আন্তর্জাতিক ইসলামী বিশ্ববিদ্যালয় চট্টগ্রাম
الجامعة الإسلامية العالمية شيتاغونغ
International Islamic University Chittagong

Department of Computer Science and Engineering

Project Report

Project Title: Smart Home Management System

Course Title: Database Management System Lab

Course Code: CSE-2424

Semester: 4th

Submitted By

Name: Nusrat Jahan Chaity
Student ID: C233470
Section: 4BF

Instructor

Ms. Mysha Sarin Kabisha
Assistant Lecturer
Department of Computer Science and Engineering
International Islamic University Chittagong

Submission Date: 13/07/2025

---

1. Abstract

This project focuses on developing a **Smart Home Management System** that helps users control and monitor household devices, manage room access, and schedule device operations efficiently. Using Oracle Database, the project includes designing an ERD, normalization to 3NF, and implementing tables with sample data. A web-based user interface supports two modes: *Normal Mode* for user interactions and *SQL Mode* for direct query execution. The system ensures secure access, real-time device status updates, and provides flexibility to manage schedules. Tools used include Oracle 10g XE Database, HTML, CSS, JavaScript for the frontend, and PHP for the backend connectivity.

2. Table of Contents

3. Introduction

3.1 Background

The rapid advancement of Internet of Things (IoT) technology has transformed traditional homes into smart environments. Smart homes integrate devices such as lights, air conditioning, and security systems to provide automated control, convenience, and energy efficiency. However, managing these interconnected devices requires a robust and well-structured database system to store, retrieve, and update data efficiently. This project addresses this need by designing and implementing a centralized database system that supports user authentication, room and device management, and scheduling of device operations. By leveraging Oracle Database, the project ensures data integrity, reliability, and scalability, while a web interface provides a user-friendly way to interact with the system.

3.2 Objective

The main objectives of this project are:

- To design and implement a relational database for managing a smart home system.
- To allow users to manage rooms, devices, and their access securely.
- To enable automated device scheduling and monitoring.
- To provide a web-based interface supporting both user-friendly interactions and direct SQL query execution.

3.3 Scope

Includes:

- User registration and login functionality. (Concept)
- Management of rooms and devices.
- Assigning user access to specific rooms.
- Scheduling device operations.
- Web interface.

Excludes:

- Direct hardware integration with real IoT devices.
- Development of a native mobile application.

---

4. Requirement Analysis

4.1 Functional Requirements

- Manage users
- Assign room access
- Add/update/delete devices
- Schedule device operations
- Query device statuses and schedules

4.2 Non-Functional Requirements

- Secure login
- Fast query response
- User-friendly UI
- Maintain referential integrity

---

5. Conceptual Model (ERD)

The conceptual model of the Smart Home Management System is designed around five core entities and their relationships:

- **Users**: Represents people who use the system. Each user can be an admin or a regular user.
- **Rooms**: Represents different rooms in the home.
- **Devices**: Devices are installed in rooms and can be controlled or scheduled.
- **Schedules**: Defines when and how devices operate automatically.
- **Room_Access**: Acts as a link between Users and Rooms to manage permissions.

Relationships:

- One **User** can have access to multiple **Rooms** through the Room_Access table (many-to-many relationship).
- One **Room** can have multiple **Devices** (one-to-many relationship).
- Each **Device** can have multiple **Schedules** (one-to-many relationship).

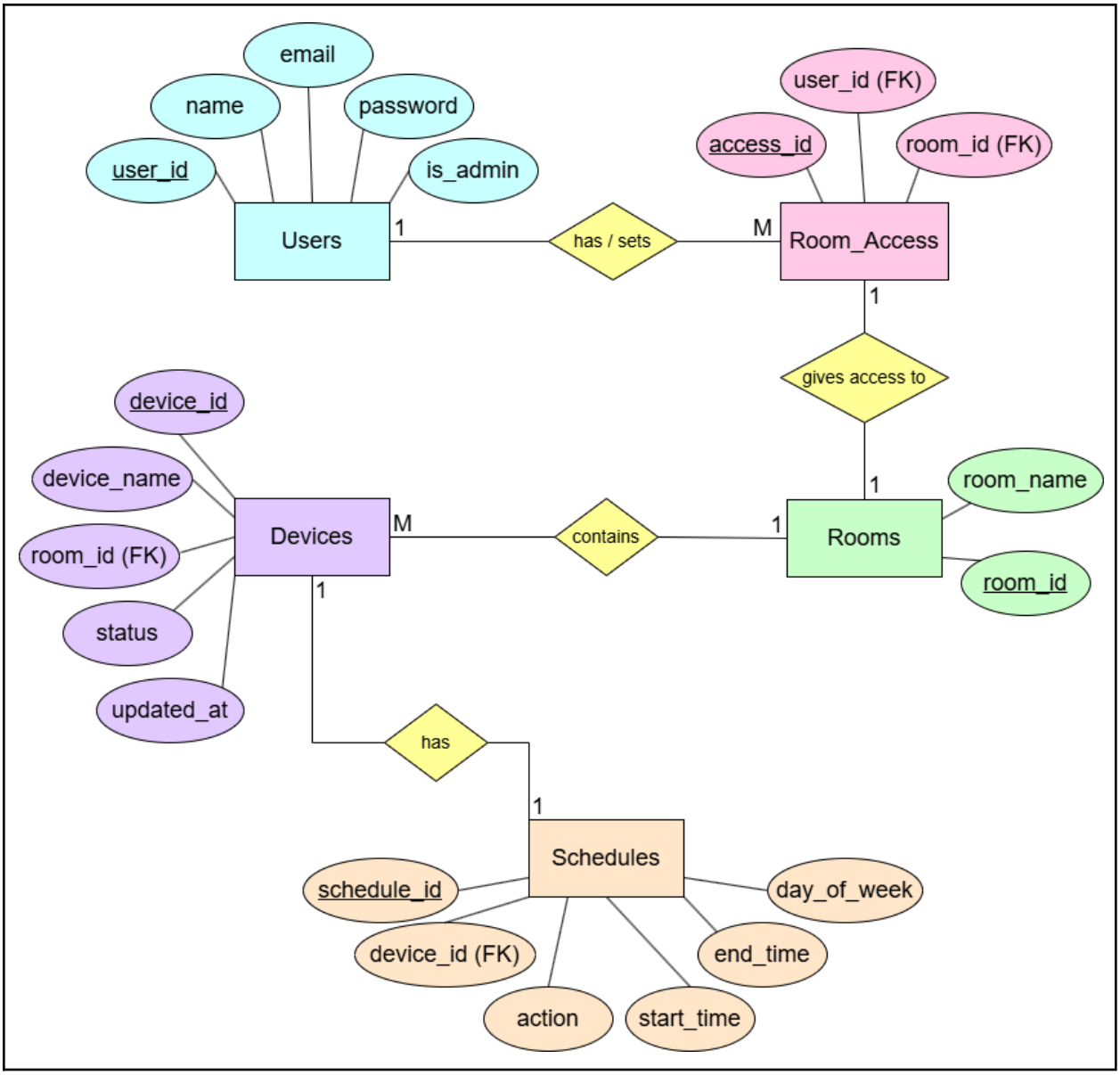


Fig: ER Diagram

---

6. Normalization Process

6.1 Step 1: Unnormalized data (UNF)

user_id	name	email	room_id	room_name	device_id	device_name	schedule_id	day_of_week	start_time	end_time	action
1	Alice Johnson	alice.j@example.com	1	Living Room	1	Ceiling Fan	1	Monday	2025-06-23 08:00:00	2025-06-23 10:00:00	1
1	Alice Johnson	alice.j@example.com	1	Living Room	2	Light Strip	2	Tuesday	2025-06-24 19:00:00	2025-06-24 20:00:00	0

6.2 Step 2: First Normal Form (1NF)

Remove repeating groups by creating separate tables:

- **Users**(user_id, name, email, password, is_admin)
- **Rooms**(room_id, room_name)
- **Devices**(device_id, device_name, room_id, status, updated_at)
- **Schedules**(schedule_id, device_id, day_of_week, start_time, end_time, action)
- **Room_Access**(access_id, user_id, room_id)

Each table now has a primary key, and data is atomic.

6.3 Step 3: Second Normal Form (2NF)

Since all tables already have single-field primary keys and no partial dependency, 2NF is achieved by:

- Ensuring that every non-key attribute is fully functionally dependent on the entire primary key.

For example, in Devices, device_name and status depend fully on device_id (the primary key).

6.4 Step 4: Third Normal Form (3NF)

Remove transitive dependencies:

- Attributes must depend only on the primary key.
- No transitive dependency like room_name in Devices table; instead, Devices table has only room_id as foreign key, and room_name stays in Rooms table.

After normalization, the final schema avoids redundancy and maintains data integrity while supporting complex queries and scalability.

---

7. Final Relational Schema

- **Users**(user_id, name, email, password, is_admin)
    - PK: user_id
    - email unique
- **Rooms**(room_id, room_name)
    - PK: room_id
- **Room_Access**(access_id, user_id, room_id)
    - PK: access_id
    - FK: user_id → Users
    - FK: room_id → Rooms
- **Devices**(device_id, device_name, room_id, status, updated_at)
    - PK: device_id
    - FK: room_id → Rooms
- **Schedules**(schedule_id, device_id, day_of_week, start_time, end_time, action)
    - PK: schedule_id
    - FK: device_id → Devices

---

8. Table Creation and Sample Data

8.1 Table Structures (DDL)

```
CREATE TABLE Users (
    user_id     NUMBER PRIMARY KEY,
    name        VARCHAR2(100) NOT NULL,
    email       VARCHAR2(100) UNIQUE NOT NULL,
    password    VARCHAR2(100) NOT NULL,
    is_admin    NUMBER(1) DEFAULT 0 CHECK (is_admin IN (0, 1))
);
CREATE TABLE Rooms (
    room_id     NUMBER PRIMARY KEY,
    room_name   VARCHAR2(100) NOT NULL
);

CREATE TABLE Room_Access (
    access_id   NUMBER PRIMARY KEY,
    user_id     NUMBER NOT NULL,
```

```
    room_id     NUMBER NOT NULL,
       FOREIGN  KEY  (user_id)  REFERENCES  Users(user_id)  ON  DELETE
CASCADE,
       FOREIGN  KEY  (room_id)  REFERENCES  Rooms(room_id)  ON  DELETE
CASCADE
);

CREATE TABLE Devices (
    device_id   NUMBER PRIMARY KEY,
    device_name VARCHAR2(50) NOT NULL,
    room_id     NUMBER NOT NULL,
    status      NUMBER(1) DEFAULT 0 CHECK (status IN (0, 1)),
    updated_at  TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
       FOREIGN  KEY  (room_id)  REFERENCES  Rooms(room_id)  ON  DELETE
CASCADE
);

CREATE TABLE Schedules (
    schedule_id NUMBER PRIMARY KEY,
    device_id   NUMBER NOT NULL,
    day_of_week VARCHAR2(10),
    start_time  TIMESTAMP NOT NULL,
    end_time    TIMESTAMP NOT NULL,
    action      NUMBER(1) NOT NULL CHECK (action IN (0, 1)),
     FOREIGN KEY (device_id) REFERENCES Devices(device_id) ON DELETE
CASCADE
);
```

8.2 Sample Data (DML)

```
-- 10 Users
INSERT INTO Users (user_id, name, email, password, is_admin) VALUES
(1, 'Alice Johnson', 'alice.j@example.com', 'alice123', 1);
INSERT INTO Users (user_id, name, email, password, is_admin) VALUES
(2, 'Bob Smith', 'bob.smith@example.com', 'bob456', 0);
INSERT INTO Users (user_id, name, email, password, is_admin) VALUES
(3, 'Clara Lee', 'clara.lee@example.com', 'clara789', 0);
INSERT INTO Users (user_id, name, email, password, is_admin) VALUES
(4, 'David Kim', 'david.k@example.com', 'david123', 0);
INSERT INTO Users (user_id, name, email, password, is_admin) VALUES
(5, 'Eva Mendes', 'eva.m@example.com', 'eva456', 1);
INSERT INTO Users (user_id, name, email, password, is_admin) VALUES
(6, 'Franklin Harris', 'franklin@gmail.com', 'frank123', 0);
INSERT INTO Users (user_id, name, email, password, is_admin) VALUES
(7, 'Grace Turner', 'grace@gmail.com', 'grace456', 0);
```

```
INSERT INTO Users (user_id, name, email, password, is_admin) VALUES
(8, 'Henry Adams', 'henry@gmail.com', 'henry789', 1);
INSERT INTO Users (user_id, name, email, password, is_admin) VALUES
(9, 'Isabella Moore', 'isabella@gmail.com', 'isa123', 0);
INSERT INTO Users (user_id, name, email, password, is_admin) VALUES
(10, 'Jack Wilson', 'jack@gmail.com', 'jack456', 0);
```

USER_ID	NAME	EMAIL	PASSWORD	IS_ADMIN
1	Alice Johnson	alice@gmail.com	alice123	1
2	Bob Smith	bob@gmail.com	bob456	0
3	Clara Lee	clara@gmail.com	clara789	0
4	David Kim	david@gmail.com	david123	0
5	Eva Mendes	eva@gmail.com	eva456	1
6	Franklin Harris	franklin@gmail.com	frank123	0
7	Grace Turner	grace@gmail.com	grace456	0
8	Henry Adams	henry@gmail.com	henry789	1
9	Isabella Moore	isabella@gmail.com	isa123	0
10	Jack Wilson	jack@gmail.com	jack456	0

Query: `SELECT * FROM Users`

```
-- 10 Rooms
INSERT INTO Rooms (room_id, room_name) VALUES (1, 'Living Room');
INSERT INTO Rooms (room_id, room_name) VALUES (2, 'Kitchen');
INSERT INTO Rooms (room_id, room_name) VALUES (3, 'Bedroom');
INSERT INTO Rooms (room_id, room_name) VALUES (4, 'Bathroom');
INSERT INTO Rooms (room_id, room_name) VALUES (5, 'Study Room');
INSERT INTO Rooms (room_id, room_name) VALUES (6, 'Garage');
INSERT INTO Rooms (room_id, room_name) VALUES (7, 'Dining Room');
INSERT INTO Rooms (room_id, room_name) VALUES (8, 'Office');
INSERT INTO Rooms (room_id, room_name) VALUES (9, 'Guest Room');
INSERT INTO Rooms (room_id, room_name) VALUES (10, 'Balcony');
```

ROOM_ID	ROOM_NAME
1	Living Room
2	Kitchen
3	Bedroom
4	Bathroom
5	Study Room
6	Garage
7	Dining Room
8	Office
9	Guest Room
10	Balcony

Query: `SELECT * FROM Rooms`

```
-- 10 Room_Access
INSERT INTO Room_Access (access_id, user_id, room_id) VALUES (1, 1,
1);
INSERT INTO Room_Access (access_id, user_id, room_id) VALUES (2, 1,
5);
INSERT INTO Room_Access (access_id, user_id, room_id) VALUES (3, 2,
2);
INSERT INTO Room_Access (access_id, user_id, room_id) VALUES (4, 2,
6);
INSERT INTO Room_Access (access_id, user_id, room_id) VALUES (5, 3,
3);
INSERT INTO Room_Access (access_id, user_id, room_id) VALUES (6, 3,
4);
INSERT INTO Room_Access (access_id, user_id, room_id) VALUES (7, 4,
1);
INSERT INTO Room_Access (access_id, user_id, room_id) VALUES (8, 4,
7);
INSERT INTO Room_Access (access_id, user_id, room_id) VALUES (9, 5,
8);
INSERT INTO Room_Access (access_id, user_id, room_id) VALUES (10,
5, 9);
```

ACCESS_ID	USER_ID	ROOM_ID
1	1	1
2	1	5
3	2	2
4	2	6
5	3	3
6	3	4
7	4	1
8	4	7
9	5	8
10	5	9

Query: `SELECT * FROM Room_Access`

```
-- 10 Devices
INSERT INTO Devices (device_id, device_name, room_id, status)
VALUES (1, 'Ceiling Fan', 1, 1);
INSERT INTO Devices (device_id, device_name, room_id, status)
VALUES (2, 'Light Strip', 1, 0);
INSERT INTO Devices (device_id, device_name, room_id, status)
VALUES (3, 'Air Conditioner', 2, 1);
INSERT INTO Devices (device_id, device_name, room_id, status)
VALUES (4, 'Refrigerator', 2, 1);
INSERT INTO Devices (device_id, device_name, room_id, status)
VALUES (5, 'Smart Oven', 2, 0);
INSERT INTO Devices (device_id, device_name, room_id, status)
VALUES (6, 'Bed Lamp', 3, 1);
INSERT INTO Devices (device_id, device_name, room_id, status)
VALUES (7, 'Night Light', 3, 0);
INSERT INTO Devices (device_id, device_name, room_id, status)
VALUES (8, 'Shower Light', 4, 1);
INSERT INTO Devices (device_id, device_name, room_id, status)
VALUES (9, 'Water Heater', 4, 1);
INSERT INTO Devices (device_id, device_name, room_id, status)
VALUES (10, 'Desk Lamp', 5, 1);
```

DEVICE_ID	DEVICE_NAME	ROOM_ID	STATUS	UPDATED_AT
1	Ceiling Fan	1	1	20-JUN-25 09.57.23.121000 AM
2	Light Strip	1	0	20-JUN-25 09.57.23.121000 AM
3	Air Conditioner	2	1	20-JUN-25 09.57.23.121000 AM
4	Refrigerator	2	1	20-JUN-25 09.57.23.121000 AM
5	Smart Oven	2	0	20-JUN-25 09.57.23.121000 AM
6	Bed Lamp	3	1	20-JUN-25 09.57.23.122000 AM
7	Night Light	3	0	20-JUN-25 09.57.23.122000 AM
8	Shower Light	4	1	20-JUN-25 09.57.23.122000 AM
9	Water Heater	4	1	20-JUN-25 09.57.23.122000 AM
10	Desk Lamp	5	1	20-JUN-25 09.57.23.122000 AM

Query: `SELECT * FROM Devices`

```
-- 10 Schedules
INSERT INTO Schedules (schedule_id, device_id, day_of_week,
start_time, end_time, action) VALUES
(1, 1, 'Monday',    TO_TIMESTAMP('2025-06-23 08:00:00', 'YYYY-MM-DD
HH24:MI:SS'), TO_TIMESTAMP('2025-06-23 10:00:00', 'YYYY-MM-DD
HH24:MI:SS'), 1),
(2, 2, 'Tuesday',   TO_TIMESTAMP('2025-06-24 19:00:00', 'YYYY-MM-DD
HH24:MI:SS'), TO_TIMESTAMP('2025-06-24 20:00:00', 'YYYY-MM-DD
HH24:MI:SS'), 0),
(3, 3, 'Wednesday', TO_TIMESTAMP('2025-06-25 12:00:00', 'YYYY-MM-DD
HH24:MI:SS'), TO_TIMESTAMP('2025-06-25 14:00:00', 'YYYY-MM-DD
HH24:MI:SS'), 1),
(4, 4, 'Thursday',  TO_TIMESTAMP('2025-06-26 09:00:00', 'YYYY-MM-DD
HH24:MI:SS'), TO_TIMESTAMP('2025-06-26 11:00:00', 'YYYY-MM-DD
HH24:MI:SS'), 1),
(5, 5, 'Friday',    TO_TIMESTAMP('2025-06-27 18:30:00', 'YYYY-MM-DD
HH24:MI:SS'), TO_TIMESTAMP('2025-06-27 20:00:00', 'YYYY-MM-DD
HH24:MI:SS'), 0),
(6, 6, 'Saturday',  TO_TIMESTAMP('2025-06-28 21:00:00', 'YYYY-MM-DD
HH24:MI:SS'), TO_TIMESTAMP('2025-06-28 22:30:00', 'YYYY-MM-DD
HH24:MI:SS'), 1),
(7, 7, 'Sunday',    TO_TIMESTAMP('2025-06-29 07:00:00', 'YYYY-MM-DD
HH24:MI:SS'), TO_TIMESTAMP('2025-06-29 08:00:00', 'YYYY-MM-DD
HH24:MI:SS'), 0),
(8, 8, 'Monday',    TO_TIMESTAMP('2025-06-30 06:30:00', 'YYYY-MM-DD
HH24:MI:SS'), TO_TIMESTAMP('2025-06-30 07:30:00', 'YYYY-MM-DD
HH24:MI:SS'), 1),
(9, 9, 'Tuesday',   TO_TIMESTAMP('2025-07-01 13:00:00', 'YYYY-MM-DD
HH24:MI:SS'), TO_TIMESTAMP('2025-07-01 14:30:00', 'YYYY-MM-DD
HH24:MI:SS'), 1),
(10, 10, 'Wednesday', TO_TIMESTAMP('2025-07-02 17:00:00',
```

```
'YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2025-07-02 18:00:00',
'YYYY-MM-DD HH24:MI:SS'), 0);
```

SCHEDULE_ID	DEVICE_ID	DAY_OF_WEEK	START_TIME	END_TIME	ACTION
3	3	Wednesday	25-JUN-25 12.00.00.000000 PM	25-JUN-25 02.00.00.000000 PM	1
4	4	Thursday	26-JUN-25 09.00.00.000000 AM	26-JUN-25 11.00.00.000000 AM	1
5	5	Friday	27-JUN-25 06.30.00.000000 PM	27-JUN-25 08.00.00.000000 PM	0
6	6	Saturday	28-JUN-25 09.00.00.000000 PM	28-JUN-25 10.30.00.000000 PM	1
7	7	Sunday	29-JUN-25 07.00.00.000000 AM	29-JUN-25 08.00.00.000000 AM	0
8	8	Monday	30-JUN-25 06.30.00.000000 AM	30-JUN-25 07.30.00.000000 AM	1
9	9	Tuesday	01-JUL-25 01.00.00.000000 PM	01-JUL-25 02.30.00.000000 PM	1
1	1	Monday	23-JUN-25 08.00.00.000000 AM	23-JUN-25 10.00.00.000000 AM	1
2	2	Tuesday	24-JUN-25 07.00.00.000000 PM	24-JUN-25 08.00.00.000000 PM	0
10	10	Wednesday	02-JUL-25 05.00.00.000000 PM	02-JUL-25 06.00.00.000000 PM	0

Query: `SELECT * FROM Schedules`

9. User Interface Design

The Smart Home Management System uses a web-based interface. This makes it easy for everyone to use. There are two modes: Normal Mode (Concept) is simple and lets regular users control devices and see information easily. SQL Mode is for administrators who need to run advanced database queries.

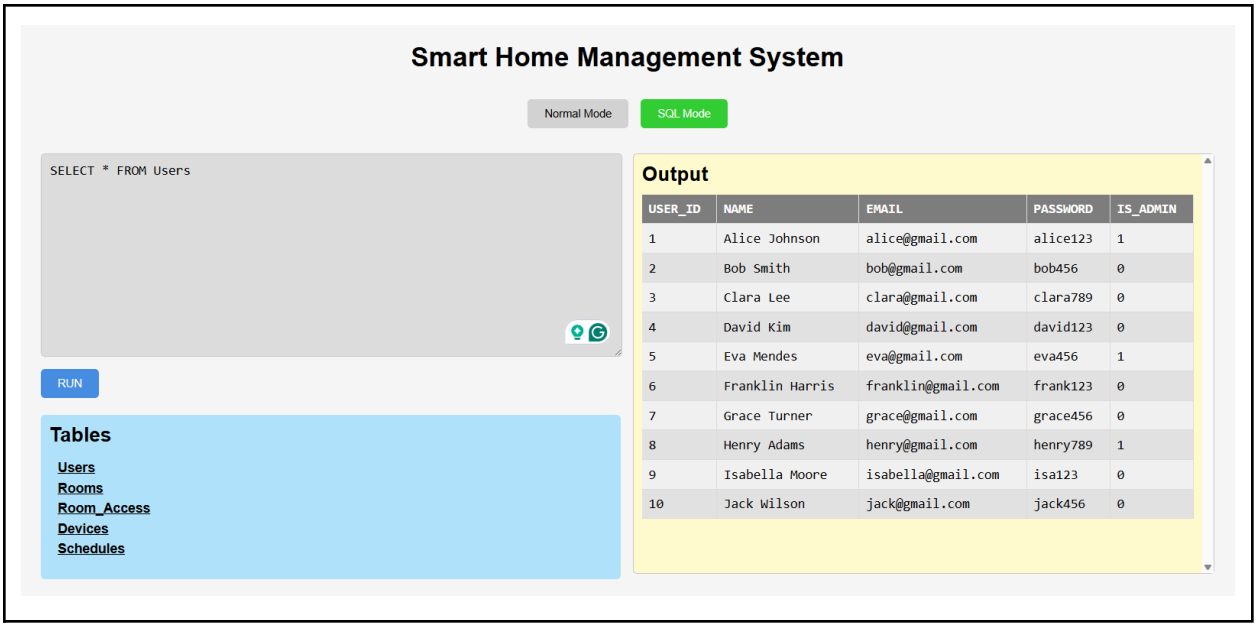


Fig: SQL Mode UI

10. DML Queries

10.1 Single Table Queries

Below are 10 single-table SELECT queries with explanations:

1. **SELECT * FROM Users;**

- Fetches all columns and records from the Users table, giving a full list of registered users.

Output

USER_ID	NAME	EMAIL	PASSWORD	IS_ADMIN
1	Alice Johnson	alice@gmail.com	alice123	1
2	Bob Smith	bob@gmail.com	bob456	0
3	Clara Lee	clara@gmail.com	clara789	0
4	David Kim	david@gmail.com	david123	0
5	Eva Mendes	eva@gmail.com	eva456	1
6	Franklin Harris	franklin@gmail.com	frank123	0
7	Grace Turner	grace@gmail.com	grace456	0
8	Henry Adams	henry@gmail.com	henry789	1
9	Isabella Moore	isabella@gmail.com	isa123	0
10	Jack Wilson	jack@gmail.com	jack456	0

2. **SELECT name, email FROM Users WHERE is_admin = 1;**

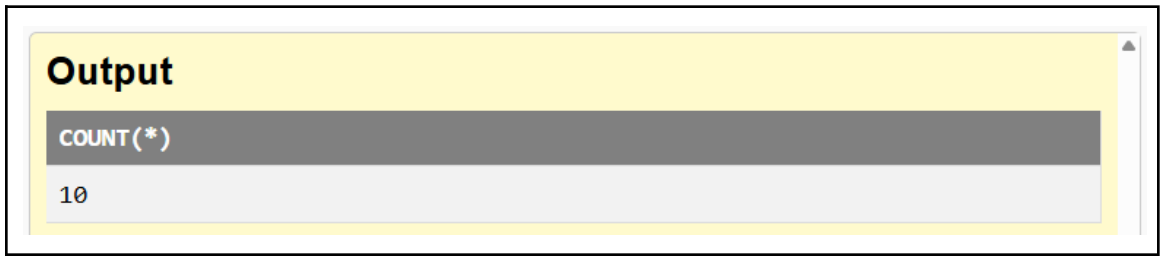
- Retrieves names and emails of users who are administrators (where is_admin equals 1). Useful for quickly identifying admin users.

Output

NAME	EMAIL
Alice Johnson	alice@gmail.com
Eva Mendes	eva@gmail.com
Henry Adams	henry@gmail.com

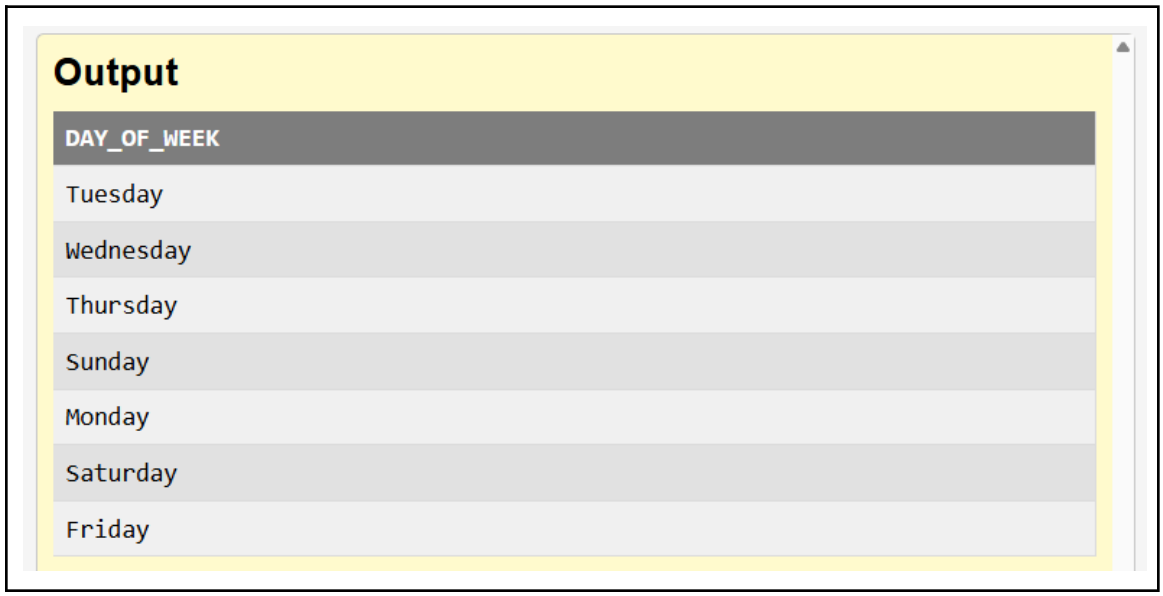
3. **SELECT COUNT(*) FROM Rooms**

- Counts the total number of rooms in the system to help with reporting or monitoring capacity.



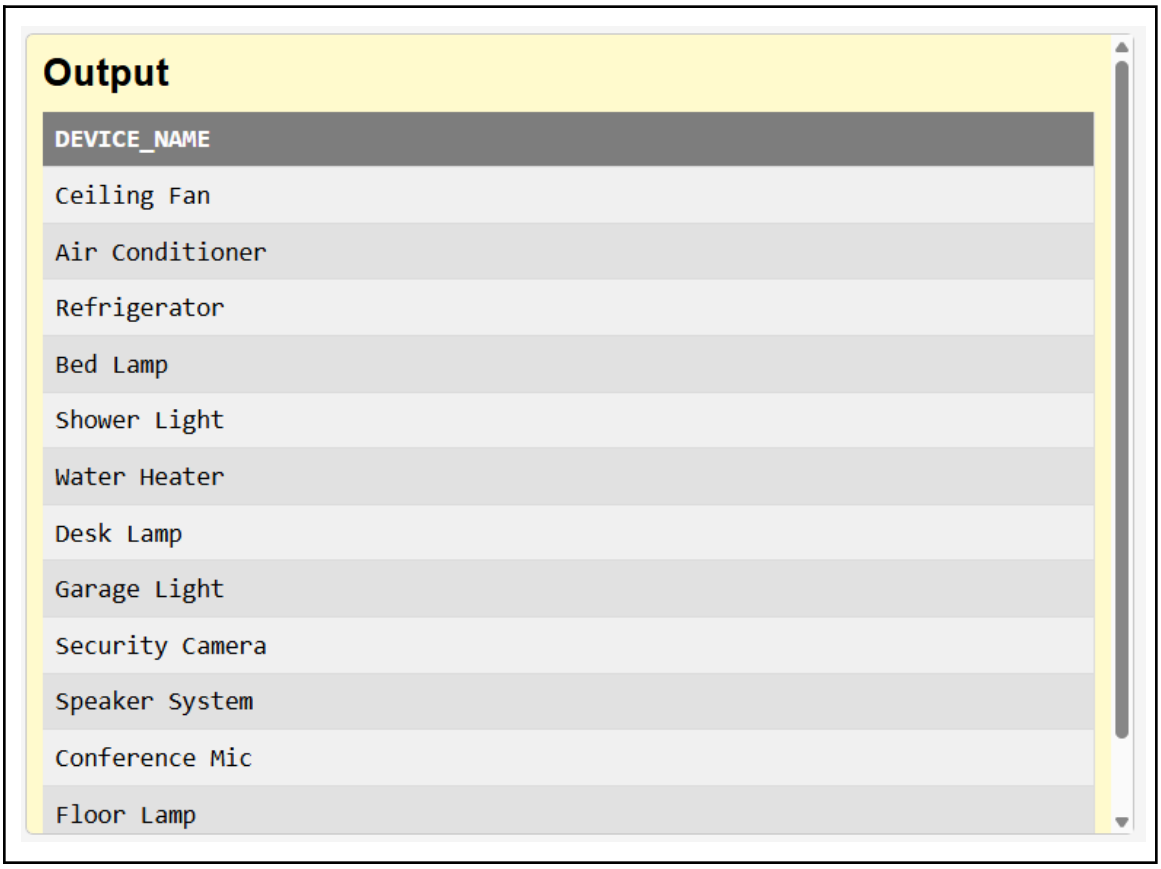
4. **SELECT DISTINCT day_of_week FROM Schedules**

- Returns unique days when any schedule exists. This shows which days devices are actively scheduled.



5. **SELECT device_name FROM Devices WHERE status = 1**

- Lists device names that are currently turned on (status equals 1). Helps monitor active devices.



Output

DEVICE_NAME
Ceiling Fan
Air Conditioner
Refrigerator
Bed Lamp
Shower Light
Water Heater
Desk Lamp
Garage Light
Security Camera
Speaker System
Conference Mic
Floor Lamp

6. **SELECT room_name FROM Rooms ORDER BY room_name ASC**

- Lists all room names in alphabetical order, making it easier to display or browse.

Output

ROOM_NAME
Balcony
Bathroom
Bedroom
Dining Room
Garage
Guest Room
Kitchen
Living Room
Office
Study Room

7. **SELECT schedule_id, start_time, end_time FROM Schedules WHERE action = 1**

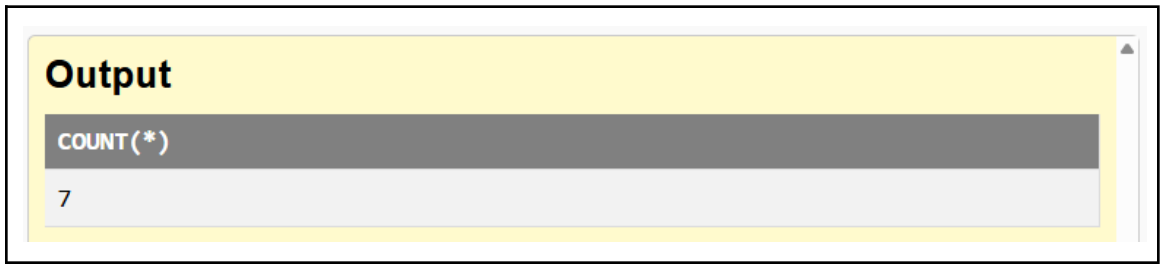
- Retrieves schedules where the action is to turn on devices, including start and end times.

Output

SCHEDULE_ID	START_TIME	END_TIME
3	25-JUN-25 12.00.00.000000 PM	25-JUN-25 02.00.00.000000 PM
4	26-JUN-25 09.00.00.000000 AM	26-JUN-25 11.00.00.000000 AM
6	28-JUN-25 09.00.00.000000 PM	28-JUN-25 10.30.00.000000 PM
8	30-JUN-25 06.30.00.000000 AM	30-JUN-25 07.30.00.000000 AM
9	01-JUL-25 01.00.00.000000 PM	01-JUL-25 02.30.00.000000 PM
1	23-JUN-25 08.00.00.000000 AM	23-JUN-25 10.00.00.000000 AM

8. **SELECT COUNT(*) FROM Users WHERE is_admin = 0**

- Counts how many users are non-admin (regular users). Useful for understanding user roles.

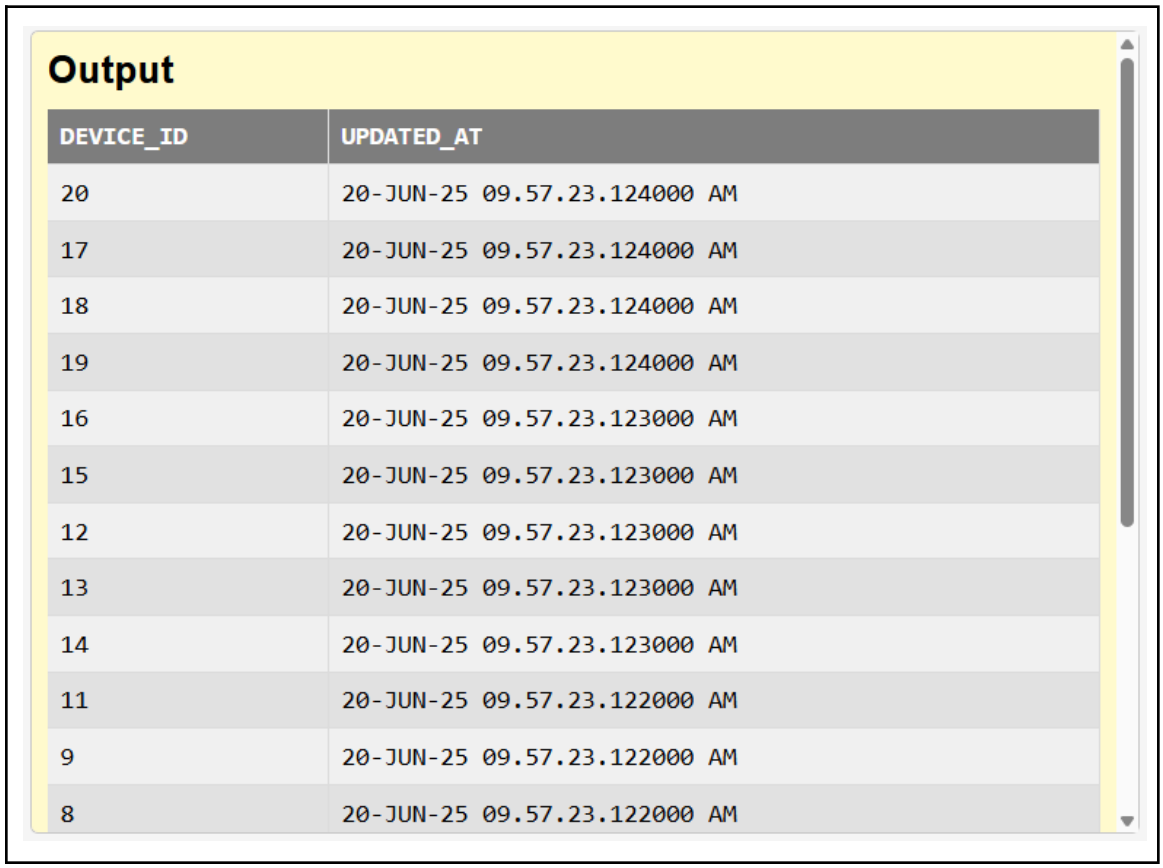


Output

COUNT(*)
7

9. **SELECT device_id, updated_at FROM Devices ORDER BY updated_at DESC**

- Shows device IDs with their most recent update times, sorted so the latest updates appear first.

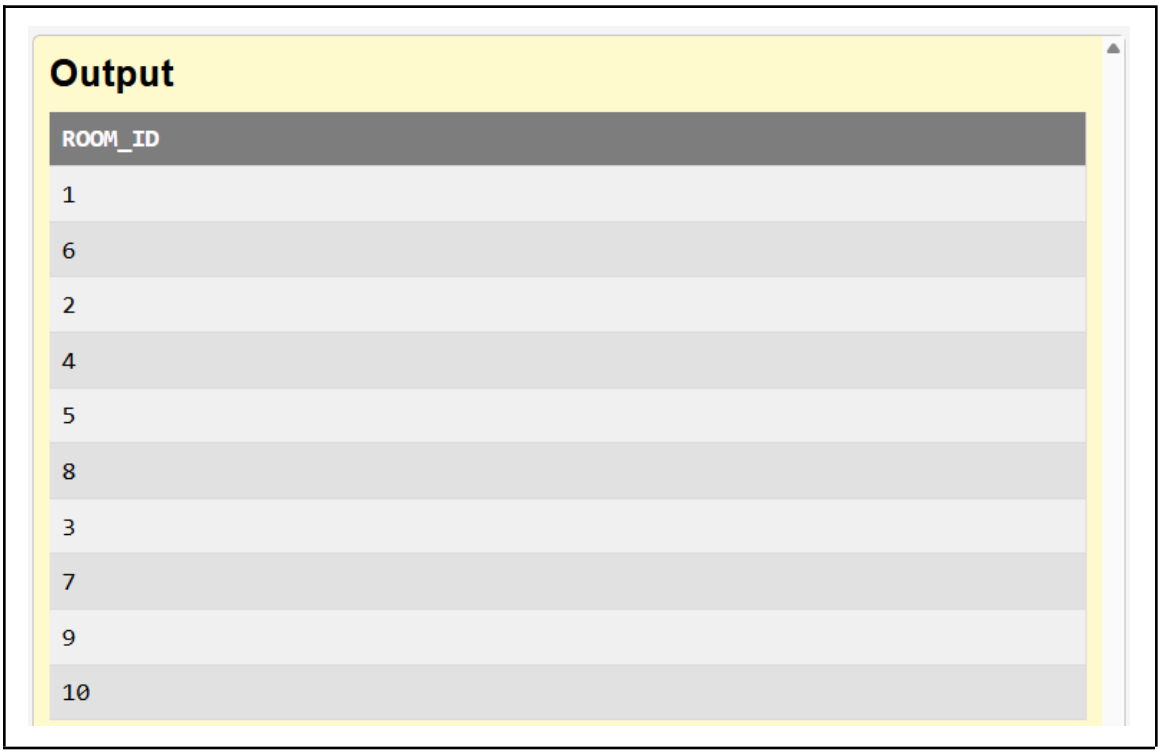


Output

DEVICE_ID	UPDATED_AT
20	20-JUN-25 09.57.23.124000 AM
17	20-JUN-25 09.57.23.124000 AM
18	20-JUN-25 09.57.23.124000 AM
19	20-JUN-25 09.57.23.124000 AM
16	20-JUN-25 09.57.23.123000 AM
15	20-JUN-25 09.57.23.123000 AM
12	20-JUN-25 09.57.23.123000 AM
13	20-JUN-25 09.57.23.123000 AM
14	20-JUN-25 09.57.23.123000 AM
11	20-JUN-25 09.57.23.122000 AM
9	20-JUN-25 09.57.23.122000 AM
8	20-JUN-25 09.57.23.122000 AM

10. SELECT room_id FROM Devices GROUP BY room_id

- Retrieves each unique room ID that has at least one device registered, showing device distribution across rooms.



Output

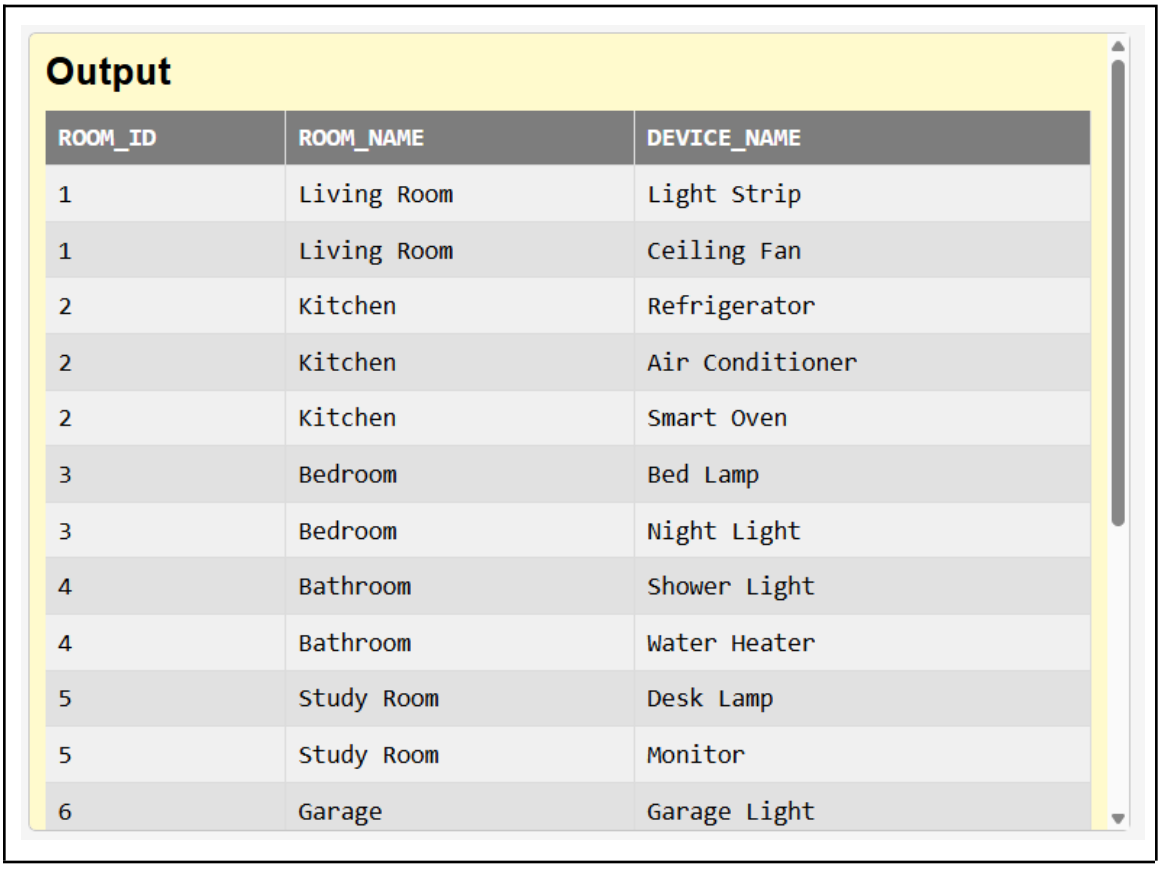
ROOM_ID
1
6
2
4
5
8
3
7
9
10

10.2 Multi-Table Queries

Below are 5 multi-table SELECT queries with explanations:

1. **SELECT Rooms.room_id, Rooms.room_name, Devices.device_name FROM Devices JOIN Rooms ON Devices.room_id = Rooms.room_id ORDER BY Rooms.room_id**

- Retrieves each device along with its room ID and room name, sorted by room ID. Useful to list all devices grouped by their corresponding rooms.



Output

ROOM_ID	ROOM_NAME	DEVICE_NAME
1	Living Room	Light Strip
1	Living Room	Ceiling Fan
2	Kitchen	Refrigerator
2	Kitchen	Air Conditioner
2	Kitchen	Smart Oven
3	Bedroom	Bed Lamp
3	Bedroom	Night Light
4	Bathroom	Shower Light
4	Bathroom	Water Heater
5	Study Room	Desk Lamp
5	Study Room	Monitor
6	Garage	Garage Light

2. **SELECT Users.name, Room_Access.room_id, Rooms.room_name FROM Users JOIN Room_Access ON Users.user_id = Room_Access.user_id JOIN Rooms ON Rooms.room_id = Room_Access.room_id ORDER BY Users.user_id**

- Shows which user has access to which room by combining data from Users, Room_Access, and Rooms tables. Sorted by user ID for easier reading.

Output

NAME	ROOM_ID	ROOM_NAME
Alice Johnson	5	Study Room
Alice Johnson	1	Living Room
Alice Johnson	4	Bathroom
Bob Smith	2	Kitchen
Bob Smith	6	Garage
Bob Smith	10	Balcony
Clara Lee	4	Bathroom
Clara Lee	6	Garage
Clara Lee	3	Bedroom
David Kim	7	Dining Room
David Kim	1	Living Room
David Kim	2	Kitchen

3. **SELECT Users.user_id, Users.name, COUNT(Room_Access.room_id) AS rooms_access_count FROM Users JOIN Room_Access ON Users.user_id = Room_Access.user_id GROUP BY Users.user_id, Users.name ORDER BY Users.user_id**

- Counts how many rooms each user has access to. Helps identify users with broader permissions.

Output

USER_ID	NAME	ROOMS_ACCESS_COUNT
1	Alice Johnson	3
2	Bob Smith	3
3	Clara Lee	3
4	David Kim	3
5	Eva Mendes	3
6	Franklin Harris	2
7	Grace Turner	2
8	Henry Adams	2
9	Isabella Moore	2
10	Jack Wilson	2

4. **SELECT Rooms.room_name, COUNT(Devices.device_id) AS number_of_devices FROM Rooms JOIN Devices ON Rooms.room_id = Devices.room_id GROUP BY Rooms.room_name ORDER BY Rooms.room_name**

- Finds how many devices are present in each room. Useful for inventory or capacity checks.

Output

ROOM_NAME	NUMBER_OF_DEVICES
Balcony	1
Bathroom	2
Bedroom	2
Dining Room	2
Garage	2
Guest Room	2
Kitchen	3
Living Room	2
Office	2
Study Room	2

5. **SELECT Devices.device_name, Schedules.start_time, Schedules.end_time FROM Devices JOIN Schedules ON Devices.device_id = Schedules.device_id ORDER BY Schedules.start_time**

- Lists each device along with its scheduled start and end times. Helps see when devices are set to turn on or off across the week.

Output

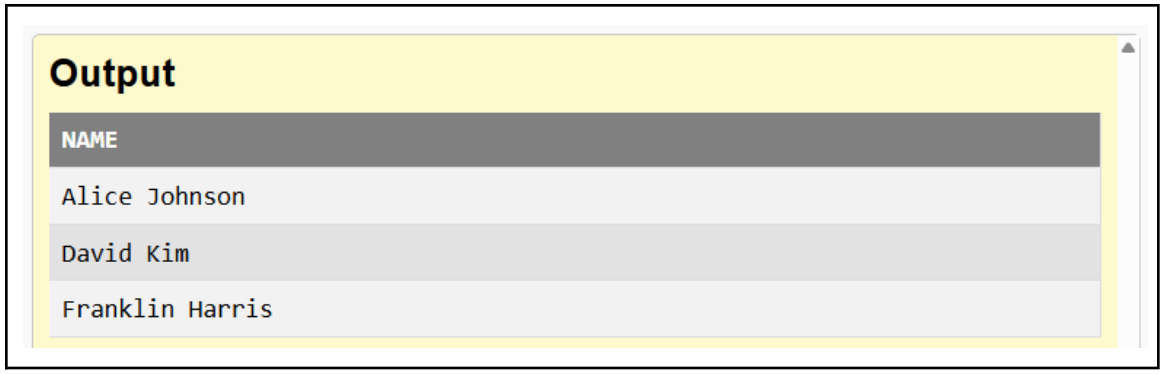
DEVICE_NAME	START_TIME	END_TIME
Ceiling Fan	23-JUN-25 08.00.00.000000 AM	23-JUN-25 10.00.00.000000 AM
Light Strip	24-JUN-25 07.00.00.000000 PM	24-JUN-25 08.00.00.000000 PM
Air Conditioner	25-JUN-25 12.00.00.000000 PM	25-JUN-25 02.00.00.000000 PM
Refrigerator	26-JUN-25 09.00.00.000000 AM	26-JUN-25 11.00.00.000000 AM
Smart Oven	27-JUN-25 06.30.00.000000 PM	27-JUN-25 08.00.00.000000 PM
Bed Lamp	28-JUN-25 09.00.00.000000 PM	28-JUN-25 10.30.00.000000 PM
Night Light	29-JUN-25 07.00.00.000000 AM	29-JUN-25 08.00.00.000000 AM
Shower Light	30-JUN-25 06.30.00.000000 AM	30-JUN-25 07.30.00.000000 AM

10.3 Subqueries

Below are 5 subqueries of different types with explanations:

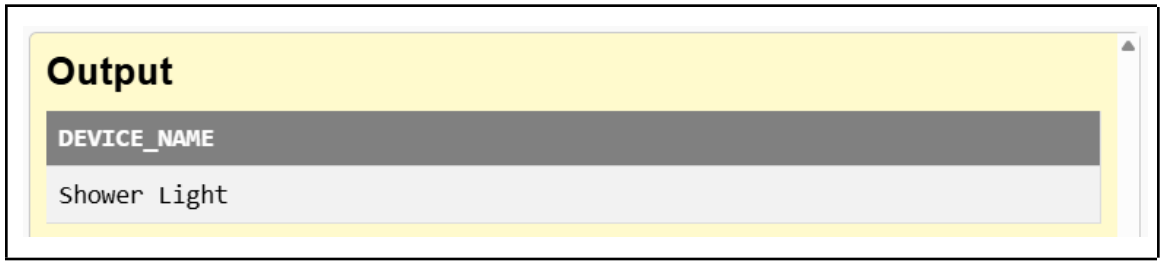
1. **SELECT name FROM Users WHERE user_id IN (SELECT user_id FROM Room_Access WHERE room_id = 1)**

- Finds the names of users who have access to room with ID 1. The inner subquery returns relevant user IDs.



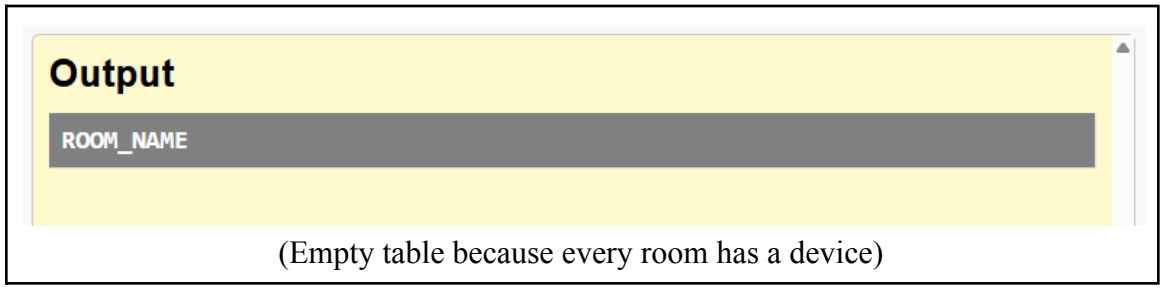
2. **SELECT device_name FROM Devices WHERE device_id = (SELECT device_id FROM Schedules WHERE day_of_week = 'Monday' AND ROWNUM = 1)**

- Retrieves the name of the first device scheduled on Monday. The subquery selects a device_id matching the condition.



3. **SELECT room_name FROM Rooms WHERE room_id NOT IN (SELECT DISTINCT room_id FROM Devices)**

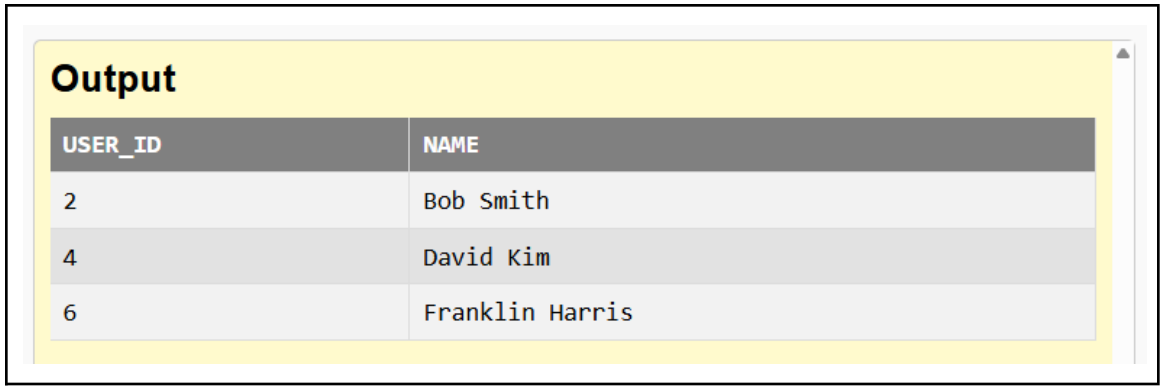
- Lists rooms that currently have no devices assigned. Helps identify unused rooms.



(Empty table because every room has a device)

4. **SELECT user_id, name FROM Users WHERE EXISTS (SELECT 1 FROM Room_Access WHERE Users.user_id = Room_Access.user_id AND room_id = 2)**

- Returns users who have access to room with ID 2. The EXISTS subquery checks if such access exists.

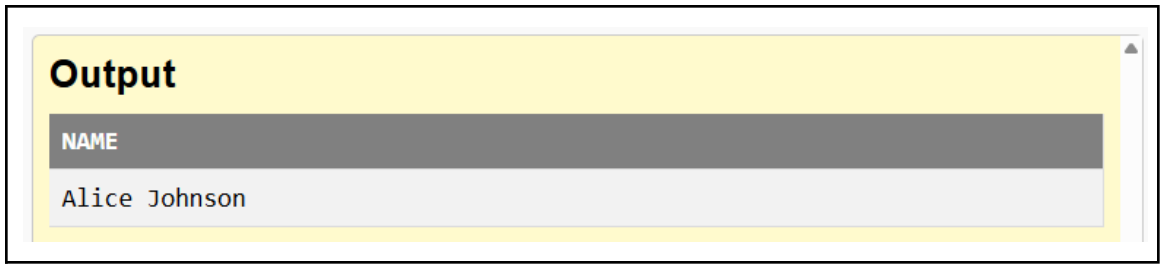


Output

USER_ID	NAME
2	Bob Smith
4	David Kim
6	Franklin Harris

5. **SELECT name FROM Users WHERE user_id = (SELECT user_id FROM (SELECT user_id FROM Room_Access GROUP BY user_id ORDER BY COUNT(room_id) DESC) WHERE ROWNUM = 1)**

- Finds the name of the user who has access to the most rooms. The inner subquery groups by user_id and sorts by count.



Output

NAME
Alice Johnson

---

11. Challenges Faced

- Installing OCI8 for Oracle to be used in PHP
- Oracle constraints syntax
- Keeping UI and DB in sync
- Ensuring referential integrity with foreign key constraints
- Debugging complex queries, especially joins and subqueries
- Handling normalization to avoid data redundancy

---

12. Conclusion

This Smart Home Management System project demonstrates the complete cycle of designing and implementing a relational database system, starting from conceptual modeling and normalization to actual data manipulation and querying. By building tables for users, rooms, devices, schedules, and access rights, it offers a practical way to manage complex relationships in a smart home environment. The project also integrates a web-based interface that supports both normal usage and direct SQL query execution, making the system versatile for both end users and administrators. Through this project, key database concepts, such as ER modeling, referential integrity, normalization up to 3NF, and advanced querying (joins and subqueries), were applied to solve real-world problems. Overall, this project has enhanced the understanding of how databases power modern applications and highlighted the importance of structured design, usability, and data consistency in building reliable smart systems.

---

13. References

[1] Silberschatz, Korth, Sudarshan. Database System Concepts (6th ed.)
[2] Oracle Documentation: https://docs.oracle.com/en/database/
[3] "How to Design a Database for Smart Home Systems," GeeksforGeeks, May 8, 2024. [Online]. Available: https://www.geeksforgeeks.org/dbms/how-to-design-a-database-for-smart-home-systems/
[4] "The Smart Home Data Model," Vertabelo Data Model Blog, Jan. 24, 2019. [Online]. Available: https://vertabelo.com/blog/the-smart-home-data-model/
[5] "ER Diagram for Smart Home Automation Systems," GeeksforGeeks, Feb. 27, 2024. [Online]. Available: https://www.geeksforgeeks.org/sql/how-to-design-er-diagrams-for-smart-home-automation-systems/

---

14. Appendix

14.1 Setup Instructions

1. Clone the Project (https://github.com/nusratjahanchaity/Smart-Home-Management-System.git)
2. Install XAMPP (win32-7.3.2-0-VC15)
3. Install Oracle 10g XE
4. Enable Oracle Support in PHP by uncommenting the following line extension=oci8.dll in XAMPP PHP Config file (you might need to download OCI8)
5. Create Database Tables and Insert Data using the given DDL and DML queries
6. Start Apache on XAMPP and run the project (http://localhost/Smart-Home-Management-System)
