Explanation

I built the output by linking the three input tables into one weekly alert pipeline.
First, I loaded the Models table and split each model (e.g., Mod1_INV) into a base model name (Mod1) and customer type (Individual/Corporate). Next, I cleaned the Rules table so each row clearly represents “this rule contributes this score to this model for this customer type.” Then I processed the Hits table by parsing the hit dates, assigning each hit into a weekly window ending on Sunday, and applying the deduplication rule so repeated hits for the same customer + same rule in the same week only count once. After that, I joined the deduplicated hits to the Rules table to convert each hit into a score, aggregated scores per customer per model per week, and finally compared each weekly total score against the corresponding threshold to decide whether an alert is generated. From the resulting alerts, I prepared the three required outputs as required.

Technical Writeup

On the technical side, I kept the Spark job predictable and scalable by defining schemas up front (instead of relying on schema inference), and by standardising column names/types early so joins and aggregations behave consistently. For performance, I only applied partitioning where it helps the heavy steps (mainly before large group-bys / joins), and used broadcast joins only for genuinely small reference tables (like rules/threshold mappings) to avoid unnecessary shuffles without forcing broadcast everywhere, since that can backfire when a table grows.