

EXERCISE 18

Program 1

Write a code in PL/SQL to develop a trigger that enforces referential integrity by preventing the deletion of a parent record if child records exist.

```
CREATE TABLE parent_table (  
    parent_id NUMBER PRIMARY KEY,  
    parent_data VARCHAR2(100)  
);  
  
CREATE TABLE child_table (  
    child_id NUMBER PRIMARY KEY,  
    parent_id NUMBER,  
    child_data VARCHAR2(100),  
    FOREIGN KEY (parent_id) REFERENCES parent_table(parent_id)  
);  
  
CREATE OR REPLACE TRIGGER trg_prevent_parent_deletion  
BEFORE DELETE ON parent_table  
FOR EACH ROW  
DECLARE  
    v_child_count NUMBER;  
BEGIN  
    SELECT COUNT(*)  
    INTO v_child_count  
    FROM child_table  
    WHERE parent_id = :OLD.parent_id;  
  
    IF v_child_count > 0 THEN  
        RAISE_APPLICATION_ERROR(-20001, 'Cannot delete parent record; child records  
exist.');
```

END IF;
END;/

Program 2

Write a code in PL/SQL to create a trigger that checks for duplicate values in a specific column and raises an exception if found.

```
CREATE TABLE unique_column_table (  
    id NUMBER PRIMARY KEY,  
    unique_data VARCHAR2(100)  
);  
  
CREATE OR REPLACE TRIGGER trg_check_duplicates  
BEFORE INSERT OR UPDATE ON unique_column_table  
FOR EACH ROW  
  
DECLARE  
    v_count NUMBER;  
  
BEGIN  
    SELECT COUNT(*)  
    INTO v_count  
    FROM unique_column_table  
    WHERE unique_data = :NEW.unique_data  
    AND id != :NEW.id;  
  
    IF v_count > 0 THEN  
        RAISE_APPLICATION_ERROR(-20002, 'Duplicate value found in unique_data column.');    END IF;  
  
END;  
  
/
```

Program 3

Write a code in PL/SQL to create a trigger that restricts the insertion of new rows if the total of a column's values exceeds a certain threshold.

```
CREATE TABLE threshold_table (  
    id NUMBER PRIMARY KEY,  
    value NUMBER  
);  
  
CREATE OR REPLACE TRIGGER trg_restrict_inserts  
BEFORE INSERT ON threshold_table  
FOR EACH ROW  
DECLARE  
    v_total NUMBER;  
    v_threshold CONSTANT NUMBER := 1000; -- Set your threshold here  
  
BEGIN  
    SELECT SUM(value)  
    INTO v_total  
    FROM threshold_table;  
  
    IF v_total + :NEW.value > v_threshold THEN  
        RAISE_APPLICATION_ERROR(-20003, 'Total value exceeds the threshold.');
```

END IF;

END;

/

Program 4

Write a code in PL/SQL to design a trigger that captures changes made to specific columns and logs them in an audit table.

```
CREATE TABLE original_table (  
    id NUMBER PRIMARY KEY,  
    sensitive_data VARCHAR2(100),  
    other_data VARCHAR2(100)  
);  
  
CREATE TABLE audit_table (  
    audit_id NUMBER PRIMARY KEY,  
    id NUMBER,  
    old_sensitive_data VARCHAR2(100),  
    new_sensitive_data VARCHAR2(100),  
    change_date DATE,  
    user_name VARCHAR2(100)  
);  
  
CREATE OR REPLACE TRIGGER trg_capture_changes  
AFTER UPDATE ON original_table  
FOR EACH ROW  
BEGIN  
    IF :OLD.sensitive_data != :NEW.sensitive_data THEN  
        INSERT INTO audit_table (audit_id, id, old_sensitive_data, new_sensitive_data,  
change_date, user_name)  
VALUES (audit_seq.NEXTVAL, :OLD.id, :OLD.sensitive_data, :NEW.sensitive_data,  
SYSDATE, USER);  
    END IF;  
END;  
/
```

Program 5

Write a code in PL/SQL to implement a trigger that records user activity (inserts, updates, deletes) in an audit log for a given set of tables.

```
CREATE TABLE activity_log (  
    log_id NUMBER PRIMARY KEY,  
    table_name VARCHAR2(100),  
    operation VARCHAR2(10),  
    id NUMBER,  
    activity_date DATE,  
    user_name VARCHAR2(100)  
);  
  
CREATE OR REPLACE TRIGGER trg_log_activity  
AFTER INSERT OR UPDATE OR DELETE ON original_table  
FOR EACH ROW  
BEGIN  
    IF INSERTING THEN  
        INSERT INTO activity_log (log_id, table_name, operation, id, activity_date, user_name)  
        VALUES (activity_seq.NEXTVAL, 'original_table', 'INSERT', :NEW.id, SYSDATE,  
        USER);  
    ELSIF UPDATING THEN  
        INSERT INTO activity_log (log_id, table_name, operation, id, activity_date, user_name)  
        VALUES (activity_seq.NEXTVAL, 'original_table', 'UPDATE', :NEW.id, SYSDATE,  
        USER);  
    ELSIF DELETING THEN  
        INSERT INTO activity_log (log_id, table_name, operation, id, activity_date, user_name)  
        VALUES (activity_seq.NEXTVAL, 'original_table', 'DELETE', :OLD.id, SYSDATE,  
        USER);  
    END IF;  
END;  
/
```

Program 7

Write a code in PL/SQL to implement a trigger that automatically calculates and updates a running total column for a table whenever new rows are inserted.

```
CREATE TABLE sales (  
    sale_id NUMBER PRIMARY KEY,  
    sale_amount NUMBER,  
    running_total NUMBER  
);  
  
CREATE OR REPLACE TRIGGER trg_update_running_total  
AFTER INSERT ON sales  
FOR EACH ROW  
DECLARE  
    v_running_total NUMBER;  
  
BEGIN  
    SELECT NVL(MAX(running_total), 0)  
    INTO v_running_total  
    FROM sales  
    WHERE sale_id != :NEW.sale_id;  
  
    UPDATE sales  
    SET running_total = v_running_total + :NEW.sale_amount  
    WHERE sale_id = :NEW.sale_id;  
  
END;  
  
/
```

Program 8

Write a code in PL/SQL to create a trigger that validates the availability of items before allowing an order to be placed, considering stock levels and pending orders.

```
CREATE TABLE items (  
    item_id NUMBER PRIMARY KEY,  
    item_name VARCHAR2(100),  
    stock_level NUMBER  
);  
  
CREATE TABLE orders (  
    order_id NUMBER PRIMARY KEY,  
    item_id NUMBER,  
    order_quantity NUMBER  
);  
  
CREATE OR REPLACE TRIGGER trg_validate_availability  
BEFORE INSERT ON orders  
FOR EACH ROW  
DECLARE  
    v_stock_level NUMBER;  
  
BEGIN  
    SELECT stock_level  
    INTO v_stock_level  
    FROM items  
    WHERE item_id = :NEW.item_id;  
  
    IF v_stock_level < :NEW.order_quantity THEN  
        RAISE_APPLICATION_ERROR(-20004, 'Insufficient stock for item.');
```

END IF;
END;
/