyId = ?
                                      AND S.QuestionId = Q.QuestionId
                                )
                        ) as t
                        ORDER BY QuestionId;'''

    param_quest_query = [currentSurveyId] * 3

    # Parameters fed to the class 'Table' to construct a dataframe from an SQL query.
    # The parameters in 'param_quest_query' are passed to 'question_query'
    # in a way that avoids SQL-injection.
    currentQuestion_param = {'query': question_query,
                             'connec': sql_connec,
                             'param': param_quest_query,
                             'Title': 'The table current question'}

    # Create the instance 'df_currentQuestion' of the class 'Table'
    # from the query contained in 'currentQuestion_param'
    df_currentQuestion = Table(**currentQuestion_param)
```

Figure 12: Segment of the function “get_all_survey_data” illustrating the creation of the Dataframe “df_currentQuestion”.

6.2 Connection to the database

The connection to the database is ensured by the package “pyodbc” which is suitable to establish connections to “MSSQL Server”. In this program we communicate to the database by sending and executing a query. This step is performed in two different ways.

- The first manner involves using the package “Pandas” and in particular the class “Table” to send the query and get back its result in the form of a Dataframe (see: Fig. 11).
This is used to create the Dataframes: “df_SurveyStructure”, “df_vw_AllSurveyData” and “df_currentQuestion”.
- The second manner involves executing the query by the cursor of the connection to the database. This is used to create/alter the view in the database. The related function is “refresh_survey_view” defined in the file “CompTools.py” (see: Fig. 13).

```
def refresh_survey_view(sql_connc, view_table_db):
    """
    This function run the SQL query on the database to create the view.
    """
    # Create the SQL query text
    str_query = get_all_svey_data(sql_connc)
    survey_data_query = 'CREATE OR ALTER VIEW {} AS ( {} );'.format(view_table_db, str_query)
    # Create a cursor from the SQL connection
    curs = sql_connc.cursor()

    try:
        # Run the created SQL query on the database
        curs.execute(survey_data_query)
        print('> The view {} has been successfully created or updated in the database.'.format(view_table_db))
    except Exception as e:
        print(e)

    # Close the cursor
    curs.close()
    return
```

Figure 13: The function “refresh_survey_view”.

6.3 About the function “install_and_import”

This function defined in the file “base.py” is used to install and import any missing package. It takes three arguments:

- “package_name”: refers to the name of the package to be used in the installation.
- “import_packg”: refers to the name of the package used in the import function. It may be different from the package_name like for instance the package “console-menu” for which “package_name = console-menu” while “import_packg = consolemenu”.
- “version_packg”: refers to the version of the package to be installed. This is used to avoid any incompatibilities with the system or between packages.

I encountered this problem at the beginning when I tried to use “Numpy 1.19.4”, the import of which turning out to be impossible after installation according to: <https://github.com/numpy/numpy/issues/17726>

7 Conclusion

This project, programmed in Python, aims to keep track of any changes in the table “SurveyStructure” to update and download accordingly the view “vw_AllSurveyData”. It involved establishing connection and sending queries to the database which are among the popular operations for communicating with SQL databases.

When doing this project, I investigated the use of two other approaches to manipulate the tables in the database.

The first one consists in downloading all the databases, creating the view in python and sending it to the database using the package “PySpark”. However, this method may be computationally expensive, especially when big databases are considered.

The second approach consists in using Object Relational Mapper (ORM) such as “SQLAlchemy ORM” to directly manipulate data on “MSSQL Server” databases from Python. This approach offers a wider scope of operations on the database but it may require more access rights.

All in all, this project allows me to strengthen my knowledge in Python by practicing it to solve a real life problem.