**Introduction to Databricks & Lakehouse**

**Lab 1: Lakehouse Concept Exploration**

**Objective:** Understand the Lakehouse architecture.  
**Steps:**

1. Open a Databricks workspace.
2. Create a new **Markdown cell** in a notebook and document:
   * What is a Data Lake
   * What is a Data Warehouse
   * How Lakehouse combines both.
3. Draw or import a diagram showing:
   * Raw Zone (Data Lake)
   * Query/Analytics Layer (Warehouse)
   * Unified Lakehouse with Delta

**Lab 2: Databricks Workspace Components**

**Objective:** Explore Repos, Jobs, Clusters, and Data tabs.  
**Steps:**

1. Go to **Workspace → Repos** → Connect GitHub repo.
2. Open **Jobs → Create Job** to schedule a simple notebook.
3. Go to **Clusters** and review existing clusters.
4. Navigate to **Data** → Explore sample datasets provided by Databricks.

**Lab 3: Cluster Types**

**Objective:** Create different cluster types.  
**Steps:**

1. Create an **All-purpose cluster** named dev-cluster.
2. Create a **Job cluster** by defining it in a scheduled job.
3. Enable **autoscaling** by checking the option while creating dev-cluster.
4. Observe the difference in usage:
   * All-purpose stays alive until terminated.
   * Job clusters spin up only when jobs run.

**Lab 4: Databricks Notebooks Basics**

**Objective:** Run Python, SQL, and Scala in Databricks notebooks.  
**Steps:**

1. Create a **Python notebook** → Run:
2. print("Hello Databricks - Python")
3. Create a **SQL notebook** → Run:
4. SHOW DATABASES;
5. In the same notebook, use magic command for Scala:
6. %scala
7. println("Hello Databricks - Scala")

**Lab 5: Working with DBFS**

**Objective:** Interact with Databricks File System.  
**Steps:**

1. List files:
2. display(dbutils.fs.ls("/databricks-datasets"))
3. Create a new directory:
4. dbutils.fs.mkdirs("/mnt/demo")
5. Write a test file:
6. dbutils.fs.put("/mnt/demo/test.txt", "Hello DBFS!", True)

**Lab 6: Mount ADLS**

**Objective:** Mount ADLS Gen2 to DBFS.  
**Steps:**  
Use the same mount code as in the earlier labs:

dbutils.fs.mount(

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

mount\_point = "/mnt/raw",

extra\_configs = configs

)

**Lab 7: Reading/Writing Data**

**Objective:** Work with CSV, Parquet, and JSON.  
**Steps:**

1. Read CSV:
2. df\_csv = spark.read.csv("/mnt/raw/sample.csv", header=True, inferSchema=True)
3. Read Parquet:
4. df\_parquet = spark.read.parquet("/mnt/raw/sample.parquet")
5. Read JSON:
6. df\_json = spark.read.json("/mnt/raw/sample.json")
7. Write back as Parquet:
8. df\_csv.write.mode("overwrite").parquet("/mnt/raw/output/data")

**Lab 8: Spark DataFrames & Transformations**

**Objective:** Perform basic transformations.  
**Steps:**

# Select columns

df\_csv.select("Name", "Salary").show()

# Filter

df\_csv.filter(df\_csv.Salary > 60000).show()

# Aggregate

df\_csv.groupBy("Department").avg("Salary").show()

**Lab 9: Medallion Architecture**

**Objective:** Implement Bronze, Silver, and Gold layers.  
**Steps:**

1. **Bronze (Raw ingestion):**
2. df\_bronze = spark.read.csv("/mnt/raw/employee.csv", header=True, inferSchema=True)
3. df\_bronze.write.format("delta").mode("overwrite").save("/mnt/bronze/employee")
4. **Silver (Cleaned data):**
5. df\_silver = df\_bronze.dropna()
6. df\_silver.write.format("delta").mode("overwrite").save("/mnt/silver/employee")
7. **Gold (Aggregated data):**
8. df\_gold = df\_silver.groupBy("Department").avg("Salary")
9. df\_gold.write.format("delta").mode("overwrite").save("/mnt/gold/employee\_salary")