Databricks: Payroll Alert

Slack-Enabled Payroll File Health Alerts

Prepared By: Sri Sivakumar Ramar
Platform: Databricks
Type: Data Quality Monitoring & Alerting Pipeline
Use Case: Prevent payroll errors by detecting and alerting on empty timesheet files during ingestion

Business Context:
Payroll systems rely on daily/weekly timesheet files from HR or project systems. Sometimes files arrive empty (e.g., header-only, failed exports, or zero-byte uploads).
If not caught early, these files can result in:

- Underpayments or missed payroll
- Compliance issues
- Loss of trust in automation

Input Folder

Location: /mnt/raw/timesheet/landing/

Example File 1: Non-Empty File

Filename: timesheet_2025-07-09.csv

Contents (2 rows):

csv
CopyEdit
```
employee_id,date,hours_worked,project_code
E001,2025-07-09,8.0,PRJ001
E002,2025-07-09,7.5,PRJ001
```

Example File 2: Empty File

Filename: timesheet_2025-07-10.csv

Contents:

csv
CopyEdit
```
employee_id,date,hours_worked,project_code
```

This file only has the **header row**, and **no data rows**.

Auto Loader + Streaming Validation Code with Comments

```
python
CopyEdit

# Step 1: Import libraries

from pyspark.sql.functions import input_file_name, count, lit, current_timestamp
from pyspark.sql.types import StructType, StringType, DoubleType

# Step 2: Define schema

schema = StructType()
    .add("employee_id", StringType())
    .add("date", StringType())
    .add("hours_worked", DoubleType())
    .add("project_code", StringType())

# Step 3: Set up paths

landing_path = "/mnt/raw/timesheet/landing/"
temp_table_path = "/mnt/temp/timesheet_bronze/"
valid_output_path = "/mnt/bronze/payroll/valid/"
invalid_output_path = "/mnt/bronze/payroll/invalid/"
log_output_path = "/mnt/logs/payroll_validation/"
checkpoint_path = "/mnt/checkpoints/timesheet/"

# Step 4: Start streaming ingestion using Auto Loader

raw_stream_df = (
    spark.readStream
    .format("cloudFiles")                          # Auto Loader format
    .option("cloudFiles.format", "csv")            # File type
    .option("header", "true")                      # Header exists in files
    .schema(schema)
    .load(landing_path)                            # Watches all new files
    .withColumn("source_file", input_file_name())  # Adds source_file column like:
                                                   # /mnt/raw/timesheet/landing/timesheet_2025-07-09.csv
)

# Step 5: Write to temp Delta table

query = (
    raw_stream_df.writeStream
    .format("delta")
    .option("checkpointLocation", checkpoint_path)
    .outputMode("append")
    .start(temp_table_path)
)
```

Batch Job to Validate (Run every 15 min as notebook or Job)

```
python
CopyEdit
# Load streamed data as batch
df = spark.read.format("delta").load(temp_table_path)
```

Databricks: Payroll Alert

```
# Count rows per file
file_counts = df.groupBy("source_file").count()

# Classify files
valid_files = file_counts.filter("count > 0").select("source_file")
invalid_files = file_counts.filter("count = 0").select("source_file")

# Filter valid and invalid rows
valid_df = df.join(valid_files, on="source_file", how="inner")
invalid_df = df.join(invalid_files, on="source_file", how="inner")

# Write to appropriate folders
valid_df.write.format("delta").mode("append").save(valid_output_path)
invalid_df.write.format("delta").mode("append").save(invalid_output_path)

# Log empty file entries (optional)
if invalid_files.count() > 0:
    log_df = (
        invalid_files.withColumn("log_time", current_timestamp())
                     .withColumn("issue", lit("EMPTY_TIMESHEET_FILE"))
    )
    log_df.write.mode("append").format("delta").save(log_output_path)
```

Example Output

If the following files are ingested:

Filename	Rows	Routed to
timesheet_2025-07-09.csv	2	/bronze/payroll/valid/
timesheet_2025-07-10.csv	0	/bronze/payroll/invalid/ + logged

Set up an **alert (to Slack or Email)** in Databricks when **empty files are detected** during your validation process.
This typically involves two components:

High-Level Architecture

Detect empty file(s) in your batch validation logic.
Trigger alert:
Via a **webhook to Slack**
Or using **email via Databricks REST API** or cloud-native notifier (e.g., Azure Logic Apps, AWS SNS)

Option 1: Send Alert to Slack via Webhook

Step 1: Create a Slack Incoming Webhook

Go to your Slack Workspace → Apps → Search for **"Incoming Webhooks"**

Create a new webhook

Choose a channel (e.g., #data-pipeline-alerts)
Copy the webhook URL (e.g., https://hooks.slack.com/services/XXXX/YYYY/ZZZZ)

Step 2: Add Slack Alert to Validation Notebook

python
CopyEdit

```
import json
import requests

def send_slack_alert(empty_files_list):
    webhook_url = "https://hooks.slack.com/services/XXXX/YYYY/ZZZZ"
    message = {
        "text": f":warning: {len(empty_files_list)} empty timesheet file(s) detected:\n" +
                "\n".join(empty_files_list)
    }
    response = requests.post(webhook_url, data=json.dumps(message),
                             headers={'Content-Type': 'application/json'})

    if response.status_code != 200:
        raise Exception(f"Slack alert failed: {response.text}")
```

Call the function after detection

python
CopyEdit

```
empty_files = [row["source_file"] for row in invalid_files.collect()]
if empty_files:
    send_slack_alert(empty_files)
```

Option 2: Send Alert via Email

Databricks doesn't have native email sending, but you can:

Option A: Use smtplib (for basic use)

python
CopyEdit

```
import smtplib
from email.mime.text import MIMEText

def send_email_alert(empty_files_list):
    msg = MIMEText("The following timesheet files are empty:\n" + "\n".join(empty_files_list))
    msg["Subject"] = "Payroll Empty File Alert"
    msg["From"] = "datapipeline@example.com"
    msg["To"] = "payroll-team@example.com"

    with smtplib.SMTP("smtp.example.com", 587) as server:
        server.starttls()
        server.login("username", "password")
        server.sendmail(msg["From"], [msg["To"]], msg.as_string())
```

Requires your cloud firewall to allow SMTP outbound traffic and credentials.

Option B: Use Cloud Notification Services

Cloud	Service	Setup Example
Azure	**Logic Apps** or **SendGrid**	Trigger HTTP webhook from notebook
AWS	**SNS** + Lambda Email	Trigger via boto3 in Databricks
GCP	**Cloud Functions** + Mailgun	Trigger via requests.post()

Summary

Alert Method	When to Use	Notes
Slack Webhook	Lightweight, fast	Easy setup, team-friendly
Email via SMTP	Controlled alerts to inbox	Requires SMTP access
Cloud-native (SNS, Logic App)	Enterprise-grade	Scalable, centralized logging

☐ Methods and code provided above

Final Note:

- Each ingested row is tagged with its full source file path using input_file_name().
- Valid file paths are maintained separately and used to filter data via an inner join.
- Only rows from valid (non-empty) files are passed into the Bronze layer for further processing.

1. Shell:

- Use Case: Industrial IoT and sensor data.
- What They Do: Validate telemetry files from thousands of rigs, separating corrupt/empty files before landing them into Delta Lake Bronze.
- Tech Stack: Azure Data Lake, Databricks, Delta Lake, Power BI.