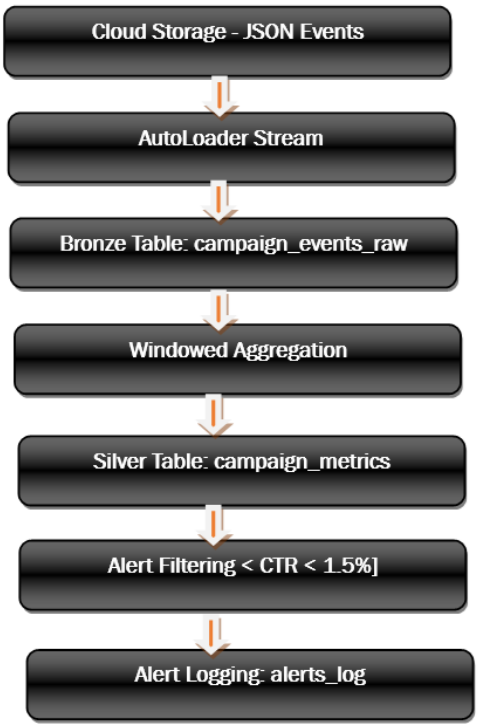
User Engagement Alerts on Campaign Performance

Use Case Summary

Real-time detection of campaign performance drop based on key user engagement metrics (CTR, impressions, conversions). Alerts are triggered when performance falls below defined thresholds.

Alerts are triggered at the *Silver Layer* — specifically after the aggregated campaign metrics (like CTR, impressions, conversions) are computed using a time window (e.g., 5 minutes).

Architecture Flow



Silver Layer

Layer	Location	What Happens
Silver Layer	silver.campaign_metrics Delta Table	Aggregated metrics are calculated per campaign and time window.
Threshold Logic	e.g., CTR < 1.5	A condition is evaluated on the computed metric columns (ctr, cvr, etc.).
Alert Trigger	If condition is met	Record is passed to alerting logic to log and notify downstream systems.

Output (per 5-min window per campaign)

window_start	window_end	campaign_id	impressions	clicks	conversions	ctr	cvr
2025-07-09 10:00	2025-07-09 10:05	cmp_101	100	12	3	12.0	25.0

---

1. Test Plan Overview

Test Case ID	Description	Input	Expected Output
TC001	Validate schema of campaign events	Raw stream	Valid structured schema
TC002	Detect performance drop	CTR below 1.5%	Alert flagged = True
TC003	Trigger alert logging	Performance breached	Entry in alerts_log table
TC004	Ensure streaming write to Delta	Valid batch	Data appended to campaign_metrics

---

2. Delta Table Design

a) campaign_events_raw (Bronze)

sql

CopyEdit

```
CREATE TABLE IF NOT EXISTS bronze.campaign_events_raw (
    campaign_id STRING,
    event_time TIMESTAMP,
    user_id STRING,
    event_type STRING, -- 'impression', 'click', 'conversion'
    channel STRING,
    region STRING
) USING DELTA;
```

b) campaign_metrics (Silver)

sql

CopyEdit

```
CREATE TABLE IF NOT EXISTS silver.campaign_metrics (
```

```
    campaign_id STRING,
    window_start TIMESTAMP,
    window_end TIMESTAMP,
    impressions LONG,
    clicks LONG,
    conversions LONG,
    ctr DOUBLE,
    cvr DOUBLE
) USING DELTA;
```

c) alerts_log (Monitoring)

```
sql

CopyEdit

CREATE TABLE IF NOT EXISTS monitoring.alerts_log (
    campaign_id STRING,
    alert_time TIMESTAMP,
    metric STRING,
    value DOUBLE,
    threshold DOUBLE,
    alert_type STRING
) USING DELTA;
```

---

3. Unit Test Cases (Pytest Style)

```
python

CopyEdit

def test_schema_validation():
    df = spark.read.json("path/to/sample_data.json")
    expected_columns = {"campaign_id", "event_time", "user_id", "event_type"}
    assert expected_columns.issubset(set(df.columns))

def test_ctr_below_threshold():
    from pyspark.sql import Row
    data = [Row(campaign_id="cmp1", clicks=10, impressions=200, conversions=3)]
    df = spark.createDataFrame(data)
    df = df.withColumn("ctr", (df.clicks / df.impressions) * 100)
    assert df.collect()[0]["ctr"] < 1.5
```

---

4. Streaming Ingestion & Aggregation (Bronze → Silver)

python

CopyEdit

```
from pyspark.sql.functions import col, window, count
raw_stream = (
   spark.readStream.format("cloudFiles")
   .option("cloudFiles.format", "json")
   .load("/mnt/campaign_events")
)
# Write to bronze table
raw_stream.writeStream.format("delta").outputMode("append") \
   .option("checkpointLocation", "/mnt/checkpoints/campaign_bronze") \
   .table("bronze.campaign_events_raw")
```

Aggregation Logic (Silver)

```
from pyspark.sql.functions import count, sum, expr
bronze_df = spark.readStream.table("bronze.campaign_events_raw")
agg_df = bronze_df.groupBy(
   window("event_time", "5 minutes"), col("campaign_id")
).agg(
            count(expr("event_type = 'impression'")).alias("impressions"),
            count(expr("event_type = 'click'")).alias("clicks"),
            count(expr("event_type = 'conversion'")).alias("conversions")
).withColumn(
            "ctr", (col("clicks") / col("impressions")) * 100
).withColumn(
            "cvr", (col("conversions") / col("clicks")) * 100
)
agg_df.selectExpr("campaign_id", "window.start as window_start", "window.end as window_end",
             "impressions", "clicks", "conversions", "ctr", "cvr")
                .writeStream.format("delta")
                .outputMode("append")
                .option("checkpointLocation", "/mnt/checkpoints/campaign_silver")
                .table("silver.campaign_metrics")
```

5. Alerting Logic

python

CopyEdit

```
from pyspark.sql.functions import current_timestamp, lit

threshold_ctr = 1.5

silver_df = spark.readStream.table("silver.campaign_metrics")

alerts_df = silver_df.filter(col("ctr") < threshold_ctr)

        .withColumn("alert_time", current_timestamp())
        .withColumn("metric", lit("CTR"))
        .withColumn("threshold", lit(threshold_ctr))
        .withColumn("alert_type", lit("performance_drop"))


.select("campaign_id", "alert_time", "metric", "ctr", "threshold", "alert_type")

alerts_df.writeStream.format("delta")
        .outputMode("append")
        .option("checkpointLocation", "/mnt/checkpoints/alerts_log")
        .table("monitoring.alerts_log")
```

Components Explained

Layer	Component	Purpose
Ingestion	Auto Loader	Real-time detection of new JSON data
Bronze	campaign_events_raw	Raw event logs (impression, click, conversion)
Silver	campaign_metrics	Aggregated campaign KPIs per window
Alerting	alerts_log	Logged alerts when performance drops
Consumption	Power BI, Email, Slack (Optional)	Notify marketing teams or display dashboard

---

Final Note:

This project demonstrates how to build a real-time, scalable alerting system in Databricks using the Delta Lake architecture (Bronze, Silver, Gold) and Structured Streaming. It empowers marketing teams to react quickly to campaign underperformance, improving agility and ROI.

Key Takeaways:

- Auto Loader simplifies real-time ingestion from cloud sources.
- Delta format ensures ACID compliance and time travel across layers.
- Windowed aggregation enables rolling KPI calculations.
- Streaming alerts are lightweight and actionable.
- Can be extended with ML for anomaly detection, Power BI dashboards, or Slack/email notifications via webhooks.

Appendix:

Brands & Campaigns Using Engagement Alerts

Spotify – "Spotify Wrapped"

- Campaign: Annual personalized summary of user listening behavior.
- Engagement Alerts: Real-time tracking of user interaction across platforms, with alerts on performance drop—e.g., shares or views dipping compared to previous launches.
- Impact:
    - 156 million users engaged in 2022
    - 425 million tweets in the first 3 days amraandelma.com+2empathyfirstmedia.com+2keyword.wordtracker.com+2
- Relevance to us: Mirrors real-time CTR/CVR tracking; we could set up alerts for drops in share rates or completion rates.

How This Aligns with Our Databricks Exercise

Feature	Our Implementation	Industry Comparison
Real-Time Metrics	Compute CTR/CVR every 5 minutes via streaming aggregation	Similar to Spotify and Softonic tracking audience interactions over time
Threshold-Based Alerts	Trigger alert when CTR < 1.5%	Softonic alerts on drops, Zomato tracks CTR dips
Segment-Level Latency	Per-campaign threshold checks	Softonic segmentation, Zomato personalization
Data Storage & Audit	Silver campaign_metrics + alert log table	Enables trend analysis, similar to Spotify's historic comparisons

Summary

- Spotify: Measures vast user interaction, alerts on share/view dips.
- Zomato: Monitors push notification CTR in real-time; alerts on drops.
- Softonic: Uses behavior/time segmentation with CTR alerts.