Create a new pandas Datasource

Use this notebook to configure a new pandas Datasource and add it to your project.

In [1]:
```
import great_expectations as ge
from great_expectations.cli.datasource import sanitize_yaml_and_save_datasource, check
context = ge.get_context()
```

Customize Your Datasource Configuration

If you are new to Great Expectations Datasources, you should check out our how-to documentation

My configuration is not so simple - are there more advanced options? Glad you asked! Datasources are versatile. Please see our How To Guides!

Give your datasource a unique name:

In [2]:
```
datasource_name = "my_pandas_datasource"
```

For files based Datasources:

Here we are creating an example configuration. The configuration contains an **InferredAssetFilesystemDataConnector** which will add a Data Asset for each file in the base directory you provided. It also contains a **RuntimeDataConnector** which can accept filepaths. This is just an example, and you may customize this as you wish!

Also, if you would like to learn more about the **DataConnectors** used in this configuration, including other methods to organize assets, handle multi-file assets, name assets based on parts of a filename, please see our docs on InferredAssetDataConnectors and RuntimeDataConnectors.

In [3]:
```
example_yaml = f"""
name: {datasource_name}
class_name: Datasource
execution_engine:
  class_name: PandasExecutionEngine
data_connectors:
  default_inferred_data_connector_name:
    class_name: InferredAssetFilesystemDataConnector
    base_directory: ..\data
    default_regex:
      group_names:
        - data_asset_name
      pattern: (.*)
  default_runtime_data_connector_name:
    class_name: RuntimeDataConnector
    assets:
      my_runtime_asset_name:
        batch_identifiers:
          - runtime_batch_identifier_name
"""
print(example_yaml)
```

```
name: my_pandas_datasource
class_name: Datasource
execution_engine:
  class_name: PandasExecutionEngine
data_connectors:
  default_inferred_data_connector_name:
    class_name: InferredAssetFilesystemDataConnector
    base_directory: ..\data
    default_regex:
      group_names:
        - data_asset_name
      pattern: (.*)
  default_runtime_data_connector_name:
    class_name: RuntimeDataConnector
    assets:
      my_runtime_asset_name:
        batch_identifiers:
          - runtime_batch_identifier_name
```

Test Your Datasource Configuration

Here we will test your Datasource configuration to make sure it is valid.

This `test_yaml_config()` function is meant to enable fast dev loops. **If your configuration is correct, this cell will show you some snippets of the data assets in the data source.** You can continually edit your Datasource config yaml and re-run the cell to check until the new config is valid.

If you instead wish to use python instead of yaml to configure your Datasource, you can use `context.add_datasource()` and specify all the required parameters.

In [4]:
```
context.test_yaml_config(yaml_config=example_yaml)
```

```
Attempting to instantiate class from config...
        Instantiating as a Datasource, since class_name is Datasource
        Successfully instantiated Datasource


ExecutionEngine class name: PandasExecutionEngine
Data Connectors:
        default_inferred_data_connector_name : InferredAssetFilesystemDataConnector

        Available data_asset_names (2 of 2):
                first_data.csv (1 of 1): ['first_data.csv']
                second_data.csv (1 of 1): ['second_data.csv']

        Unmatched data_references (0 of 0):[]

        default_runtime_data_connector_name:RuntimeDataConnector

        default_runtime_data_connector_name : RuntimeDataConnector

        Available data_asset_names (1 of 1):
                my_runtime_asset_name (0 of 0): []

        Unmatched data_references (0 of 0):[]
```

```
Out[4]: <great_expectations.datasource.new_datasource.Datasource at 0x1ab437ba250>
```

Save Your Datasource Configuration

Here we will save your Datasource in your Data Context once you are satisfied with the configuration. Note that `overwrite_existing` defaults to False, but you may change it to True if you wish to overwrite. Please note that if you wish to include comments you must add them directly to your `great_expectations.yml`.

```
In [5]: sanitize_yaml_and_save_datasource(context, example_yaml, overwrite_existing=False)
        context.list_datasources()
```

```
**WARNING** A Datasource named "my_pandas_datasource" already exists in this Data Con
text. The Datasource has *not* been saved. Please use a different name or set overwri
te_existing=True if you want to overwrite!
Out[5]: [{'class_name': 'Datasource',
  'data_connectors': {'default_inferred_data_connector_name': {'class_name': 'Inferre
dAssetFilesystemDataConnector',
    'default_regex': {'group_names': ['data_asset_name'], 'pattern': '(.*)'},
    'base_directory': '..\\data',
    'module_name': 'great_expectations.datasource.data_connector'},
   'default_runtime_data_connector_name': {'class_name': 'RuntimeDataConnector',
    'assets': {'my_runtime_asset_name': {'class_name': 'Asset',
      'batch_identifiers': ['runtime_batch_identifier_name'],
      'module_name': 'great_expectations.datasource.data_connector.asset'}},
    'module_name': 'great_expectations.datasource.data_connector'}},
  'execution_engine': {'class_name': 'PandasExecutionEngine',
   'module_name': 'great_expectations.execution_engine'},
  'module_name': 'great_expectations.datasource',
  'name': 'my_pandas_datasource'}]
```

Now you can close this notebook and delete it!