## 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 from pyspark.sql import SparkSession

### # SLA thresholds per layer

### SLA\_BRONZE\_MIN = 30

### SLA\_SILVER\_MIN = 20

### SLA\_GOLD\_MIN = 10

### # Widgets passed by Databricks Job

### job\_id\_bronze = "bronze\_ingestion"

### job\_id\_silver = "customer\_cleaning"

### job\_id\_gold = "customer\_aggregation"

### run\_id = dbutils.widgets.get("run\_id") # e.g., "20250708\_001"

### # Source path for raw data

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

### Step 2: Bronze Layer – Raw Ingestion

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

### # Read raw CSV data

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

### # Add metadata

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

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

### .withColumn("run\_id", lit(run\_id))

### # Write to Bronze table

### bronze\_df.write.format("delta")

### .mode("append")

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

### row\_count\_bronze = bronze\_df.count()

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

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

### # Log to monitoring

### bronze\_log = [{

### "job\_id": job\_id\_bronze,

### "run\_id": run\_id,

### "start\_time": start\_time\_bronze,

### "end\_time": end\_time\_bronze,

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

### "sla\_minutes": SLA\_BRONZE\_MIN,

### "breached": duration\_bronze > SLA\_BRONZE\_MIN,

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

### "source\_path": SOURCE\_PATH,

### "row\_count": row\_count\_bronze

### }]

### spark.createDataFrame(bronze\_log)

### .write.mode("append").format("delta")

### .saveAsTable("monitoring.job\_runtime\_log")

### Step 3: Silver Layer – Data Cleaning

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

### # Read from Bronze for current run

### bronze\_df\_filtered = spark.read.table("bronze.customers\_raw")

### .filter(f"run\_id = '{run\_id}'")

### # Clean and transform

### silver\_df = bronze\_df\_filtered.dropDuplicates(["customer\_id"])

### .filter("status = 'active'") \

### .withColumnRenamed("cust\_id", "customer\_id")

### .withColumn("run\_id", lit(run\_id))

### # Write to Silver table

### silver\_df.write.format("delta")

### .mode("append")

### .saveAsTable("silver.customers\_clean")

### row\_count\_silver = silver\_df.count()

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

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

### # Log to monitoring

### silver\_log = [{

### "job\_id": job\_id\_silver,

### "run\_id": run\_id,

### "start\_time": start\_time\_silver,

### "end\_time": end\_time\_silver,

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

### "sla\_minutes": SLA\_SILVER\_MIN,

### "breached": duration\_silver > SLA\_SILVER\_MIN,

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

### "source\_path": "bronze.customers\_raw",

### "row\_count": row\_count\_silver

### }]

### spark.createDataFrame(silver\_log)

### .write.mode("append").format("delta")

### .saveAsTable("monitoring.job\_runtime\_log")

### Step 4: Gold Layer – Aggregation

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

### # Read from Silver

### silver\_df\_filtered = spark.read.table("silver.customers\_clean")

### .filter(f"run\_id = '{run\_id}'")

### # Aggregate: Count customers per region

### gold\_df = silver\_df\_filtered.groupBy("region")

### .count()

### .withColumn("run\_id", lit(run\_id))

### # Write to Gold table

### gold\_df.write.format("delta")

### .mode("append")

### .saveAsTable("gold.customer\_summary")

### row\_count\_gold = gold\_df.count()

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

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

### # Log to monitoring

### gold\_log = [{

### "job\_id": job\_id\_gold,

### "run\_id": run\_id,

### "start\_time": start\_time\_gold,

### "end\_time": end\_time\_gold,

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

### "sla\_minutes": SLA\_GOLD\_MIN,

### "breached": duration\_gold > SLA\_GOLD\_MIN,

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

### "source\_path": "silver.customers\_clean",

### "row\_count": row\_count\_gold

### }]

### spark.createDataFrame(gold\_log)

### .write.mode("append").format("delta")

### .saveAsTable("monitoring.job\_runtime\_log")

### Monitoring Table Schema (Run once)

### 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).