Real Estate Property Management

SQL PROJECT

1.1 INTRODUCTION

The Real Estate Property Management System is an SQL-driven application designed to manage and streamline the operations of real estate businesses. This system efficiently handles various critical functions, including property listings, unit management, tenant information, lease agreements, maintenance requests, and payment tracking. By leveraging SQL, the system ensures robust data management, facilitating seamless storage, retrieval, and manipulation of complex data related to real estate operations. This project addresses the need for an organized and efficient way to manage real estate properties, ensuring that property managers can maintain accurate records, optimize rental income, and provide better service to tenants. The Real Estate Property Management System not only simplifies day-to-day tasks but also provides insightful reports and analytics to support informed decision-making, ultimately enhancing the overall efficiency and profitability of the real estate business.

1.2 OBJECTIVE

The primary objective of the Real Estate Property Management System is to design and implement a comprehensive SQL-based platform that addresses the multifaceted needs of property management. This system is intended to centralize the management of various real estate operations, providing an all-in-one solution for property managers to effectively oversee their portfolio.

1.2.1 Streamlining Property Listings and Unit Management:

The system aims to create an organized and accessible database for all properties and their respective units. This includes detailed records of property attributes, unit types, rental prices, and availability status, ensuring that property managers can easily track and update property information in real-time.

1.2.2 Enhancing Tenant and Lease Management:

A critical objective is to maintain accurate and up-to-date records of tenant information, lease agreements, and related documentation. The system seeks to facilitate seamless tenant onboarding, track lease terms and conditions, and automate the renewal or termination processes, thereby reducing manual errors and ensuring compliance with legal standards.

1.2.3 Efficient Maintenance Request Tracking and Resolution:

The system is designed to handle maintenance requests efficiently, from submission to completion. By providing a streamlined process for tenants to report issues and for property managers to assign and monitor maintenance tasks, the system aims to improve response times and ensure that properties are well-maintained.

1.2.4 Optimizing Financial Transactions and Reporting:

Managing financial transactions is a core objective of the system. This includes tracking rent payments, security deposits, and other fees, generating automated reminders for due payments, and providing detailed financial reports. The goal is to ensure transparency and accuracy in financial dealings, which is essential for both property managers and tenants.

1.2.5 Supporting Informed Decision-Making:

By offering comprehensive data analytics and reporting tools, the system empowers property managers to make informed decisions. Whether it's analyzing occupancy rates, monitoring revenue trends, or evaluating maintenance costs, the system is designed to provide the insights necessary to optimize property management strategies and improve overall profitability. Enhancing

1.2.6 User Experience and Satisfaction:

Ultimately, the system aims to improve the experience for all users—property managers, tenants, and maintenance staff. By automating routine tasks, reducing paperwork, and providing easy access to critical information, the system seeks to enhance operational efficiency and ensure a high level of service and satisfaction for tenants.

1.3 QUERY

Properties TABLE

```
CREATE TABLE Properties (
    PropertyID INT PRIMARY KEY AUTO_INCREMENT,
    PropertyName VARCHAR(255) NOT NULL,
    Address VARCHAR(255) NOT NULL,
    City VARCHAR(100),
    State VARCHAR(50),
    ZipCode VARCHAR(10),
    PropertyType VARCHAR(50) -- e.g., Residential, Commercial
);
```

```
INSERT INTO Properties (PropertyName, Address, City, State, ZipCode, PropertyType) VALUES
('Green Valley Apartments', '123 Oak Street', 'Springfield', 'IL', '62704', 'Residential'),
('Sunset Villas', '456 Pine Avenue', 'Los Angeles', 'CA', '90001', 'Residential'),
('Downtown Office Complex', '789 Main Street', 'Chicago', 'IL', '60601', 'Commercial'),
('Lakeside Condominiums', '101 Maple Drive', 'Madison', 'WI', '53703', 'Residential'),
('Park Plaza', '202 Elm Street', 'Denver', 'CO', '80203', 'Commercial'),
('Riverside Towers', '303 Birch Lane', 'Austin', 'TX', '78701', 'Residential'),
('Oceanview Estates', '404 Cedar Road', 'Miami', 'FL', '33101', 'Residential'),
('Corporate Center', '505 Oak Avenue', 'San Francisco', 'CA', '94102', 'Commercial'),
('Suburban Plaza', '606 Spruce Street', 'Atlanta', 'GA', '30303', 'Commercial'),
('City Heights', '707 Walnut Lane', 'Seattle', 'WA', '98101', 'Residential'),
('Mountain Ridge', '808 Willow Street', 'Boulder', 'CO', '80302', 'Residential'),
('The Grand', '909 Pine Street', 'New York', 'NY', '10001', 'Commercial'),
('Bayview Apartments', '1010 Palm Avenue', 'San Diego', 'CA', '92101', 'Residential'),
('Metro Office Park', '1111 Cypress Road', 'Philadelphia', 'PA', '19103', 'Commercial'),
('Summit Towers', '1212 Ash Boulevard', 'Portland', 'OR', '97201', 'Residential'),
('Lakefront Residences', '1313 Fir Street', 'Minneapolis', 'MN', '55401', 'Residential'),
('Gateway Business Center', '1414 Redwood Drive', 'Dallas', 'TX', '75201', 'Commercial'),
('Westside Plaza', '1515 Cherry Lane', 'Phoenix', 'AZ', '85001', 'Commercial'),
('Skyline Apartments', '1616 Magnolia Avenue', 'Las Vegas', 'NV', '89101', 'Residential'),
('Silver Sands Resort', '1717 Poplar Street', 'Orlando', 'FL', '32801', 'Residential');
```

UNITS TABLE

```
CREATE TABLE Units (
    UnitID INT PRIMARY KEY AUTO_INCREMENT,
    PropertyID INT,
    UnitNumber VARCHAR(50),
    UnitType VARCHAR(50), -- e.g., Apartment, Office
    RentAmount DECIMAL(10, 2),
    IsAvailable BOOLEAN DEFAULT TRUE,
    FOREIGN KEY (PropertyID) REFERENCES Properties(PropertyID)
);

INSERT INTO Units (PropertyID, UnitNumber, UnitType, RentAmount, IsAvailable) VALUES
(1, 'A101', 'Apartment', 1200.00, TRUE),
(1, 'A102', 'Apartment', 1250.00, FALSE),
(2, 'B201', 'Apartment', 1500.00, TRUE),
(3, 'C301', 'Office', 3000.00, TRUE),
(4, 'D401', 'Apartment', 1800.00, TRUE),
(5, 'E501', 'Office', 4000.00, FALSE),
(6, 'F601', 'Apartment', 2000.00, TRUE),
(7, 'G701', 'Apartment', 2200.00, TRUE),
(8, 'H801', 'Office', 3500.00, FALSE),
(9, 'I901', 'Apartment', 1700.00, TRUE);
```

TENENTS TABLE

```
CREATE TABLE Tenants (
    TenantID INT PRIMARY KEY AUTO_INCREMENT,
```

```
    FirstName VARCHAR(100),
    LastName VARCHAR(100),
    Phone VARCHAR(15),
    Email VARCHAR(100),
    StartDate DATE
);
INSERT INTO Tenants (FirstName, LastName, Phone, Email, StartDate) VALUES
('John', 'Doe', '555-1234', 'john.doe@example.com', '2023-01-15'),
('Jane', 'Smith', '555-5678', 'jane.smith@example.com', '2023-02-01'),
('Robert', 'Johnson', '555-8765', 'robert.johnson@example.com', '2023-03-10'),
('Emily', 'Davis', '555-4321', 'emily.davis@example.com', '2023-04-20'),
('Michael', 'Brown', '555-9876', 'michael.brown@example.com', '2023-05-25'),
('Sarah', 'Wilson', '555-6543', 'sarah.wilson@example.com', '2023-06-05'),
('David', 'Martinez', '555-3456', 'david.martinez@example.com', '2023-07-15'),
('Jessica', 'Taylor', '555-7890', 'jessica.taylor@example.com', '2023-08-12'),
('Daniel', 'Anderson', '555-0987', 'daniel.anderson@example.com', '2023-09-01'),
('Laura', 'Thomas', '555-4567', 'laura.thomas@example.com', '2023-10-01');
```

LEASES TABLE

```
CREATE TABLE Leases (
    LeaseID INT PRIMARY KEY AUTO_INCREMENT,
    UnitID INT,
    TenantID INT,
    StartDate DATE,
    EndDate DATE,
    MonthlyRent DECIMAL(10, 2),
    DepositAmount DECIMAL(10, 2),
    LeaseStatus VARCHAR(50), -- e.g., Active, Terminated
    FOREIGN KEY (UnitID) REFERENCES Units(UnitID),
    FOREIGN KEY (TenantID) REFERENCES Tenants(TenantID)
);
INSERT INTO Leases (UnitID, TenantID, StartDate, EndDate, MonthlyRent, DepositAmount,
LeaseStatus) VALUES
(1, 1, '2023-01-15', '2024-01-15', 1200.00, 1200.00, 'Active'),
(2, 2, '2023-02-01', '2024-02-01', 1250.00, 1250.00, 'Terminated'),
(3, 3, '2023-03-10', '2024-03-10', 1500.00, 1500.00, 'Active'),
(4, 4, '2023-04-20', '2024-04-20', 3000.00, 3000.00, 'Active'),
(5, 5, '2023-05-25', '2024-05-25', 1800.00, 1800.00, 'Terminated'),
(6, 6, '2023-06-05', '2024-06-05', 2000.00, 2000.00, 'Active'),
(7, 7, '2023-07-15', '2024-07-15', 2200.00, 2200.00, 'Active'),
(8, 8, '2023-08-12', '2024-08-12', 3500.00, 3500.00, 'Terminated'),
(9, 9, '2023-09-01', '2024-09-01', 1700.00, 1700.00, 'Active'),
(10, 10, '2023-10-01', '2024-10-01', 1700.00, 1700.00, 'Active');
```

MaintenanceRequests TABLE

```
CREATE TABLE MaintenanceRequests (
    RequestID INT PRIMARY KEY AUTO_INCREMENT,
    UnitID INT,
```

```
    TenantID INT,
    RequestDate DATE,
    RequestDescription TEXT,
    Status VARCHAR(50), -- e.g., Pending, Completed
    CompletionDate DATE,
    FOREIGN KEY (UnitID) REFERENCES Units(UnitID),
    FOREIGN KEY (TenantID) REFERENCES Tenants(TenantID)
);

INSERT INTO MaintenanceRequests (UnitID, TenantID, RequestDate, RequestDescription, Status, CompletionDate) VALUES
(1, 1, '2023-02-01', 'Leaky faucet in the kitchen.', 'Completed', '2023-02-03'),
(2, 2, '2023-03-15', 'Broken window in the living room.', 'Completed', '2023-03-17'),
(3, 3, '2023-04-20', 'Air conditioning not working.', 'Pending', NULL),
(4, 4, '2023-05-05', 'Electrical issue in the bedroom.', 'Completed', '2023-05-07'),
(5, 5, '2023-06-10', 'Mold in the bathroom.', 'Completed', '2023-06-12'),
(6, 6, '2023-07-18', 'Garage door malfunction.', 'Pending', NULL),
(7, 7, '2023-08-22', 'Heating system needs servicing.', 'Completed', '2023-08-24'),
(8, 8, '2023-09-01', 'Pest control needed in the basement.', 'Completed', '2023-09-03'),
(9, 9, '2023-10-10', 'Plumbing issue in the laundry room.', 'Pending', NULL),
(10, 10, '2023-11-05', 'Ceiling leak in the hallway.', 'Completed', '2023-11-07');
```

Payments TABLE

```
CREATE TABLE Payments (
    PaymentID INT PRIMARY KEY AUTO_INCREMENT,
    LeaseID INT,
    PaymentDate DATE,
    AmountPaid DECIMAL(10, 2),
    PaymentType VARCHAR(50), -- e.g., Rent, Maintenance Fee
    FOREIGN KEY (LeaseID) REFERENCES Leases(LeaseID)
);

INSERT INTO Payments (LeaseID, PaymentDate, AmountPaid, PaymentType) VALUES
(1, '2023-02-01', 1200.00, 'Rent'),
(1, '2023-03-01', 1200.00, 'Rent'),
(2, '2023-02-15', 1250.00, 'Rent'),
(3, '2023-03-10', 1500.00, 'Rent'),
(4, '2023-04-20', 3000.00, 'Rent'),
(5, '2023-05-25', 1800.00, 'Rent'),
(6, '2023-06-05', 2000.00, 'Rent'),
(7, '2023-07-15', 2200.00, 'Rent'),
(8, '2023-08-12', 3500.00, 'Rent'),
(9, '2023-09-01', 1700.00, 'Rent');
```

Employees TABLE

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY AUTO_INCREMENT,
    FirstName VARCHAR(100),
    LastName VARCHAR(100),
```

```
    Role VARCHAR(50),
    Phone VARCHAR(15),
    Email VARCHAR(100),
    HireDate DATE
);

INSERT INTO Employees (FirstName, LastName, Role, Phone, Email, HireDate) VALUES
('Alice', 'Johnson', 'Property Manager', '555-1111', 'alice.johnson@example.com', '2022-01-15'),
('Bob', 'Smith', 'Maintenance Technician', '555-2222', 'bob.smith@example.com', '2022-02-20'),
('Carol', 'Williams', 'Leasing Agent', '555-3333', 'carol.williams@example.com', '2022-03-10'),
('David', 'Brown', 'Property Manager', '555-4444', 'david.brown@example.com', '2022-04-25'),
('Eve', 'Davis', 'Maintenance Supervisor', '555-5555', 'eve.davis@example.com', '2022-05-30'),
('Frank', 'Miller', 'Accountant', '555-6666', 'frank.miller@example.com', '2022-06-05'),
('Grace', 'Wilson', 'Receptionist', '555-7777', 'grace.wilson@example.com', '2022-07-12'),
('Henry', 'Moore', 'Security Guard', '555-8888', 'henry.moore@example.com', '2022-08-18'),
('Isabel', 'Taylor', 'Leasing Agent', '555-9999', 'isabel.taylor@example.com', '2022-09-25'),
('Jack', 'Anderson', 'Maintenance Technician', '555-0000', 'jack.anderson@example.com', '2022-10-01');
```

1.4 SIMPLE OPERATIONS

List All Properties

```
SELECT PropertyName, Address, City, State, ZipCode, PropertyType

FROM Properties;
```

Get Available Units by Property

```
SELECT PropertyName, UnitNumber, UnitType, RentAmount

FROM Units

JOIN Properties ON Units.PropertyID = Properties.PropertyID

WHERE IsAvailable = TRUE;
```

Monthly Rent Roll

```
SELECT Properties.PropertyName, Units.UnitNumber, Tenants.FirstName, Tenants.LastName, Leases.MonthlyRent, Payments.AmountPaid

FROM Leases JOIN Units ON Leases.UnitID = Units.UnitID

JOIN Properties ON Units.PropertyID = Properties.PropertyID

JOIN Tenants ON Leases.TenantID = Tenants.TenantID

LEFT JOIN Payments ON Leases.LeaseID = Payments.LeaseID AND MONTH(Payments.PaymentDate) = MONTH(CURRENT_DATE)

WHERE Leases.LeaseStatus = 'Active';
```

Overdue Payments

```
SELECT Tenants.FirstName, Tenants.LastName, Units.UnitNumber, Leases.MonthlyRent, (Leases.MonthlyRent - COALESCE(SUM(Payments.AmountPaid), 0)) AS AmountDue

FROM Leases

JOIN Units ON Leases.UnitID = Units.UnitID

JOIN Tenants ON Leases.TenantID = Tenants.TenantID

LEFT JOIN Payments ON Leases.LeaseID = Payments.LeaseID AND MONTH(Payments.PaymentDate) = MONTH(CURRENT_DATE)

WHERE Leases.LeaseStatus = 'Active'

GROUP BY Leases.LeaseID

HAVING AmountDue > 0;
```

1.5 Subqueries

Listing Properties with No Active Leases

```
SELECT t.FirstName,t.LastName,l.StartDate,l.EndDate

FROM Tenants t

JOIN Leases l ON t.TenantID = l.TenantID

WHERE l.LeaseID = (SELECT LeaseID FROM Leases WHERE LeaseStatus = 'Active'

    ORDER BY DATEDIFF(EndDate, StartDate) DESC

    LIMIT 1);
```

Finding the Most Expensive Rent Paid by a Tenant

```
SELECT p.PropertyName, p.Address, p.City, p.State

FROM Properties p WHERE p.PropertyID NOT IN (

    SELECT DISTINCT l.PropertyID

    FROM Leases l

    WHERE l.LeaseStatus = 'Active' );
```

Identifying Tenants with No Maintenance Requests

```
SELECT  t.FirstName t.LastName

FROM Tenants t

WHERE t.TenantID NOT IN (

SELECT DISTINCT TenantID

FROM MaintenanceRequests );
```

Calculating Average Rent for a Specific Property

```
SELECT
   p.PropertyName,AVG(u.RentAmount) AS AverageRent
FROM Properties p
JOIN Units u ON p.PropertyID = u.PropertyID
WHERE p.PropertyID = (
 SELECT PropertyID
  FROM   properties
     WHERE PropertyName = 'Sunset Apartments');
```

1.6 view

```
CREATE VIEW ActiveLeasesView AS
SELECT
l.LeaseID,t.FirstName,t.LastName,u.UnitNumber,u.UnitType,l.StartDate,l.EndDate,l.MonthlyRent
FROM Leases l
JOIN Tenants t ON l.TenantID = t.TenantID
JOIN Units u ON l.UnitID = u.UnitID
WHERE l.LeaseStatus = 'Active';

SELECT * FROM ActiveLeasesView;

UPDATE ActiveLeasesView
SET MonthlyRent = 1300.00
WHERE LeaseID = 1;

DROP VIEW ActiveLeasesView;
```

1.7 Stored Procedure

Creating a Stored Procedure

```
DELIMITER $$
CREATE PROCEDURE AddTenant(
   IN p_FirstName VARCHAR(100),
   IN p_LastName VARCHAR(100),
   IN p_Phone VARCHAR(15),
   IN p_Email VARCHAR(100),
   IN p_StartDate DATE
)
BEGIN
```

```
    INSERT INTO Tenants (FirstName, LastName, Phone, Email, StartDate)
    VALUES (p_FirstName, p_LastName, p_Phone, p_Email, p_StartDate);
END$$
DELIMITER ;
```

Executing a Stored Procedure

```
CALL AddTenant('John', 'Doe', '555-1234', 'john.doe@example.com', '2023-08-01');
```

Stored Procedure with Output Parameter

```
DELIMITER $$
CREATE PROCEDURE GetTotalRentPaid(
    IN p_TenantID INT,
    OUT p_TotalRent DECIMAL(10, 2)
)
BEGIN
    SELECT SUM(AmountPaid)
    INTO p_TotalRent
    FROM Payments
    WHERE LeaseID IN (
        SELECT LeaseID FROM Leases WHERE TenantID = p_TenantID
    );
END $$
DELIMITER ;
CALL GetTotalRentPaid(1, @TotalRent);
SELECT @TotalRent;
```

Conditional Logic in Stored Procedures

```
DELIMITER $$
CREATE PROCEDURE UpdateTenantInfo(
    IN p_TenantID INT,
    IN p_Phone VARCHAR(15),
    IN p_Email VARCHAR(100)
)
BEGIN
```

```
    IF EXISTS (SELECT 1 FROM Tenants WHERE TenantID = p_TenantID) THEN
        UPDATE Tenants
        SET Phone = p_Phone, Email = p_Email
        WHERE TenantID = p_TenantID;
    ELSE
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Tenant not found';
    END IF;
END $$
DELIMITER ;
```

1.8JOINS

INNER JOIN

```
SELECT t.FirstName,t.LastName,u.UnitNumber, u.UnitType, l.StartDate, l.EndDate
FROM Tenants t
INNER JOIN Leases l ON t.TenantID = l.TenantID
INNER JOIN Units u ON l.UnitID = u.UnitID
WHERE l.LeaseStatus = 'Active';
```

LEFT JOIN (or LEFT OUTER JOIN)

```
SELECT p.PropertyName, p.Address, mr.RequestDate, mr.RequestDescription, mr.Status
FROM Properties p
LEFT JOIN Units u ON p.PropertyID = u.PropertyID
LEFT JOIN MaintenanceRequests mr ON u.UnitID = mr.UnitID;
```

RIGHT JOIN (or RIGHT OUTER JOIN)

```
SELECT t.FirstName, t.LastName, p.PaymentDate, p.AmountPaid
FROM Tenants t
RIGHT JOIN Leases l ON t.TenantID = l.TenantID
RIGHT JOIN Payments p ON l.LeaseID = p.LeaseID;
```

FULL OUTER JOIN

```
SELECT u.UnitNumber, u.UnitType, l.StartDate, l.EndDate
FROM Units u
FULL OUTER JOIN Leases l ON u.UnitID = l.UnitID;
```

CROSS JOIN

```
SELECT t.FirstName,t.LastName, u.UnitNumber, u.UnitType
FROM Tenants t
CROSS JOIN Units u;
SELF JOIN
SELECT t1.FirstName AS FirstName1, t1.LastName AS LastName1,
   t2.FirstName AS FirstName2, t2.LastName AS LastName2
FROM Tenants t1
INNER JOIN Tenants t2 ON t1.LastName = t2.LastName
AND t1.TenantID != t2.TenantID;
```

1.9CONCLUSION

The Real Estate Property Management System successfully demonstrates the power and flexibility of SQL in managing complex real estate operations. Through the implementation of various SQL queries, joins, subqueries, views, and stored procedures, the system effectively handles critical aspects such as property listings, tenant management, lease administration, maintenance requests, and financial transactions.

This project highlights the importance of structured data management in ensuring accurate, efficient, and scalable operations within the real estate industry. By centralizing information and automating routine tasks, the system not only enhances operational efficiency but also supports better decision-making and improves tenant satisfaction.

Overall, the project showcases how SQL can be utilized to build a robust, reliable, and user-friendly property management solution, offering significant benefits to property managers, tenants, and other stakeholders involved in real estate management.