Practical 09 Part 02

Aim: Write and execute triggers using PL/SQL.

1. BEFORE INSERT Trigger (Row Level)

Goal: Print a message before inserting into `Employee`.

```
CREATE OR REPLACE TRIGGER trg_before_insert_emp
BEFORE INSERT ON Employee
FOR EACH ROW
BEGIN
  DBMS_OUTPUT.PUT_LINE('Inserting employee: ' ||
:NEW.Name);
END;
/
```

Test:

```
INSERT INTO Employee VALUES (201, 'Ravi',
'Clerk');
```

2. AFTER UPDATE Trigger (Row Level)

Goal: Show old and new balances after update.

```
CREATE OR REPLACE TRIGGER
trg_after_update_account
AFTER UPDATE ON Account
FOR EACH ROW
BEGIN
  DBMS_OUTPUT.PUT_LINE('Account updated from '
|| :OLD.Balance || ' to ' || :NEW.Balance);
END;
/
```

Test:

```
UPDATE Account SET Balance = 3000 WHERE
AccountID = 1001;
```

3. BEFORE DELETE Trigger (Row Level)

Goal: Prevent deleting accounts with balance > 0.

```
CREATE OR REPLACE TRIGGER
trg_block_high_bal_delete
BEFORE DELETE ON Account
FOR EACH ROW
BEGIN
  IF :OLD.Balance > 0 THEN
    RAISE_APPLICATION_ERROR(-20001, 'Cannot
delete account with positive balance');
  END IF;
END;
/
```

Test:

```
DELETE FROM Account WHERE AccountID = 1001;
```

4. AFTER UPDATE (Statement-Level Trigger)

Goal: Notify that an update has occurred.

```
CREATE OR REPLACE TRIGGER
trg_stmt_update_customer
AFTER UPDATE ON Customer
BEGIN
  DBMS_OUTPUT.PUT_LINE('Customer table has been
updated.');
```

```
END;
/
```

Test:

```
UPDATE Customer SET Name = 'Ajay' WHERE
CustomerID = 1;
```

5. INSTEAD OF Trigger on View

```
-- Create view
CREATE OR REPLACE VIEW acc_view AS
SELECT AccountID, Balance FROM Account;

-- Trigger to allow updates
CREATE OR REPLACE TRIGGER trg_update_acc_view
INSTEAD OF UPDATE ON acc_view
FOR EACH ROW
BEGIN
  UPDATE Account
  SET Balance = :NEW.Balance
  WHERE AccountID = :OLD.AccountID;
END;
/
```

Test:

```
UPDATE acc_view SET Balance = 6000 WHERE
AccountID = 1001;
```

Lab Tasks (SQL*Plus Based)

1. **Create** a `BEFORE INSERT` trigger to print employee department.

2. **Create** an `AFTER UPDATE` trigger to display old and new account balances.

3. **Write** a trigger to **prevent deletion** of customers if they have accounts.

4. **Create** a **statement-level trigger** to log updates on the `Customer` table.

5. **Write** an **INSTEAD OF trigger** on a view for updating balances.

Submit the following in PDF

- Submit the .sql script with all trigger definitions.

- Provide screen output showing successful trigger execution.

- Comment on each trigger with its **type**, **purpose**, and **test query**.