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 OR Replace Trigger prevent_parent_delete
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_primary_key;

    IF V_child_count > 0 THEN
        RAISE_APPLICATION_ERROR (-20001, 'Cannot delete parent row: 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 OR REPLACE TRIGGER check_duplicate_value
BEFORE INSERT OR UPDATE ON your_table
FOR EACH ROW
    v_count NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO v_count
    FROM your_table
    WHERE some_unique_column = :new.some_unique_column;
    IF v_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Duplicate value: This
                                value already exists.');
    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 OR REPLACE TRIGGER check_total_threshold
AFTER INSERT ON your_table
DECLARE
    v_total NUMBER;
    v_threshold NUMBER := 100000;
BEGIN
    SELECT SUM(some_column)
    INTO v_total
    FROM your_table;
    IF v_total > v_threshold THEN
        RAISE_APPLICATION_ERROR(-20003, 'Insertion failed:
                                Total exceeds 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 column_audit log (
        log_id NUMBER GENERATED AS IDENTITY,
        table_audited VARCHAR2(30),
        column_audited VARCHAR2(30),
        row_pk VARCHAR2(100),
        old_value VARCHAR2(1000),
        new_value VARCHAR2(1000),
        changed_by VARCHAR2(30),
        change_date TIMESTAP
);

CREATE OR REPLACE TRIGGER log_column_changes
AFTER UPDATE ON your_table
FOR EACH ROW
BEGIN
    If :old.salary != :new.salary THEN
        INSERT INTO column_audit log(table_audited, column_
audited, row_pk, old_value, new_value, changed_by, change_date)
        VALUES ('your_table', 'salary', :old.primary_key,
        :old.salary, :new.salary, USER, SYSTIMESTAMP);
    END IF;
```

```
IF :old.job_id != :new.job_id THEN
    INSERT INTO column_audit_log (table_audited, column_
    audited, row_pk, old_value, new_value, changed_by,
    change_date)
    VALUES ('your_table', 'job_id', :old.primary_key,
    :old.job_id, :new.job_id, USER, SYSTIMESTAMP);
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 OR REPLACE TRIGGER log_user_activity
AFTER INSERT OR UPDAE OR DELETE ON your_table
FOR EACH ROW
DECLARE
    v_action VARCHAR2(10);
BEGIN
    IF INSERTING THEN
        v_action := 'INSERT';
        INSERT INTO activity_audit_log( table_audited, dml_action,
        row_pk, action by, actin_date)
        VALUES ('your_table', v_action, :new.primary_key, USER,
        SYSTIMESTAMP);

    ELSIF UPDATING THEN
        v_action := 'UPDATE';
        INSERT INTO activity_audit_log (table_audited,
        dml_action, row_pk, action_by, action_date)
        VALUES ('your_table', v_action, :old.primary_by,
        USER, SYS TIMESTAMP);
```

```
ELSIF DELETING THEN
    V_action := 'DELETE';
    INSERT INTO activity_audit_log (table_audited, dml_action,
    row_pk, action_by, action_date)
    VALUES ('your_table', V_action, :old.primary_key,
    USER, SYSTIMESTAMP);
    ENDIF;
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 OR REPLACE TRIGGER update_summary_total
AFTER INSERT ON orders
FOR EACH ROW
BEGIN
    UPDATE sales_summary
    SET total_sales = total_sales + :new.amount
    WHERE summary_id = 1;
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 OR REPLACE TRIGGER validate_stock_level
BEFORE INSERT ON order_items
FOR EACH ROW
DECLARE
    v_stock NUMBER;
BEGIN
    SELECT stock_level
    INTO v_stock
    FROM products
    WHERE product_id = :new.product_id;
    IF v_stock < :new.quantity THEN
        RAISE_APPLICATION_ERROR (-20004, 'Cannot
        place order : Insufficient stock for product' ||
        new.product_id);
    ELSE
        UPDATE products
        SET stock_level = stock_level - :new.quantity
        WHERE product_id = :new.product_id;
    END IF;
END;
```

Evaluation Procedure	Marks awarded
PL/SQL Procedure(5)	5
Program/Execution (5)	5
Viva(5)	5
Total (15)	15
Faculty Signature	