Create a new sqlalchemy Datasource

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

In [ ]:
```
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 [ ]:
```
datasource_name = "my_datasource"
```

For SQL based Datasources:

Here we are creating an example configuration based on the database backend you specified in the CLI. The configuration contains an **InferredAssetSqlDataConnector**, which will add a Data Asset for each table in the database, and a **RuntimeDataConnector** which can accept SQL queries. 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, please see our docs on InferredAssetDataConnectors and RuntimeDataConnectors.

Credentials will not be saved until you run the last cell. The credentials will be saved in `uncommitted/config_variables.yml` which should not be added to source control.

In [ ]:
```
host = "YOUR_HOST"  # The account name (include region -- ex 'ABCD.us-east-1')
username = "YOUR_USERNAME"
database = ""  # The database name
schema = ""  # The schema name
warehouse = ""  # The warehouse name
role = ""  # The role name
authenticator_url = "externalbrowser"  # A valid okta URL or 'externalbrowser' used to
```

In [ ]:
```
example_yaml = f"""
name: {datasource_name}
class_name: Datasource
execution_engine:
  class_name: SqlAlchemyExecutionEngine
  credentials:
```

```
    host: {host}
    username: {username}
    database: {database}
    query:
      schema: {schema}
      warehouse: {warehouse}
      role: {role}
    connect_args:
      authenticator: {authenticator_url}
    drivername: snowflake
data_connectors:
  default_runtime_data_connector_name:
    class_name: RuntimeDataConnector
    batch_identifiers:
      - default_identifier_name
  default_inferred_data_connector_name:
    class_name: InferredAssetSqlDataConnector
    include_schema_name: True"""
print(example_yaml)
```

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

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

Now you can close this notebook and delete it!