**Dashboard Validation AI**

**Introduction**

The aim of the project is to validate the widgets/visual present on the power bi report against the L5 database. The validation of reports is required as a slight mistake in the reports may have humongous business impacts and may harm our reputation in the industry.

**Tools Used**

* powerbiclient,
* sql\_metadata,
* psycopg2

**PowerBiClient**

The powerbiclient package lets you embed Power BI reports in Jupyter notebooks easily. You’ll be able to export data from visuals in a Power BI report to the Jupyter notebook for in-depth data exploration. You can also **filter** the report for quick analysis or use bookmarks to apply a saved view. You can install the Power BI Client for Jupyter from [PyPI](https://pypi.org/project/powerbiclient/) and find the open-sourced Python package and associate TypeScript widget on [GitHub](https://github.com/microsoft/powerbi-jupyter).

* Visuals in powerbiclient: Anything visible on the report is a visual.

**report.visuals\_on\_page**() function returns a dictionary of all visuals.

* Filters in powerbiclient: You can apply page-level filters on your report

**report.updated\_filters**() function applies filters to the report.

Page level filter format:

filtr = { '$schema': "<http://powerbi.com/product/schema#basic>",

 'target': {  'table': table,

                   'column': column},

         'operator': "In",

                'values': values}

!! Note : Hurdles in Embedding Power BI Report in jupyter notebook and their possible Solutions

* Enabling powerbiclient extension in jupyter notebook

“ !jupyter nbextension enable --py powerbiclient “

* Go to browser settings and disable JavaScript Source Map and Css Source map

Graphical user interface, application

Description automatically generated

**SQL\_Metadata**

Extracts column names and tables used by the query. Automatically conduct column alias resolution, sub queries aliases resolution as well as tables aliases resolving.

Provides also a helper for normalization of SQL queries

Supported queries syntax:

* MySQL
* PostgreSQL
* Sqlite
* MSSQL
* [Apache Hive](https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DML)

!! Installation Guide -> pip install sql-metadata

A close-up of a computer keyboard

Description automatically generated with low confidence

**Psycopg2:**

Psycopg is the most popular PostgreSQL database adapter for the Python programming language. Its main features are the complete implementation of the Python DB API 2.0 specification and the thread safety (several threads can share the same connection). It was designed for heavily multi-threaded applications that create and destroy lots of cursors and make a large number of concurrent “INSERTs or “Updates”s.

!! Installation Guide -> pip install psycopg2

Diagram

Description automatically generated

**DataBase Used - L5 Data Model**

L5, or the Dashboard data model, is the data layer that sits between the analytical outputs and the BI visualisations. It has been designed as a multi-layered fact constellation data model with the following group of entities:

* Dimensions
* Facts
  + Aggregated at Patient level
  + Aggregated at PCP level

Along with the ones above there also are a collection of intermediate tables (of the form l5.mv\_\*) that store the output of specific logical transformations and hence become the source of truth for any entity actually referencing them in the fact or dimension tables.

Lastly, L5 also has a host of lookup tables (of the form l5.lookup\_\*) that stores crosswalks, metadata and any other form of external data that might be needed to be referenced in L5

Innovaccer has taken the decision to move from Sisense to a superior BI tool that can support the vast extent of use-cases our customers throw at us. At the same time, as we acquire larger customers, the data volume in the BI layer has become unsustainably large, leading to sluggish dashboard performance and low customer satisfaction. The current L5 is a non-aggregated fact constellation data model that needs to evolve to a set of aggregated data models that can fulfil all current and future use-cases. These data models should be able to power up data at varying levels of granularity with seamless integration to provide superior performance on the dashboards.

**Some Prerequisites**

* SQL Query Format Should be same as mentioned in the Script Usage guide
* Slicers CSV file contain all slicers title or “header of the column” should have identical name as it is in database originally.

**Approach:**

* To validate the dashboard we validate every widget for all the possible combinations of slicer values given by the user.
* Our code framework applies the page level filters automatically and extract the data for all the widgets one by one from the report and simultaneously we run the corresponding SQL query and extract the data from L5 database as our source of truth.
* Once we have the data from both the sources we validate the report data and whenever there are unmatching values we dump them into a .csv file along with the filter values for which they are not matching.
* The csv file which we get as an output from the code is considered as a validation report and the user can use it to debug why values were not matching.

**Code Workflow - Helper Functions:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No.** | **Function** | **Input** | **Output** | **Explanation** |
| 1 | HelperRender | Report | Report Status | Everytime when the report is updated this function prints the status of the report. |
| 2 | Input\_BI | Report Credential:   * Username * Password * Group\_ID * Report\_ID | Report Instance | It gives report instance in jupyter notebook or it helps in embedding power BI report in jupyter notebook  itself |
| 3. | SlicerCombo | Csv file name, dictionary developed from query input | Combination of slicers values in required format. | To make queries dynamic the where values are to be passed in a required format that is achieved by this function. |
| 4 | bi\_filter | table\_name,column\_name,value | Specified filter | It gives filter for given table and column that would apply on power BI with passed value |
| 5 | FilterValues | Slicer combination list generated by SlicerCombo,dictionary generated by GetWhereSql Function,one iterator ,widget title\_name,connection for database | Validation Score | This functions keeps apply filters for all specified combinations and  also fetches data from database with changing sql every time for running slicer Combo and gives validation score. |
| 6 | GetWhereSql | Query | Dictionary with key as column names and value as table name. | Since the csv input has column names the same as the column names in the L5 database, we make a dictionary which can retrieve the tables from the query itself. |
| 7 | ChangeQuery | Query , Combination | Dynamic or Changed query | For a given combination of slicers we need a corresponding query to be run on the database that is developed by ChangeQuery function. |
| 8 | get\_active\_page | report | Dictionary of active pages on the report | To get the current active page of the report. |
| 9 | GetSqlData | Connection of database,query | Data for query | This function takes sql query and pass it to the database and fetch its resultant data |
| 10 | Extract\_Visual | report,visual\_name | visual\_ID | This function returns visual\_ID by looking at its title using visuals Information |

**GetSQLData Func:**

For a given query:

Data from database is extracted here.

**ChangeQuery Func:**

For a particular slicer combination:

Query updated here.

**Validate Function:**

Validating the data.

**Filter Values Function:**

Works recursively for all the possible combination of filters.

For a particular filter combination.

Get the data from the report using power-bi-client.

Update the query(ChangeQuery func) and get the data from L5(GetSqlData func)

Validate by calling validate helper function.

**Main Function:**

Take inputs:🡪 config.json, slicer.csv, queries.csv

Having authenticated from power-bi service, embed the report.

Calls **Filter\_Values function**