Generate a ‘Data Quality Validation Framework’ using ‘Great Expectation’ library which will run Data Quality Checks like :-

1. Count Validation
2. Checksum Validation
3. Business Rule Validation
4. Reconciliation Logic
5. Duplicate Check
6. Pattern Check
7. Enumeration Check
8. Mandatory Check
9. Range Check
10. Type Check
11. Unique Check

Data Points could be any like :-

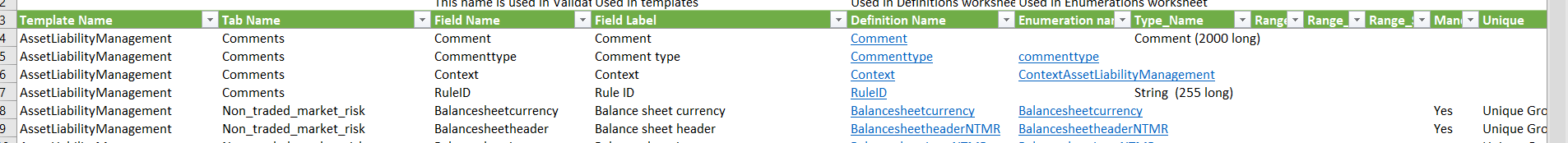
* SQL Server Table
* Feed File
* FlatFile
* Parque file
* AWS S3 Bucket Feed Files

This framework should run based on rules Dictionary:-

1. **Validations** :- In This TAB There is a- List of Unique Rules which will define which type of rules needs to be run
   1. Rule\_ID
   2. Template Name
   3. Columns Name
   4. RuleType
      1. Mandatory
      2. Range
      3. Type
      4. Unique
      5. Enumerations
      6. BusinessRules
      7. Pattern Rule

RuleType :- Range,Type,Unique :- Should pich expectations from Tab ‘Template’

1. **Template** : In This tab given is the description of Template and following are the columns:
   1. Feed Name
   2. Column Name
   3. Definition
   4. Enumeration
   5. Type\_Name
   6. Range:-
      1. Range Bottom
      2. Range Top
      3. Range Scope
   7. Mandatory
   8. Unique



1. **Enumerations** :- In this tab list of expected values (‘ENUMS’) for column get saved for each ‘non-Numeric’ field . If this is BLANK the any value is acceptable
   1. ‘Enumeration name’ should get picked from ‘Template’ Tab 🡪 Enumeration column
   2. Based on ‘Enumeration name’ , enums should be picked as an expectation for a column
2. **Pattern** : In this tab expectation of Pattern is given . Like
   1. If pattern for Feed1(Table) is :
      1. A = ‘text’ ,B = ‘texts’ , Column C = ‘text123’
      2. A = ‘text2’ ,B = ‘texts2’ , Column C = ‘text1234’
      3. A = ‘text3’ ,B = ‘texts3’ , Column C = ‘text12345’

So here expectation is: For Feed1, values of column A, column B, column C

Should be same as given pattern a, other than this rule should get fail

1. **Business Rule** : In this tab Business Rule should get define like
   1. Where Country in (India,China,Srilanka) then Sum of ColumnsNum1 and ColumnNum2 should be greater than or equal to CoulumnNum3

This Rule should run on distinct values of SDSID , ExtractMonth

Which will derive like:-

Select

SDSID,

Extractmonth,

Case if Country in (‘India’,’China’,’Srilanka’) then

Column1+Columns2 = Coumn3

Else ‘Fail’

End As “RuleID”

from stg.DataNumTable

Group by SDSID , ExtractMonth

This Rule will run for all Rows one By One and give result like failed for how many Rows and also give the Row details

1. Reconciliations : This Tab is like business Rules
   1. Rec\_ID
   2. Source/Target :- Based on this select source and target Feed details
   3. Feed Name
   4. Dimensions :- Column name (Semi Comma Seperated)
      1. Like Column1;Column2;Column3
   5. DimensionValues: Values of above mention Dimensions
      1. Ankita;(amol,Viraj);Ayushmaan

This means Column1 shuould have value ‘Ankita’

Columns2 should have values(‘Amol’,Viraj)

Column3 should have values ‘Ayushmaan’

* 1. Mesure\_Fieldname
     1. This means All **summation** of Mesure\_Fieldname filtered rows

This Reconciliation will decide summation of Measure Field should be Equal in Source and Target based on its filter

Now let's create the validation module that will handle all the data quality checks:

**blackbox wants to create a new file:**

Rollback

dq\_framework/validations.py‎

**BLACKBOXAI**

Let's create a sample usage script to demonstrate how to use the framework:

**blackbox wants to create a new file:**

Rollback

example\_usage.py‎

**BLACKBOXAI**

Let's create a README.md file to document the framework and provide usage instructions:

**blackbox wants to create a new file:**

Rollback

README.md‎

**BLACKBOXAI**

Now that we have created all the necessary files for the Data Quality Validation Framework, let me summarize what has been implemented:

**Task Completed**

I have created a comprehensive Data Quality Validation Framework with the following components:

1. Core Framework Files:

* requirements.txt: Lists all required dependencies
* config/rules\_dictionary.yaml: Configuration file for validation rules
* dq\_framework/main.py: Main framework implementation
* dq\_framework/validations.py: Implementation of all validation types
* dq\_framework/reporting/report\_generator.py: Report generation functionality
* dq\_framework/reporting/templates/report\_template.html: Beautiful HTML report template
* example\_usage.py: Example script showing framework usage
* README.md: Detailed documentation

1. Features Implemented:

* 11 types of validations (Count, Checksum, Business Rules, etc.)
* Support for multiple data sources
* Beautiful, interactive HTML reports
* PDF and Excel report generation
* Email distribution capability
* Comprehensive error handling
* Detailed logging

1. Report Features:

* Executive summary with key metrics
* Interactive visualizations
* Detailed validation results
* Error samples
* Modern, responsive design using Tailwind CSS
* Multiple export formats (HTML, PDF, Excel)

To use the framework:

1. Install dependencies: pip install -r requirements.txt
2. Configure validation rules in config/rules\_dictionary.yaml
3. Run example: python example\_usage.py

The framework is now ready to use and can be extended with additional validation types or customized reporting as needed.

<https://agent.blackbox.ai/?sandbox=9ftwl7>

<https://agent.blackbox.ai/?sandbox=9ftwl7>