# Integration of DQL library into data validation components

## Key validation checks:

* Data Quality Check
  + We use statistical analysis to validate the following checks:
    - response\_exposure\_check
    - saturation\_check
    - humid\_check
* Schema Validation Check
  + We use hard defined rules to ensure input files follow a specified format:
    - \_\_check\_columns()
    - \_\_constant\_col\_val\_check()
    - \_\_check\_time\_columns()
    - \_\_check\_temp\_humid\_range()
    - \_\_check\_format()
    - \_\_sensors\_validation()
    - \_\_label\_validation()

## How is the DQL library is integrated?

The data quality source code is downloaded and placed in the validation component folder.

## Key changes required for integration:

* Created following wrapper functions (understood by pyTest) in the validation component to instantiate relevant classes of the DQL library:
  + test\_schema\_validation()
  + test\_data\_analyser()
  + test\_data\_quality()
* For each class in the DQL created parametrized constructor to give user the control over the input and output paths
  + schema\_validator = dql.SchemaValidationClass(input\_folder, output\_folder, log\_folder, log\_config)
  + data\_analyser = dql.DataAnalyticsClass(input\_folder, statistical\_analysis)
  + data\_validator = dql.DataQualityClass(input\_folder, output\_folder, statistical\_analysis,
  + log\_folder, log\_config)
* For each class in the DQL wrote interface function that allows users to initiate the quality checks:
  + schema\_validator.validate\_schema()
  + data\_analyser.create\_analytics()
  + data\_validator.validate\_data\_quality()
* Updated the DQL library such that it raises an AssertionError whenever a quality check fails