**Labs for Synapse, Fabric & Airflow**

**Lab 1: Overview of Synapse Components (SQL Pools, Spark Pools, Pipelines)**

**Objective:** Get familiar with Synapse Studio and its components.  
**Steps:**

1. Go to **Azure Portal → Create Resource → Synapse Analytics**.
2. Provision a **Synapse Workspace** with Managed VNet and ADLS Gen2 as the default storage.
3. Navigate to **Synapse Studio**.
4. Explore:
   * **Data Hub** (linked data lakes, SQL DBs).
   * **Develop Hub** (Notebooks, SQL scripts, Data Flows).
   * **Integrate Hub** (pipelines).
   * **Monitor Hub** (pipeline runs).
5. Run a sample SQL query in **SQL on-demand pool** (serverless).

**Lab 2: Dedicated vs Serverless SQL Pools**

**Objective:** Compare performance and usage of both pool types.  
**Steps:**

1. In Synapse Studio, open **SQL scripts**.
2. For **Serverless SQL Pool**:
   * Query Parquet data in ADLS without importing:
   * SELECT TOP 10 \*
   * FROM OPENROWSET(
   * BULK 'https://<storage>.dfs.core.windows.net/sampledata/nyctaxi/\*.parquet',
   * FORMAT = 'PARQUET'
   * ) AS [result];
3. For **Dedicated SQL Pool**:
   * Create a table:
   * CREATE TABLE TaxiData (
   * trip\_id INT,
   * passenger\_count INT,
   * fare\_amount FLOAT
   * ) WITH (DISTRIBUTION = ROUND\_ROBIN, CLUSTERED COLUMNSTORE INDEX);
   * Load data using **COPY INTO** from ADLS.
   * Run aggregation queries and compare performance with serverless.

**Lab 3: Creating and Managing Databases, Tables, and Views**

**Objective:** Learn schema management in Synapse SQL.  
**Steps:**

1. Create a new database:
2. CREATE DATABASE SalesDB;
3. Create a fact table and a dimension table.
4. Load sample CSV data from ADLS.
5. Create a **view** that joins both tables.
6. CREATE VIEW vw\_SalesReport AS
7. SELECT f.SalesAmount, d.Region
8. FROM FactSales f
9. JOIN DimRegion d ON f.RegionId = d.Id;

**Dataset:** Sample Sales CSV (columns: RegionId, SalesAmount, Date).

**Lab 4: Introduction to Microsoft Fabric**

**Objective:** Explore Fabric workspace and data ecosystem.  
**Steps:**

1. Open **Microsoft Fabric portal (app.powerbi.com)**.
2. Create a **Fabric Workspace**.
3. Use **Lakehouse** to ingest a CSV file (Sales data).
4. Explore **OneLake integration** (data accessible across Fabric).
5. Build a quick Power BI report using ingested data.

**Dataset:** SalesData.csv (columns: Product, Region, Amount, Date).

**Lab 5: Partitioning & Distribution Strategies**

**Objective:** Optimize queries with partitioning.  
**Steps:**

1. In **Dedicated SQL Pool**, create a partitioned table:
2. CREATE TABLE SalesPartitioned (
3. SaleId INT,
4. SaleDate DATE,
5. Amount FLOAT
6. )
7. WITH
8. (
9. CLUSTERED COLUMNSTORE INDEX,
10. DISTRIBUTION = HASH (SaleId),
11. PARTITION (SaleDate RANGE RIGHT FOR VALUES ('2022-01-01', '2022-06-01', '2023-01-01'))
12. );
13. Load sales data.
14. Run queries filtered by date and compare performance vs non-partitioned table.

**Dataset:** Sales with Date column across 2 years.

**Lab 6: Snowflake Query Profiling & Query History**

**Objective:** Learn query optimization in Snowflake.  
**Steps:**

1. Create a free **Snowflake trial account**.
2. Load **NYC Taxi data** into Snowflake using COPY INTO.
3. Run sample queries.
4. Go to **Query History** and check execution plan, stages, and partitions scanned.
5. Compare performance of **SELECT \*** vs **projected columns only**.

**Dataset:** NYC Taxi Yellow Tripdata.

**Lab 7: Airflow Overview**

**Objective:** Build a simple Airflow DAG for ETL.  
**Steps:**

1. Set up Airflow using Docker or Astronomer.io.
2. Create a DAG (nyc\_taxi\_dag.py):
3. from airflow import DAG
4. from airflow.operators.python import PythonOperator
5. from datetime import datetime
6. def extract():
7. print("Extracting NYC taxi data...")
8. def transform():
9. print("Transforming data...")
10. def load():
11. print("Loading data into Synapse...")
12. with DAG('nyc\_taxi\_dag', start\_date=datetime(2023,1,1), schedule\_interval='@daily', catchup=False) as dag:
13. t1 = PythonOperator(task\_id='extract', python\_callable=extract)
14. t2 = PythonOperator(task\_id='transform', python\_callable=transform)
15. t3 = PythonOperator(task\_id='load', python\_callable=load)
16. t1 >> t2 >> t3
17. Run the DAG and monitor in Airflow UI.

**Dataset:** Use dummy NYC taxi data for pipeline simulation.