```sql
-- DROP TABLES
BEGIN
  EXECUTE IMMEDIATE 'DROP TABLE Payments CASCADE CONSTRAINTS';
  EXECUTE IMMEDIATE 'DROP TABLE Transactions CASCADE CONSTRAINTS';
  EXECUTE IMMEDIATE 'DROP TABLE Terminals CASCADE CONSTRAINTS';
  EXECUTE IMMEDIATE 'DROP TABLE Bikes CASCADE CONSTRAINTS';
  EXECUTE IMMEDIATE 'DROP TABLE Admins CASCADE CONSTRAINTS';
  EXECUTE IMMEDIATE 'DROP TABLE Users CASCADE CONSTRAINTS';
EXCEPTION
  WHEN OTHERS THEN NULL;
END;
/

-- USERS
CREATE TABLE Users (
    Email VARCHAR2(100) PRIMARY KEY,
    First_Name VARCHAR2(50),
    Last_Name VARCHAR2(50),
    Phone VARCHAR2(15),
    Password VARCHAR2(100),
    DL_Number VARCHAR2(20) UNIQUE
);

-- ADMINS
CREATE TABLE Admins (
    Email VARCHAR2(100) PRIMARY KEY,
    Password VARCHAR2(100)
);

-- BIKES
CREATE TABLE Bikes (
    Bike_ID NUMBER PRIMARY KEY,
    Bike_Name VARCHAR2(100),
    Model VARCHAR2(50),
    Color VARCHAR2(30),
    Availability VARCHAR2(10) CHECK (Availability IN ('Available', 'Rented')),
    Bike_Type VARCHAR2(50),
    Price NUMBER(10, 2) CHECK (Price >= 0)
);

-- TERMINALS
CREATE TABLE Terminals (
    Terminal_ID NUMBER PRIMARY KEY,
    Terminal_Name VARCHAR2(100),
    No_of_Bikes NUMBER DEFAULT 0
);

-- TRANSACTIONS
CREATE TABLE Transactions (
    Transaction_ID NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    Email VARCHAR2(100),
    Bike_ID NUMBER,
    Terminal_ID NUMBER,
    Start_Time TIMESTAMP,
    End_Time TIMESTAMP,
    Trans_Date DATE,
    Actual_Return_Time TIMESTAMP,
    FOREIGN KEY (Email) REFERENCES Users(Email) ON DELETE CASCADE,
    FOREIGN KEY (Bike_ID) REFERENCES Bikes(Bike_ID),
    FOREIGN KEY (Terminal_ID) REFERENCES Terminals(Terminal_ID)
);

-- PAYMENTS
CREATE TABLE Payments (
    Receipt_No NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    Transaction_ID NUMBER,
    Cost NUMBER(10, 2),
    Mode_of_Payment VARCHAR2(50),
    Paid_On DATE,
    Penalty NUMBER(10, 2) DEFAULT 0,
    FOREIGN KEY (Transaction_ID) REFERENCES Transactions(Transaction_ID)
);

-- SAMPLE DATA
INSERT INTO Users VALUES ('alice@example.com', 'Alice', 'Smith', '1234567890', 'alice123',
'DL001');
INSERT INTO Users VALUES ('bob@example.com', 'Bob', 'Johnson', '9876543210',
'bobpass', 'DL002');
INSERT INTO Users VALUES ('charlie@example.com', 'Charlie', 'Williams', '9812345678',
'charliepass', 'DL003');
INSERT INTO Users VALUES ('diana@example.com', 'Diana', 'Brown', '9801122334',
'diana321', 'DL004');
INSERT INTO Users VALUES ('eric@example.com', 'Eric', 'Davis', '9871112233', 'ericpass',
'DL005');
INSERT INTO Users VALUES ('fiona@example.com', 'Fiona', 'Wilson', '9844112255',
'fiona456', 'DL006');
INSERT INTO Users VALUES ('george@example.com', 'George', 'Moore', '9866223344',
'george789', 'DL007');

INSERT INTO Admins VALUES ('admin@bikehub.com', 'admin123');

INSERT INTO Bikes VALUES (1, 'Thunder 200X', '2023', 'Red', 'Available', 'Sports', 150.00);
INSERT INTO Bikes VALUES (2, 'Cruiser 150', '2022', 'Black', 'Available', 'Cruiser', 120.00);
INSERT INTO Bikes VALUES (3, 'Speedster 3000', '2023', 'Blue', 'Available', 'Sports',
180.00);

INSERT INTO Terminals VALUES (101, 'Central Terminal', 20);
INSERT INTO Terminals VALUES (102, 'City Park Terminal', 15);

COMMIT;

-- PROCEDURE: BOOK_BIKE
CREATE OR REPLACE PROCEDURE Book_Bike (
    p_email IN VARCHAR2,
    p_bike_id IN NUMBER,
    p_terminal_id IN NUMBER,
    p_start_time IN TIMESTAMP,
    p_end_time IN TIMESTAMP
) AS
    v_available VARCHAR2(10);
BEGIN
    SELECT Availability INTO v_available FROM Bikes WHERE Bike_ID = p_bike_id;

    IF v_available = 'Available' THEN
        INSERT INTO Transactions (Email, Bike_ID, Terminal_ID, Start_Time, End_Time,
Trans_Date)
```

```
        VALUES (p_email, p_bike_id, p_terminal_id, p_start_time, p_end_time, SYSDATE);

        UPDATE Bikes SET Availability = 'Rented' WHERE Bike_ID = p_bike_id;
        UPDATE Terminals SET No_of_Bikes = No_of_Bikes - 1
        WHERE Terminal_ID = p_terminal_id AND No_of_Bikes > 0;

        DBMS_OUTPUT.PUT_LINE('Bike booked successfully.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Bike is not available.');
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Bike not found.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/

-- PROCEDURE: RETURN_BIKE
CREATE OR REPLACE PROCEDURE Return_Bike (
    p_transaction_id IN NUMBER
) AS
    v_bike_id NUMBER;
    v_terminal_id NUMBER;
    v_end_time TIMESTAMP;
    v_actual_return_time TIMESTAMP;
    v_penalty NUMBER(10, 2) := 0;
    v_hours_late NUMBER(10, 2);
    v_bike_type VARCHAR2(50);
    v_start_time TIMESTAMP;
    v_price NUMBER(10, 2);
    v_cost NUMBER(10, 2);
BEGIN
    SELECT Bike_ID, Terminal_ID, End_Time, Actual_Return_Time, Start_Time
    INTO v_bike_id, v_terminal_id, v_end_time, v_actual_return_time, v_start_time
    FROM Transactions
    WHERE Transaction_ID = p_transaction_id;

    IF v_actual_return_time IS NOT NULL THEN
        SELECT Bike_Type, Price INTO v_bike_type, v_price FROM Bikes WHERE Bike_ID =
v_bike_id;

        IF v_actual_return_time > v_end_time THEN
            v_hours_late := (CAST(v_actual_return_time AS DATE) - CAST(v_end_time AS DATE))
* 24;

            IF v_hours_late > 0.25 THEN
                IF v_bike_type = 'Sports' THEN
                    v_penalty := (v_hours_late - 0.25) * 75;
                ELSE
                    v_penalty := (v_hours_late - 0.25) * 50;
                END IF;
            END IF;

            IF v_penalty > 1000 THEN v_penalty := 1000; END IF;
        END IF;

        v_cost := (CAST(v_end_time AS DATE) - CAST(v_start_time AS DATE)) * 24 * v_price;
        INSERT INTO Payments (Transaction_ID, Cost, Mode_of_Payment, Paid_On, Penalty)
        VALUES (p_transaction_id, v_cost, 'Card', SYSDATE, v_penalty);

        UPDATE Bikes SET Availability = 'Available' WHERE Bike_ID = v_bike_id;
        UPDATE Terminals SET No_of_Bikes = No_of_Bikes + 1 WHERE Terminal_ID =
v_terminal_id;

        DBMS_OUTPUT.PUT_LINE('Bike return processed. Penalty: ₹' || ROUND(v_penalty, 2));
    ELSE
        DBMS_OUTPUT.PUT_LINE('Error: Actual Return Time is NULL.');
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Transaction not found.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/

-- PROCEDURE: UPDATE_BIKE_COUNT
CREATE OR REPLACE PROCEDURE Update_Bike_Count (
    p_terminal_id IN NUMBER
) AS
    v_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_count
    FROM Transactions T
    JOIN Bikes B ON T.Bike_ID = B.Bike_ID
    WHERE T.Terminal_ID = p_terminal_id
      AND B.Availability = 'Available'
      AND T.Actual_Return_Time IS NOT NULL;

    UPDATE Terminals
    SET No_of_Bikes = v_count
    WHERE Terminal_ID = p_terminal_id;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/



-- Transaction 1
BEGIN
    Book_Bike('alice@example.com', 1, 101,
      TO_TIMESTAMP('2025-05-02 10:00:00', 'YYYY-MM-DD HH24:MI:SS'),
      TO_TIMESTAMP('2025-05-02 12:00:00', 'YYYY-MM-DD HH24:MI:SS'));
END;
/
UPDATE Transactions SET Actual_Return_Time = TO_TIMESTAMP('2025-05-02 12:30:00',
'YYYY-MM-DD HH24:MI:SS')
WHERE Transaction_ID = 1;
BEGIN
    Return_Bike(1);
END;
/
```

```
-- Transaction 2
BEGIN
   Book_Bike('bob@example.com', 2, 101,
     TO_TIMESTAMP('2025-05-01 09:00:00', 'YYYY-MM-DD HH24:MI:SS'),
     TO_TIMESTAMP('2025-05-01 11:00:00', 'YYYY-MM-DD HH24:MI:SS'));
END;
/
UPDATE Transactions SET Actual_Return_Time = TO_TIMESTAMP('2025-05-01 11:45:00',
'YYYY-MM-DD HH24:MI:SS')
WHERE Transaction_ID = 2;
BEGIN
   Return_Bike(2);
END;
/

-- Transaction 3
BEGIN
   Book_Bike('charlie@example.com', 3, 102,
     TO_TIMESTAMP('2025-05-01 14:00:00', 'YYYY-MM-DD HH24:MI:SS'),
     TO_TIMESTAMP('2025-05-01 16:00:00', 'YYYY-MM-DD HH24:MI:SS'));
END;
/
UPDATE Transactions SET Actual_Return_Time = TO_TIMESTAMP('2025-05-01 16:30:00',
'YYYY-MM-DD HH24:MI:SS')
WHERE Transaction_ID = 3;
BEGIN
   Return_Bike(3);
END;
/

--1. View All Bikes with Terminal Info
SELECT B.Bike_ID, B.Bike_Name, B.Model, B.Color, B.Bike_Type, B.Price, B.Availability,
       T.Terminal_Name
FROM Bikes B
LEFT JOIN Transactions T1 ON B.Bike_ID = T1.Bike_ID
LEFT JOIN Terminals T ON T1.Terminal_ID = T.Terminal_ID;

--2. View All Users with Their Current/Recent Bookings
SELECT U.Email, U.First_Name, U.Last_Name, T.Transaction_ID, T.Bike_ID,
       T.Start_Time, T.End_Time, T.Actual_Return_Time
FROM Users U
LEFT JOIN Transactions T ON U.Email = T.Email;

-- 3. List All Terminals with Number of Available Bikes

SELECT Terminal_ID, Terminal_Name, No_of_Bikes FROM Terminals;

-- 4. Total Amount Spent by Each Customer (with penalties)
SELECT U.Email, U.First_Name || ' ' || U.Last_Name AS Full_Name,
       NVL(SUM(P.Cost + P.Penalty), 0) AS Total_Spent
FROM Users U
LEFT JOIN Transactions T ON U.Email = T.Email
LEFT JOIN Payments P ON T.Transaction_ID = P.Transaction_ID
GROUP BY U.Email, U.First_Name, U.Last_Name;

--  5. Bikes Currently Rented
SELECT B.Bike_Name, COUNT(*) AS Rental_Count
```

```
FROM Bikes B
JOIN Transactions T ON B.Bike_ID = T.Bike_ID
GROUP BY B.Bike_Name;

-- 6. Available Bikes by Terminal
SELECT T.Terminal_Name, COUNT(B.Bike_ID) AS Available_Bikes
FROM Bikes B
JOIN Transactions TR ON B.Bike_ID = TR.Bike_ID
JOIN Terminals T ON TR.Terminal_ID = T.Terminal_ID
WHERE B.Availability = 'Available'
GROUP BY T.Terminal_Name;


SELECT DISTINCT U.Email, U.First_Name, U.Last_Name, P.Penalty
FROM Users U
JOIN Transactions T ON U.Email = T.Email
JOIN Payments P ON T.Transaction_ID = P.Transaction_ID
WHERE P.Penalty > 0;


-- 8. Revenue by Bike Type
SELECT B.Bike_Type, SUM(P.Cost + P.Penalty) AS Total_Revenue
FROM Bikes B
JOIN Transactions T ON B.Bike_ID = T.Bike_ID
JOIN Payments P ON T.Transaction_ID = P.Transaction_ID
GROUP BY B.Bike_Type;

--  9. Most Frequently Rented Bike
   SELECT B.Bike_Name, COUNT(*) AS Rental_Count
   FROM Bikes B
   JOIN Transactions T ON B.Bike_ID = T.Bike_ID
   GROUP BY B.Bike_Name
   ORDER BY Rental_Count DESC
   FETCH FIRST 1 ROWS ONLY;
```