1. **Trigger**
2. Write a update, delete trigger on clientmstr table. The System should keep track of the records that ARE BEING updated or deleted. The old value of updated or deleted records should be added in audit\_trade table. (separate implementation using both row and statement triggers).
3. Write a before trigger for Insert, update event considering following requirement: Emp(e\_no, e\_name, salary) I) Trigger action should be initiated when salary is tried to be inserted is less than Rs. 50,000/- II) Trigger action should be initiated when salary is tried to be updated for value less than Rs. 50,000/- Action should be rejection of update or Insert operation by displaying appropriate error message. Also the new values expected to be inserted will be stored in new table Tracking(e\_no, salary).

a) Update and Delete Triggers with Audit Trail using Both Row and Statement Triggers:

1. Row Trigger for Update:

```sql

CREATE OR REPLACE TRIGGER audit\_update\_clientmstr

BEFORE UPDATE ON clientmstr FOR EACH ROW

BEGIN

IF :OLD.client\_status IS NOT NULL THEN

-- Insert the old record into the audit\_trade table

INSERT INTO audit\_trade (client\_id, client\_name, action, action\_date)

VALUES (:OLD.client\_id, :OLD.client\_name, 'UPDATE', SYSDATE);

END IF;

END;

/

```

2. Row Trigger for Delete:

```sql

CREATE OR REPLACE TRIGGER audit\_delete\_clientmstr

BEFORE DELETE ON clientmstr FOR EACH ROW

BEGIN

-- Insert the old record into the audit\_trade table before deletion

INSERT INTO audit\_trade (client\_id, client\_name, action, action\_date)

VALUES (:OLD.client\_id, :OLD.client\_name, 'DELETE', SYSDATE);

END;

/

```

3. Statement Trigger for Update:

```sql

CREATE OR REPLACE TRIGGER audit\_update\_clientmstr\_stmt

AFTER UPDATE ON clientmstr

BEGIN

-- Insert the old records into the audit\_trade table using a statement trigger

FOR rec IN (SELECT client\_id, client\_name FROM clientmstr WHERE client\_status IS NOT NULL) LOOP

INSERT INTO audit\_trade (client\_id, client\_name, action, action\_date)

VALUES (rec.client\_id, rec.client\_name, 'UPDATE', SYSDATE);

END LOOP;

END;

/

```

4. Statement Trigger for Delete:

```sql

CREATE OR REPLACE TRIGGER audit\_delete\_clientmstr\_stmt

AFTER DELETE ON clientmstr

BEGIN

-- Insert the old records into the audit\_trade table using a statement trigger

FOR rec IN (SELECT client\_id, client\_name FROM clientmstr) LOOP

INSERT INTO audit\_trade (client\_id, client\_name, action, action\_date)

VALUES (rec.client\_id, rec.client\_name, 'DELETE', SYSDATE);

END LOOP;

END;

/

```

In the above triggers, we have both row triggers and statement triggers for update and delete operations on the `clientmstr` table. These triggers capture the old records before updates or deletions and insert them into the `audit\_trade` table.

b) Before Trigger for Insert and Update with Salary Validation:

```sql

CREATE OR REPLACE TRIGGER salary\_validation

BEFORE INSERT OR UPDATE ON emp

FOR EACH ROW

DECLARE

v\_min\_salary NUMBER := 50000; -- Minimum salary threshold

BEGIN

IF :NEW.salary < v\_min\_salary THEN

-- Display an error message and reject the operation

RAISE\_APPLICATION\_ERROR(-20001, 'Salary must be greater than or equal to Rs. 50,000.');

-- Store the new values in the Tracking table (assumes Tracking table exists)

INSERT INTO Tracking (e\_no, salary)

VALUES (:NEW.e\_no, :NEW.salary);

END IF;

END;

/

```

In this before trigger, we validate the salary during insert or update operations. If the salary is less than Rs. 50,000, it raises an application error with an error message and rejects the operation. The new values that were expected to be inserted or updated are stored in the `Tracking` table (assuming the `Tracking` table exists).