Deequ tutorial: <https://aws.amazon.com/ru/blogs/big-data/test-data-quality-at-scale-with-deequ/>

As a result:

**Auto:** a maven project with Deequ checks should be provided

**Manual:** screenshots from spark-shell should be provided

Note that deequ is scala framework.

1. Download file from AWS: s3://amazon-reviews-pds/tsv/amazon\_reviews\_us\_Camera\_v1\_00.tsv.gz

Use [AWS CLI](https://docs.amazonaws.cn/en_us/cli/latest/userguide/install-windows.html) (in case of installed pyhton the easiest way is to install via pip) or download it from AWS S3 like in deequ tutorial

1. Check that column *verified\_purchase* contains only “N” and “Y” as values
2. Check that column *review\_date* contains only dates (use regex)
3. Check that column *review\_id* contains unique not null values
4. Check that column *total\_votes* contains only integers

# Check file header

!head amazon\_reviews\_us\_Camera\_v1\_00.tsv

#Read file and create a df

US\_CAM = spark.read.csv("amazon\_reviews\_us\_Camera\_v1\_00.tsv", sep=r'\t', header=True)

#2. Check that column verified\_purchase contains only “N” and “Y” as values

#4. Check that column review\_id contains unique not null values

#5. Check that column total\_votes contains only integers

from pydeequ.checks import \*

from pydeequ.verification import \*

check = Check(spark, CheckLevel.Warning, "Review Check")

checkResult\_os = VerificationSuite(spark) \

.onData(US\_CAM) \

.addCheck(

check.isContainedIn("verified\_purchase", ["Y", "N"]) \

.isUnique("review\_id")

.isComplete("review\_id")

.hasDataType("total\_votes", ConstrainableDataTypes.Integral)) \

.run()

checkResult\_os\_df = VerificationResult.checkResultsAsDataFrame(spark, checkResult\_os)

checkResult\_os\_df.show()

#3. Check that column review\_date contains only dates (use regex)

from pydeequ.analyzers import \*

analysisResult\_os = AnalysisRunner(spark) \

.onData(US\_CAM) \

.addAnalyzer(hasPattern("review\_date","""^(19|20)\d\d([- /.])(0[1-9]|1[012])\2(0[1-9]|[12][0-9]|3[01])$""")) \

.run()

analysisResult\_os\_df = AnalyzerContext.successMetricsAsDataFrame(spark, analysisResult\_os)

analysisResult\_os\_df.show()

Table

Description automatically generated

Graphical user interface, text, application, Word, email

Description automatically generated