**Design normalized schema**

CREATE TABLE users (

user\_id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50) NOT NULL UNIQUE,

password\_hash VARCHAR(255) NOT NULL,

role VARCHAR(20) NOT NULL,

email VARCHAR(100) UNIQUE,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE departments (

dept\_id INT AUTO\_INCREMENT PRIMARY KEY,

dept\_name VARCHAR(100) NOT NULL,

description TEXT

);

CREATE TABLE categories (

category\_id INT AUTO\_INCREMENT PRIMARY KEY,

category\_name VARCHAR(100) NOT NULL,

description TEXT

);

CREATE TABLE complaints (

complaint\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

dept\_id INT NOT NULL,

category\_id INT NOT NULL,

title VARCHAR(200) NOT NULL,

description TEXT,

status VARCHAR(20) DEFAULT 'OPEN',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(user\_id) ON DELETE CASCADE,

FOREIGN KEY (dept\_id) REFERENCES departments(dept\_id),

FOREIGN KEY (category\_id) REFERENCES categories(category\_id)

);

CREATE TABLE responses (

response\_id INT AUTO\_INCREMENT PRIMARY KEY,

complaint\_id INT NOT NULL,

user\_id INT NOT NULL,

comment TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (complaint\_id) REFERENCES complaints(complaint\_id) ON DELETE CASCADE,

FOREIGN KEY (user\_id) REFERENCES users(user\_id) ON DELETE CASCADE

);

--Add indexes for performance

CREATE INDEX idx\_complaints\_user\_id ON complaints(user\_id);

CREATE INDEX idx\_complaints\_dept\_id ON complaints(dept\_id);

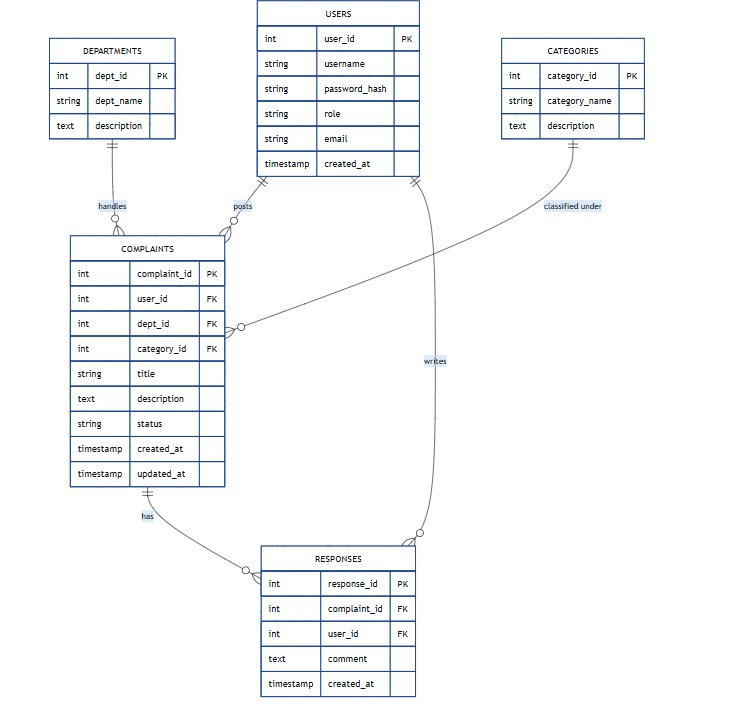
CREATE INDEX idx\_complaints\_category\_id ON complaints(category\_id);

CREATE INDEX idx\_complaints\_status ON complaints(status);

CREATE INDEX idx\_responses\_complaint\_id ON responses(complaint\_id);

CREATE INDEX idx\_responses\_user\_id ON responses(user\_id);

**ER Diagram:**



**SQL Queries Implementation**

**A. DML – Insert, Update, Delete**

**1. Insert Sample Users**

INSERT INTO users (username, password\_hash, role, email)

VALUES

('alice', 'pass1hash', 'citizen', 'alice@mail.com'),

('bob', 'pass2hash', 'employee', 'bob@mail.com'),

('admin', 'pass3hash', 'admin', 'admin@mail.com');

**2. Insert Departments**

INSERT INTO departments (dept\_name, description)

VALUES

('IT', 'Handles technical issues'),

('HR', 'Human resources'),

('Admin', 'Administrative tasks');

**3. Insert Categories**

INSERT INTO categories (category\_name, description)

VALUES

('Network Issue', 'Problems with internet and WiFi'),

('Salary', 'Payroll-related issues'),

('Cleanliness', 'Hygiene and sanitation');

**4. Insert a Complaint**

INSERT INTO complaints (user\_id, dept\_id, category\_id, title, description)

VALUES (1, 1, 1, 'WiFi not working', 'The internet is down in the main building.');

**5. Insert a Response**

INSERT INTO responses (complaint\_id, user\_id, comment)

VALUES (1, 2, 'We have informed the network team. Will be resolved soon.');

**🔹 B. DQL – Select Queries**

**1. View All Complaints**

SELECT \* FROM complaints;

**2. Get Complaints with User, Department, and Category**

SELECT

c.complaint\_id,

u.username AS user,

d.dept\_name AS department,

cat.category\_name AS category,

c.title,

c.status,

c.created\_at

FROM complaints c

JOIN users u ON c.user\_id = u.user\_id

JOIN departments d ON c.dept\_id = d.dept\_id

JOIN categories cat ON c.category\_id = cat.category\_id;

**3. All Responses to a Complaint**

SELECT

r.response\_id,

r.comment,

r.created\_at,

u.username AS responded\_by

FROM responses r

JOIN users u ON r.user\_id = u.user\_id

WHERE r.complaint\_id = 1;

**🔹 C. JOINS – Combine Data Across Tables**

**1. Join Complaints and Responses**

SELECT

c.title,

r.comment,

r.created\_at,

responder.username AS responded\_by

FROM complaints c

JOIN responses r ON c.complaint\_id = r.complaint\_id

JOIN users responder ON r.user\_id = responder.user\_id;

**🔹 D. Subqueries**

**1. Complaints Submitted to HR Department**

SELECT \* FROM complaints

WHERE dept\_id = (

SELECT dept\_id FROM departments WHERE dept\_name = 'HR'

);

**2. Users with Multiple Complaints**

SELECT u.username, COUNT(c.complaint\_id) AS total\_complaints

FROM users u

JOIN complaints c ON u.user\_id = c.user\_id

GROUP BY u.user\_id

HAVING COUNT(c.complaint\_id) > 1;

**3. Most Recent Complaint per User**

SELECT \* FROM complaints c

WHERE created\_at = (

SELECT MAX(created\_at)

FROM complaints

WHERE user\_id = c.user\_id

);

**🔹 E. Rollback and Transaction Example**

**Use when modifying data cautiously:**

START TRANSACTION;

-- Mistakenly updating the wrong complaint

UPDATE complaints

SET status = 'Resolved'

WHERE complaint\_id = 100;

-- Realize the mistake

ROLLBACK;

-- Correct update

START TRANSACTION;

UPDATE complaints

SET status = 'Resolved'

WHERE complaint\_id = 1;

COMMIT;