**Lab 1: Mapping Data Flow – Join and Aggregate**

**Objective**

Create a mapping data flow to join two datasets and perform aggregations.

**Steps**

1. **Open ADF Studio**
   * Go to the Azure Portal → Find your Data Factory → Click **Open Studio**.
   * Navigate to the **Author hub**.
2. **Create Mapping Data Flow**
   * Click **+ (plus)** → **Data Flow**.
   * Select **Mapping Data Flow** → Name it EmployeeDeptAggregation.
3. **Add Source Transformations**
   * Click **Add Source** → select Employee\_CSV\_Dataset.
     + Schema: EmployeeID, DepartmentID, Salary.
   * Add another source → select Department\_JSON\_Dataset.
     + Schema: DepartmentID, DepartmentName.
4. **Join Transformation**
   * Add **Join** transformation.
   * Left input: Employee.
   * Right input: Department.
   * Join condition: Employee.DepartmentID = Department.DepartmentID.
   * Join type: **Inner Join**.
5. **Aggregate Transformation**
   * Add **Aggregate** transformation.
   * Group by: DepartmentName.
   * Aggregates:
     + AvgSalary = avg(Salary).
     + EmpCount = count(EmployeeID).
6. **Sink Transformation**
   * Add **Sink** → Azure SQL Database dataset → Table: DeptSalarySummary.
   * Set **Write Behavior = Truncate table**.
7. **Debug & Validate**
   * Turn on **Data Flow Debug** (takes ~5 mins to warm up).
   * Preview results → Check that each department has AvgSalary and EmpCount.
8. **Publish & Execute**
   * Save, Publish All.
   * Create pipeline with this data flow, trigger it, and check DeptSalarySummary in SQL DB:
   * SELECT \* FROM DeptSalarySummary;

**Lab 2: Parameterized Pipeline – Dynamic File Names**

**Objective**

Pass file names dynamically to datasets for reusability.

**Steps**

1. **Create Pipeline**
   * In Author hub → Pipelines → **New pipeline**.
   * Name it ParameterizedFilePipeline.
2. **Add Pipeline Parameter**
   * In pipeline → Parameters tab → Add:
     + Name: FileName
     + Type: String
3. **Parameterize Dataset**
   * Open Employee\_CSV\_Dataset.
   * Add a **dataset parameter**:
     + Name: FileNameParam.
   * File path → For file name, set @dataset().FileNameParam.
4. **Map Dataset Parameter in Pipeline**
   * Add **Copy Data** activity.
   * Source dataset = Employee\_CSV\_Dataset.
   * Under **Parameters**, set FileNameParam = @pipeline().parameters.FileName.
5. **Test Run**
   * Trigger pipeline manually → In **Parameters**, enter:
     + employee\_data\_2023.csv
   * Run pipeline.
   * Validate in sink location that the correct file is processed.

**Lab 3: Control Flow – ForEach Loop for Multiple Files**

**Objective**

Process multiple files in a folder using ForEach.

**Steps**

1. **Get Metadata Activity**
   * Add **Get Metadata** activity.
   * Source: ADLS/Blob folder containing multiple CSVs.
   * Field list: **Child Items**.
2. **ForEach Activity**
   * Add **ForEach** activity → Connect after Get Metadata.
   * Items: @activity('Get Metadata1').output.childItems.
3. **Inside ForEach**
   * Add **Copy Data** activity.
   * Source dataset = parameterized CSV dataset (Employee\_CSV\_Dataset).
   * Map dataset parameter: FileNameParam = @item().name.
4. **Publish & Test**
   * Save & Publish.
   * Trigger pipeline.
   * Monitor → Each file in folder processed individually.

**Lab 4: Incremental Load – Watermark Strategy**

**Objective**

Implement incremental loading from SQL to ADLS using a watermark column.

**Steps**

1. **Check SQL Source Table**
2. SELECT TOP 10 \* FROM Employee ORDER BY LastModifiedDate DESC;

Ensure LastModifiedDate exists.

1. **Create Pipeline Parameter**
   * Name: WatermarkValue (Type: String or DateTime).
2. **Parameterize SQL Dataset Query**
   * In SQL dataset → Use query:
   * SELECT \*
   * FROM Employee
   * WHERE LastModifiedDate > '@{pipeline().parameters.WatermarkValue}'
3. **Copy Activity**
   * Source: SQL dataset with query filter.
   * Sink: ADLS → Folder incremental\_load/.
4. **Store Latest Watermark**
   * Add **Lookup Activity**:
   * SELECT MAX(LastModifiedDate) as LatestDate FROM Employee;
   * Store in pipeline variable → Use for next run.
5. **Test Incremental Load**
   * Run with watermark = yesterday’s date.
   * Verify only changed records get loaded.

**Lab 5: Git Integration – CI/CD Setup**

**Objective**

Connect ADF to Git repository for version control.

**Steps**

1. **Open Git Config**
   * In ADF Studio → Manage hub → **Git Configuration**.
2. **Configure Git**
   * Select **Azure DevOps Git** or **GitHub**.
   * Provide:
     + Repository URL
     + Collaboration branch (e.g., main)
     + Root folder path (e.g., /adf)
3. **Save Configuration**
4. **Commit Changes**
   * Go to Author hub.
   * Make a pipeline change (add a dummy activity).
   * Click **Save All** → Provide commit message → **Commit**.
5. **Verify in Repo**
   * Open Azure DevOps/GitHub repo → Confirm ADF JSON files are pushed.