

**Group Members:**

* **Muhammad Fasih Farhaj (BSE233116)**
* **Hamna Shahid (BSE233070)**
* **Ibtesam Ul Haque (BSE233102)**

**Assignment Made By: Muhammad Fasih Farhaj Section: 02**

**Assignment #: 04**

**Date: 13-01-2024**

**Subject: Introduction to Database Systems**

**Description: Project Report**

**GitHub Link:** https://github.com/Fasih-131/Real-Estate-Management-System

**Submitted To: Ma’am Hina Rashid**

Table of Contents

[**1. Database Implementation** 3](#_Toc187648403)

[**Key Tables and Relationships** 3](#_Toc187648404)

[**Foreign Key Constraints** 3](#_Toc187648405)

[**2. Data Population** 3](#_Toc187648406)

[**3. Querying Capabilities** 4](#_Toc187648407)

[**Basic Queries** 4](#_Toc187648408)

[**Joins** 4](#_Toc187648409)

[**Aggregations** 4](#_Toc187648410)

[**Advanced Queries** 4](#_Toc187648411)

[**Sample Queries** 4](#_Toc187648412)

[**Efficiency and Flexibility** 5](#_Toc187648413)

[**Conclusion** 5](#_Toc187648414)

# **1. Database Implementation**

The Real Estate Database was meticulously designed to streamline the management of tenants, properties, leases, agents, and payments. It incorporates a normalized schema with well-defined relationships, ensuring data integrity and efficient querying.

## **Key Tables and Relationships**

* **Tenants:** This table stores tenant details such as tenant ID, name, contact information, and other relevant data.
* **Properties:** This table includes property information, including property ID, address, type, and the associated managing agent.
* **Leases:** This is the central table that connects tenants and properties, containing lease details like lease start and end dates, rent amounts, and associated tenant and property IDs.
* **Agents:** This table represents agents responsible for managing properties, with details such as agent ID and name.
* **Payments:** This table tracks all payments made by tenants, linking each payment to a lease through the lease ID.

## **Foreign Key Constraints**

To ensure data integrity and establish relationships:

* **Leases.tenant\_id** references **Tenants.tenant\_id**.
* **Leases.property\_id** references **Properties.id**.
* **Properties.agent\_id** references **Agents.agent\_id**.
* **Payments.lease\_id** references **Leases.lease\_id**.

Cascading operations (ON DELETE CASCADE) were implemented to handle deletions, ensuring that dependent records are automatically updated or removed as needed.

# **2. Data Population**

The database was populated using SQL INSERT statements. The dataset was crafted to represent realistic scenarios, including:

* **Tenants:** Individuals with unique tenant IDs and contact details.
* **Properties:** A variety of properties managed by specific agents.
* **Leases:** Records linking tenants and properties, including lease durations and rent amounts.
* **Agents:** Profiles of agents managing properties.
* **Payments:** Transactions linked to specific leases, including payment dates and amounts.

The use of foreign keys ensured that data integrity was maintained during population. All entries adhered to the constraints set in the schema, preventing invalid or inconsistent data.

# **3. Querying Capabilities**

The Real Estate Database was designed to handle a wide range of queries, supporting both basic and advanced data retrieval. Below are the primary types of queries supported:

## **Basic Queries**

* Retrieve all tenants, properties, leases, agents, or payments.
* Filter records based on specific conditions, such as tenants with high rent or properties managed by a particular agent.

## **Joins**

* Combine data across multiple tables to derive insights, such as retrieving all tenants and their leased properties.

## **Aggregations**

* Summarize data, such as calculating total rent collected, average rent per property, or the number of properties managed by each agent.

## **Advanced Queries**

* Nested queries to extract complex relationships, such as identifying properties not currently leased.
* Relational algebra queries, translated into SQL, for operations like selection (σ), projection (π), joins (⋉), and division.

## **Sample Queries**

* List all tenants who have a lease starting in 2025.
* Retrieve properties with rent below the average rent.
* Calculate the total rent collected by each property.
* Identify tenants whose lease duration exceeds 12 months.

## **Efficiency and Flexibility**

The database schema and query capabilities demonstrate:

* **Efficiency:** Optimized relationships and indexes ensure quick query execution.
* **Flexibility:** The system can handle diverse real-world scenarios, making it adaptable to various requirements.

# **Conclusion**

The Real Estate Database was successfully implemented and populated with realistic data. Its robust schema ensures data integrity, and its querying capabilities provide flexibility for various use cases. The combination of SQL and relational algebra queries highlights the database’s power to handle complex operations while maintaining efficiency and accuracy. This system is well-suited for real estate management, offering scalability for future enhancements.