

## Introduction

This phase of my project focuses on extending the Student Housing and Maintenance Management System by incorporating automated database operations through triggers and stored procedures.

The main goal here is to ensure that frequently occurring operations, consistency rules, and administrative tasks are handled automatically at the database level which enhances database automation.

A trigger is a code that executes when a database event such as INSERT, UPDATE, or DELETE occurs. Triggers increase data integrity, enforce business rules, and eliminate human error by reacting instantly to changes in the database.

A stored procedure is a group of SQL codes stored together that can be reused over and over. Stored procedures help encapsulate repetitive operations into a single function. They help improve performance, by reducing code duplication, and make the system easier to maintain.

In this phase, one trigger and one stored procedure were implemented to automate maintenance request assignment and to generate dorm occupancy reports.

## Trigger

**Trigger Name:** trg\_request\_auto\_assign

**Table:** MaintenanceRequest

**Type:** BEFORE INSERT Trigger

### **Purpose and Functionality:**

This trigger ensures that every maintenance request recorded in the system has an assigned staff member, even if the user submitting the request does not specify one. Students usually do not know which technician should handle an issue, and leaving a request unassigned would result in delays and inconsistent data.

The trigger executes before a row is inserted into the MaintenanceRequest table.

If the assignee\_staff\_id value of the new record is NULL, the trigger automatically assigns the default technician:

- **Staff ID:** 501
- **Name:** Ahmet Usta
- **Role:** Electrician

This approach guarantees seamless processing of maintenance requests and prevents incomplete or invalid records from being stored.

### **Trigger Script:**

```
USE housing_mgmt;

DELIMITER //

CREATE TRIGGER trg_request_auto_assign
BEFORE INSERT ON MaintenanceRequest
FOR EACH ROW
BEGIN
    IF NEW.assignee_staff_id IS NULL THEN
        SET NEW.assignee_staff_id = 501;
    END IF;
END//

DELIMITER ;
```

### **Screenshots:**

#### **Before Trigger:**

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

recitation1

sys

Administration Schemas

Information

No object selected

Object Info Session

Query 1

```
140 BEGIN
141 IF NEW.assignee_staff_id IS NULL THEN
142 SET NEW.assignee_staff_id = 501;
143 END IF;
144 END//
145
146 DELIMITER ;
147 SELECT * FROM MaintenanceRequest;
148
```

Result Grid

request_id	dorm_id	roomno	reporter_stud_id	assignee
3001	1	101	1001	502
3002	1	102	1002	501
3003	2	201	1003	503
3004	2	202	1004	505
3005	3	305	1005	507
3006	4	410	1006	506
3007	5	115	1007	508
3008	8	220	1008	501
3009	9	130	1009	509
3010	10	510	1010	503

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Context Help Snippets

Windows'u Etkinleştir

Windows'u etkinleştirmek için Ayarlar'a gidin.

Query Completed

## After Trigger:

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

recitation1

sys

Administration Schemas

Information

No object selected

Object Info Session

Query 1

```
151 SELECT * FROM MaintenanceRequest WHERE request_id = 4001;
152
```

Result Grid

request_id	dorm_id	roomno	reporter_stud_id	assignee
4001	1	101	1001	501

MaintenanceRequest 2 MaintenanceRequest 3 Apply Revert Context Help Snippets

Output

Action Output

#	Time	Action	Message	Duration / Fetch
63	01:28:36	INSERT INTO Contract (c_id, stud_id, dorm_id, roomno) VALUES (9001, 1001, 1, ...)	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.000 sec
64	01:28:36	CREATE TABLE MaintenanceRequest ( request_id INT PRIMARY KEY, dor...	0 row(s) affected	0.062 sec
65	01:28:36	INSERT INTO MaintenanceRequest (request_id, dorm_id, roomno, reporter_stud_...	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.016 sec
66	01:28:36	USE housing_mngmt	0 row(s) affected	0.000 sec
67	01:28:36	CREATE TRIGGER trg_request_auto_assign BEFORE INSERT ON Maintenance...	0 row(s) affected	0.015 sec
68	01:28:36	SELECT * FROM MaintenanceRequest LIMIT 0, 1000	10 row(s) returned	0.016 sec / 0.000 sec
69	01:28:36	INSERT INTO MaintenanceRequest (request_id, dorm_id, roomno, reporter_stud_...	1 row(s) affected	0.000 sec
70	01:28:36	SELECT * FROM MaintenanceRequest WHERE request_id = 4001 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Windows'u Etkinleştir

Windows'u etkinleştirmek için Ayarlar'a gidin.

Query Completed

# Stored Procedure

Name: sp\_room\_occupancy\_report

## Purpose:

This stored procedure provides an administrative report showing room usage for a given dormitory. Dorm management frequently needs to know how many students are currently living in each room, how many beds are free, and whether rooms are fully occupied.

## The procedure takes a dorm ID as input and returns:

- The room numbers in that dorm
- Each room's assigned capacity
- The number of students currently assigned through the [Contract](#) table
- The number of free beds remaining

## Procedure Script:

```
USE housing_mgmt;
```

```
DELIMITER //
```

```
CREATE PROCEDURE sp_room_occupancy_report(IN p_dorm_id INT)
```

```
BEGIN
```

```
    SELECT
```

```
        r.dorm_id,
```

```
        r.roomno,
```

```
        r.capacity,
```

```
        COUNT(c.c_id) AS current_occupancy,
```

```
        (r.capacity - COUNT(c.c_id)) AS free_beds
```

```
FROM Room r
```

```
LEFT JOIN Contract c
```

```
    ON r.dorm_id = c.dorm_id
```

```
    AND r.roomno = c.roomno
```

```
WHERE r.dorm_id = p_dorm_id
GROUP BY r.dorm_id, r.roomno, r.capacity
ORDER BY r.roomno;
END//

DELIMITER ;
```