Dealing with data quality and checks

Let's say that you have a problem at hand, where the external tool is displaying much different numbers than what people are used to from the internal database. You can export the data from an external source, and you also have data available from the database. Both are available in CSV.

- *What tools would you choose for the task (which programs, libraries)?*
- *What would be your approach for merging the two datasets, so you could discover which rows are missing either in the 3rd party destination or the internal database?*
- *How would you clean the data or transform the values? For example to put all values in lowercase, or to transform the string values into numeric ones.*

Quite an interesting scenario yet a real-world problem. I've done some research online and came to the conclusion that the best tools to complete this type of issue are using scripts. The tools are as follows:

- *What tools would you choose for the task (which programs, libraries)?*
    - Programming language: Python
    - Library which can be used: Pandas
    - IDE: Visual studio or Jupyter Notebook
    - Test CSV files for demonstration of how it will work

- *What would be your approach for merging the two datasets, so you could discover which rows are missing either in the 3rd party destination or the internal database?*

  With the help of the Python program and Pandas library, get the two CSV files in a data frame variable.

```
import pandas as pd

file_1_df = pd.read_csv('file1.csv') #This is the file from our database
that contains the true information and is complete

file_2_df = pd.read_csv('file2.csv') #This is the file from the other
database that has inconsistencies
```

Now that we have the two CSV files together, we will be merging them since all columns and rows should match as outer joints. But for us to identify which columns belong to which file, we will add a prefix to every column within a file.

```
file_1_df = file_1_df.add_prefix('file_1_')
file_2_df = file_2_df.add_prefix('file_2_')

merged_files_df =
pd.merge(file_1_df,file_2_df,how='outer',left_on='file_1_serialnumber',
right_on='serialnumber')
```

Now that the dataset is merged, we can write a function to identify if one column of file1 matches with the date on the same column of file2. For that, we create a function that will compare the columns. If they match, return 1 else return 0.

```
def compare_emailAddress(df):
      if df['file_1_emailaddress'] == df ['file_2_emailaddress']:
            return 1
          else:
            return 0
```

Once the function is created, we will send the merged file into this function and add a column in the merge file as emailaddress_compare, which will hold boolean values that indicate 1 as matched and 0 as not matched.

```
merged_files_df['emailaddress_compare'] =
merged_files_df.apply(compare_emailAddress,axis=1)
```

To filter down further the discrepancies, we can only show the rows that have a boolean value as 0 (this indicates the row with discrepancy) on column emailaddress_compare.

```
merged_files_df[merged_files_df['emailaddress_compare'] == 0]
```

By doing this, we get the exact row which can be corrected with the true information. The next step is to just copy all the true information into the one with discrepancies.

- *How would you clean the data or transform the values? For example to put all values in lowercase, or to transform the string values into numeric ones.*

Assuming that the file1.csv has a column payment where all numbers are strings. We can change their data type to an integer with the following command. The same applies to converting a column to string

```
file_2_df['payment'] = file_2_df['payment'].astype(int)
file_2_df['postalcode']= frame['postalcode'].astype(str)
```