**SAGARAMITRA**

*Report submitted as a part of Activity Based Assessment for*

**Database Management System**

**[MDBML105]**

of

first semester Master of Computer Applications

Submitted by

# Basanagoud Appasaheb Patil [1AT24MC010]

# Bsavakiran Digge [1AT24MC011]

# Ganapathi Gowda [1AT24MC025]

# Manoranjan Behera [1AT24MC044]

Under the guidance of

Dr. Gomathy Prathima E

Academic Year: 2024 – 2025

**EVALUATION SHEET – ACTIVITY BASED ASSESSMENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Evaluation criteria | 1AT24MC0–  Name1 | 1AT24MC0–  Name2 | 1AT24MC0–  Name3 | 1AT24MC0–  Name4 |
| ER Model |  |  |  |  |
| ER – to - Relation Mapping |  |  |  |  |
| Schema diagram and Table design |  |  |  |  |
| Normalization |  |  |  |  |
| Report | . |  |  |  |
| Demonstration and Presentation |  |  |  |  |
| Q & A |  |  |  |  |
| Infosys Springboard Certificate |  |  |  |  |
| Total |  |  |  |  |
| Signature of student |  |  |  |  |

Course Coordinator HOD

1. **INTRODUCTION**

**Database-Driven Boat Booking System for Honnavar Tourism(SagaraMitra)** Honnavar is a well-known boating destination due to its strategic location where the **Sharavathi River meets the Arabian Sea**, creating a breathtaking network of **backwaters, islands, and mangrove forests**. This unique geographical feature makes Honnavar a paradise for boating enthusiasts, nature lovers, and tourists seeking a peaceful and scenic water adventure.

**Honnavar Boating Experience in SagaraMitra Boat Booking Application**

To enhance the boating experience in Honnavar, the **SagaraMitra Boat Booking Application** provides an easy-to-use platform for **booking private and public boats** while ensuring smooth slot management and availability tracking.

* **Backwater Formation & Scenic Beauty**  
  The SagaraMitra application allows users to book boats for exploring **Honnavar's backwaters**, surrounded by lush green landscapes and peaceful waters, ensuring a memorable journey through nature’s beauty.
* **Biodiversity & Mangrove Exploration**  
  Visitors booking through the platform can enjoy **boat rides through mangrove forests**, witnessing diverse flora and fauna while experiencing the rich biodiversity of Honnavar. The integration of **private and public boat booking** ensures a seamless experience for adventure seekers.
* **Slot-Based Booking & Availability Tracking**  
  The app efficiently manages **boat slots**, ensuring that users can select their preferred time slots without overbooking. The system automatically updates **boat availability**, reflecting real-time data on open slots for public and private boats.
* **Mangrove Boardwalk & Eco-Tourism**  
  Honnavar is home to the famous **Mangrove Boardwalk (Kandla Vana)**, which attracts eco-tourists. Through **SagaraMitra**, visitors can book boats specifically designed for **eco-tourism and guided mangrove exploration**, making the experience even more enriching.
* **User-Friendly Booking System**  
  The SagaraMitra Boat Booking Application offers an **efficient, real-time booking system** where tourists can check available boats, book their preferred slots, and receive confirmation instantly. Additionally, boat owners can manage their fleet, update slot availability, and track bookings conveniently.

By integrating the natural beauty and tourism significance of Honnavar with an advanced **database-driven** boat booking system, SagaraMitra ensures a seamless, well-organized, and enjoyable boating experience for visitors while promoting eco-friendly and sustainable tourism in the region.

1. **E R MODEL**

**Entity-Relationship Model for Boat Owners, Users, Bookings, and Availability**

**1. Users**

* **Attributes:**
  + user\_id (PK) – Unique identifier for a user
  + user\_name – Name of the user
  + user\_age – Age of the user
  + user\_phone – Unique phone number of the user
  + user\_password – Password for login
* **Relationships:**
  + A User can make multiple Bookings (Public or Private) → (1:M)
  + Participation: Total (Every user can book at least one boat)

**2. Boat\_Owners**

* **Attributes:**
  + owner\_id (PK) – Unique identifier for a boat owner
  + owner\_name – Name of the owner
  + owner\_phone – Unique phone number of the owner
* **Relationships:**
  + A **Boat Owner** can own multiple **Boats (Public and Private)** → (1:M)
  + Participation: **Total** (Every boat must have an owner)

**3. Public\_Boats**

* **Attributes:**
  + boat\_id (PK) – Unique identifier for the boat
  + boat\_name – Name of the boat
  + capacity – Maximum number of passengers
  + pickup\_drop\_location – Boat location
  + owner\_id (FK) – References Boat\_Owners(owner\_id)
* **Relationships:**
  + A **Public Boat** can have multiple **Bookings** → (1:M)
  + A **Public Boat** is owned by **one Boat Owner** → (M:1)
  + Participation: **Total** (Every boat must belong to an owner)

**4. Private\_Boats**

* **Attributes:**
  + boat\_id (PK) – Unique identifier for the private boat
  + boat\_name – Name of the boat
  + pickup\_drop\_location – Location details
  + owner\_id (FK) – References Boat\_Owners(owner\_id)
* **Relationships:**
  + A **Private Boat** is owned by **one Boat Owner** → (M:1)
  + A **Private Boat Slot** can be booked by only **one User** → **(**1:1**)**
  + Participation: **Total** (Every private boat must belong to an owner)

**5. Public\_Boat\_Slots**

* **Attributes:**
  + slot\_id (PK) – Unique identifier for the slot
  + slot\_time – Time slot description
* **Relationships:**
  + A **Slot** can be associated with multiple **Bookings** → (1:M)
  + Participation: **Partial** (Not every slot will always be booked)

**6. Private\_Boat\_Slots**

* **Attributes:**
  + slot\_id (PK) – Unique identifier for the slot
  + slot\_time – Time slot description
* **Relationships:**
* A **Slot** can be associated with only one **Booking** → (1:1)
* Participation: **Partial** (Not every slot will always be booked)

**7. Public\_Boat\_Booking**

* **Attributes:**
  + booking\_id (PK) – Unique identifier for the booking
  + user\_id (FK) – References Users(user\_id)
  + boat\_id (FK) – References Public\_Boats(boat\_id)
  + booking\_date – Date of booking
  + slot\_id (FK) – References Public\_Boat\_Slots(slot\_id)
  + seats\_booked – Number of seats booked
* **Relationships:**
  + A **User** can book multiple **Public Boats** → (1:M)
  + A **Public Boat** can be booked by multiple **Users** → (M:1)
  + Participation: **Total** (Every booking must be linked to a user, boat, and slot)

**8. Private\_Boat\_Booking**

* **Attributes:**
  + booking\_id (PK) – Unique identifier for the booking
  + user\_id (FK) – References Users(user\_id)
  + boat\_id (FK) – References Private\_Boats(boat\_id)
  + booking\_date – Date of booking
  + slot\_id (FK) – References Private\_Boat\_Slots(slot\_id)
* **Relationships:**
  + A **User** can book multiple **Private Boats** → (1:M)
  + A **Private Boat** **slot** can be booked by only one **User** → (1:1)
  + Participation: **Total** (Every booking must be linked to a user, boat, and slot)

**9. Public\_Boat\_Status**

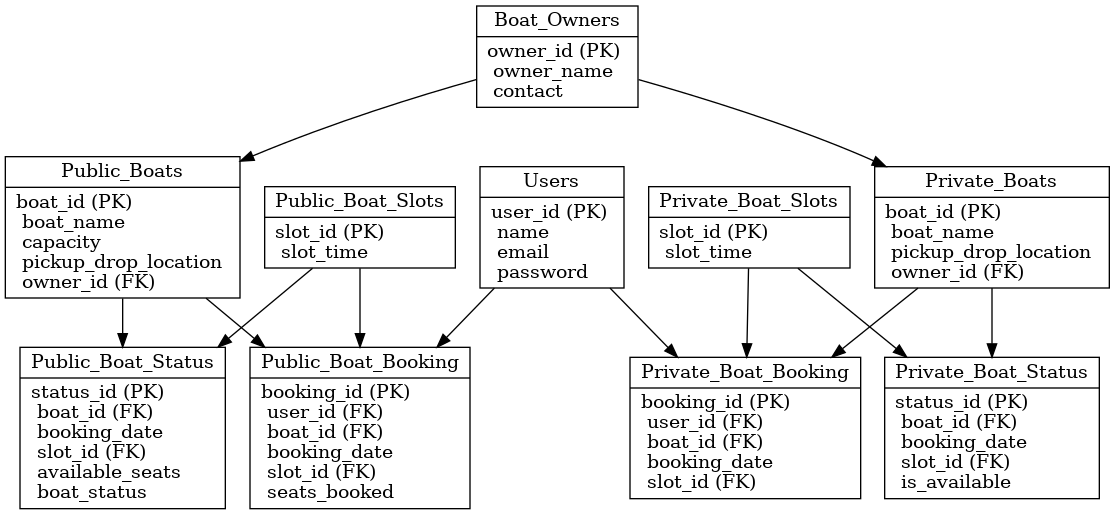
* **Attributes:**
  + status\_id (PK) – Unique identifier for the status entry
  + boat\_id (FK) – References Public\_Boats(boat\_id)
  + booking\_date – Date of booking
  + slot\_id (FK) – References Public\_Boat\_Slots(slot\_id)
  + available\_seats – Number of seats available
  + boat\_status – Availability status (Available/Full)
* **Relationships:**
  + A **Public Boat** has a **Status** record for each slot and date → (1:M)
  + A **Slot** is associated with multiple **Boat Status records** → (1:M)
  + Participation: **Total** (Every public boat must have status tracking)

**10. Private\_Boat\_Status**

* **Attributes:**
  + status\_id (PK) – Unique identifier for the status entry
  + boat\_id (FK) – References Private\_Boats(boat\_id)
  + booking\_date – Date of booking
  + slot\_id (FK) – References Private\_Boat\_Slots(slot\_id)
  + is\_available – Boolean flag indicating availability
* **Relationships:**
  + A **Private Boat** has a **Status** record for each slot and date → (1:M)
  + A **Slot** is associated with single **Boat Status records** → (1:1)
  + Participation: **Total** (Every private boat must have status tracking)

1. **ER MODEL TO RELATIONAL MODEL MAPPING**
2. **DATABASE DESIGN**

* **Schema diagram**



**Entity Creation and Database Structure**

CREATE TABLE **Users** (

user\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_name VARCHAR(100) NOT NULL,

user\_age INT NOT NULL,

user\_phone VARCHAR(15) UNIQUE NOT NULL

user\_password varchar(10)

);

CREATE TABLE **Boat\_Owners** (

owner\_id INT AUTO\_INCREMENT PRIMARY KEY,

owner\_name VARCHAR(100) NOT NULL,

owner\_phone VARCHAR(15) UNIQUE NOT NULL

);

CREATE TABLE **Public\_Boats** (

boat\_id INT AUTO\_INCREMENT PRIMARY KEY,

boat\_name VARCHAR(100) UNIQUE NOT NULL,

capacity INT NOT NULL DEFAULT 8,

pickup\_drop\_location VARCHAR(255) NOT NULL,

owner\_id INT NOT NULL,

FOREIGN KEY (owner\_id) REFERENCES Boat\_Owners(owner\_id)

);

CREATE TABLE **Private\_Boats** (

boat\_id INT AUTO\_INCREMENT PRIMARY KEY,

boat\_name VARCHAR(100) UNIQUE NOT NULL,

pickup\_drop\_location VARCHAR(255) NOT NULL,

owner\_id INT NOT NULL,

FOREIGN KEY (owner\_id) REFERENCES Boat\_Owners(owner\_id)

);

CREATE TABLE **Public\_Boat\_Slots** (

slot\_id INT AUTO\_INCREMENT PRIMARY KEY,

slot\_time VARCHAR(50) NOT NULL

);

INSERT INTO **Public\_Boat\_Slots** (slot\_time) VALUES

('Morning (6 AM - 9 AM)'),

('Midday (9 AM - 12 PM)'),

('Afternoon (12 PM - 3 PM)'),

('Evening (3 PM - 6 PM)');

CREATE TABLE **Private\_Boat\_Slots** (

slot\_id INT AUTO\_INCREMENT PRIMARY KEY,

slot\_time VARCHAR(50) NOT NULL

);

INSERT INTO **Private\_Boat\_Slots** (slot\_time) VALUES

('Morning (6 AM - 10 AM)'),

('Afternoon (10 AM - 2 PM)'),

('Evening (2 PM - 6 PM)');

CREATE TABLE **Public\_Boat\_Booking** (

booking\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

boat\_id INT NOT NULL,

booking\_date DATE NOT NULL,

slot\_id INT NOT NULL,

seats\_booked INT NOT NULL,

FOREIGN KEY (user\_id) REFERENCES Users(user\_id,

FOREIGN KEY (boat\_id) REFERENCES Public\_Boats(boat\_id),

FOREIGN KEY (slot\_id) REFERENCES Public\_Boat\_Slots(slot\_id)

);

CREATE TABLE **Private\_Boat\_Booking** (

booking\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

boat\_id INT NOT NULL,

booking\_date DATE NOT NULL,

slot\_id INT NOT NULL,

FOREIGN KEY (user\_id) REFERENCES Users(user\_id),

FOREIGN KEY (boat\_id) REFERENCES Private\_Boats(boat\_id),

FOREIGN KEY (slot\_id) REFERENCES Private\_Boat\_Slots(slot\_id)

);

CREATE TABLE **Public\_Boat\_Status** (

status\_id INT AUTO\_INCREMENT PRIMARY KEY,

boat\_id INT NOT NULL,

booking\_date DATE NOT NULL,

slot\_id INT NOT NULL,

available\_seats INT NOT NULL,

boat\_status ENUM('Available', 'Full') NOT NULL,

FOREIGN KEY (boat\_id) REFERENCES Public\_Boats(boat\_id),

FOREIGN KEY (slot\_id) REFERENCES Public\_Boat\_Slots(slot\_id)

);

CREATE TABLE **Private\_Boat\_Status** (

status\_id INT AUTO\_INCREMENT PRIMARY KEY,

boat\_id INT NOT NULL,

booking\_date DATE NOT NULL,

slot\_id INT NOT NULL,

is\_available BOOLEAN NOT NULL DEFAULT TRUE,

FOREIGN KEY (boat\_id) REFERENCES Private\_Boats(boat\_id),

FOREIGN KEY (slot\_id) REFERENCES Private\_Boat\_Slots(slot\_id)

);

**Stored Procedure for User Registration**

The RegisterUser procedure checks if a phone number already exists in the Users table. If found, it returns "User already exists!"; otherwise, it inserts the new user and returns "Registration successful". It ensures uniqueness using the phone number as a key.

DELIMITER //

CREATE PROCEDURE RegisterUser(

IN p\_user\_name VARCHAR(100),

IN p\_user\_age INT,

IN p\_user\_phone VARCHAR(15),

IN p\_user\_password VARCHAR(255)

)

BEGIN

DECLARE user\_count INT;

-- Check if the phone number already exists

SELECT COUNT(\*) INTO user\_count FROM Users WHERE user\_phone = p\_user\_phone;

IF user\_count > 0 THEN

SELECT 'User already exists!' AS MESSAGE;

ELSE

-- Insert new user with hashed password

INSERT INTO Users (user\_name, user\_age, user\_phone, user\_password)

VALUES (p\_user\_name, p\_user\_age, p\_user\_phone, p\_user\_password);

SELECT 'Registration successful' AS MESSAGE;

END IF;

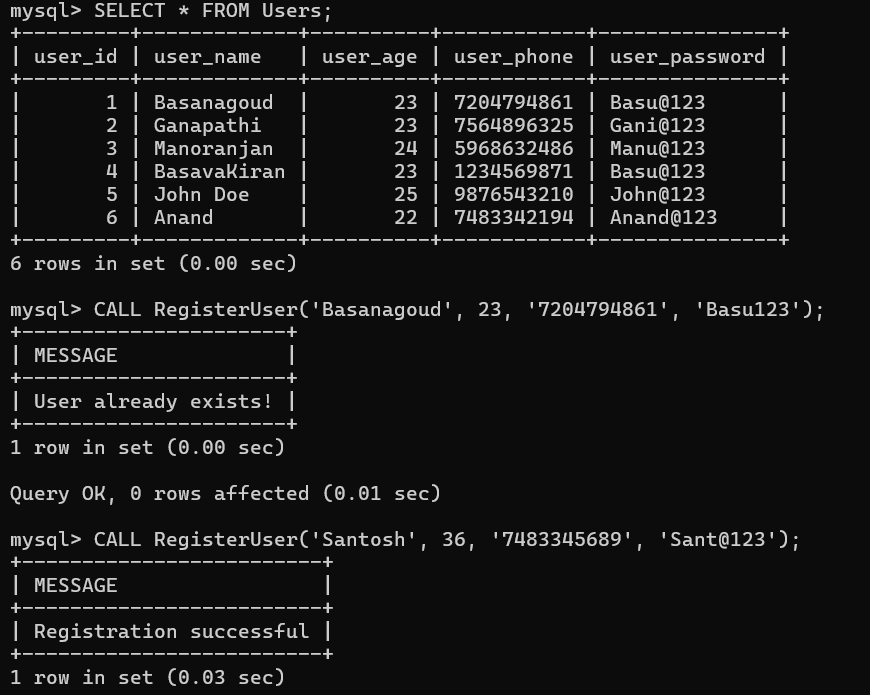
END //

DELIMITER ;

CALL RegisterUser('Basanagoud', 23, '7204794861', 'Basu123');

The following output is generated when a user attempts to register through the

**RegisterUser procedure:**



**User Login Procedure**

This procedure verifies user login by checking the provided phone number and password in the database. If a matching record is found, it returns "Login successful"; otherwise, it returns "Invalid phone number or password." It ensures only registered users can access the system. The procedure is executed using the CALL UserLogin statement.

DELIMITER //

CREATE PROCEDURE UserLogin(

IN p\_user\_phone VARCHAR(15),

IN p\_user\_password VARCHAR(255)

)

BEGIN

DECLARE user\_count INT;

-- Check if the user exists with the correct password

SELECT COUNT(\*) INTO user\_count

FROM Users

WHERE user\_phone = p\_user\_phone

AND user\_password = p\_user\_password;

IF user\_count > 0 THEN

SELECT 'Login successful' AS Message;

ELSE

SELECT 'Invalid phone number or password' AS Message;

END IF;

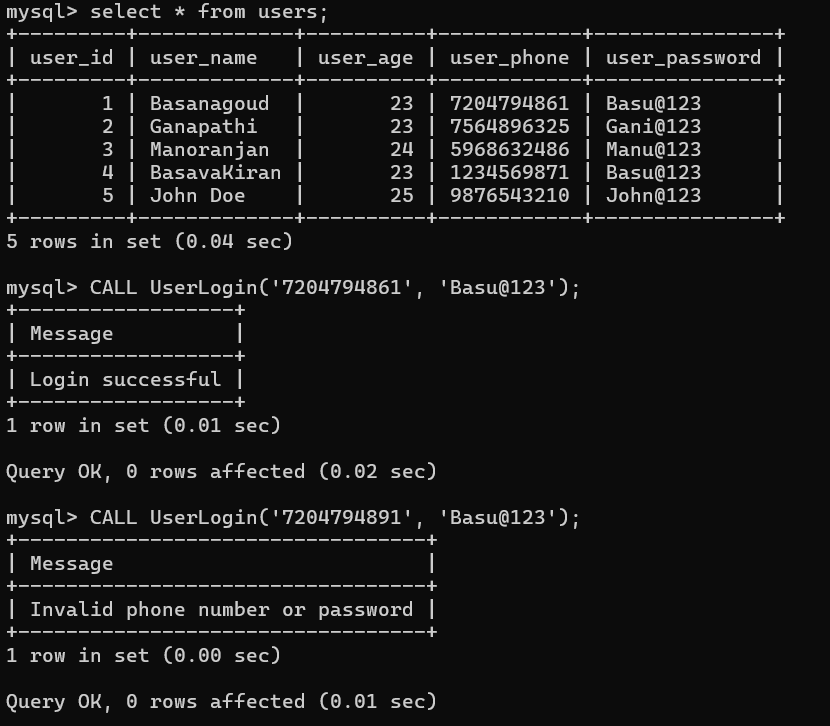
END //

DELIMITER ;

CALL UserLogin('9876543210', 'securepassword123');

The following output is generated when a user attempts to login through the

**UserLogin procedure:**

****

**Private Boat Booking Procedure**

This stored procedure handles private boat booking by verifying the requested date and slot availability. It first checks if the booking date is in the past and restricts such entries. Then, it ensures the boat is not already booked for the same date and slot. If available, the procedure inserts a new booking record into Private\_Boat\_Booking and updates the Private\_Boat\_Status table to mark the boat as unavailable. If the boat is already booked, it notifies the user to choose another slot or boat.

DELIMITER //

CREATE PROCEDURE Book\_Private\_Boat(

IN p\_user\_id INT,

IN p\_boat\_id INT,

IN p\_booking\_date DATE,

IN p\_slot\_id INT

)

BEGIN

DECLARE booking\_exists INT;

-- Check if the provided date is in the past

IF p\_booking\_date < CURDATE() THEN

SELECT 'Please enter a valid date. Past dates are not allowed.' AS message;

ELSE

-- Check if the boat is already booked for the same date and slot

SELECT COUNT(\*) INTO booking\_exists

FROM Private\_Boat\_Status

WHERE boat\_id = p\_boat\_id

AND booking\_date = p\_booking\_date

AND slot\_id = p\_slot\_id

AND is\_available = FALSE; -- Boat is already booked

-- If the boat is already booked, return an error

IF booking\_exists > 0 THEN

SELECT 'Boat is already booked for this date and slot. Choose another boat or slot.' AS message;

ELSE

-- Insert booking into Private\_Boat\_Booking

INSERT INTO Private\_Boat\_Booking (user\_id, boat\_id, booking\_date, slot\_id)

VALUES (p\_user\_id, p\_boat\_id, p\_booking\_date, p\_slot\_id);

-- Insert or update Private\_Boat\_Status to mark boat as unavailable

INSERT INTO Private\_Boat\_Status (boat\_id, booking\_date, slot\_id, is\_available)

VALUES (p\_boat\_id, p\_booking\_date, p\_slot\_id, FALSE)

ON DUPLICATE KEY UPDATE is\_available = FALSE;

-- Return success message

SELECT 'Booking successful!' AS message;

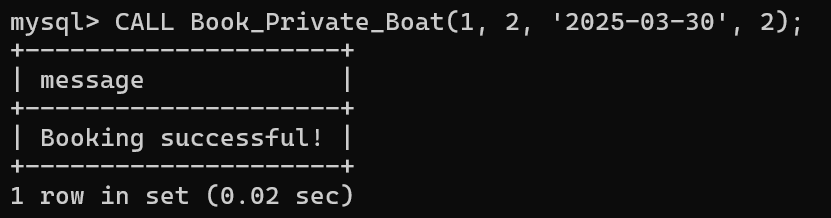
END IF;

END IF;

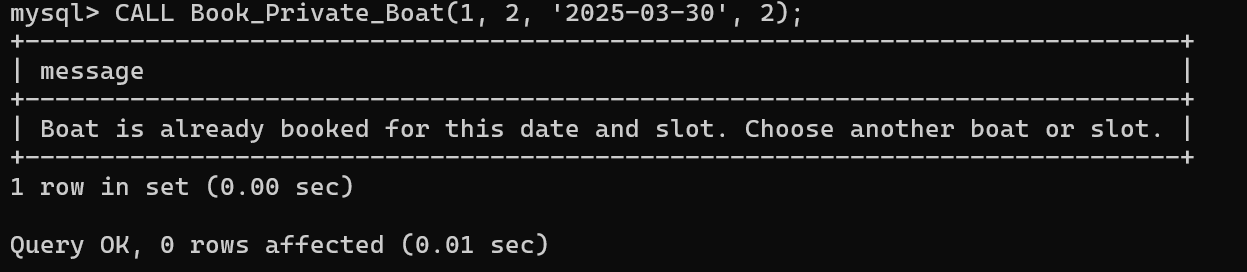
END //

DELIMITER ;

**If the boat is not already booked:**

****

**If the boat is already booked(Using already booked data):**

****

**Removing Expired Private Boat Slots**

This event automatically deletes expired private boat booking records every hour. It removes entries where the booking date has passed or the slot time has expired for the current day. The scheduler ensures that outdated records do not accumulate, maintaining database efficiency. This helps in keeping the booking system updated and free of unnecessary data.

SET GLOBAL event\_scheduler = ON;

This command enables MySQL's event scheduler to execute scheduled tasks automatically. It ensures time-based events, like removing expired bookings, run as expected.

DELIMITER //

CREATE EVENT Remove\_Expired\_Private\_Boat\_Status

ON SCHEDULE EVERY 1 HOUR

DO

BEGIN

DELETE FROM Private\_Boat\_Status

WHERE booking\_date < CURDATE()

OR (booking\_date = CURDATE() AND slot\_id IN (

SELECT slot\_id FROM Private\_Boat\_Slots

WHERE (slot\_id = 1 AND TIME(NOW()) > '10:00:00') -- Morning slot expired

OR (slot\_id = 2 AND TIME(NOW()) > '14:00:00') -- Afternoon slot expired

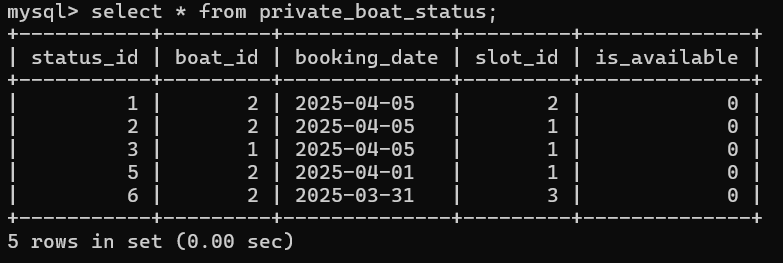
OR (slot\_id = 3 AND TIME(NOW()) > '18:00:00') -- Evening slot expired

));

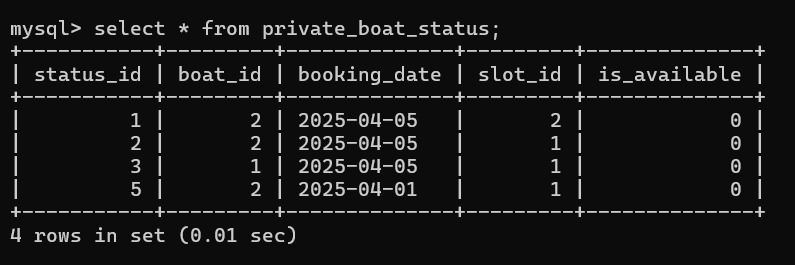
END //

DELIMITER ;

**Data Before expiry:**



**Data After expiry:**



**Public Boat Booking Procedure**

This stored procedure handles booking for public boats by checking seat availability for a given date and slot. It first verifies that the booking date is not in the past. If no existing status record is found, it initializes available seats based on the boat's capacity. If enough seats are available, the booking is confirmed, and the remaining seats are updated. If there aren't enough seats, an appropriate message is returned.

DELIMITER //

CREATE PROCEDURE Book\_Public\_Boat(

IN p\_user\_id INT,

IN p\_boat\_id INT,

IN p\_booking\_date DATE,

IN p\_slot\_id INT,

IN p\_seats\_booked INT

)

BEGIN

DECLARE v\_available\_seats INT;

-- Check if the entered date is in the past

IF p\_booking\_date < CURDATE() THEN

SELECT 'Please enter a valid date. Past dates are not allowed.' AS message;

ELSE

-- Check available seats from Public\_Boat\_Status

SELECT available\_seats INTO v\_available\_seats

FROM Public\_Boat\_Status

WHERE boat\_id = p\_boat\_id

AND booking\_date = p\_booking\_date

AND slot\_id = p\_slot\_id;

-- If no entry exists, initialize status with full capacity

IF v\_available\_seats IS NULL THEN

SELECT capacity INTO v\_available\_seats FROM Public\_Boats WHERE boat\_id = p\_boat\_id;

-- Insert initial status

INSERT INTO Public\_Boat\_Status (boat\_id, booking\_date, slot\_id, available\_seats, boat\_status)

VALUES (p\_boat\_id, p\_booking\_date, p\_slot\_id, v\_available\_seats, 'Available');

END IF;

-- Check if enough seats are available

IF v\_available\_seats >= p\_seats\_booked THEN

-- Insert into Public\_Boat\_Booking

INSERT INTO Public\_Boat\_Booking (user\_id, boat\_id, booking\_date, slot\_id, seats\_booked)

VALUES (p\_user\_id, p\_boat\_id, p\_booking\_date, p\_slot\_id, p\_seats\_booked);

-- Update available seats

UPDATE Public\_Boat\_Status

SET available\_seats = available\_seats - p\_seats\_booked,

boat\_status = CASE

WHEN available\_seats - p\_seats\_booked = 0 THEN 'Full'

ELSE 'Available'

END

WHERE boat\_id = p\_boat\_id

AND booking\_date = p\_booking\_date

AND slot\_id = p\_slot\_id;

SELECT 'Booking Successful' AS message;

ELSE

-- Not enough seats available

SELECT 'Not enough seats available. Choose another boat or slot.' AS message;

END IF;

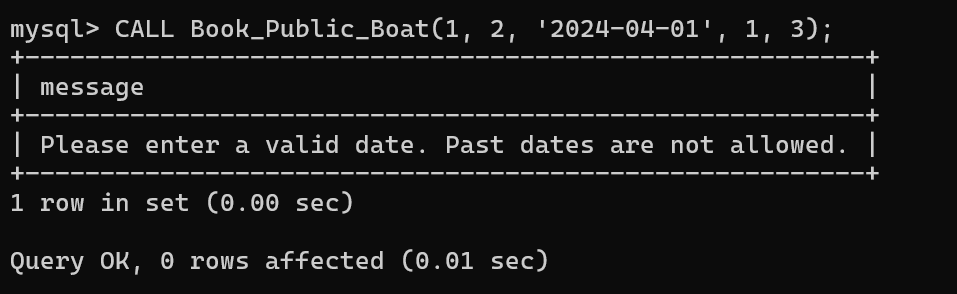
END IF;

END //

DELIMITER ;

**Date Validation in Public Boat Booking**

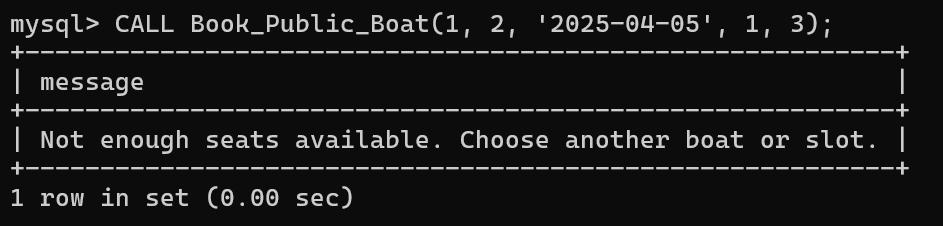
The procedure checks if the entered booking date is in the past using IF p\_booking\_date < CURDATE(). If true, it returns a message:  
"Please enter a valid date. Past dates are not allowed."



**Boat Booking Process**

The procedure retrieves available seats for the selected boat, date, and slot. If no record exists, it initializes the status. If enough seats are available, it books the seats, updates the availability, and marks the boat as full if necessary. If seats are insufficient, it notifies the user to select different slot or boat.

CALL Book\_public\_Boat(1, ‘2025-04-05’, 1, 3);



**Automatic Removal of Expired Public Boat Slots**

This event runs every hour to remove expired bookings from Public\_Boat\_Status. It deletes records where the booking date has passed or where the slot time has expired on the current day. This ensures the system maintains accurate availability records.

DELIMITER //

CREATE EVENT Remove\_Expired\_Public\_Boat\_Status

ON SCHEDULE EVERY 1 HOUR

DO

BEGIN

DELETE FROM Public\_Boat\_Status

WHERE booking\_date < CURDATE()

OR (booking\_date = CURDATE() AND slot\_id IN (

SELECT slot\_id FROM Public\_Boat\_Slots

WHERE (slot\_id = 1 AND TIME(NOW()) > '09:00:00') -- Morning slot expired

OR (slot\_id = 2 AND TIME(NOW()) > '12:00:00') -- Midday slot expired

OR (slot\_id = 3 AND TIME(NOW()) > '15:00:00') -- Afternoon slot expired

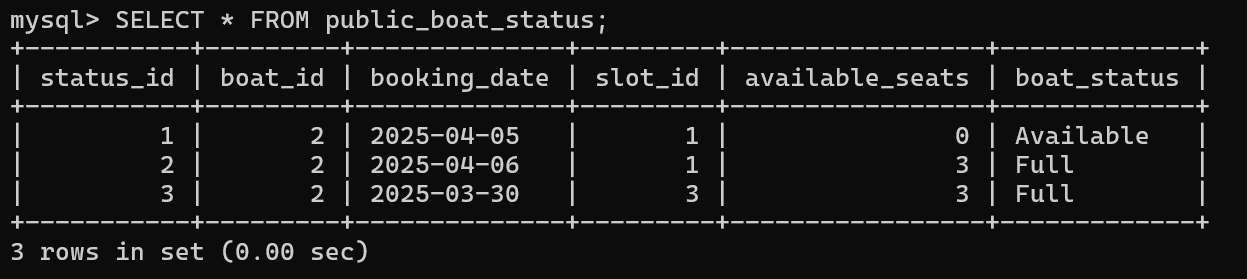
OR (slot\_id = 4 AND TIME(NOW()) > '18:00:00') -- Evening slot expired

));

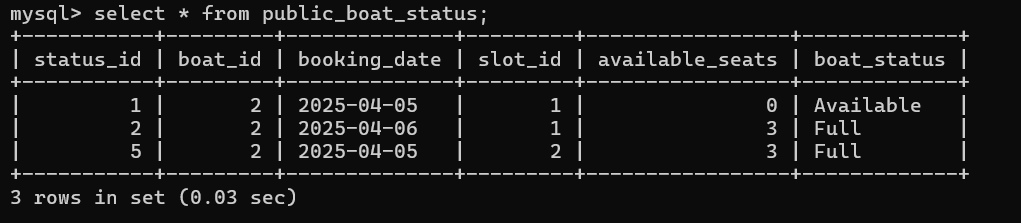
END //

DELIMITER ;

**Data Before expiry:**



**Data After expiry:**

****

Compare both tables, and in the second table, expired dates are automatically deleted.

1. **NORMALIZATION**

**SagaraMitra Boat Booking Application** database structure follows **Third Normal Form (3NF)** and is well-suited for efficient data storage and retrieval. **Here's why:**

**Normalization Analysis**

**1st Normal Form (1NF) - Ensuring Atomicity**

* Each column contains atomic values (no multiple values in a single field).
* Each row is uniquely identifiable by a Primary Key (PK).

**Example:**

* The Users table does not store multiple phone numbers or names in a single column.
* The Public\_Boat\_Slots table stores each slot in a separate row instead of combining multiple slots in one field.

**2nd Normal Form (2NF) - Removing Partial Dependency**

* All non-key attributes depend on the whole primary key, not just part of it.
* Composite primary keys (if any) are structured correctly.

**Example:**

* In Public\_Boat\_Booking, user\_id, boat\_id, slot\_id, and booking\_date together define a unique booking.
* The Public\_Boat\_Status table ensures available\_seats depends on boat\_id and slot\_id.

**3rd Normal Form (3NF) - Removing Transitive Dependencies**

* No transitive dependencies exist (non-key attributes only depend on primary keys).
* Every non-key column depends directly on the primary key.

**Example:**

* Boat\_Owners(owner\_name) only depends on owner\_id (not indirectly through another column).
* Public\_Boats (pickup\_drop\_location) directly depends on boat\_id, ensuring no redundant data.

**Why 3NF is Suitable to our SagaraMitra Database?**

* Eliminates Data Redundancy – No unnecessary duplicate data (e.g., user details are stored only once).
* Ensures Data Integrity – Changes in one table (e.g., Boat\_Owners) won’t cause inconsistencies elsewhere.
* Avoids Update Anomalies – Updating a boat name or owner does not require changing multiple tables**.**
* To implement the SagaraMitra Boat Booking Application, we designed the relational database schema using 3rd Normal Form (3NF) to ensure data integrity, eliminate redundancy, and optimize performance.