## JOB Runtime SLA Breach Databricks

### Scenario:

## Use Case: Job Runtime Logging & SLA Breach Dashboarding

Track and log job start time, end time, duration, and SLA status → store in Delta table → visualize in Power BI/Looker.

### ****Step 1: Import libraries and Define Parameter****

### from datetime import datetime, timezone from pyspark.sql.functions import current\_timestamp, input\_file\_name, lit

# Define SLA (expected runtime in minutes)

JOB\_SLA\_MINUTES = 30 # Customize per job , expected duration of SLA Compliance

### ****Step 2: Log Start Time at Job Begin****

# Databricks job parameters (passed as widgets)

job\_id = dbutils.widgets.get("job\_id") #**identifier of the specific ingestion/transformation job.**

run\_id = dbutils.widgets.get("run\_id") **#unique identifier for this execution (timestamp, UUID)**

**# Source data path**

**SOURCE\_PATH = "/mnt/raw/customers/"**

Capture Start Time

start\_time = datetime.now(timezone.utc)

### ****Step 3: Ingest Data into Bronze Table for Tracebility****

### df = spark.read.option("header", True).csv(SOURCE\_PATH)

### # Read raw customer data

### df = spark.read.option("header", True).csv(SOURCE\_PATH)

### # Add metadata columns (traceability)

### df = df.withColumn("ingested\_at", current\_timestamp()) \

### .withColumn("source\_file", input\_file\_name()) \

### .withColumn("run\_id", lit(run\_id)) # Enables linking with log

### # Write to Bronze table (append mode)

### df.write.format("delta") \

### .mode("append") \

### .saveAsTable("bronze.customers\_raw")

### # Capture row count for monitoring

### row\_count = df.count() # Optional: useful for tracking volume

### ****Step 4: Capture End Time and Calculate Duration****

end\_time = datetime.now(timezone.utc)

duration\_minutes = (end\_time - start\_time).total\_seconds() / 60

### ****Step 5: SLA Evaluation + Logging to Delta Table****

from pyspark.sql import SparkSession

from pyspark.sql.functions import lit

spark = SparkSession.builder.getOrCreate()

log\_data = [{

"job\_id": job\_id,

"run\_id": run\_id,

"start\_time": start\_time,

"end\_time": end\_time,

"duration\_minutes": round(duration\_minutes, 2),

"sla\_minutes": JOB\_SLA\_MINUTES,

"breached": duration\_minutes > JOB\_SLA\_MINUTES,

"logged\_at": datetime.now(timezone.utc),

"source\_path": SOURCE\_PATH,

"row\_count": row\_count

}]

log\_df = spark.createDataFrame(log\_data)

# Write to Monitoring Delta Table

log\_df.write.mode("append").format("delta").saveAsTable("monitoring.job\_runtime\_log")

Create Monitoring Table (DDL) # Run this once before Write Delta Table

CREATE OR REPLACE TABLE monitoring.job\_runtime\_log (

job\_id STRING,

run\_id STRING,

start\_time TIMESTAMP,

end\_time TIMESTAMP,

duration\_minutes DOUBLE,

sla\_minutes INT,

breached BOOLEAN,

logged\_at TIMESTAMP,

source\_path STRING,

row\_count LONG

) USING DELTA;

### ****Step 6: Create Power BI/Looker Dashboard (Summary)****

### SELECT

### b.customer\_id,

### b.ingested\_at,

### b.source\_file,

### m.duration\_minutes,

### m.breached,

### m.row\_count

### FROM bronze.customers\_raw b

### JOIN monitoring.job\_runtime\_log m

### ON b.run\_id = m.run\_id

### WHERE m.breached = TRUE

### ORDER BY m.start\_time DESC

**Connect Power BI or Looker to Delta Lake** via Databricks SQL endpoint or connector.

**Suggested Columns to Use in Dashboard**:

* job\_id, run\_id
* start\_time, end\_time
* duration\_minutes
* breached (Boolean → SLA violated)
* Add visualizations:
  + Bar chart: Avg duration vs SLA
  + Filter: breached == True
  + Trend line: Daily average runtime

## Summary

|  |  |  |
| --- | --- | --- |
| **Layer** | **Table** | **Purpose** |
| Bronze | bronze.customers\_raw | Ingested data with run\_id |
| Monitoring | monitoring.job\_runtime\_log | SLA metrics and performance logging |
| Dashboard | SQL Join (Step #6) | Trace and analyze job behavior |

### Optional: Email Alert / Slack on SLA Breach

You can trigger an alert with dbutils.notebook.exit() and pass breach info to orchestration (e.g., Azure Data Factory, Airflow, or Alert API).