**Lab Document: Great Expectations CLI & Enterprise Integration**

This lab is designed to help you **build practical experience** with Great Expectations (GE) by exploring key capabilities including CLI usage, rule creation, scheduling checkpoints, webhook alerts, lineage mapping, and Azure Purview integration.

**Lab Prerequisites**

* Python 3.8 or higher
* pip and virtualenv installed
* Access to a dataset (CSV or DB)
* Optional: Slack webhook, Azure account with Purview enabled

**Lab 1: Environment Setup**

**Step 1.1 – Create a Virtual Environment**

python -m venv ge\_lab\_env

source ge\_lab\_env/bin/activate # Linux/macOS

ge\_lab\_env\Scripts\activate # Windows

**Step 1.2 – Install Great Expectations**

pip install great\_expectations

**Step 1.3 – Initialize a GE Project**

great\_expectations init

* This creates the project structure:
  + great\_expectations/
  + expectations/, checkpoints/, plugins/

**Lab 2: Auto-Profile Inference**

**Step 2.1 – Load a Dataset**

Place your dataset (e.g., sales\_data.csv) in the data/ folder.

**Step 2.2 – Create a Datasource**

Run:

great\_expectations datasource new

Follow prompts for a Pandas/Spark/SQL datasource.

**Step 2.3 – Create Expectation Suite using Auto-Profile**

great\_expectations suite scaffold sales\_suite

* GE scans your dataset and auto-generates expectations.
* Edit or refine expectations/sales\_suite.json if needed.

**Step 2.4 – Build and View Data Docs**

great\_expectations docs build

View: great\_expectations/uncommitted/data\_docs/local\_site/index.html

**Lab 3: Create Custom Expectation Rules**

**Step 3.1 – Regex Expectation**

Add this in your suite or via notebook:

expect\_column\_values\_to\_match\_regex(column="email", regex="^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$")

**Step 3.2 – Range Expectation**

expect\_column\_values\_to\_be\_between("total\_amount", min\_value=0, max\_value=10000)

**Step 3.3 – Completeness Rule**

expect\_column\_values\_to\_not\_be\_null("customer\_id")

**Step 3.4 – Save and Rebuild Docs**

great\_expectations docs build

**Lab 4: Checkpoint Configuration and Scheduling**

**Step 4.1 – Create a Checkpoint**

great\_expectations checkpoint new sales\_checkpoint

Define validation suite and batch in the generated YAML.

**Step 4.2 – Run Checkpoint**

great\_expectations checkpoint run sales\_checkpoint

**Step 4.3 – Schedule with Cron (Linux Example)**

Edit crontab:

crontab -e

Add:

0 6 \* \* \* /path/to/venv/bin/great\_expectations checkpoint run sales\_checkpoint

**Lab 5: Webhook Alert Integration**

**Step 5.1 – Generate Slack Webhook (or Teams/PagerDuty)**

* Slack: Go to Slack App > Incoming Webhooks > Generate URL

**Step 5.2 – Configure Action List in Checkpoint YAML**

action\_list:

- name: send\_slack\_on\_validation

action: great\_expectations.action.SendSlackNotificationAction

slack\_webhook: https://hooks.slack.com/services/XXX/YYY/ZZZ

**Step 5.3 – Re-run Checkpoint and Observe Alerts**

great\_expectations checkpoint run sales\_checkpoint

**Lab 6: Source→Target Mapping**

**Step 6.1 – Simulate ETL Layers**

* Use sample files for bronze\_sales.csv and silver\_sales.csv
* Create expectation suites for each file

great\_expectations suite scaffold bronze\_sales\_suite

great\_expectations suite scaffold silver\_sales\_suite

**Step 6.2 – Validate Field Mapping**

* Compare key metrics across layers
* Use expectations like:
  + Row count consistency
  + Key field existence
  + Value integrity (transformation logic checks)

**Step 6.3 – Document Layer Tags**

expect\_table\_row\_count\_to\_equal(10000).meta = {"layer": "silver"}

**Lab 7: Azure Purview Lineage Registration (Advanced)**

**Step 7.1 – Setup Azure Purview**

* Create a Purview account in Azure Portal
* Register your data source (e.g., Azure Blob, SQL DB)

**Step 7.2 – Use REST API to Register Metadata**

Install necessary libraries:

pip install azure-identity azure-purview-catalog

Use script (simplified):

from azure.identity import DefaultAzureCredential

from azure.purview.catalog import PurviewCatalogClient

client = PurviewCatalogClient(endpoint="https://<your-purview-name>.purview.azure.com", credential=DefaultAzureCredential())

asset\_payload = {

"name": "sales\_data",

"typeName": "azure\_datalake\_gen2\_resource\_set",

"attributes": {"qualifiedName": "...", "description": "Validated with GE"}

}

client.entity.create\_or\_update(entity=asset\_payload)

**Step 7.3 – Link Expectation Results**

* Embed GE suite info in asset attributes
* Optionally register checkpoint outcomes using Atlas lineage model

**Lab Wrap-Up: Review & Discussion**

* Validate and document each lab output
* Discuss integration opportunities in real pipelines
* Identify areas to expand expectations or improve alerting

**Deliverables**

* Expectation Suites for source/target datasets
* YAML checkpoint config
* Webhook setup confirmation
* (Optional) Purview registration script

**Additional Resources**

* [Great Expectations Docs](https://docs.greatexpectations.io/)
* [GE CLI Reference](https://docs.greatexpectations.io/docs/cli/)
* [Azure Purview Overview](https://learn.microsoft.com/en-us/azure/purview/)