UE22CS351A – DBMS

AUG - DEC 2025

Mini Project Report on

REAL ESTATE MANAGEMENT

Submitted to

Dr. Geetha D
Associate Professor

Submitted by:

SRN & Name

PES2UG23AM046&JALANTH S
PES2UG23AM036&G VISHWAS

Department of CSE(AI&ML)

PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

TABLE OF CONTENTS

INTRODUCTION

The Real Estate Database Management System is a comprehensive software solution designed to streamline the operations of a real estate agency. It manages critical data entities such as clients, agents, properties, contracts, and payments, ensuring data integrity and providing valuable insights through complex queries and automated processes. This project aims to replace manual record-keeping with a robust, scalable, and secure digital platform.

PROBLEM DEFINITION WITH USER REQUIREMENT SPECIFICATIONS

Purpose

The purpose of this project is to develop a centralized database system for a real estate firm to efficiently manage its property listings, client interactions, agent performance, and financial transactions. The system aims to reduce data redundancy, improve data accuracy, and provide real-time reporting capabilities.

Scope

The scope of this project includes the design and implementation of a relational database, a backend API for data access and manipulation, and a frontend user interface for easy interaction. Key features include managing client and agent profiles, tracking property details, recording contracts and payments, and automatically calculating agent commissions.

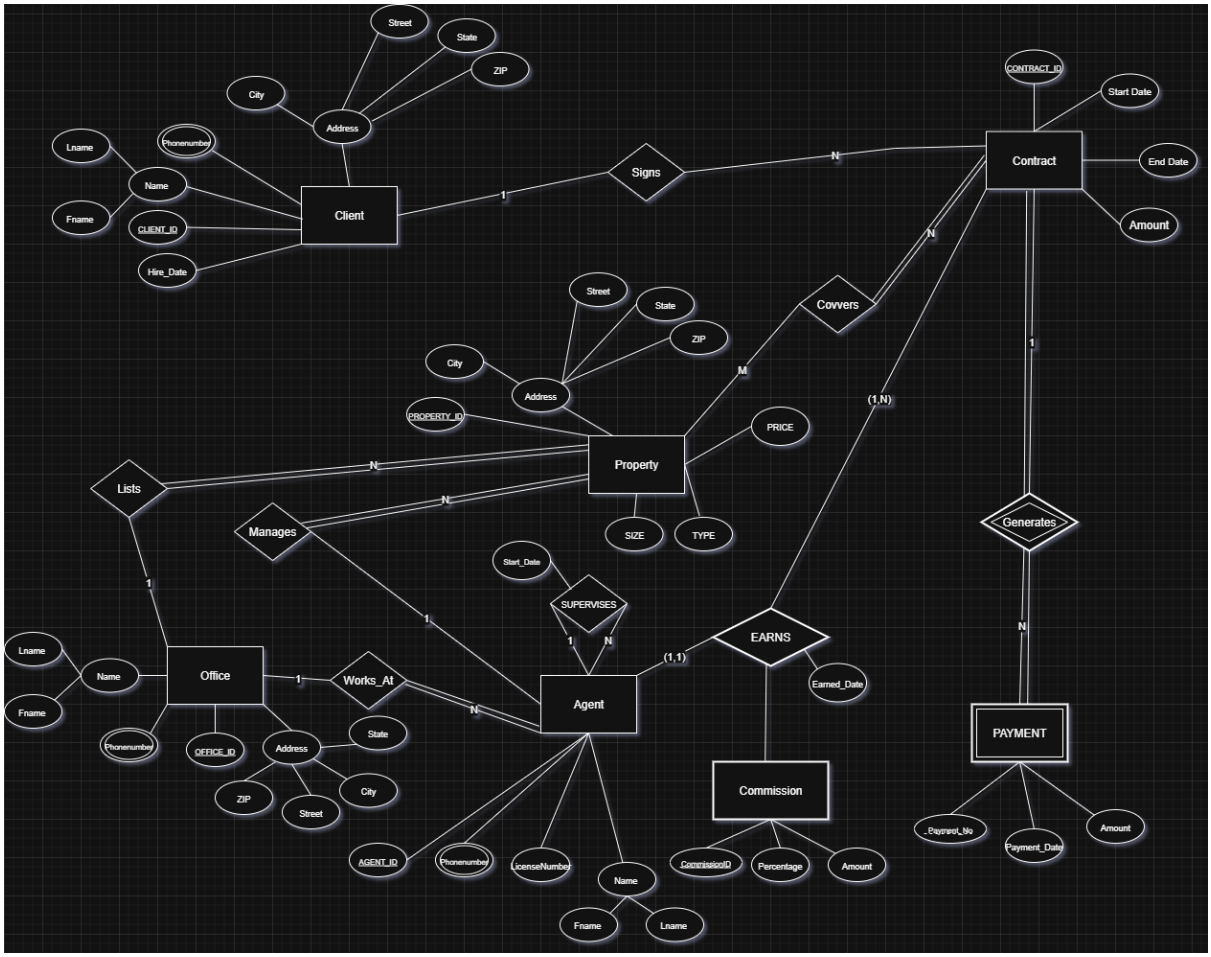
Functional Requirements

- **User Authentication:** Secure login and registration for authorized access to the system.
- **Client Management:** Ability to add, view, and delete client records.
- **Contract Management:** Functionality to create new contracts with automatic date validation and delete existing contracts.
- **Payment Processing:** Recording payments against specific contracts, which automatically updates commission calculations.
- **Agent Performance Tracking:** Viewing total earnings (potential vs. actual) for each agent.
- **Financial Reporting:** Generating reports on total payments received per contract.
- **High-Value Client Identification:** Identifying clients with contracts valued above the average.

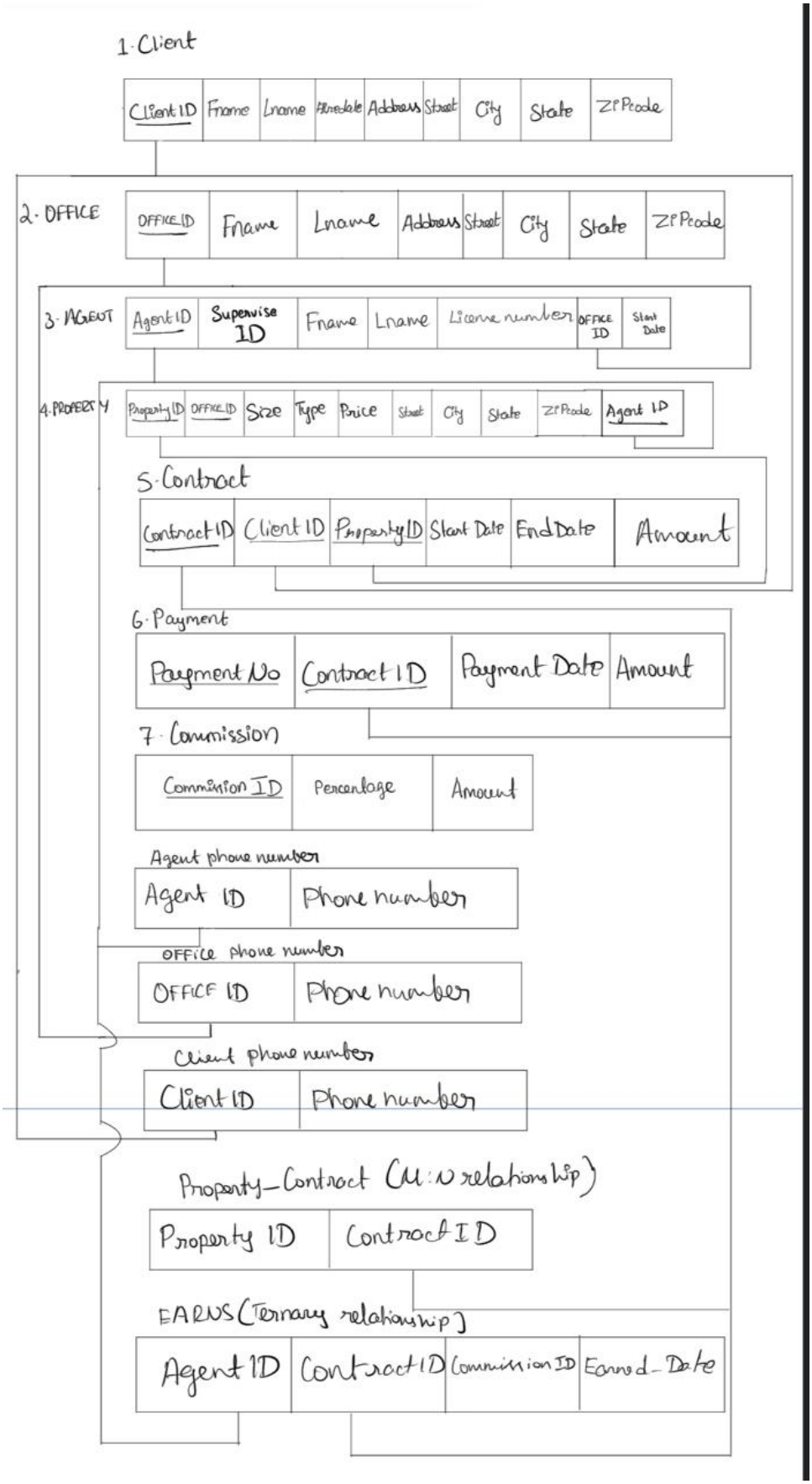
LIST OF SOFTWARES/TOOLS/PROGRAMMING LANGUAGES USED

- **Database:** MySQL 8.0
- **Backend Framework:** Flask (Python 3.x)
- **Frontend Technologies:** HTML5, CSS3 (Tailwind CSS), JavaScript (ES6+)
- **API Testing:** Postman / Browser DevTools
- **IDE:** Visual Studio Code

ER MODEL



ER TO RELATIONAL MAPPING



DDL STATEMENTS

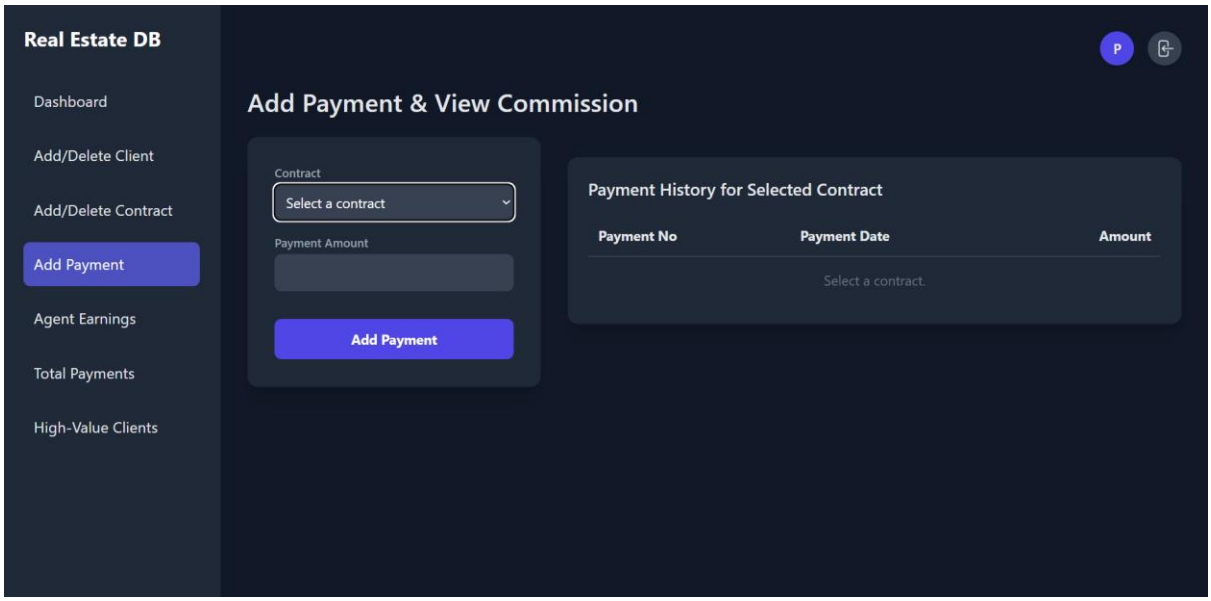
```
CREATE DATABASE IF NOT EXISTS RealEstateDB;
USE RealEstateDB;

CREATE TABLE IF NOT EXISTS Client (
    ClientID INT PRIMARY KEY AUTO_INCREMENT,
    Fname VARCHAR(50),
    Lname VARCHAR(50),
    HireDate DATE,
    AddressStreet VARCHAR(100),
    City VARCHAR(50),
    State VARCHAR(50),
    ZIPCode VARCHAR(10)
);

CREATE TABLE IF NOT EXISTS Office (
    OfficeID INT PRIMARY KEY AUTO_INCREMENT,
    Fname VARCHAR(50),
    Lname VARCHAR(50),
    AddressStreet VARCHAR(100),
    City VARCHAR(50),
    State VARCHAR(50),
    ZIPCode VARCHAR(10)
);

CREATE TABLE IF NOT EXISTS Agent (
    AgentID INT PRIMARY KEY AUTO_INCREMENT,
    SuperviseID INT NULL,
    Fname VARCHAR(50),
    Lname VARCHAR(50),
    LicenseNumber VARCHAR(30),
    OfficeID INT,
    StartDate DATE,
    FOREIGN KEY (OfficeID) REFERENCES Office(OfficeID),
    FOREIGN KEY (SuperviseID) REFERENCES Agent(AgentID)
);

CREATE TABLE IF NOT EXISTS Property (
    PropertyID INT PRIMARY KEY AUTO_INCREMENT,
    OfficeID INT,
    Size DECIMAL(10,2),
    Type VARCHAR(50),
```

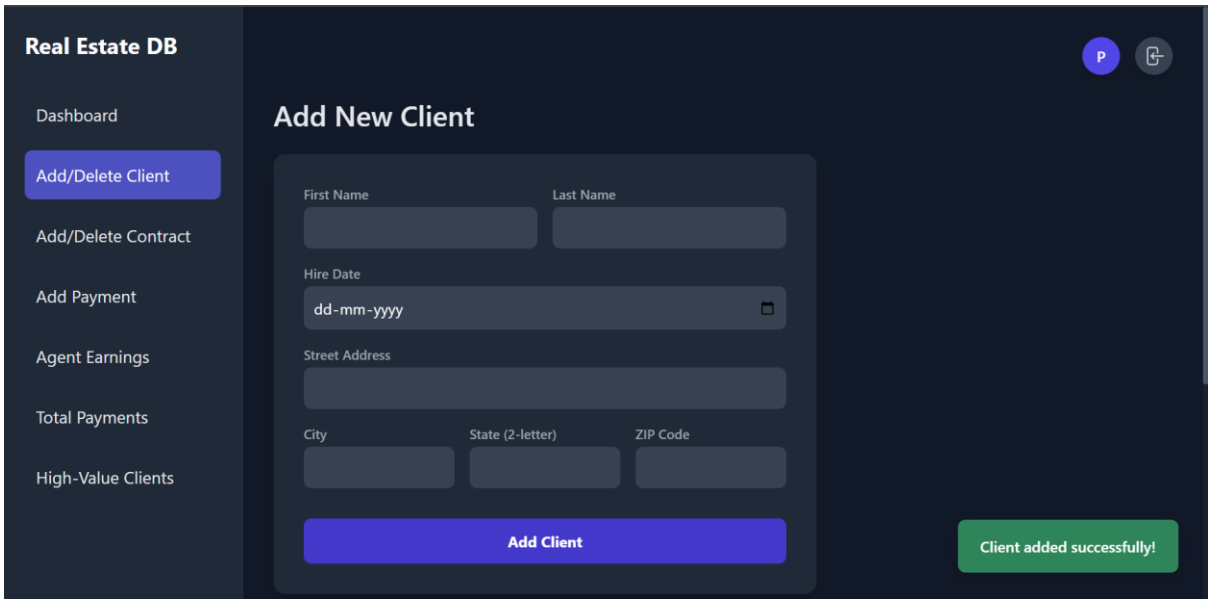
```
    Price DECIMAL(12,2),
    Street VARCHAR(100),
    City VARCHAR(50),
    State VARCHAR(50),
    ZIPCode VARCHAR(10),
    AgentID INT,
    FOREIGN KEY (OfficeID) REFERENCES Office(OfficeID),
    FOREIGN KEY (AgentID) REFERENCES Agent(AgentID)
);

CREATE TABLE IF NOT EXISTS Contract (
    ContractID INT PRIMARY KEY AUTO_INCREMENT,
    ClientID INT,
    StartDate DATE,
    EndDate DATE,
    Amount DECIMAL(12,2),
    FOREIGN KEY (ClientID) REFERENCES Client(ClientID)
);

CREATE TABLE IF NOT EXISTS Payment (
    PaymentNo INT PRIMARY KEY AUTO_INCREMENT,
    ContractID INT,
    PaymentDate DATE,
    Amount DECIMAL(12,2),
    FOREIGN KEY (ContractID) REFERENCES Contract(ContractID)
);

CREATE TABLE IF NOT EXISTS Commission (
    CommissionID INT PRIMARY KEY AUTO_INCREMENT,
    Percentage DECIMAL(5,2),
    Amount DECIMAL(12,2)
);

CREATE TABLE IF NOT EXISTS users (
    UserID INT AUTO_INCREMENT PRIMARY KEY,
    Username VARCHAR(100) UNIQUE NOT NULL,
    PasswordHash VARCHAR(255) NOT NULL,
    CreatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Multivalued attributes and relationships
CREATE TABLE IF NOT EXISTS AgentPhone (
    AgentID INT,
    PhoneNumber VARCHAR(15),
```

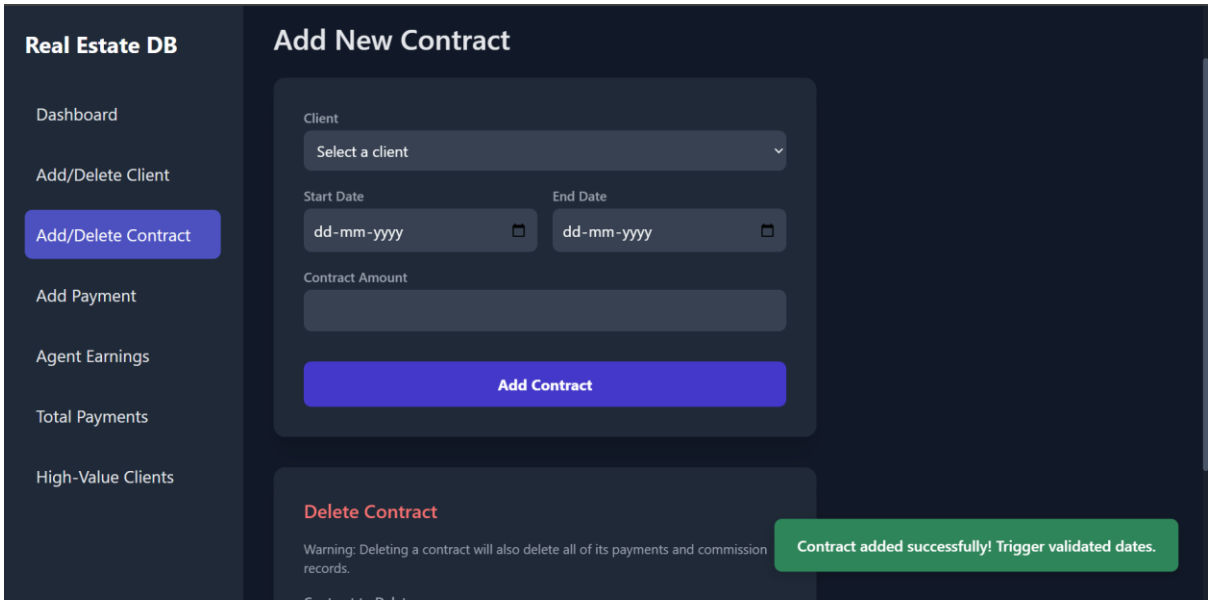
```
    PRIMARY KEY (AgentID, PhoneNumber),
    FOREIGN KEY (AgentID) REFERENCES Agent(AgentID)
);

CREATE TABLE IF NOT EXISTS OfficePhone (
    OfficeID INT,
    PhoneNumber VARCHAR(15),
    PRIMARY KEY (OfficeID, PhoneNumber),
    FOREIGN KEY (OfficeID) REFERENCES Office(OfficeID)
);

CREATE TABLE IF NOT EXISTS ClientPhone (
    ClientID INT,
    PhoneNumber VARCHAR(15),
    PRIMARY KEY (ClientID, PhoneNumber),
    FOREIGN KEY (ClientID) REFERENCES Client(ClientID)
);

CREATE TABLE IF NOT EXISTS PropertyContract (
    PropertyID INT,
    ContractID INT,
    PRIMARY KEY (PropertyID, ContractID),
    FOREIGN KEY (PropertyID) REFERENCES Property(PropertyID),
    FOREIGN KEY (ContractID) REFERENCES Contract(ContractID)
);

CREATE TABLE IF NOT EXISTS Earns (
    AgentID INT,
    ContractID INT,
    CommissionID INT,
    EarnedDate DATE,
    PRIMARY KEY (AgentID, ContractID, CommissionID),
    FOREIGN KEY (AgentID) REFERENCES Agent(AgentID),
    FOREIGN KEY (ContractID) REFERENCES Contract(ContractID),
    FOREIGN KEY (CommissionID) REFERENCES Commission(CommissionID)
);
```

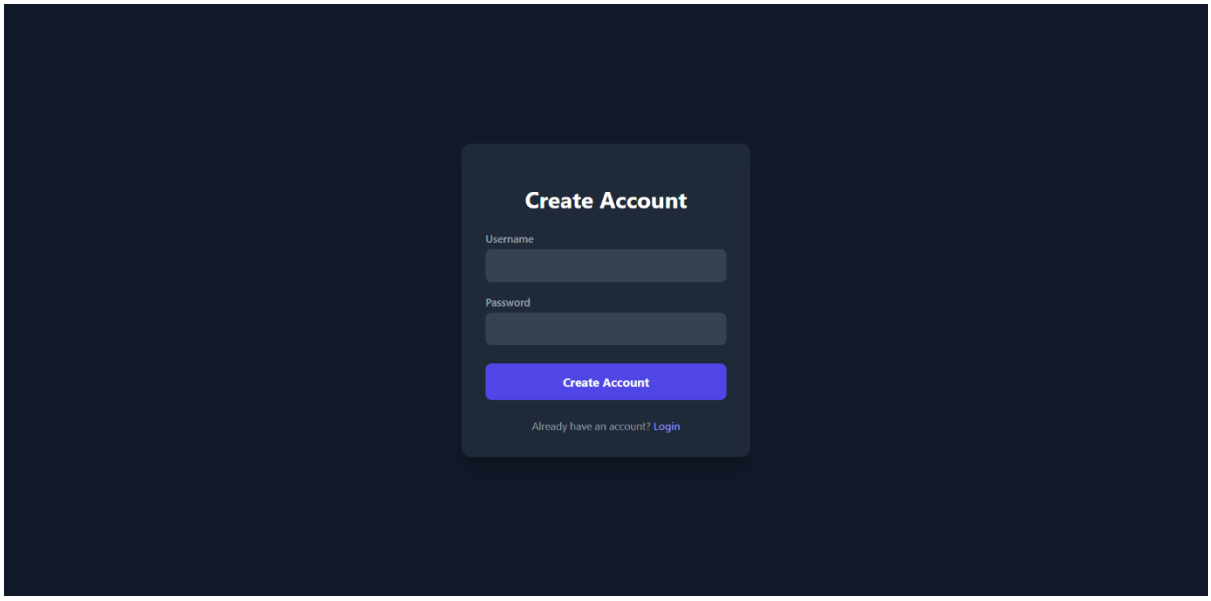
DML STATEMENTS (CRUD OPERATION SCREENSHOTS)

Create(C):

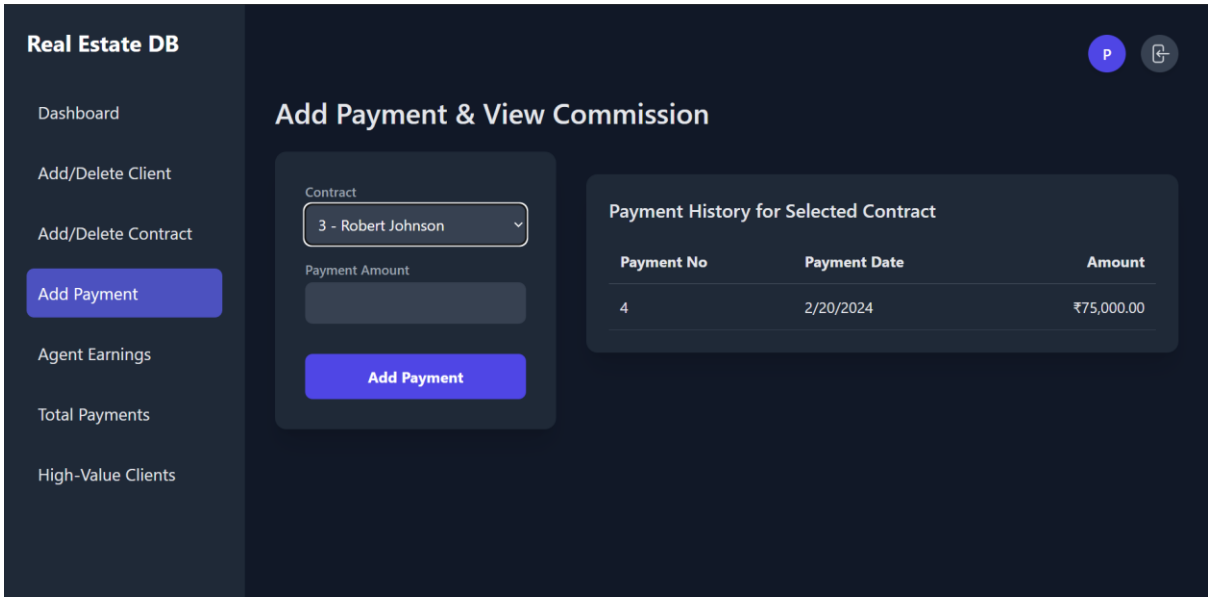


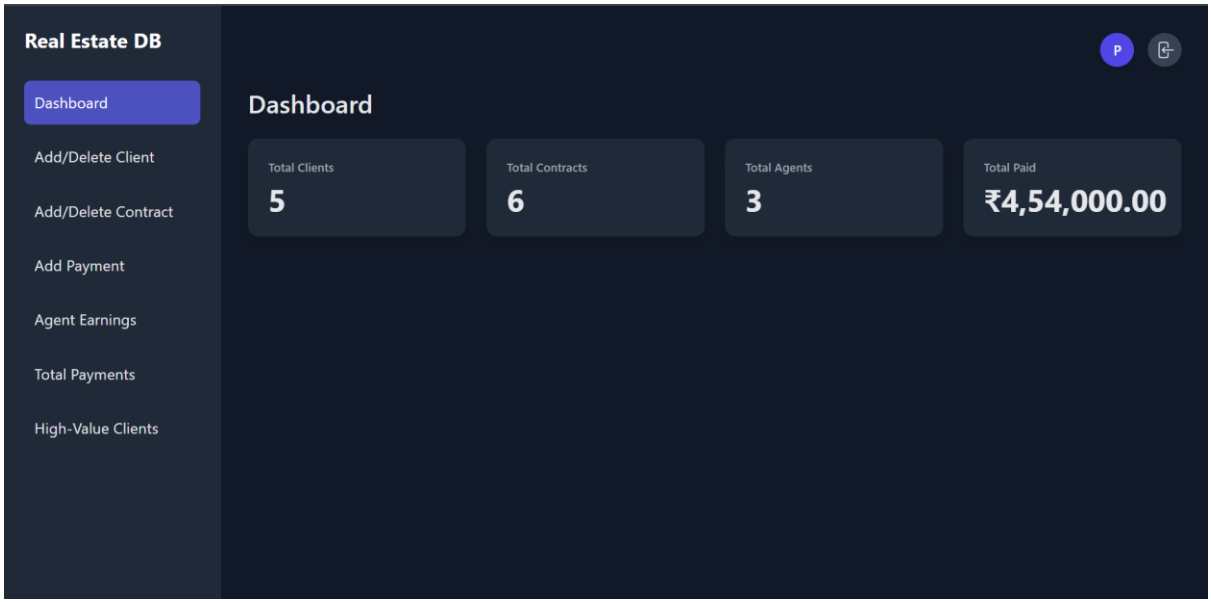



Real Estate DB
Dashboard
Add/Delete Client
Add/Delete Contract
Add Payment
Agent Earnings
Total Payments
High-Value Clients
Add New Contract
Client
Select a client
Start Date
dd-mm-yyyy
End Date
dd-mm-yyyy
Contract Amount
Add Contract
Delete Contract
Warning: Deleting a contract will also delete all of its payments and commission records.
Contract added successfully! Trigger validated dates.

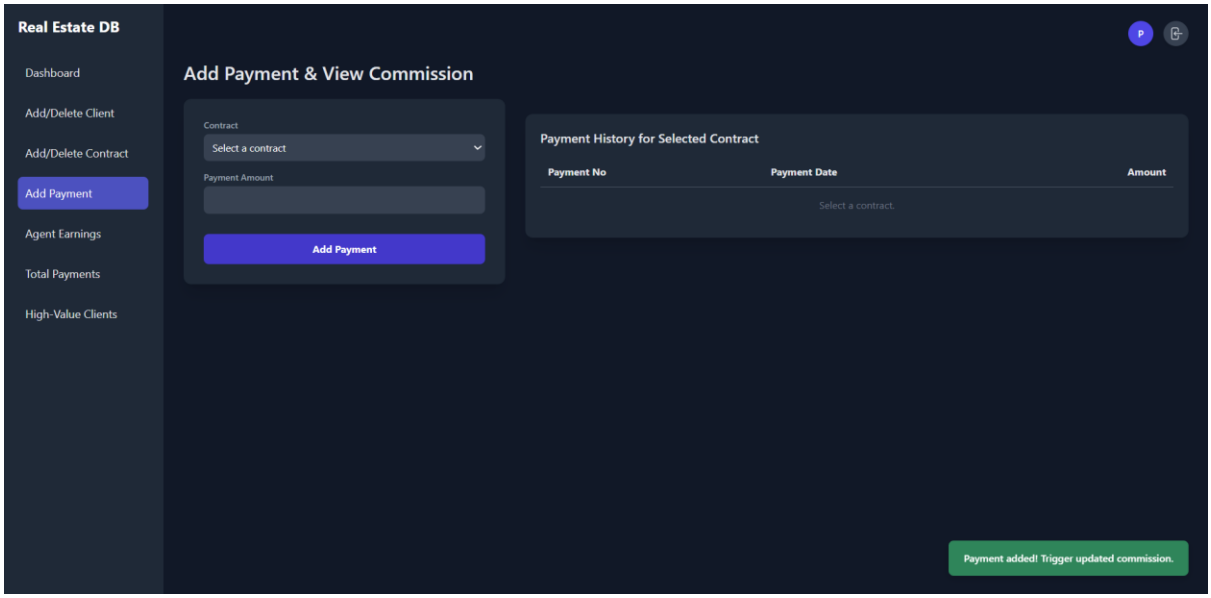

Create Account
Username
Password
Create Account
Already have an account? Login

READ(R):

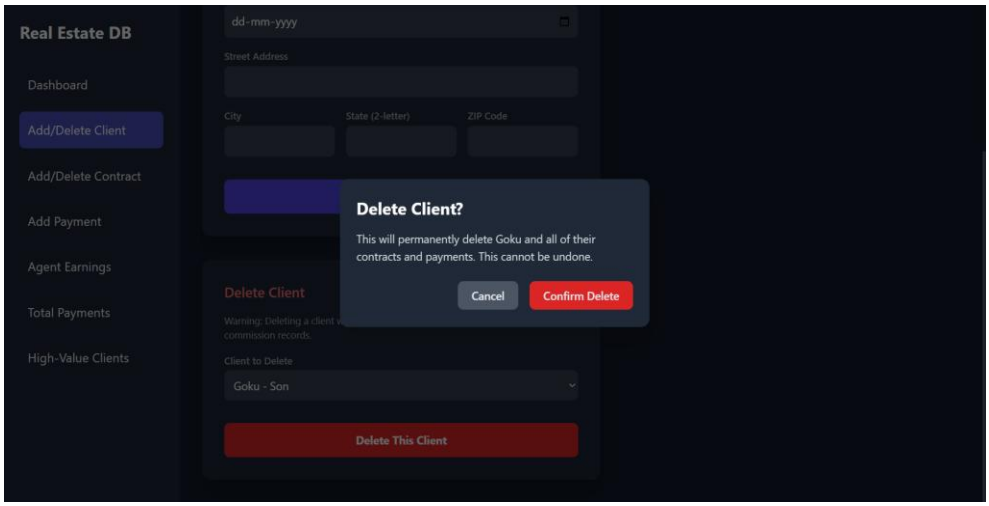


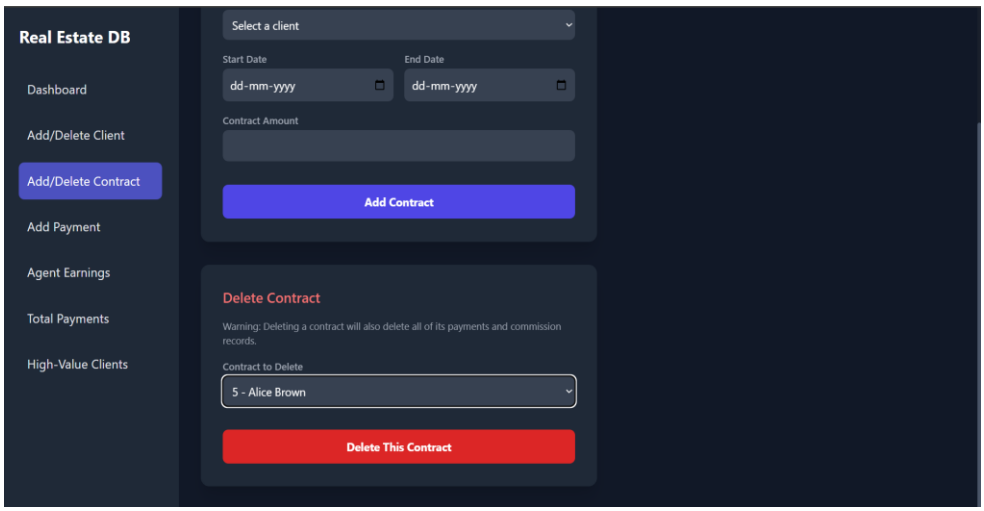


Update (U):


Real Estate DB
Dashboard
Add/Delete Client
Add/Delete Contract
Add Payment
Agent Earnings
Total Payments
High-Value Clients
Add Payment & View Commission
Contract
Select a contract
Payment Amount
Add Payment
Payment History for Selected Contract
Payment No
Payment Date
Amount
Select a contract.
Payment added! Trigger updated commission.

Delete (D):


Real Estate DB
Dashboard
Add/Delete Client
Add/Delete Contract
Add Payment
Agent Earnings
Total Payments
High-Value Clients
dd-mm-yyyy
Street Address
City
State (2-letter)
ZIP Code
Delete Client?
This will permanently delete Goku and all of their contracts and payments. This cannot be undone.
Cancel
Confirm Delete
Delete Client
Client to Delete
Goku - Son
Delete This Client


Real Estate DB
Dashboard
Add/Delete Client
Add/Delete Contract
Add Payment
Agent Earnings
Total Payments
High-Value Clients
Select a client
Start Date
End Date
dd-mm-yyyy
dd-mm-yyyy
Contract Amount
Add Contract
Delete Contract
Warning: Deleting a contract will also delete all of its payments and commission records.
Contract to Delete
5 - Alice Brown
Delete This Contract

QUERIES (JOIN QUERY, AGGREGATE FUNCTION QUERIES AND NESTED QUERY)

Join Query
Retrieves agent earnings by joining Earns, Contract, and Commission tables.

```
SELECT e.AgentID, c.Amount, co.Percentage, co.Amount
FROM earns e
JOIN contract c ON e.ContractID = c.ContractID
JOIN commission co ON e.CommissionID = co.CommissionID
WHERE e.AgentID = p_AgentID;
```

Aggregate Query
Calculates total stats for the dashboard.

```
SELECT COUNT(*) FROM client;
SELECT SUM(Amount) FROM payment;
```

Nested Query
Identifies clients with high-value contracts.

```
SELECT c.Fname, c.Lname, co.Amount
FROM client c
JOIN contract co ON c.ClientID = co.ClientID
WHERE co.Amount > (SELECT AVG(Amount) FROM contract);
```

STORED PROCEDURE, FUNCTIONS AND TRIGGERS

Stored Procedure: GetClientsWithHighValueContracts

```
CREATE PROCEDURE GetClientsWithHighValueContracts()
BEGIN
   SELECT c.ClientID, c.Fname, c.Lname, co.ContractID, co.Amount
   FROM client c
   JOIN contract co ON c.ClientID = co.ClientID
   WHERE co.Amount > (SELECT AVG(Amount) FROM contract)
   ORDER BY co.Amount DESC;
END;
```

Function: GetTotalPayment

```
CREATE FUNCTION GetTotalPayment(p_contractId INT)
RETURNS DECIMAL(12,2)
DETERMINISTIC
BEGIN
   DECLARE total DECIMAL(12,2);
   SELECT IFNULL(SUM(Amount), 0) INTO total
   FROM payment WHERE ContractID = p_contractId;
   RETURN total;
END;
```

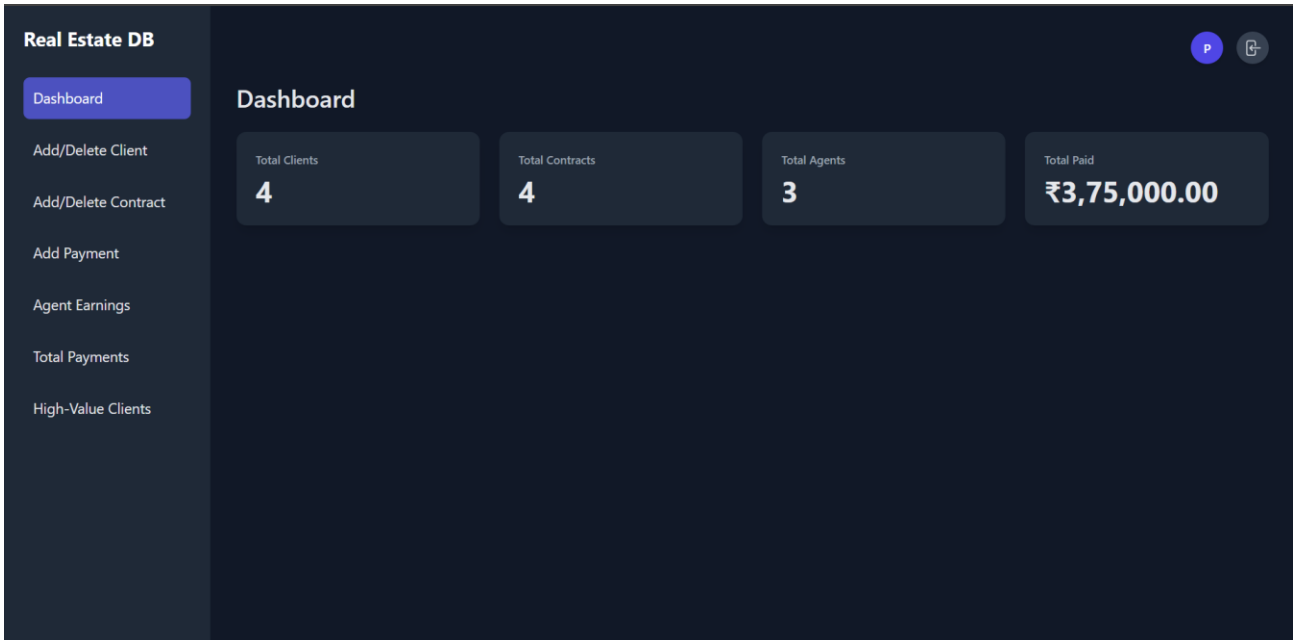
Trigger: update_commission_after_payment

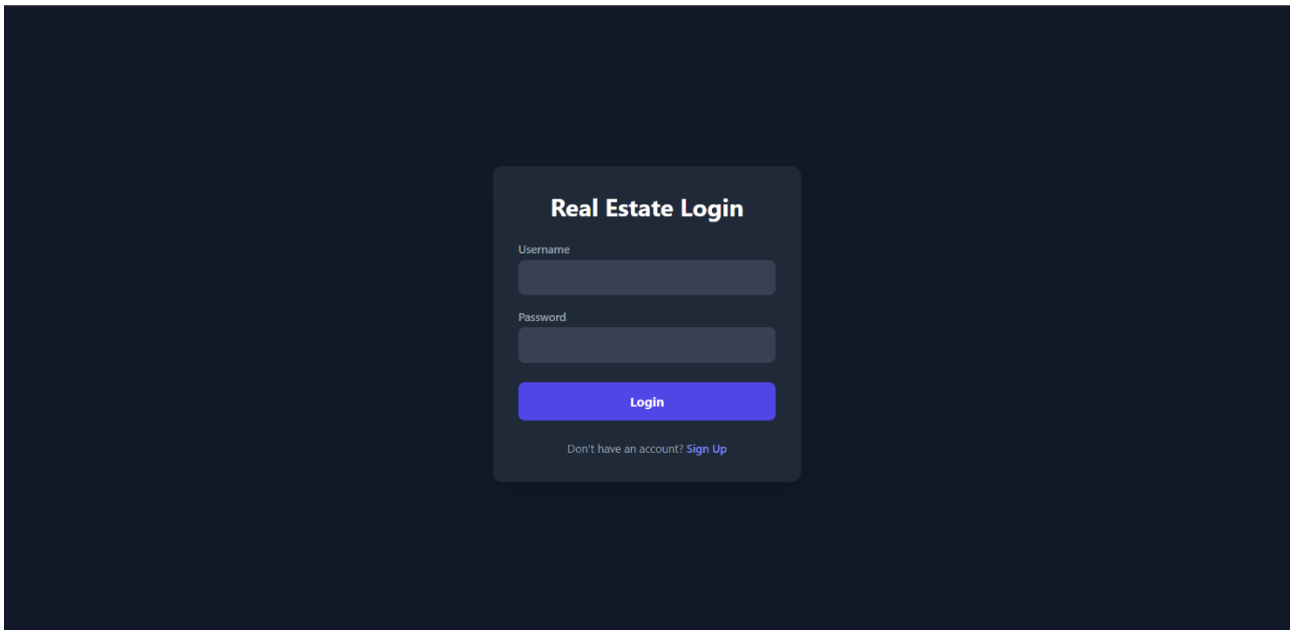
```
CREATE TRIGGER update_commission_after_payment
AFTER INSERT ON payment
FOR EACH ROW
BEGIN
   UPDATE commission AS co
   JOIN earns AS e ON co.CommissionID = e.CommissionID
   SET co.Amount = co.Amount + (NEW.Amount * (co.Percentage / 100))
   WHERE e.ContractID = NEW.ContractID;
END;
```

FRONT END DEVELOPMENT (FUNCTIONALITIES/FEATURES OF THE APPLICATION)

The frontend is a single-page application built with HTML, Tailwind CSS, and JavaScript. Key features include:

- **Dashboard:** Displays real-time aggregate statistics.
- **Forms:** Intuitive forms for adding clients, contracts, and payments with client-side validation.
- **Interactive Reports:** Dynamic tables for viewing agent earnings and high-value clients.
- **Secure Authentication:** JWT-based login system.





SQL QUERIES USED IN THE PROJECT (.SQL FILE)

01_schema.sql

```
-- Create Database
CREATE DATABASE IF NOT EXISTS RealEstateDB;

-- Select Database
USE RealEstateDB;
```

```
-- ============================
-- CREATE TABLES
-- ============================

-- 1. Client
CREATE TABLE IF NOT EXISTS Client (
    ClientID INT PRIMARY KEY AUTO_INCREMENT,
    Fname VARCHAR(50),
    Lname VARCHAR(50),
    HireDate DATE,
    AddressStreet VARCHAR(100),
    City VARCHAR(50),
    State VARCHAR(50),
    ZIPCode VARCHAR(10)
);

-- 2. Office
CREATE TABLE IF NOT EXISTS Office (
    OfficeID INT PRIMARY KEY AUTO_INCREMENT,
    Fname VARCHAR(50),
    Lname VARCHAR(50),
    AddressStreet VARCHAR(100),
    City VARCHAR(50),
    State VARCHAR(50),
    ZIPCode VARCHAR(10)
);

-- 3. Agent
CREATE TABLE IF NOT EXISTS Agent (
    AgentID INT PRIMARY KEY AUTO_INCREMENT,
    SuperviseID INT NULL,
    Fname VARCHAR(50),
    Lname VARCHAR(50),
    LicenseNumber VARCHAR(30),
    OfficeID INT,
    StartDate DATE,
    FOREIGN KEY (OfficeID) REFERENCES Office(OfficeID),
    FOREIGN KEY (SuperviseID) REFERENCES Agent(AgentID)
);

-- 4. Property
CREATE TABLE IF NOT EXISTS Property (
    PropertyID INT PRIMARY KEY AUTO_INCREMENT,
```

```
    OfficeID INT,
    Size DECIMAL(10,2),
    Type VARCHAR(50),
    Price DECIMAL(12,2),
    Street VARCHAR(100),
    City VARCHAR(50),
    State VARCHAR(50),
    ZIPCode VARCHAR(10),
    AgentID INT,
    FOREIGN KEY (OfficeID) REFERENCES Office(OfficeID),
    FOREIGN KEY (AgentID) REFERENCES Agent(AgentID)
);

-- 5. Contract
CREATE TABLE IF NOT EXISTS Contract (
    ContractID INT PRIMARY KEY AUTO_INCREMENT,
    ClientID INT,
    StartDate DATE,
    EndDate DATE,
    Amount DECIMAL(12,2),
    FOREIGN KEY (ClientID) REFERENCES Client(ClientID)
);

-- 6. Payment
CREATE TABLE IF NOT EXISTS Payment (
    PaymentNo INT PRIMARY KEY AUTO_INCREMENT,
    ContractID INT,
    PaymentDate DATE,
    Amount DECIMAL(12,2),
    FOREIGN KEY (ContractID) REFERENCES Contract(ContractID)
);

-- 7. Commission
CREATE TABLE IF NOT EXISTS Commission (
    CommissionID INT PRIMARY KEY AUTO_INCREMENT,
    Percentage DECIMAL(5,2),
    Amount DECIMAL(12,2)
);

-- 8. Users (FOR LOGIN)
CREATE TABLE IF NOT EXISTS users (
    UserID INT AUTO_INCREMENT PRIMARY KEY,
    Username VARCHAR(100) UNIQUE NOT NULL,
    PasswordHash VARCHAR(255) NOT NULL,
```

```
    CreatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Phone numbers (multivalued attributes)
CREATE TABLE IF NOT EXISTS AgentPhone (
    AgentID INT,
    PhoneNumber VARCHAR(15),
    PRIMARY KEY (AgentID, PhoneNumber),
    FOREIGN KEY (AgentID) REFERENCES Agent(AgentID)
);

CREATE TABLE IF NOT EXISTS OfficePhone (
    OfficeID INT,
    PhoneNumber VARCHAR(15),
    PRIMARY KEY (OfficeID, PhoneNumber),
    FOREIGN KEY (OfficeID) REFERENCES Office(OfficeID)
);

CREATE TABLE IF NOT EXISTS ClientPhone (
    ClientID INT,
    PhoneNumber VARCHAR(15),
    PRIMARY KEY (ClientID, PhoneNumber),
    FOREIGN KEY (ClientID) REFERENCES Client(ClientID)
);

-- Property-Contract (M:N relationship)
CREATE TABLE IF NOT EXISTS PropertyContract (
    PropertyID INT,
    ContractID INT,
    PRIMARY KEY (PropertyID, ContractID),
    FOREIGN KEY (PropertyID) REFERENCES Property(PropertyID),
    FOREIGN KEY (ContractID) REFERENCES Contract(ContractID)
);

-- Earns (Ternary Relationship: Agent–Contract–Commission)
CREATE TABLE IF NOT EXISTS Earns (
    AgentID INT,
    ContractID INT,
    CommissionID INT,
    EarnedDate DATE,
    PRIMARY KEY (AgentID, ContractID, CommissionID),
    FOREIGN KEY (AgentID) REFERENCES Agent(AgentID),
    FOREIGN KEY (ContractID) REFERENCES Contract(ContractID),
    FOREIGN KEY (CommissionID) REFERENCES Commission(CommissionID)
```

```
);

-- ============================
-- INSERT SAMPLE DATA
-- ============================

-- Clear tables before inserting (optional, good for testing)
-- SET FOREIGN_KEY_CHECKS = 0;
-- TRUNCATE TABLE Earns;
-- TRUNCATE TABLE PropertyContract;
-- TRUNCATE TABLE ClientPhone;
-- TRUNCATE TABLE OfficePhone;
-- TRUNCATE TABLE AgentPhone;
-- TRUNCATE TABLE Commission;
-- TRUNCATE TABLE Payment;
-- TRUNCATE TABLE Contract;
-- TRUNCATE TABLE Property;
-- TRUNCATE TABLE Agent;
-- TRUNCATE TABLE Office;
-- TRUNCATE TABLE Client;
-- SET FOREIGN_KEY_CHECKS = 1;

-- Clients
INSERT INTO Client (Fname, Lname, HireDate, AddressStreet, City, State, ZIPCode)
VALUES
('John', 'Smith', '2021-01-15', '123 Main St', 'New York', 'NY', '10001'),
('Alice', 'Brown', '2022-03-20', '456 Oak Ave', 'Los Angeles', 'CA', '90001'),
('Robert', 'Johnson', '2023-05-10', '789 Pine Rd', 'Chicago', 'IL', '60601');

-- Offices
INSERT INTO Office (Fname, Lname, AddressStreet, City, State, ZIPCode)
VALUES
('Michael', 'Davis', '12 Wall St', 'New York', 'NY', '10005'),
('Sophia', 'Taylor', '34 Sunset Blvd', 'Los Angeles', 'CA', '90028');

-- Agents
INSERT INTO Agent (SuperviseID, Fname, Lname, LicenseNumber, OfficeID, StartDate)
VALUES
(NULL, 'James', 'Williams', 'LIC123', 1, '2020-06-01'),
(1, 'Emma', 'Martinez', 'LIC456', 1, '2021-07-15'),
(NULL, 'Liam', 'Garcia', 'LIC789', 2, '2022-02-10');

-- Properties
INSERT INTO Property (OfficeID, Size, Type, Price, Street, City, State, ZIPCode, AgentID)
```

```
VALUES
(1, 1200.50, 'Apartment', 300000.00, '101 Park Ave', 'New York', 'NY', '10010', 1),
(1, 2000.00, 'House', 550000.00, '202 Lexington Ave', 'New York', 'NY', '10011', 2),
(2, 1500.75, 'Condo', 400000.00, '303 Sunset Dr', 'Los Angeles', 'CA', '90012', 3);

-- Contracts
INSERT INTO Contract (ClientID, StartDate, EndDate, Amount)
VALUES
(1, '2023-01-01', '2023-12-31', 500000.00),
(2, '2023-06-01', '2024-05-31', 300000.00),
(3, '2024-01-01', '2024-12-31', 450000.00);

-- Payments
INSERT INTO Payment (ContractID, PaymentDate, Amount)
VALUES
(1, '2023-02-15', 100000.00),
(1, '2023-06-15', 150000.00),
(2, '2023-07-10', 50000.00),
(3, '2024-02-20', 75000.00);

-- Commissions
INSERT INTO Commission (Percentage, Amount)
VALUES
(2.5, 0.00), -- Base amount should probably be 0, and updated by trigger
(3.0, 0.00),
(1.5, 0.00);

-- Phone Numbers
INSERT INTO AgentPhone (AgentID, PhoneNumber)
VALUES
(1, '111-222-3333'),
(2, '222-333-4444'),
(3, '333-444-5555');

INSERT INTO OfficePhone (OfficeID, PhoneNumber)
VALUES
(1, '555-123-4567'),
(2, '555-987-6543');

INSERT INTO ClientPhone (ClientID, PhoneNumber)
VALUES
(1, '999-111-2222'),
(2, '999-222-3333'),
(3, '999-333-4444');
```

```
-- Property-Contract (M:N relationship)
INSERT INTO PropertyContract (PropertyID, ContractID)
VALUES
(1, 1),
(2, 2),
(3, 3);

-- Earns (ternary relationship)
INSERT INTO Earns (AgentID, ContractID, CommissionID, EarnedDate)
VALUES
(1, 1, 1, '2023-02-15'),
(2, 2, 2, '2023-07-10'),
(3, 3, 3, '2024-02-20');
```

02_logic.sql

```
USE realestatedb;

-- ============================================================
--  TRIGGERS
-- ============================================================

DELIMITER //

--  ◇ Trigger 1: Validate that EndDate > StartDate in Contract
DROP TRIGGER IF EXISTS validate_contract_dates;
CREATE TRIGGER validate_contract_dates
BEFORE INSERT ON contract
FOR EACH ROW
BEGIN
   IF NEW.EndDate <= NEW.StartDate THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = '❌ End Date must be greater than Start Date';
   END IF;
END;
//

--  ◇ Trigger 2: Automatically update Commission Amount after new Payment
DROP TRIGGER IF EXISTS update_commission_after_payment;
CREATE TRIGGER update_commission_after_payment
AFTER INSERT ON payment
FOR EACH ROW
```

```
BEGIN
   UPDATE commission AS co
   JOIN earns AS e ON co.CommissionID = e.CommissionID
   SET co.Amount = co.Amount + (NEW.Amount * (co.Percentage / 100))
   WHERE e.ContractID = NEW.ContractID;
END;
//

DELIMITER ;

-- ============================================================
--  FUNCTIONS
-- ============================================================

DELIMITER //

--  ◇ Function: Get total payment amount for a contract
DROP FUNCTION IF EXISTS GetTotalPayment;
CREATE FUNCTION GetTotalPayment(p_contractId INT)
RETURNS DECIMAL(12,2)
DETERMINISTIC
BEGIN
   DECLARE total DECIMAL(12,2);
   SELECT IFNULL(SUM(Amount), 0)
   INTO total
   FROM payment
   WHERE ContractID = p_contractId;
   RETURN total;
END;
//

DELIMITER ;

-- ============================================================
--  STORED PROCEDURES
-- ============================================================

DELIMITER //

--  ◇ Procedure 1: Add new Contract safely
DROP PROCEDURE IF EXISTS AddNewContract;
CREATE PROCEDURE AddNewContract(
   IN p_ClientID INT,
   IN p_StartDate DATE,
```

```
    IN p_EndDate DATE,
    IN p_Amount DECIMAL(12,2)
)
BEGIN
    INSERT INTO contract (ClientID, StartDate, EndDate, Amount)
    VALUES (p_ClientID, p_StartDate, p_EndDate, p_Amount);
END;
//

-- ◇ Procedure 2: View Agent Earnings (joins earns + contract + commission)
DROP PROCEDURE IF EXISTS GetAgentEarnings;
CREATE PROCEDURE GetAgentEarnings(IN p_AgentID INT)
BEGIN
    SELECT
        e.AgentID,
        e.ContractID,
        c.Amount AS ContractAmount,
        e.CommissionID,
        co.Percentage,
        (c.Amount * co.Percentage / 100) AS PotentialEarning,
        co.Amount AS ActualEarnedAmount
    FROM earns e
    JOIN contract c ON e.ContractID = c.ContractID
    JOIN commission co ON e.CommissionID = co.CommissionID
    WHERE e.AgentID = p_AgentID;
END;
//

-- ◇ Procedure 3: Get High-Value Clients (Nested Query)
-- This fulfills the "Nested Query" requirement for your project.
DROP PROCEDURE IF EXISTS GetClientsWithHighValueContracts;
CREATE PROCEDURE GetClientsWithHighValueContracts()
BEGIN
    SELECT
        c.ClientID,
        c.Fname,
        c.Lname,
        co.ContractID,
        co.Amount
    FROM client c
    JOIN contract co ON c.ClientID = co.ClientID
    WHERE co.Amount > (
        -- Nested Query: Calculates average contract amount
        SELECT AVG(Amount)
```

```
        FROM contract
    )
    ORDER BY co.Amount DESC;
END;
//

DELIMITER ;
```

GITHUB REPO LINK

https://github.com/GVishwasReddy/Real_Estate_Management_DBMS_MiniProject