Log of tested solutions

Project BCI Global Data Cleaning

Jeron Kamp, Antonella Portugal, Jakub Marcinek, Maxwell Agbana, Sophie Hu, Max Verwijmeren

Contents

[Create a document for keeping track of our Tested solution 3](#_Toc155613066)

[Python Scripts for file Integrity Check 4](#_Toc155613067)

[Test Great Expectations Library 5](#_Toc155613068)

[Test pydqc library 6](#_Toc155613069)

[Test Soda Library 7](#_Toc155613070)

[Normalize city names (version 1) 8](#_Toc155613071)

|  |  |  |  |
| --- | --- | --- | --- |
| Executed by: | Date: | Task/Goal: | Context: (files/scripts) |
| Jeron | 20/11/2023 | Create a document for keeping track of our Tested solution | Log of Tested solutions: Github |
| Key points: | | | |
| * Format * Different kind of information: who, when, what, why. | | | |
| Conclusions: | | | |
| * Using this document will help us keeping track of all solution we have explored | | | |
| Sources: “Apa source” | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Executed by: | Date: | Task: | Context: (files/scripts) |
| Antonella and Max | 13-11-2023 | Python Scripts for file Integrity Check | Integrity Check Parcel File+Sales Order Anonymized.ipynb  Product Master + Interfacility - Integrity checks.ipynb |
| Key points: | | | |
| * 2 Python Scripts that both do integrity checks for the different received data files such as:   + Share of negative values   + Report for textual and ID Columns   + Report for Numerical Columns | | | |
| Conclusions: | | | |
| * Both scripts are applicable to any future files. However They do Require user input to know which columns in the dataset might be “ID” columns instead of numerical for example | | | |
| Sources: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Executed by: | Date: | Task: | Context: (files/scripts) |
| Antonella | 21-11-23 | Test Great Expectations Library | Test/Great Expectations test script.ipynb |
| Key points: | | | |
| * Tested Great Expectations Library on python notebook. * Didn’t figure out the use of batches * Didn’t figure out how to set expectations | | | |
| Conclusions: | | | |
| * We could re-explore this option if we think it could actually be useful for the client and if it is the simplest of the proposed solutions * It might be too big and complex of a solution for the kind of data we are using | | | |
| Sources:  <https://docs.greatexpectations.io/docs/> | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Executed by: | Date: | Task: | Context: (files/scripts) |
| Antonella | 29-11-23 | Test pydqc library | File:test/pydqc\_test |
| Key points: | | | |
| * Generates Friendly Excel Files * The library is able to make plot of numerical columns * It can also make comparisons and compare consistency between tables * Requires a little human intervention for defining the column types in Excel | | | |
| Conclusions: | | | |
| * It’s good for summarizing data, but still haven’t figured out the consistency and comparison checks as it requires more than one table. | | | |
| Sources:  <https://github.com/SauceCat/pydqc/tree/master> | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Executed by: | Date: | Task: | Context: (files/scripts) |
| Jeron | 29-11-23 | Test Soda Library |  |
| Key points: | | | |
| * Soda Core executes the checks to find invalid, missing, or unexpected data. * Manual selection * Works with SQL | | | |
| Conclusions: | | | |
| * Its fine for checking for missing data but is a lot of manual input when using different databases. Also is primarily focussed on missing data. | | | |
| Sources:  <https://github.com/sodadata/soda-core> | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Executed by: | Date: | Task: | Context: (files/scripts) |
| Antonella | 13/12/2023 | Normalize city names (version 1) | City normalization.ipynb |
| Key points: | | | |
| * Created 2 Python functions:   + Clean\_and\_standardize\_cities: this function standardizes cities based on whether they have the same postal code. It removes any weird characters such as dots, commas, dashes, etc. to reduce the amount of cities with different names. It’s not completely fool proof as a lot of cities still remain with different names.   + Find\_similar\_cities: based on textual similarity, this function prints how similar city names are to each other (provided they are in the same country) * This was done with 2 datasets that contain City, Country and Postal Code information: Customer Master Anonymized and Facility Master Anonymized | | | |
| Conclusions: | | | |
| * This solution is not completely fool proof yet. It might be a good idea to create a new dataset for all the cities (standardized) such as this one: <https://pkgstore.datahub.io/core/world-cities/world-cities_csv/data/6cc66692f0e82b18216a48443b6b95da/world-cities_csv.csv> and based on this give each city a unique name. | | | |
| Sources: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Executed by: | Date: | Task: | Context: (files/scripts) |
|  |  |  |  |
| Key points: | | | |
|  | | | |
| Conclusions: | | | |
|  | | | |
| Sources: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Executed by: | Date: | Task: | Context: (files/scripts) |
|  |  |  |  |
| Key points: | | | |
|  | | | |
| Conclusions: | | | |
|  | | | |
| Sources: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Executed by: | Date: | Task: | Context: (files/scripts) |
|  |  |  |  |
| Key points: | | | |
|  | | | |
| Conclusions: | | | |
|  | | | |
| Sources: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Executed by: | Date: | Task: | Context: (files/scripts) |
|  |  |  |  |
| Key points: | | | |
|  | | | |
| Conclusions: | | | |
|  | | | |
| Sources: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Executed by: | Date: | Task: | Context: (files/scripts) |
|  |  |  |  |
| Key points: | | | |
|  | | | |
| Conclusions: | | | |
|  | | | |
| Sources: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Executed by: | Date: | Task: | Context: (files/scripts) |
|  |  |  |  |
| Key points: | | | |
|  | | | |
| Conclusions: | | | |
|  | | | |
| Sources: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Executed by: | Date: | Task: | Context: (files/scripts) |
|  |  |  |  |
| Key points: | | | |
|  | | | |
| Conclusions: | | | |
|  | | | |
| Sources: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Executed by: | Date: | Task: | Context: (files/scripts) |
|  |  |  |  |
| Key points: | | | |
|  | | | |
| Conclusions: | | | |
|  | | | |
| Sources: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Executed by: | Date: | Task: | Context: (files/scripts) |
|  |  |  |  |
| Key points: | | | |
|  | | | |
| Conclusions: | | | |
|  | | | |
| Sources: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Executed by: | Date: | Task: | Context: (files/scripts) |
|  |  |  |  |
| Key points: | | | |
|  | | | |
| Conclusions: | | | |
|  | | | |
| Sources: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Executed by: | Date: | Task: | Context: (files/scripts) |
|  |  |  |  |
| Key points: | | | |
|  | | | |
| Conclusions: | | | |
|  | | | |
| Sources: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Executed by: | Date: | Task: | Context: (files/scripts) |
|  |  |  |  |
| Key points: | | | |
|  | | | |
| Conclusions: | | | |
|  | | | |
| Sources: | | | |