**Ex. No.: 13 WORKING WITH TRIGGER** DEEPA S

230701065

Initial:

CREATE TABLE orders ( order\_id NUMBER PRIMARY KEY, item\_id

NUMBER, quantity NUMBER,

order\_date DATE, running\_total NUMBER, user\_id NUMBER,

FOREIGN KEY (item\_id) REFERENCES items(item\_id)

);

INSERT INTO orders (order\_id, item\_id, quantity, order\_date, running\_total, user\_id) VALUES (1, 1, 20, SYSDATE, 20, 101);

INSERT INTO orders (order\_id, item\_id, quantity, order\_date, running\_total, user\_id) VALUES (2, 2, 30, SYSDATE, 50, 102);

CREATE TABLE items ( item\_id NUMBER PRIMARY KEY, item\_name VARCHAR2(50), stock\_level NUMBER, pending\_orders NUMBER DEFAULT 0

);

INSERT INTO items (item\_id, item\_name, stock\_level, pending\_orders) VALUES (1, 'Item A', 100, 0);

INSERT INTO items (item\_id, item\_name, stock\_level, pending\_orders) VALUES (2, 'Item B', 50, 0);

INSERT INTO items (item\_id, item\_name, stock\_level, pending\_orders) VALUES (3, 'Item C', 150, 0);

CREATE TABLE audit\_log (

log\_id NUMBER PRIMARY KEY,

table\_name VARCHAR2(50), operation VARCHAR2(10),

change\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

user\_id NUMBER, details VARCHAR2(200)

);

CREATE SEQUENCE audit\_log\_seq START WITH 1

# INCREMENT BY 1;

1. 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 items

# FOR EACH ROW DECLARE

child\_count NUMBER;

# BEGIN

SELECT COUNT(\*) INTO child\_count FROM orders WHERE item\_id = :OLD.item\_id;

IF child\_count > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Cannot delete item; dependent orders exist.');

# END IF;

END; /

1. 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\_for\_duplicates BEFORE INSERT OR UPDATE ON orders

# FOR EACH ROW DECLARE

duplicate\_count NUMBER;

# BEGIN

SELECT COUNT(\*) INTO duplicate\_count FROM orders

WHERE item\_id = :NEW.item\_id AND order\_id != :NEW.order\_id;

IF duplicate\_count > 0 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Duplicate item entry found in orders.');

# END IF;

END; /

1. 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 restrict\_insertion BEFORE INSERT ON orders

# FOR EACH ROW DECLARE

total\_quantity NUMBER;

# BEGIN

SELECT SUM(quantity) INTO total\_quantity FROM orders; IF (total\_quantity + :NEW.quantity) > 500 THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Cannot insert order; total quantity exceeds threshold.');

# END IF;

END; /

1. 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 OR REPLACE TRIGGER log\_changes AFTER UPDATE ON orders

# FOR EACH ROW BEGIN

INSERT INTO audit\_log (log\_id, table\_name, operation, user\_id, details) VALUES (audit\_log\_seq.NEXTVAL, 'orders', 'UPDATE', :NEW.user\_id, 'Order ' ||

:NEW.order\_id || ' changed from ' || :OLD.quantity || ' to ' || :NEW.quantity ); END;

/

1. 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 DELETE OR UPDATE ON orders FOR EACH ROW

# BEGIN

INSERT INTO audit\_log (log\_id, table\_name, operation, user\_id, details) VALUES (audit\_log\_seq.NEXTVAL, 'orders',

CASE

WHEN INSERTING THEN 'INSERT' WHEN UPDATING THEN 'UPDATE' WHEN DELETING THEN 'DELETE'

END,

NVL(:NEW.user\_id, :OLD.user\_id), 'User action recorded on order ' || NVL(:NEW.order\_id, :OLD.order\_id));

# END; /

1. 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\_running\_total AFTER INSERT ON orders

# FOR EACH ROW BEGIN

UPDATE orders SET running\_total = (SELECT SUM(quantity) FROM orders) WHERE order\_id = :NEW.order\_id;

# END; /

1. 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\_item\_availability BEFORE INSERT ON orders

# FOR EACH ROW DECLARE

available\_stock NUMBER;

# BEGIN

SELECT stock\_level - pending\_orders INTO available\_stock FROM items WHERE item\_id = :NEW.item\_id;

IF :NEW.quantity > available\_stock THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Insufficient stock available for the order.');

# END IF;

END; /

UPDATE items SET pending\_orders = pending\_orders + :NEW.quantity WHERE item\_id = :NEW.item\_id;

Result:

The given programs are performed successfully.