**Monitoring & Troubleshooting a Data Platform in Azure**

**Scenario**

You are part of the engineering team at **RetailX**, which runs an Azure-based data platform. The platform ingests **sales data** through **ADF pipelines**, processes it using **Databricks**, and exposes APIs monitored with **Application Insights**.

Business users report **slower data refresh** and **delayed dashboards** during peak hours. Your task is to:

1. Configure **logs, metrics, and alerts** in **Azure Monitor**.
2. Enable **Application Insights** for monitoring data services.
3. Diagnose performance issues in **ADF pipelines** and **Databricks jobs**.
4. Build a **troubleshooting playbook** for production issues.

**Dataset (Sample Sales Data)**

Create a CSV file named sales\_data.csv:

TransactionID,CustomerID,ProductID,Quantity,Price,TransactionDate,Region

T1001,C001,P101,2,150,2025-09-01,North

T1002,C002,P102,1,200,2025-09-01,East

T1003,C003,P101,5,150,2025-09-02,South

T1004,C004,P103,3,300,2025-09-02,West

T1005,C005,P104,10,50,2025-09-03,North

T1006,C001,P102,2,200,2025-09-03,East

T1007,C002,P105,1,400,2025-09-04,South

T1008,C003,P101,8,150,2025-09-04,West

T1009,C004,P104,15,50,2025-09-05,North

T1010,C005,P103,2,300,2025-09-05,East

* **File size**: ~1000+ rows (for realism, replicate rows with slight variations in Quantity/Price/Date/Region).
* **Storage location**: Upload to **Azure Blob Storage** (container: salesraw).

**Lab Tasks**

**Task 1: Logs, Metrics, and Alerts in Azure Monitor**

1. Enable **Diagnostic Settings** on:
   * **ADF** → send logs to **Log Analytics Workspace**.
   * **Databricks** → send cluster logs to **Log Analytics**.
2. Create **Alerts**:
   * ADF pipeline failure count > 3 in 30 mins.
   * Databricks cluster memory utilization > 80% for 10 mins.

**Checkpoint:** Verify alert triggers when you run a failing ADF pipeline (simulate by pointing to a wrong storage path).

**Task 2: Application Insights for Data Services**

1. Enable **Application Insights** on a simple API (mock API or Azure Function that returns sales summary).
2. Monitor:
   * Request/response times.
   * SQL dependency (if API queries a DB).
3. Open **Application Map** → identify dependencies.

**Checkpoint:** Trigger a slow query (use SELECT \* FROM sales\_data without index) and observe response time in App Insights.

**Task 3: Diagnosing Performance Issues in ADF & Databricks**

1. **ADF Pipeline**:
   * Build pipeline: Copy sales data → Data Lake → SQL DB.
   * Run pipeline with **low DIUs** (2) → note execution time.
   * Increase to **8 DIUs** → compare performance.
2. **Databricks Notebook**:

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("SalesProcessing").getOrCreate()

df = spark.read.csv("path-to-salesraw/sales\_data.csv", header=True, inferSchema=True)

# Intentional skew (bad join)

product\_ref = df.select("ProductID").distinct()

joined = df.join(product\_ref, on="ProductID", how="inner")

print("Row count:", joined.count())

* + Monitor Spark UI → see skew in shuffle stages.
  + Fix skew → joined = df.join(product\_ref.hint("broadcast"), on="ProductID", how="inner").

**Checkpoint:** Compare runtime before vs after optimization.

**Task 4: Troubleshooting Playbook**

Create a **Playbook Document** with these sections:

1. **Detection** → Alerts in Azure Monitor, App Insights anomalies.
2. **Diagnosis** → Log Analytics queries, Spark UI, pipeline run history.
3. **Resolution** → Scale DIUs, repartition data, optimize SQL queries.
4. **RCA** → Identify root causes (bad joins, missing indexes, low resources).
5. **Prevention** → Auto-scaling clusters, partition strategy, staged copy.