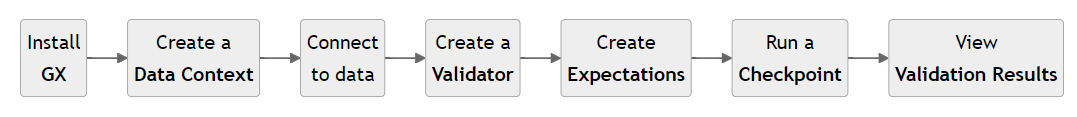
<https://docs.greatexpectations.io/docs/oss/tutorials/quickstart/>

## Data validation workflow[​](https://docs.greatexpectations.io/docs/oss/tutorials/quickstart/#data-validation-workflow)

The following diagram illustrates the end-to-end GX OSS data validation workflow that you'll implement with this quickstart. Click a workflow step to view the related content.

## Prerequisites[​](https://docs.greatexpectations.io/docs/oss/tutorials/quickstart/#prerequisites)

* An installation of Python, version 3.8 to 3.11. To download and install Python, see [Python downloads](https://www.python.org/downloads/).
* pip
* An internet browser



Install GX OSS

Run the following command in an empty base directory inside a Python virtual environment:

pip install great\_expectations

Run the following Python code to import the great\_expectations module:

Python

import great\_expectations as gx

Create a Data Context

Run the following command to create a Data Context object:

Python

context = gx.get\_context()

Connect to data

Run the following command to connect to existing .csv data stored in the great\_expectations GitHub repository and create a Validator object:

Python

validator = context.sources.pandas\_default.read\_csv(

"https://raw.githubusercontent.com/great-expectations/gx\_tutorials/main/data/yellow\_tripdata\_sample\_2019-01.csv"

)

The code example uses the default Data Context Data Source for Pandas to access the .csv data from the file at the specified URL path.

Create Expectations

Run the following commands to create two Expectations and save them to the Expectation Suite:

Python

validator.expect\_column\_values\_to\_not\_be\_null("pickup\_datetime")

validator.expect\_column\_values\_to\_be\_between(

"passenger\_count", min\_value=1, max\_value=6

)

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

The first Expectation uses domain knowledge (the pickup\_datetime shouldn't be null).

The second Expectation uses explicit kwargs along with the passenger\_count column.

The basic workflow when creating an Expectation Suite is to populate it with Expectations that accurately describe the state of the associated data. Therefore, when an Expectation Suite is saved failed Expectations are not kept by default. However, the discard\_failed\_expectations parameter of save\_expectation\_suite(...) can be used to override this behavior if you have created Expectations that describe the ideal state of your data rather than its current state.

Validate data

Run the following command to define a Checkpoint and examine the data to determine if it matches the defined Expectations:

Python

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

name="my\_quickstart\_checkpoint",

validator=validator,

)

Run the following command to return the Validation Results:

Python

checkpoint\_result = checkpoint.run()

Run the following command to view an HTML representation of the Validation Results in the generated Data Docs:

Python

context.view\_validation\_result(checkpoint\_result)

1. Install GE

pip install great\_expectations

1. In cmd init GE

great\_expectations init

It create necessary folders structure:

A screenshot of a computer

Description automatically generated

1. In cmd line run (Create new datasourse )

great\_expectations datasource new

3.1 What data would you like Great Expectations to connect to?

2. Relational database (SQL)

3.2 Which database backend are you using?:

9. other - Do you have a working SQLAlchemy connection string?

A screenshot of a computer screen

Description automatically generated

It will open automatically generated notebook where you just need to put your settings

Pay attention on:

connection\_string = f"""mssql+pyodbc://{user}:{password}@localhost/{db}?driver=ODBC+Driver+17+for+SQL+Server&charset=utf&autocommit=true"""

this notebook exist in folder uncommitedA screenshot of a computer

Description automatically generated

In Jupiter notebook run all commands from file datasource\_new.py.

1. Then Create new suite

In cmd line

great\_expectations suite new

there appears questions:

4.1 How would you like to create your Expectation Suite?

1. Manually, without interacting with a sample Batch of data (default)

2. Interactively, with a sample Batch of data

3. Automatically, using a Data Assistant

:3

4.2 Select data\_connector

1. default\_runtime\_data\_connector\_name

2. default\_inferred\_data\_connector\_name

3. default\_configured\_data\_connector\_name

: 3

4.3 Which data asset (accessible by data connector "default\_configured\_data\_connector\_name") would you like to use?

1. employee

4.4 give name for you suite:

employee\_suite

A screenshot of a computer

Description automatically generated

Run Jupiter notebook

Here add your expectations which you would like to save in suite

A screenshot of a computer

Description automatically generated

The results of checks will be present in data\_docs

Index.html

A screenshot of a computer

Description automatically generated

1. Create checkpoint:

great\_expectations checkpoint new employee\_checkpoint

run Jupiter notebook

For running already saved tests:

great\_expectations checkpoint run employee\_checkpoint