# Enabling Batch Inference for FX Deals on Databricks

## 1. Objective

The current inference script (`infer.py`) processes one FX deal at a time to detect anomalies using ML models and business rules. The goal is to extend this capability to handle batch inference — processing multiple deals at once — either from an incoming payload or by reading from a predefined table.

## 2. Proposed Approach (Simple Flow)

### Step 1: Input Source

We will store incoming deals in a Delta table on Databricks, e.g. fx\_deals\_inference\_queue.  
- Each row represents one deal.  
- A column status tracks the progress: NEW, PROCESSING, DONE, or ERROR.  
- Initially, all new deals will have status = NEW.  
  
Optionally, we can allow APIs or jobs to write new records into this table.

### Step 2: Batch Processing Job

A scheduled Databricks job (e.g. every 5–15 minutes) will:  
1. Read all records with status = NEW.  
2. Mark them as PROCESSING to avoid duplicates.  
3. Run inference using the existing infer.py script — modified slightly to accept multiple deals at once.  
4. Write results (e.g. anomaly status, scores, rule violations) back into the same table.  
5. Update status = DONE once complete.  
  
If any record fails, mark it as ERROR with a message.

### Step 3: Output and Completion

Once a deal is processed, its result is available in the same table.  
External systems can poll this table or use a dashboard/view to know when processing is complete.  
No separate listener API is required — status in the table is the source of truth.

## 3. Why Table-Driven Approach?

Simple and robust — no need for complex APIs or queues.  
 Easy to monitor — use Databricks SQL or dashboards to see progress.  
 Scalable — Databricks can handle large batches efficiently.  
 Transparent — single table shows all inputs, outputs, and status.

## 4. Example Table Structure

CREATE TABLE fx\_deals\_inference\_queue (  
 deal\_id STRING,  
 payload STRING, -- JSON or columns with deal details  
 status STRING, -- NEW / PROCESSING / DONE / ERROR  
 result STRING, -- JSON with model predictions and anomalies  
 model\_version STRING,  
 processed\_at TIMESTAMP,  
 error\_message STRING  
);

## 5. Example Workflow Summary

Step | Action | Description  
------|--------|-------------  
1 | Insert new deals | Source systems insert deals with status='NEW'  
2 | Scheduler triggers job | Databricks Workflow runs every few minutes  
3 | Claim batch | Job reads all NEW rows and updates them to PROCESSING  
4 | Run inference | Call infer.py with batch data (DataFrame or list)  
5 | Write results | Save predictions & anomalies back into the table  
6 | Update status | Set status='DONE' or 'ERROR'

## 6. Client Interaction

- Clients can poll the table using deal\_id or batch\_id.  
- Once status = DONE, they can read result for outputs.  
- Optional: build a view or dashboard summarizing status and anomalies.

## 7. Benefits

- No need for listener service — status column indicates completion.  
- No data loss — all data stored in Delta table.  
- Flexible — can be scaled, retried, or reprocessed easily.  
- Auditable — every record has status and result history.

## 8. Next Steps

1. Modify infer.py to accept batch input (list/DataFrame).  
2. Create the Delta table and define schema.  
3. Build a Databricks job to read NEW rows and process them.  
4. Test with small sample batches.  
5. Add monitoring (count of NEW, DONE, ERROR).