**Step 5: Power BI dashboards + Case Integration** for your Databricks-first AML MVP. It’s split into two tracks you can run in parallel:

* **5A. BI Serving (Synapse Serverless + Power BI)**
* **5B. Minimal Case Management (Azure SQL + Databricks CDF job)**

**5A) BI serving (Synapse Serverless + Power BI)**

**1) Create Synapse Serverless database, credential, data source**

Point to your **gold** container where aml.gold.alerts (Delta) lives.

-- Connect to your Synapse Serverless endpoint

CREATE DATABASE aml\_bi;

GO

USE aml\_bi;

GO

IF NOT EXISTS (SELECT \* FROM sys.symmetric\_keys WHERE name = '##MS\_DatabaseMasterKey##')

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'Temp#Password\_Only\_For\_This\_Step';

CREATE DATABASE SCOPED CREDENTIAL msi\_cred WITH IDENTITY = 'Managed Identity';

IF NOT EXISTS (SELECT \* FROM sys.external\_data\_sources WHERE name = 'ds\_gold')

CREATE EXTERNAL DATA SOURCE ds\_gold

WITH ( TYPE = HADOOP,

LOCATION = 'abfss://gold@<STORAGE>.dfs.core.windows.net',

CREDENTIAL = msi\_cred );

GO

**2) Analytical views for Power BI**

These views read your **Delta** tables via OPENROWSET … FORMAT='DELTA'.

USE aml\_bi;

GO

-- Base alerts view

CREATE OR ALTER VIEW vw\_alerts AS

SELECT

TRY\_CAST(JSON\_VALUE(evidence, '$.sum7') AS DECIMAL(18,2)) AS sum7,

TRY\_CAST(JSON\_VALUE(evidence, '$.sum30') AS DECIMAL(18,2)) AS sum30,

JSON\_VALUE(evidence, '$.origin\_country') AS origin\_country,

JSON\_VALUE(evidence, '$.beneficiary\_country') AS beneficiary\_country,

a.alert\_id,

a.subject\_id,

a.typology,

a.risk\_score,

a.created\_ts,

CAST(a.created\_ts AS date) AS created\_d

FROM OPENROWSET(

BULK 'alerts',

DATA\_SOURCE = 'ds\_gold',

FORMAT = 'DELTA'

) WITH (

alert\_id VARCHAR(128),

subject\_id VARCHAR(128),

typology VARCHAR(64),

risk\_score FLOAT,

evidence VARCHAR(8000),

created\_ts DATETIME2

) AS a;

GO

-- Daily counts by typology

CREATE OR ALTER VIEW vw\_typology\_daily AS

SELECT created\_d, typology, COUNT(\*) AS alerts

FROM vw\_alerts

GROUP BY created\_d, typology;

GO

-- 7-day investigator queue (latest)

CREATE OR ALTER VIEW vw\_investigator\_queue AS

SELECT \*

FROM vw\_alerts

WHERE created\_ts >= DATEADD(day, -7, SYSUTCDATETIME());

GO

-- Corridor heatmap (origin -> beneficiary)

CREATE OR ALTER VIEW vw\_corridors AS

SELECT

UPPER(origin\_country) AS origin\_country,

UPPER(beneficiary\_country) AS beneficiary\_country,

COUNT(\*) AS alerts

FROM vw\_alerts

WHERE origin\_country IS NOT NULL AND beneficiary\_country IS NOT NULL

GROUP BY UPPER(origin\_country), UPPER(beneficiary\_country);

GO

*(Optional performance tip: If Power BI is set to Import, you can persist summaries with CETAS later. For MVP, DirectQuery is fine.)*

**3) Power BI dataset & measures (DirectQuery)**

* **Connect** Power BI Desktop → *Azure Synapse Analytics (SQL)* → your Serverless endpoint → aml\_bi.
* **Select tables**: vw\_alerts, vw\_investigator\_queue, vw\_typology\_daily, vw\_corridors.

Add baseline **DAX measures**:

Alerts := COUNTROWS('vw\_alerts')

Alerts 7D := COUNTROWS('vw\_investigator\_queue')

Avg Risk := AVERAGE('vw\_alerts'[risk\_score])

Alerts by Typology (last 7d) :=

CALCULATE([Alerts],

FILTER('vw\_alerts', 'vw\_alerts'[created\_ts] >= NOW() - 7)

)

High-Risk Share :=

DIVIDE(

CALCULATE([Alerts], 'vw\_alerts'[risk\_score] >= 0.8),

[Alerts]

)

**Pages to build (MVP)**

* **Investigator Queue:** table of alert\_id, subject\_id, typology, risk\_score, created\_ts + slicers (date, typology). Conditional formatting on risk\_score.
* **Typology Trends:** line chart alerts by created\_d, legend = typology.
* **Corridor Heatmap:** matrix/filled-map using origin\_country x beneficiary\_country → alerts.

**Row-Level Security (optional MVP)**

* Create a small table Users(UPN, Team) and a column AssignedTeam on alerts (or use a mapping).
* Define role filter: Users[UPN] = USERPRINCIPALNAME() and relate Users.Team → Alerts.AssignedTeam.

Publish to Power BI Service; set **DirectQuery** refresh (default live), or schedule if Import.

**5B) Minimal Case Management (Azure SQL + Delta CDF → upsert)**

*Goal:* every new alert in aml.gold.alerts creates/updates a **case** in Azure SQL for investigators to work on (and for integration with ticketing later).

**1) Create Azure SQL DB + schema**

-- In Azure SQL (e.g., "amlcases")

CREATE SCHEMA aml AUTHORIZATION dbo;

GO

CREATE TABLE aml.cases (

case\_id UNIQUEIDENTIFIER NOT NULL DEFAULT NEWID() PRIMARY KEY,

alert\_id VARCHAR(128) NOT NULL UNIQUE,

subject\_id VARCHAR(128) NOT NULL,

typology VARCHAR(64) NOT NULL,

risk\_score FLOAT NOT NULL,

status VARCHAR(20) NOT NULL DEFAULT 'Open', -- Open|InProgress|Closed

assigned\_to VARCHAR(256) NULL, -- UPN or group

created\_ts DATETIME2 NOT NULL DEFAULT SYSUTCDATETIME(),

updated\_ts DATETIME2 NOT NULL DEFAULT SYSUTCDATETIME()

);

CREATE TABLE aml.case\_events (

event\_id BIGINT IDENTITY(1,1) PRIMARY KEY,

case\_id UNIQUEIDENTIFIER NOT NULL,

event\_type VARCHAR(64) NOT NULL, -- Created|Assignment|Note|Disposition

details\_json NVARCHAR(MAX) NULL,

created\_ts DATETIME2 NOT NULL DEFAULT SYSUTCDATETIME(),

created\_by VARCHAR(256) NULL,

CONSTRAINT fk\_case FOREIGN KEY (case\_id) REFERENCES aml.cases(case\_id)

);

CREATE INDEX IX\_cases\_status ON aml.cases(status);

CREATE INDEX IX\_cases\_assigned ON aml.cases(assigned\_to);

**2) Enable Delta Change Data Feed (CDF) on aml.gold.alerts**

Run once in Databricks:

ALTER TABLE aml.gold.alerts SET TBLPROPERTIES (delta.enableChangeDataFeed = true);

**3) Databricks job: stream CDF → upsert into Azure SQL**

Use a small notebook scheduled every 5 minutes (or continuous) to merge new alerts.

**Cluster libraries:** built-in JDBC.  
**Secrets:** store SQL connection string/credentials in **Key Vault** and reference via Databricks secret scope (or use AAD token flow if you prefer).

# /Repos/aml/case\_sync/alerts\_to\_cases.py

from pyspark.sql.functions import col, current\_timestamp

import os

JDBC\_URL = dbutils.secrets.get("kv-aml", "sql-jdbc-url") # e.g., jdbc:sqlserver://<server>.database.windows.net:1433;database=amlcases

JDBC\_USER = dbutils.secrets.get("kv-aml", "sql-user")

JDBC\_PWD = dbutils.secrets.get("kv-aml", "sql-pass")

# Read CDF (inserts & updates) since checkpoint

df = (spark.readStream

.format("delta")

.option("readChangeData", "true")

.table("aml.gold.alerts")

.filter("\_change\_type IN ('insert','update\_postimage')") # CDF metadata columns

.selectExpr("alert\_id","subject\_id","typology","risk\_score","created\_ts")

)

# Write to staging in-memory per microbatch and MERGE into SQL

def upsert\_to\_sql(batch\_df, batch\_id):

batch\_df.createOrReplaceTempView("v\_new\_alerts")

# Write to a temporary table in SQL (staging) then MERGE into aml.cases

tmp\_table = "##incoming\_alerts" # SQL global temp table (session-scoped)

(batch\_df

.write

.format("jdbc")

.mode("overwrite")

.option("url", JDBC\_URL)

.option("user", JDBC\_USER)

.option("password", JDBC\_PWD)

.option("dbtable", tmp\_table)

.save())

merge\_sql = """

MERGE aml.cases AS tgt

USING (SELECT alert\_id, subject\_id, typology, risk\_score, created\_ts FROM ##incoming\_alerts) AS src

ON tgt.alert\_id = src.alert\_id

WHEN NOT MATCHED THEN

INSERT (alert\_id, subject\_id, typology, risk\_score, status, created\_ts, updated\_ts)

VALUES (src.alert\_id, src.subject\_id, src.typology, src.risk\_score, 'Open', src.created\_ts, SYSUTCDATETIME())

WHEN MATCHED THEN

UPDATE SET

tgt.risk\_score = src.risk\_score,

tgt.updated\_ts = SYSUTCDATETIME();

"""

# Execute MERGE via JDBC (small trick: use the same connection with a dummy write to run a query)

from pyspark.sql import SparkSession

spark.\_jsparkSession.sessionState().conf().setConfString("spark.datasource.jdbc.pushDownAggregate.enabled", "false")

(spark.read

.format("jdbc")

.option("url", JDBC\_URL)

.option("user", JDBC\_USER)

.option("password", JDBC\_PWD)

.option("query", merge\_sql)

.load()

)

(df.writeStream

.foreachBatch(upsert\_to\_sql)

.outputMode("update")

.option("checkpointLocation", "abfss://logs@<STORAGE>.dfs.core.windows.net/checkpoints/case\_sync")

.start()

)

Prefer AAD auth? Replace secrets with an AAD access token and set Authentication=ActiveDirectoryAccessToken on the JDBC connection—fine to add later.

**4) Lightweight case UI (choose one)**

**Option A — Power BI “workbench” (fastest)**

* Add a **cases** page: bind to amlcases.aml.cases via the Azure SQL connector.
* Actions:
  + “Assign to me” → use a **Power Automate** flow button calling a stored proc to set assigned\_to = USERPRINCIPALNAME().
  + “Add note” → Power Automate to insert into aml.case\_events.

**Option B — Power Apps Canvas (simple form)**

* Data source: Azure SQL amlcases.
* Screens:
  + Queue (filter status='Open', sort by risk\_score desc).
  + Case details (fields + a gallery of case\_events).
  + Buttons: Assign, Add Note, Close Case (writes to SQL).
* Security: AAD integrated; basic CRUD to those two tables.

*(If you already own a vendor case tool, swap Step 3 to push into their API instead of Azure SQL.)*

**What you have after Step 5**

* **Live dashboards** in Power BI (queue, trends, corridors) backed by Synapse Serverless views.
* **Operational case store** in Azure SQL, kept in sync from **Delta CDF**.
* A **simple investigator UI** (Power BI actions or Power Apps) to triage/assign/close.