**Data Quality Framework**

**Problem Statement :** Identify the data issue from big volume of data to validate in very less time with defined constraints (possibly with threshold value too i.e. null percentage).

**Solution :** Develop a Rule based generic config-based Data Quality framework to check the quality of data by checking against the validation rules, it is possible to test whether the data meet the defined criteria and possess the required attributes. In this way, provided data quality rules help detect potential weak points in processes and derive recommendations for action. This can be used for all the data ingestion/reporting to identify the issue without making any change in existed codebase. Only Config File will be changed as per the input data and required different quality check.

1. We can develop it in a way to accept different type of Data/Source as Csv, Xml, Json, Parquet, Hive, SQL, logs basically anything that we can fit into a Spark DataFrame and run the Data Quality Check over it and collect Report in a table/file to use for further reporting/analysis purpose.

This Spark based framework will be able to analyze issue very quickly from GB and TB of data. It can be designed in such a way that multiple table with different columns and rules can be check at a time.

Data quality rules allow for the measurement of different data quality dimensions, such as :

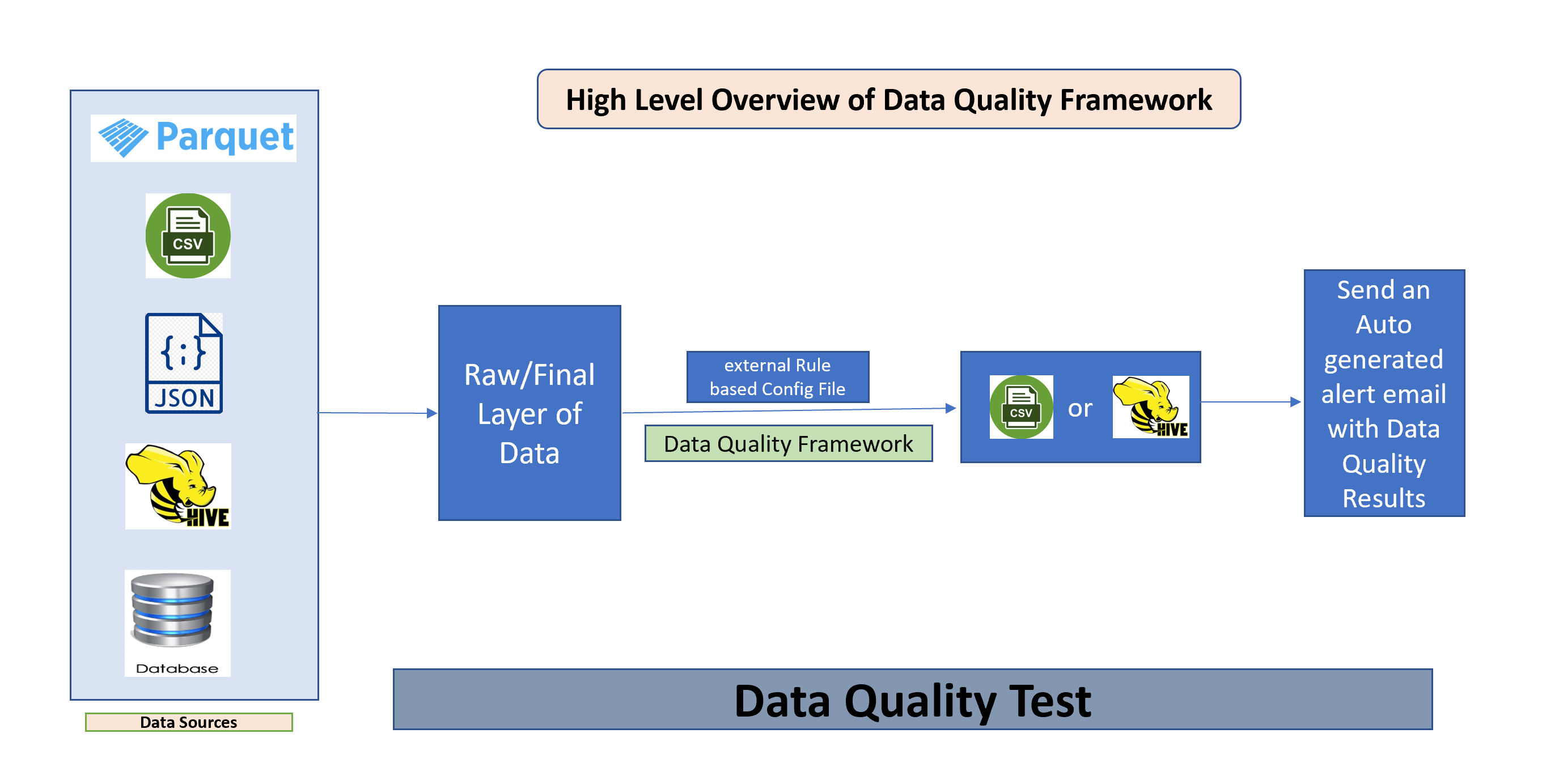
* The contextual accuracy of values (correctness, accuracy)
* The consistency among values (consistency)
* The allowed format of values (representational consistency, accuracy)
* The completeness of values

A list of most common data quality metrics is given below, but we need to select the ones that are helpful in our case and figure out the least percentile values that represent good data quality.

1. **Accuracy**: How well do data values depict reality/correctness?
2. **Lineage**: How trustworthy is the originating source of data values?
3. **Semantic**: Are data values true to their meaning?
4. **Structure**: Do data values exist in the correct pattern and/or format?
5. **Completeness**: Is your data as comprehensive as you need it to be?
6. **Consistency**: Do disparate data stores have the same data values for the same records?
7. **Currency**: Is your data acceptably up to date?
8. **Timeliness**: How quickly is the requested data made available?
9. **Reasonableness**: Do data values have the correct data type and size?
10. **Identifiability**: Does every record represent a unique identity and is not a duplicate?

**Some Examples of Data Quality Rules** :

1. Size or Count Check
2. Uniqueness
3. Duplicate
4. DataType
5. Data
6. Regular Expression
7. Primary Key
8. Distinctness
9. Number Of Distinct Values
10. Min Length
11. Max Length
12. Min
13. Mean
14. Max
15. Sum
16. Satisfies
17. Contains (any Value)
18. Negative and NonNegative
19. Positive
20. Less Than
21. Greater Than
22. Greater Than Or Equal To
23. Equal To
24. Value Contained In
25. Data Threshold value
26. Data Check with its type i.e. String or Integer
27. Length



We can use the Data Quality Constraints like above examples to build Rules which can be use to check the quality of the given data. Rules can be Changed for different type of Data Sets.

|  |  |
| --- | --- |
| **Rule File/Config File Example :**  [{  "entity\_name": "db.table\_1",  "rules": [  {  "name": "isNonNegative",  "Failure\_Flag": "Error",  "args": {  "column": "col\_2"  }  },  {  "name": "satisfies",  "args" : {  "column": "col\_name IS NULL OR col\_name IN ('BUY','SELL')",  }  },  {  "name": "hasSize",  "args": {  "condition": ">= 100000"  }  }]},    { "entity\_name": "db.table\_2",  "rules": [  {  "name": "satisfies",  "args" : {  "column": "column\_name IS NULL OR column\_name IN('value')",  }  },  {  "name": "hasSize",  "args": {  "condition": ">= 10000"  }  },  {  "name": "hasDistinctness",  "Failure\_Flag": "Info",  "args": {  "column": ["coll"],  }}  ]}  ] |  |
|  |  |
|  |
| **Sample Output :**  **Seq validation\_rule column\_name constraint\_msg result\_status total\_rows pass\_count fail\_count rule\_name input\_data**  1 .isNonNegative("customer\_id") customer\_id ‘customer\_id’ has no negative values Success 100 100 0 Non Negetive select \* from database.table\_1  2 .isContainedIn("country", Array("US", "UK", "DE", "JP", "FR")) country ‘country’ has value range ‘US’, ‘UK’, ‘DE’, ‘JP’,'FR' Failure 100 40 60 Contained In select \* from database.table\_2 |  |
|  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |