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 Gouda	[1AT24MC025]
Manoranjan Behera	[1AT24MC044]

Under the guidance of

Dr. Gomathy Prathima E

Academic Year: 2024 – 2025

ATRIA INSTITUTE OF TECHNOLOGY
(AN AUTONOMOUS INSTITUTION)
BENGALURU

EVALUATION SHEET – ACTIVITY BASED ASSESSMENT

Evaluation criteria	1AT24MC010 Basanagoud	1AT24MC011 Basavakiran	1AT24MC025 Ganapathi	1AT24MC044 Manoranjan
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.

2. 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

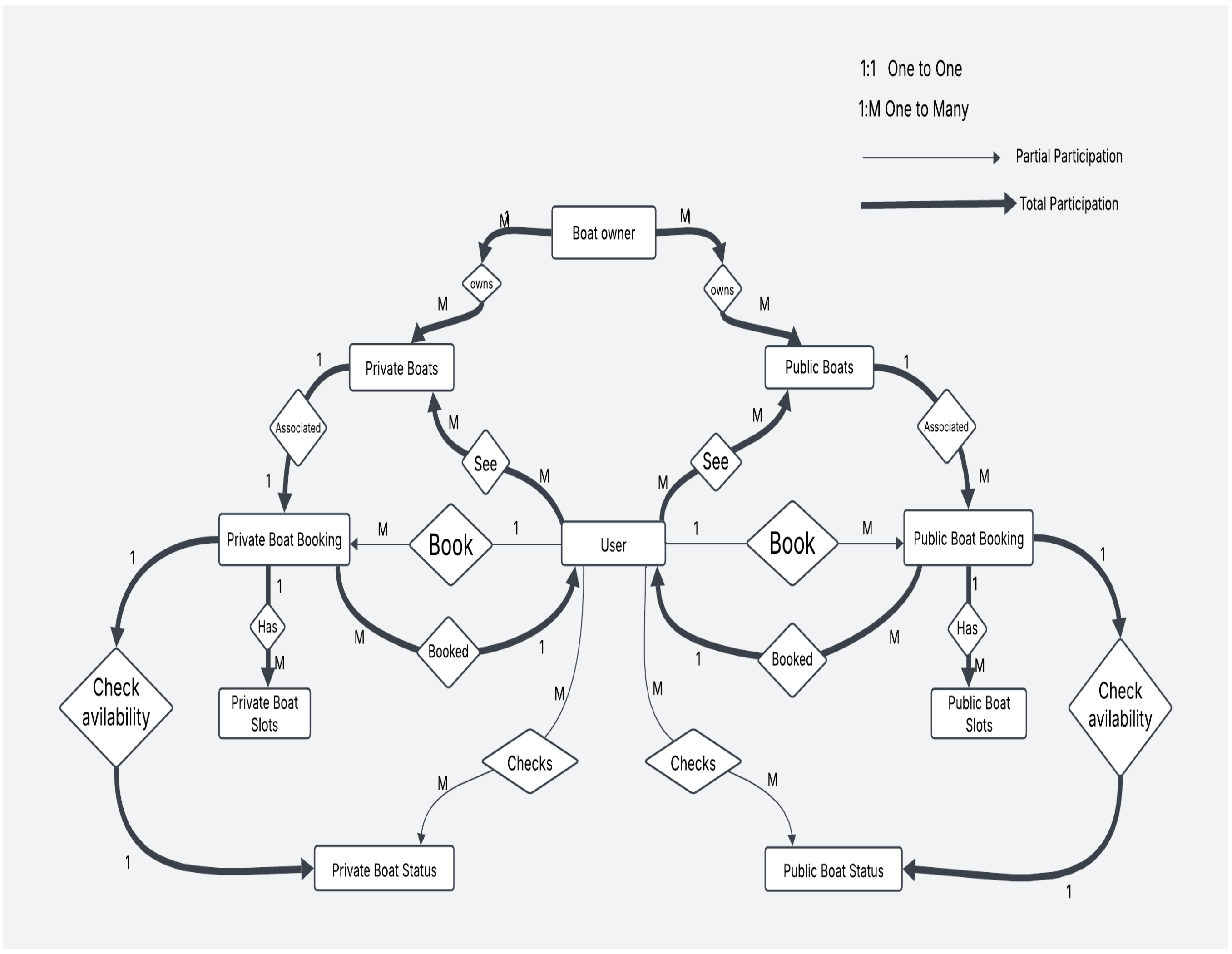
- **Attributes:**
  - status_id (PK) – Unique identifier for the status entry
  - boat_id (FK) – References Public_Boats(boat_id)
  - booking_date – Date of booking
  - slot_id (FK) – References Public_Boat_Slots(slot_id)

    - 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

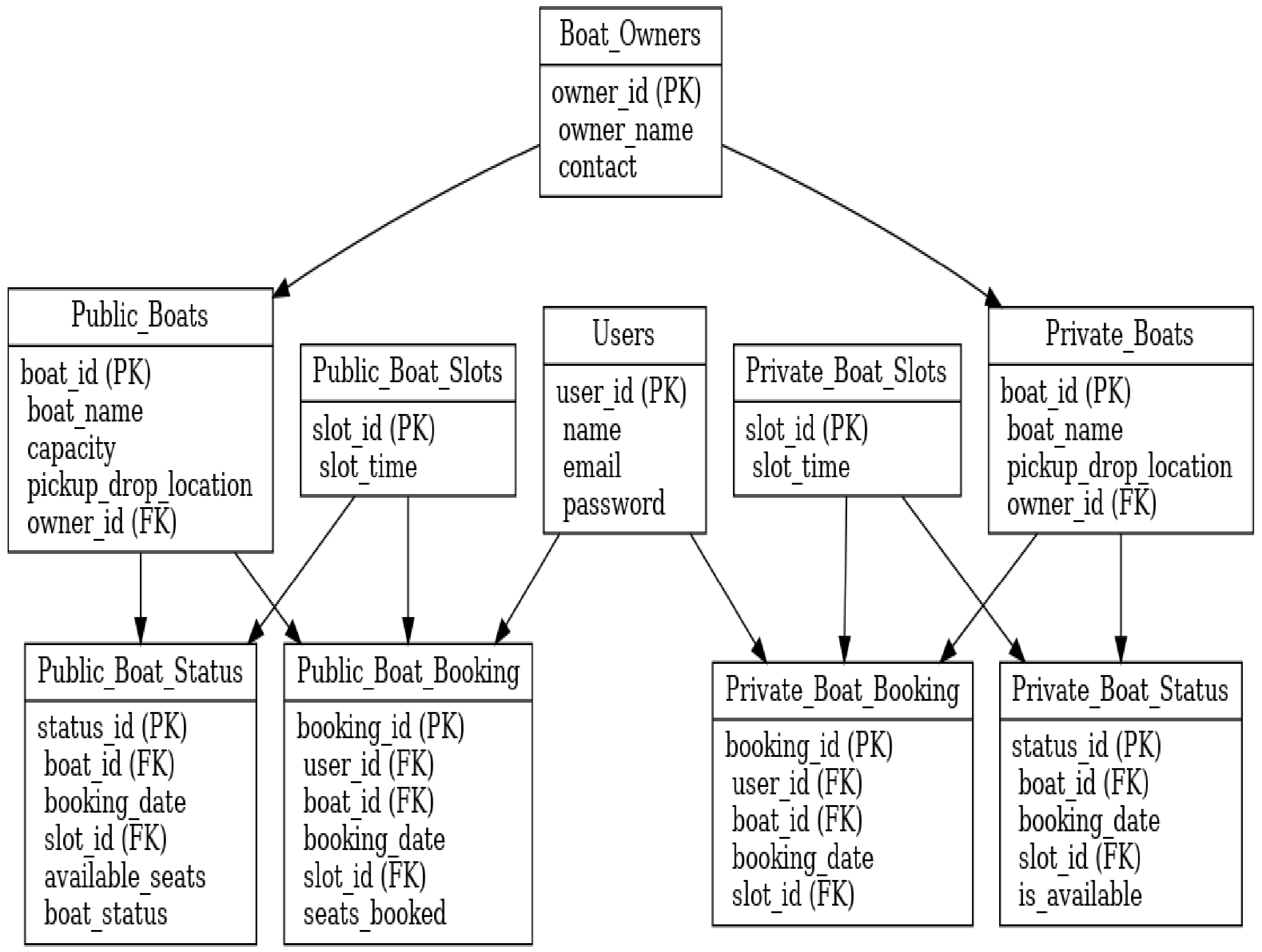
- **Attributes:**
    - status_id (PK) – Unique identifier for the status entry
    - boat_id (FK) – References Private_Boats(boat_id)
    - booking_date – Date of booking
    - slot_id (FK) – References Private_Boat_Slots(slot_id)
    - is_available – Boolean flag indicating availability
- **Relationships:**
    - 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)

3. ER MODEL TO RELATIONAL MODEL MAPPING



4. 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)
```

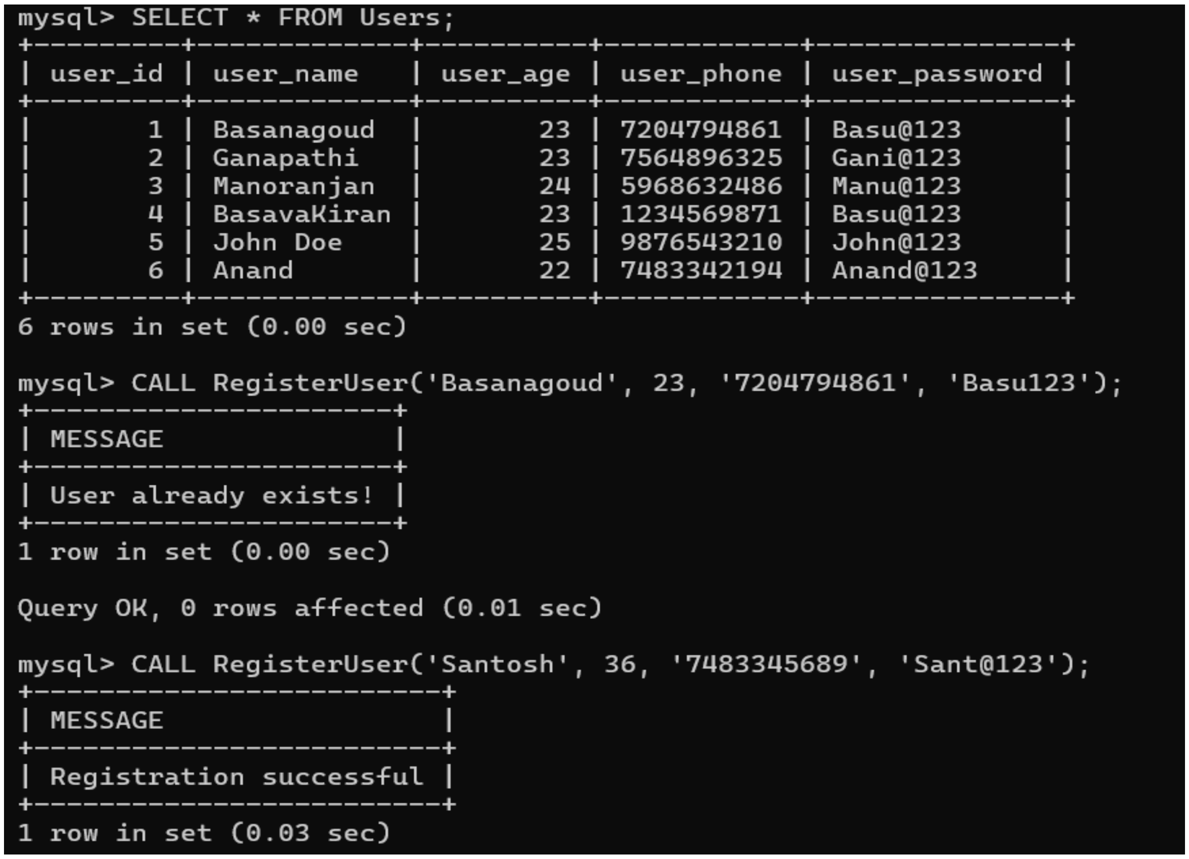
```
);

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:**



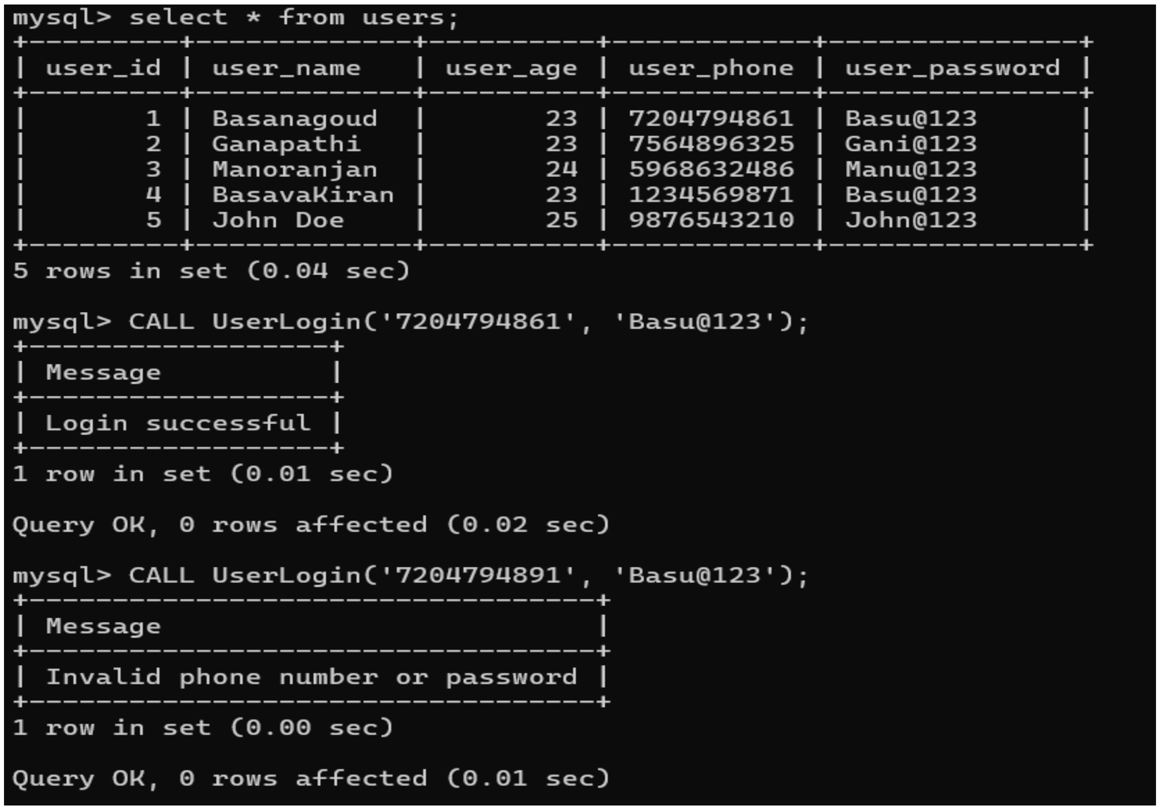
```
mysql> SELECT * FROM Users;
+---------+-------------+----------+------------+---------------+
| user_id | user_name   | user_age | user_phone | user_password |
+---------+-------------+----------+------------+---------------+
|       1 | Basanagoud  |       23 | 7204794861 | Basu@123      |
|       2 | Ganapathi   |       23 | 7564896325 | Gani@123      |
|       3 | Manoranjan  |       24 | 5968632486 | Manu@123      |
|       4 | BasavaKiran |       23 | 1234569871 | Basu@123      |
|       5 | John Doe    |       25 | 9876543210 | John@123      |
|       6 | Anand       |       22 | 7483342194 | Anand@123     |
+---------+-------------+----------+------------+---------------+
6 rows in set (0.00 sec)

mysql> CALL RegisterUser('Basanagoud', 23, '7204794861', 'Basu123');
+----------------------+
| MESSAGE              |
+----------------------+
| User already exists! |
+----------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> CALL RegisterUser('Santosh', 36, '7483345689', 'Sant@123');
+-------------------------+
| MESSAGE                 |
+-------------------------+
| Registration successful |
+-------------------------+
1 row in set (0.03 sec)
```

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:**



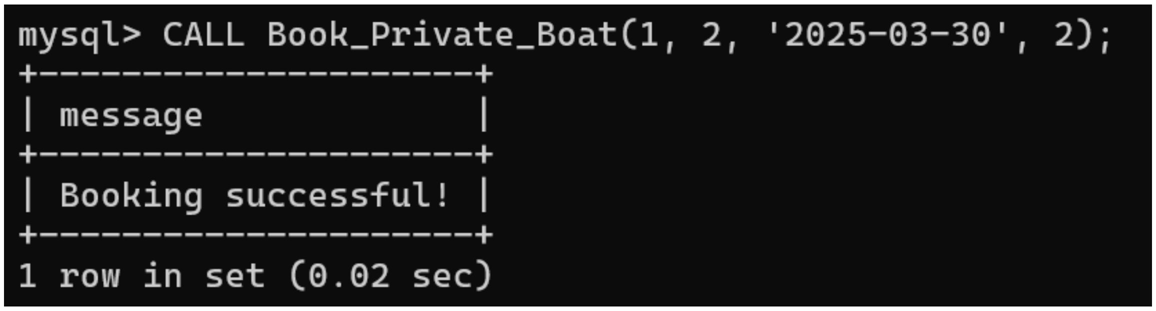
```
mysql> select * from users;
+---------+-------------+----------+------------+---------------+
| user_id | user_name   | user_age | user_phone | user_password |
+---------+-------------+----------+------------+---------------+
|       1 | Basanagoud  |       23 | 7204794861 | Basu@123      |
|       2 | Ganapathi   |       23 | 7564896325 | Gani@123      |
|       3 | Manoranjan  |       24 | 5968632486 | Manu@123      |
|       4 | BasavaKiran |       23 | 1234569871 | Basu@123      |
|       5 | John Doe    |       25 | 9876543210 | John@123      |
+---------+-------------+----------+------------+---------------+
5 rows in set (0.04 sec)

mysql> CALL UserLogin('7204794861', 'Basu@123');
+------------------+
| Message          |
+------------------+
| Login successful |
+------------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.02 sec)

mysql> CALL UserLogin('7204794891', 'Basu@123');
+----------------------------------+
| Message                          |
+----------------------------------+
| Invalid phone number or password |
+----------------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```

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.

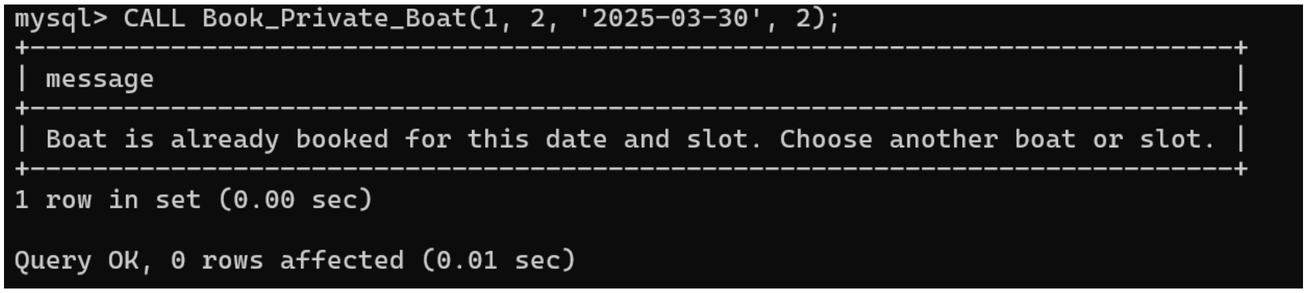
```
DELIMITER //

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:



```
mysql> CALL Book_Private_Boat(1, 2, '2025-03-30', 2);
+---------------------+
| message             |
+---------------------+
| Booking successful! |
+---------------------+
1 row in set (0.02 sec)
```

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



```
mysql> CALL Book_Private_Boat(1, 2, '2025-03-30', 2);
+-----------------------------------------------------------------------------+
| message                                                                     |
+-----------------------------------------------------------------------------+
| Boat is already booked for this date and slot. Choose another boat or slot. |
+-----------------------------------------------------------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```

Removing Expired Private Boat Slots

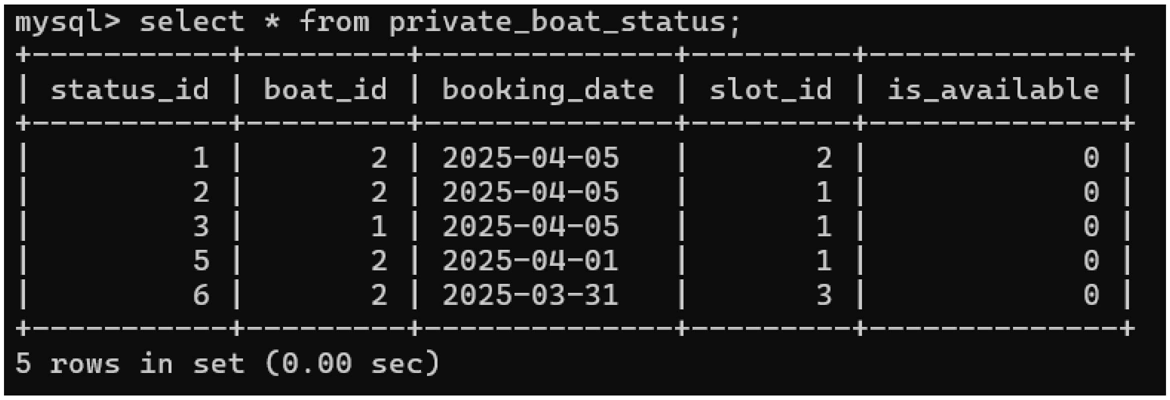
This event automatically deletes expired private boat booking records every hour. It removes entries where the booking date has passed or the slot time has expired for the current day. The scheduler ensures that outdated records do not accumulate, maintaining database efficiency. This helps in keeping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

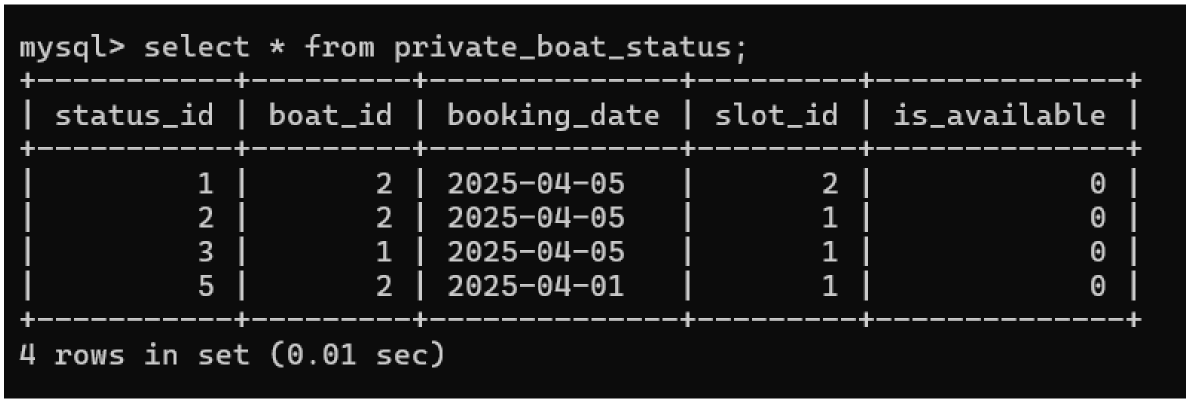
```
DELIMITER //

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:



```
mysql> select * from private_boat_status;
+-----------+---------+--------------+---------+--------------+
| status_id | boat_id | booking_date | slot_id | is_available |
+-----------+---------+--------------+---------+--------------+
|         1 |       2 | 2025-04-05   |       2 |            0 |
|         2 |       2 | 2025-04-05   |       1 |            0 |
|         3 |       1 | 2025-04-05   |       1 |            0 |
|         5 |       2 | 2025-04-01   |       1 |            0 |
|         6 |       2 | 2025-03-31   |       3 |            0 |
+-----------+---------+--------------+---------+--------------+
5 rows in set (0.00 sec)
```

Data After expiry:



```
mysql> select * from private_boat_status;
+-----------+---------+--------------+---------+--------------+
| status_id | boat_id | booking_date | slot_id | is_available |
+-----------+---------+--------------+---------+--------------+
|         1 |       2 | 2025-04-05   |       2 |            0 |
|         2 |       2 | 2025-04-05   |       1 |            0 |
|         3 |       1 | 2025-04-05   |       1 |            0 |
|         5 |       2 | 2025-04-01   |       1 |            0 |
+-----------+---------+--------------+---------+--------------+
4 rows in set (0.01 sec)
```

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
```

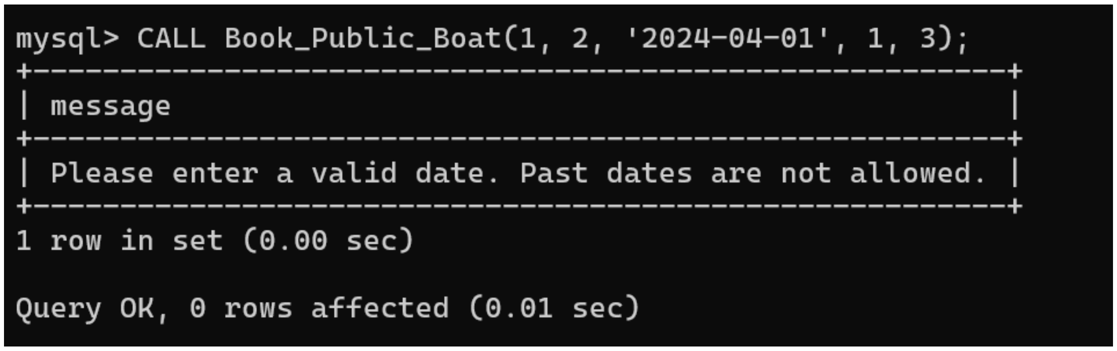
```
            -- Insert into Public_Boat_Booking
            INSERT INTO Public_Boat_Booking (user_id, boat_id, booking_date, slot_id,
seats_booked)
            VALUES (p_user_id, p_boat_id, p_booking_date, p_slot_id, p_seats_booked);

            -- 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."

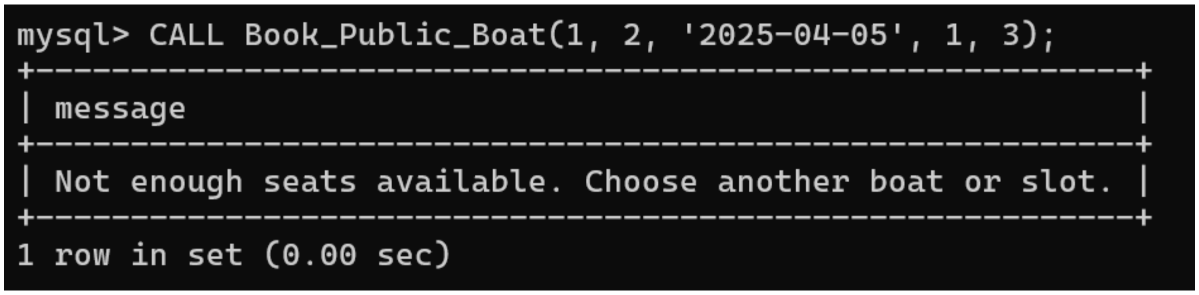


```
mysql> CALL Book_Public_Boat(1, 2, '2024-04-01', 1, 3);
+--------------------------------------------------------+
| message                                                |
+--------------------------------------------------------+
| Please enter a valid date. Past dates are not allowed. |
+--------------------------------------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```

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);



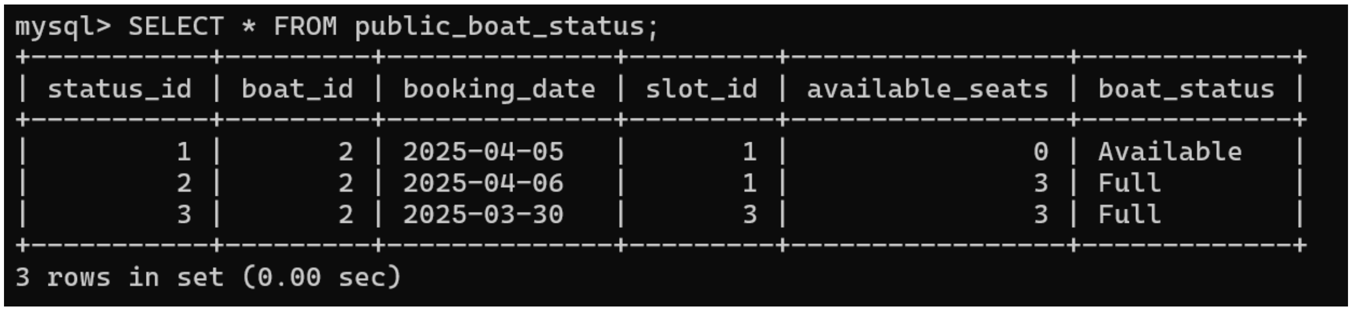
```
mysql> CALL Book_Public_Boat(1, 2, '2025-04-05', 1, 3);
+-----------------------------------------------------------+
| message                                                   |
+-----------------------------------------------------------+
| Not enough seats available. Choose another boat or slot.  |
+-----------------------------------------------------------+
1 row in set (0.00 sec)
```

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.

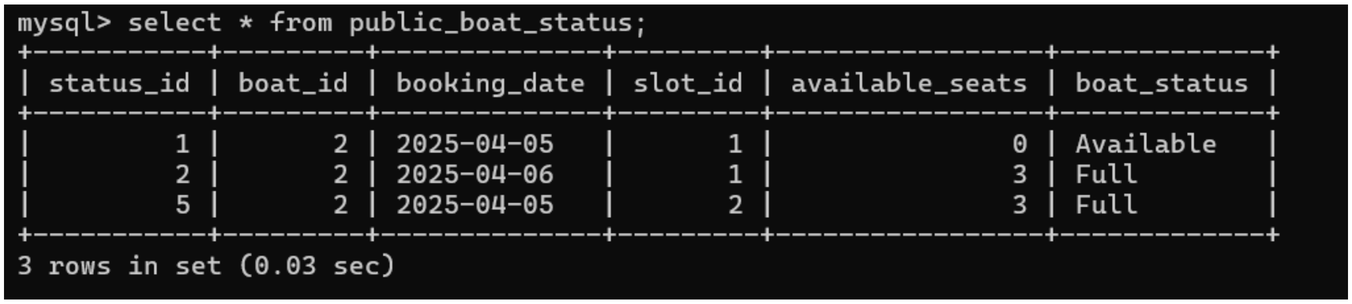
```
DELIMITER //

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:



```
mysql> SELECT * FROM public_boat_status;
+-----------+---------+--------------+---------+-----------------+-------------+
| status_id | boat_id | booking_date | slot_id | available_seats | boat_status |
+-----------+---------+--------------+---------+-----------------+-------------+
|         1 |       2 | 2025-04-05   |       1 |               0 | Available   |
|         2 |       2 | 2025-04-06   |       1 |               3 | Full        |
|         3 |       2 | 2025-03-30   |       3 |               3 | Full        |
+-----------+---------+--------------+---------+-----------------+-------------+
3 rows in set (0.00 sec)
```

Data After expiry:



```
mysql> select * from public_boat_status;
+-----------+---------+--------------+---------+-----------------+-------------+
| status_id | boat_id | booking_date | slot_id | available_seats | boat_status |
+-----------+---------+--------------+---------+-----------------+-------------+
|         1 |       2 | 2025-04-05   |       1 |               0 | Available   |
|         2 |       2 | 2025-04-06   |       1 |               3 | Full        |
|         5 |       2 | 2025-04-05   |       2 |               3 | Full        |
+-----------+---------+--------------+---------+-----------------+-------------+
3 rows in set (0.03 sec)
```

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

5. 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.

Conclusion:

The SagaraMitra Boat Booking System offers a database-driven platform for booking boats in Honnavar, enhancing tourism with easy slot management. It enables public and private boat reservations, ensuring real-time availability tracking. The system supports tourism and a user-friendly booking process. The ER model defines structured relationships between users, boat owners, boats, slots, and bookings, ensuring efficient data management.

Secure user authentication is implemented with stored procedures for registration and login. The booking procedures prevent overbooking and ensure smooth transactions for private boat booking. A scheduled event automatically removes expired bookings, maintaining database efficiency. The database design ensures data scalability. Overall, SagaraMitra promotes sustainable tourism by integrating technology with Honnavar’s natural beauty. This system ensures a well-organized, and friendly boating experience for tourists and boat owners.