**Databricks Basics with ADLS & DataFrames**

**Lab 1: Create Workspace & Cluster**

**Objective:** Provision a Databricks workspace and create an all-purpose cluster for development.

**Steps:**

1. Sign in to the **Azure Portal**.
2. Search for **Azure Databricks** and click **+ Create**.
3. Provide details:
   * **Workspace name**: adb-lab-ws
   * **Resource group**: Create/select one
   * **Region**: Choose same region as ADLS
   * **Pricing tier**: Standard or Premium
4. Click **Review + Create → Create**.
5. Once deployed, go to the resource and click **Launch Workspace**.
6. Inside Databricks, go to **Clusters → Create Cluster**.
   * Name: lab-cluster
   * Cluster Mode: Standard
   * Databricks Runtime: Latest stable (e.g., 13.x)
   * Worker Type: Standard\_DS3\_v2 (example)
   * Autoscaling: Enabled
7. Click **Create Cluster**. Wait for cluster to be running.

**Lab 2: Mount ADLS**

**Objective:** Mount an Azure Data Lake Storage (ADLS Gen2) container to DBFS.

**Steps:**

1. Ensure you have:
   * Storage account name
   * Container name (e.g., raw)
   * Service Principal (Client ID, Secret, Tenant ID)
2. In Databricks **Notebook (Python)**, run:

configs = {

"fs.azure.account.auth.type": "OAuth",

"fs.azure.account.oauth.provider.type": "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",

"fs.azure.account.oauth2.client.id": "<client-id>",

"fs.azure.account.oauth2.client.secret": "<client-secret>",

"fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/<tenant-id>/oauth2/token"

}

dbutils.fs.mount(

source = "abfss://raw@<storage-account>.dfs.core.windows.net/",

mount\_point = "/mnt/raw",

extra\_configs = configs

)

1. Verify:

display(dbutils.fs.ls("/mnt/raw"))

**Lab 3: Notebook Basics**

**Objective:** Work with Python & SQL notebooks.

**Steps:**

1. Go to **Workspace → Create → Notebook**.
   * Name: lab-python
   * Language: Python
   * Cluster: Attach lab-cluster
2. Run a Python cell:

print("Hello Databricks!")

1. Create another notebook:
   * Name: lab-sql
   * Language: SQL
2. Run a SQL query on sample database:

SHOW DATABASES;

**Lab 4: Data Read/Write**

**Objective:** Load CSV & Parquet data from ADLS, process, and write back.

**Steps:**

1. Upload employee.csv and department.parquet to /mnt/raw.
2. In Python notebook:

# Read CSV

emp\_df = spark.read.option("header", True).csv("/mnt/raw/employee.csv")

# Read Parquet

dept\_df = spark.read.parquet("/mnt/raw/department.parquet")

# Show schema

emp\_df.printSchema()

dept\_df.printSchema()

1. Write processed data back in Parquet format:

emp\_df.write.mode("overwrite").parquet("/mnt/raw/processed/employees")

**Lab 5: DataFrame Transformations**

**Objective:** Perform filtering, selection, and aggregation on datasets.

**Steps:**

1. Filter employees with salary > 50,000:

high\_salary = emp\_df.filter(emp\_df.salary > 50000)

display(high\_salary)

1. Select employee name and department ID:

emp\_selected = emp\_df.select("emp\_name", "dept\_id")

display(emp\_selected)

1. Aggregate average salary per department:

avg\_salary = emp\_df.groupBy("dept\_id").avg("salary")

display(avg\_salary)

1. Join employees with departments:

joined = emp\_df.join(dept\_df, emp\_df.dept\_id == dept\_df.dept\_id, "inner")

display(joined)