Bank Transaction Auditing and Flagging

Introduction and Problem Summary

This document details the implementation of a critical security measure within banking transaction processing. The main goal of this PL/SQL block is to process a batch of daily bank transactions and apply a critical security check. The system must ensure that if a high-value withdrawal (over $50,000) is detected, the entire batch processing stops immediately to allow for human review, preventing any further automated actions on the remaining transactions.

In simple terms: we must check every transaction in a list. If we find a large withdrawal, the program must halt all automated steps for the remaining transactions in the list and raise an alert.

Attribute	Description
Project Goal	To enforce a strict security protocol: **instant termination of automated processing** upon the detection of a high-risk withdrawal ($50,000+).
Solution Approach	Solved using **PL/SQL Collections** (for batch data), **Records** (for individual transaction structure), and the **GOTO statement** (for emergency flow control).

Core PL/SQL Concepts Demonstrated

This solution uses these fundamental concepts to structure the data and control the program's flow.

Concept	Structure Used	Technical Role and Functionality
Record	`t_transaction_rec`	Defines a single transaction item, grouping fields like account number and amount into one variable.
Collection	`t_transaction_batch`	Holds the entire list of transactions in memory for batch processing.
GOTO Statement	`GOTO CRITICAL_AUDIT_HALT`	**Critical Control Flow:** Immediately diverts execution out of the processing loop to an alert point, instantly bypassing all remaining logic.

Conceptual Data Model

The code uses in-memory data types to simulate interacting with production database tables.

Test Case	Transaction Details	Expected System Behavior	Proof of Validation
Normal Flow	T1 (Deposit, $1,000)	Processed completely; status is set to `CLEARED`.	Output shows processing details and the final `-> Status: CLEARED` line.

Trigger Event	T2 (Withdrawal, $55,000)	Status is set to FLAGGED FOR REVIEW, and **GOTO** is immediately executed.	The GOTO bypasses the printing of T2's final status line (-> Status: CLEARED).
Halt Confirmation	Flow Control	Program execution jumps directly to the **<<CRITICAL_AUDIT_HALT>>** label.	The output immediately displays the *** AUDIT HALTED *** message.
Skipped Processing	T3 (Transfer, $500)	The processing loop is aborted, and T3 is never loaded or analyzed.	**Crucial:** No output lines related to ACC300 are displayed.

Complete PL/SQL Implementation

This anonymous block defines the necessary types, initializes the sample data, and executes the audit logic.

First : SET SERVEROUTPUT ON; “to tell your session to display the results from the `DBMS_OUTPUT.PUT_LINE` calls.

```
DECLARE

    TYPE t_code_map IS TABLE OF VARCHAR2(20) INDEX BY VARCHAR2(1);

    v_transaction_types t_code_map;
    TYPE t_transaction_rec IS RECORD (

        account_number VARCHAR2(15),

        transaction_type_code VARCHAR2(1),

        amount NUMBER,

        status VARCHAR2(50)

    );


    TYPE t_transaction_batch IS TABLE OF t_transaction_rec;

    v_batch t_transaction_batch;


    c_high_value_limit CONSTANT NUMBER := 50000;

    v_current_transaction t_transaction_rec;


BEGIN

    -- Initialize lookup map

    v_transaction_types('W') := 'WITHDRAWAL';

    v_transaction_types('D') := 'DEPOSIT';

    v_transaction_types('T') := 'TRANSFER';


    -- Initialize test data (Batch Collection)

    v_batch := t_transaction_batch(

        t_transaction_rec('ACC100', 'D', 1000, NULL),

        t_transaction_rec('ACC200', 'W', 55000, NULL), -- GOTO Trigger

        t_transaction_rec('ACC300', 'T', 500, NULL)    -- Expected to be skipped
```

```
   );


   DBMS_OUTPUT.PUT_LINE('--- Starting Batch Audit ---');


   -- Iterate through the Collection
   FOR i IN 1 .. v_batch.COUNT LOOP
      v_current_transaction := v_batch(i);


      DBMS_OUTPUT.PUT_LINE('Processing ' ||
v_transaction_types(v_current_transaction.transaction_type_code) ||
                           ' for $' || v_current_transaction.amount);


      -- CRITICAL HALT CONDITION
      IF v_current_transaction.transaction_type_code = 'W' AND v_current_transaction.amount >
c_high_value_limit THEN
         v_current_transaction.status := 'FLAGGED FOR REVIEW';
         GOTO CRITICAL_AUDIT_HALT; -- Execute immediate jump
      ELSE
         v_current_transaction.status := 'CLEARED';
      END IF;


      DBMS_OUTPUT.PUT_LINE('  -> Status: ' || v_current_transaction.status);
   END LOOP;


   -- Normal exit path (Only reached if NO high-value withdrawal is found)
   DBMS_OUTPUT.PUT_LINE(CHR(10) || '--- Batch Audit Complete: All transactions cleared. ---');

   GOTO END_PROCEDURE;


<<CRITICAL_AUDIT_HALT>>

   -- GOTO Destination: Code executed upon critical failure

   DBMS_OUTPUT.PUT_LINE(CHR(10) || '*** AUDIT HALTED ***');

   DBMS_OUTPUT.PUT_LINE('Critical Flag: High-value withdrawal detected on Account ' || v_current_transaction.account_number);

   DBMS_OUTPUT.PUT_LINE('Processing stopped immediately to prevent further automated actions.');


<<END_PROCEDURE>>

   NULL; -- Clean exit point


END;

/
```

Testing and Validation (Proof of Logic)

The validation plan confirms that the `GOTO` statement provides the required, non-sequential control flow diversion necessary for the emergency halt.