Declarative Data Testing using Great Expectations using Python3

Below is a guide for using Great expectations to validate your data using Python 3 and Visual Studio code IDE on your database or data set in csv format or excel format.

Please follow the steps to set up environments , I am suing mac with Zsh terminal. Make sure you have python3 or greater version installed.

1.First create a folder like data\_quality or nay name suitable in your machine .

2. Cd into your folder created and run the following command

python3 -m venv dq\_venv

this installs virtual env

3. Run the following command to activate or enter the env

source dq\_venv/bin/activate

4. Copy using copy commands or drag and drop the example.csv file or data you want to validate into this data\_quality file you created

At this point you can open VS code on thiw folder or use

source dq\_venv/bin/activate

command in terminal of VS code to get into venv

5. Install great expecataions onto this env

pip install great\_expectations

6. You can use the sample csv file or connect to postgresse or myswl using below ocmmands

pip install sqlalchemy psycopg2-binary

for sql use below

pip install pymysql

6. Now run this command

great\_expectations init

This genereated gx file

7. Create new file into data\_quality folder and Type this code inot your hopistlas.py file and

import great\_expectations as gx

# Run this file anytime against your data

context = gx.get\_context()

# Create an expectation suite

expectation\_suite\_name = "CSV\_HOSPITALS\_EXPECTATIONS\_SUITE"

context.add\_or\_update\_expectation\_suite(expectation\_suite\_name=expectation\_suite\_name)

datasource = context.sources.add\_or\_update\_pandas(name="csv\_hospitals")#any name

# Add one or more assets

asset = datasource.add\_csv\_asset(name="Hospitals",

filepath\_or\_buffer="Hospitals.csv",

encoding="ISO-8859-1")

#Test everythign in that asset or set of data in batches?Prefer batches for large sets

batch\_request = datasource.get\_asset("Hospitals").build\_batch\_request()

#connects test cases to data set -Validator

validator = context.get\_validator(batch\_request=batch\_request,expectation\_suite\_name=expectation\_suite\_name,)

validator.expect\_column\_values\_to\_not\_be\_null(column = "ID")

#Create Expectations or Tests

validator.save\_expectation\_suite(discard\_failed\_expectations=False)

#Save expectatin suite

# Create a checkpoint to run expectations and store the results

checkpoint = context.add\_or\_update\_checkpoint(

name="CSV\_HOSPITALS\_CHECKPOINT",

run\_name\_template="%Y%m%d-%H%M%S-csv-hosptials-run",

validations=[

{

"batch\_request": batch\_request,

"expectation\_suite\_name": expectation\_suite\_name,

"action\_list": [

{

"name": "store\_validation\_result",

"action": {"class\_name": "StoreValidationResultAction"},

},

{

"name": "update\_data\_docs",

"action": {"class\_name": "UpdateDataDocsAction"}

},

]

}

]

)

# Run the checkpoint

checkpoint\_result = checkpoint.run()

# Open the data docs

context.open\_data\_docs()

run the python command

Python hopistlas.py , this opens a dashboard with test case success or failure rate

You will see the generated expectations file in expectations in gx folder like below

{

"data\_asset\_type": null,

"expectation\_suite\_name": "CSV\_HOSPITALS\_EXPECTATIONS\_SUITE",

"expectations": [

{

"expectation\_type": "expect\_column\_values\_to\_not\_be\_null",

"kwargs": {

"column": "ID"

},

"meta": {}

}

],

"ge\_cloud\_id": null,

"meta": {

"great\_expectations\_version": "0.18.9"

}

}

You can add env variables file in yml for db details and use them according to prod , dev envs

You can see the validations in uncommitted folder under gx directory/folder

In epxecations.yml file you can update to S3 bucket and give team members access to it .Here currently it is in expectations folder

A screen shot of a computer

Description automatically generated

See documentation to connect to dbs.