**Microsoft Purview Governance & Automation – Hands-on Lab Guide**

**Lab 1: Setting up Collection Hierarchy & RBAC**

**Objective**: Create hierarchical collections and assign RBAC roles.  
**Governance Portal** → **Collections**

**Steps:**

1. Go to https://purview.microsoft.com and log in.
2. Click **“Manage Collections”** (left nav).
3. Click **“+ New Collection”**, enter:
   * **Name**: Finance-Data
   * **Parent Collection**: Optional (select if needed)
4. After creation, click on the collection → **Access Control** tab.
5. Click **“Add role assignments”** and assign roles:
   * Collection Admin – full access
   * Data Curator – metadata enrichment
   * Data Reader – view-only

**Expected Outcome:**

A structured collection tree with user/group roles assigned.

**Lab 2: Configure and Run a Scan**

**Objective**: Scan metadata from ADLS Gen2 or Azure SQL.  
**Data Map** → **Sources**

**Steps:**

1. Navigate to **Data Sources** > **+ Register Source**.
2. Choose source type (e.g., Azure Data Lake Gen2).
3. Input name, subscription, and storage account.
4. Choose authentication method:
   * **Managed Identity** or **Key Vault + SPN**
5. Post-registration, click **+ New Scan**:
   * Choose or create scan rule set
   * Set **scope**, **frequency**, and **filters**
6. Run the scan.
7. Monitor under **Data Map → Scans → Scan History**

**Expected Outcome:**

Assets and metadata from the source are ingested and visible.

**Lab 3: Create Custom Classification Rule**

**Objective**: Detect sensitive patterns using regex.  
**New UI Navigation**: **Data Map** → **Classifications** → **Classification Rules**

**Steps:**

1. Navigate to **Classification Rules** → **+ New Rule**.
2. Name: PAN-Classifier
3. Pattern Type: **Custom regex**
   * Regex: [A-Z]{5}[0-9]{4}[A-Z]
4. Confidence: Medium or High
5. Scope: Files or columns (apply at scan rule set level)
6. Save and link to an existing **scan rule set** (via Scan Rules tab)

**Sample Data:**

Name, PAN, Salary

Ravi Kumar, AJCPK1234M, 75000

Seema Rao, ABCDE5678Z, 62000

Invalid, 1234567890, 50000

**Expected Outcome:**

Records matching the PAN regex are flagged and tagged.

**Lab 4: Define and Assign Glossary Terms**

**Objective**: Link business definitions to technical assets.  
**Data Catalog** → **Glossary**

**Steps:**

1. Go to **Glossary** → **+ New Term**.
2. Enter term:
   * Name: Customer
   * Description: "A person or entity that purchases a product or service"
   * Synonyms: "Client", "Buyer"
3. Save the term.
4. Go to **Data Catalog** → Select an asset → **Manage Terms**
5. Tag columns like CustomerID, CustomerName with the glossary term.

**Expected Outcome:**

Data assets enriched with business metadata for improved discovery.

**Lab 5: Push Lineage from Databricks to Purview**

**Objective**: Capture process-level lineage.  
**Lineage** (Post scan or via API)

**Steps:**

1. Set up Databricks-to-Purview connection:
   * Register App in AAD with Purview Data Curator and Data Source Admin roles.
   * Use that identity in Databricks (via Service Principal or Managed Identity)
2. Use Databricks to process and capture lineage via REST API:
   * Use the /lineage/processes API to register:
     + Source: raw.customers
     + Process: notebook/job
     + Target: cleansed.customers\_gold
3. After sending, verify under **Data Map** → **Lineage View**

**Sample Python for Lineage (via REST API):**

# Python pseudocode for posting lineage

import requests

token = get\_azure\_token()

payload = {

"name": "Customer Cleansing",

"inputs": ["raw/customers"],

"outputs": ["gold/customers"],

...

}

requests.put(

"https://<purview-name>.purview.azure.com/lineage/processes",

headers={"Authorization": f"Bearer {token}"},

json=payload

)

**Expected Outcome:**

Lineage graphs show source-to-target with process in the middle.

**Lab 6: Monitor Governance Scorecard**

**Objective**: Visualize governance metrics.  
**Insights** → **Governance Scorecard**

**Steps:**

1. Navigate to **Insights** > **Scorecard**
2. Review metrics:
   * **Asset coverage**
   * **Classification coverage**
   * **Glossary linkage**
3. Filter by collection, asset type, or time

**Expected Outcome:**

Highlights scan progress, coverage gaps, and enrichment levels.

**Lab 7: Automate Scan via DevOps Pipeline**

**Objective**: Deploy scan configurations using CI/CD.  
Uses REST APIs or ARM Templates

**Steps:**

1. Define scan parameters in a JSON ARM template or PowerShell script.
2. Store script/template in a DevOps repo.
3. Create a DevOps pipeline with a task to call:
   * az rest (for REST API)
   * or ARM deploy task
4. Use a Service Principal with Purview contributor roles.
5. Run and validate under **Scan History**

**Example YAML Task:**

- task: AzureCLI@2

inputs:

azureSubscription: '<your-sp-connection>'

scriptType: bash

scriptLocation: inlineScript

inlineScript: |

az rest --method post \

--uri "https://<your-purview>.purview.azure.com/scans/..." \

--body @scanconfig.json \

--headers "Content-Type=application/json"

**Expected Outcome:**

Scan is initiated through DevOps automation and logs show success.