VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi-590018.

A Mini Project Report on

Database Management Systems

[BCS403]

Submitted in partial fulfillment of the requirements for the Degree of

BACHELOR OF ENGINEERING
IN

COMPUTER SCIENCE & ENGINEERING

(DATA SCIENCE)

Submitted by,

NAME: DISHA N H

USN: 4AI22CD018

Under the Guidance of

Prof. Gagana Deepa J B.E, MTech

Asst. Professor

Dept. of Data Science

Dept. of Computer Science and Engineering (DATA SCIENCE)

Adichunchanagiri Institute of Technology

Chikkamagaluru – 577102, Karnataka, India

2023-24

ADICHUNCHANAGIRI INSTITUTE OF TECHNOLOGY

(Affiliated to VTU, Belagavi, Approved by AICTE, New Delhi and Accredited by NAAC)

CHIKKAMAGALURU–577102, KARNATAKA, INDIA.

CERTIFICATE

This is to certify that the report on Database Management Systems [BCS403] is prepared by DISHA N H Bearing USN 4AI22CD018 of 4th semester B.E in partial fulfillment for the award of Bachelor of Engineering degree in Computer Science and Engineering (DATA SCIENCE) of the Visvesvaraya Technological University, Belagavi, during the year 2023-2024. The report is prepared according to the norms specified by the university. All the corrections suggested are incorporated. Their report has been approved as it is the academic requirements prescribed for the said Degree and a copy of the same is deposited in the library.

Signature of the Guide **Signature of the HOD**

Visvesvaraya Technological University, Belagavi

DBMS Mini Project (BCS403)

Marks Sheet

Weightage	Max Marks	Marks Obtained
Seminar	**15**	
Report	**10**	
Total Marks	**25**	
Scaled to	**05**	

Signature of the Faculty **Signature of the HOD**

Signature of the Principal

ACKNOWLEDGEMENT

I express my humble Pranamas to his holiness **Parama Poojya Jagadguru Padmabushana Sri Sri Sri Dr.Balagangadharanatha Maha Swamiji** and **Parama Poojya Jagadguru Sri SriSri Dr.NirmalanandanathaMahaSwamiji**and also to **Sri Sri Gunanatha Swamiji** Sringeri Branch Chikkamagaluru, for blessing us to achieve success and become noble citizens of our country.

I am deeply indebted to our honorable Director **Dr. C K Subbaraya** for creating the right kind of care and Infrastructure for us.

I am thankful to our beloved Principal **Dr. C T Jayadeva** for inspiring us to achieve greater endeavors in all aspects of Learning.

I express my deepest gratitude to **Dr.Adarsh M J,**Assoc.Professor & Head, **Dept. of Computer Science & Engineering (DATA SCIENCE)** for his valuable guidance, suggestions, and constant encouragement in all our academic activities.

I would like to thank my guide **Ms. Gagana Deepa J,** Asst. Professor, Dept. of Data Science for her patience with our irregularities and timely correspondence with suggestions and corrections.

I would like to thank my beloved parents for their support, encouragement, and blessings.

I express my gratitude to all the teaching and non-teaching staff of the Department of Computer Science & Engineering (DATA SCIENCE) and to all the people who have helped me in preparing the report on database management systems.

DISHA N H (4AI22CD018)

ABSTRACT

A real estate management system is an advanced and integrated platform designed to streamline the processes involved in managing real estate properties. It aims to enhance the efficiency and effectiveness of property management through advanced technologies like databases, web applications, and mobile solutions. By leveraging these technologies, the system enables property managers to collect, analyze, and manage real-time data related to property listings, tenant management, lease agreements, maintenance schedules, and financial transactions.

The primary goals of a real estate management system are to improve tenant satisfaction, optimize property management processes, and maximize the financial performance of real estate assets. By automating routine tasks, providing insights through data analytics, and facilitating better communication between property managers, tenants, and landlords, the system contributes to enhanced decision-making and operational efficiency.

Overall, a real estate management system is a vital component of modern property management practices, and its significance is expected to grow as the real estate industry continues to evolve. The adoption and implementation of such systems should be prioritized by property managers and stakeholders to ensure the sustainable growth and profitability of real estate investments. Moreover, a well-implemented REMS can serve as a competitive advantage by differentiating property management businesses and attracting more clients.

INDEX

01. Introduction:

What is data:

In a Database Management System (DBMS), data refers to the information stored, organized, and managed within the system. This data is structured in a way that allows efficient retrieval, manipulation, and storage. Here is a breakdown of the concept of data in a DBMS:

1. Types of Data

- **Structured Data**: This is data that adheres to a specific format or schema, such as tables in a relational database. Examples include customer names, addresses, and transaction records.
- **Unstructured Data**: Data that does not have a predefined format, such as emails, documents, images, and videos. DBMS systems that handle unstructured data are often called NoSQL databases.
- **Semi-Structured Data**: This data does not fit neatly into a table but contains tags or markers to separate data elements (e.g., XML or JSON).

2. Data Components in a DBMS

- **Tables**: The primary structure in a relational DBMS where data is stored in rows and columns. Each table represents a different entity (e.g., customers, products).
- **Records (Rows)**: Individual entries within a table that represent specific instances of an entity.
- **Fields (Columns)**: Attributes or properties of the entity that describe the data (e.g., customer name, order date).
- **Indexes**: Data structures that improve the speed of data retrieval operations on a database table.
- **Views**: Virtual tables created by querying one or more tables. They provide a way to present data in a specific format or structure.
- **Schemas**: Define the structure of the database, including tables, fields, relationships, and constraints.

3. Data Operations in a DBMS

- **Data Definition**: Creating, altering, and deleting tables and schemas.
- **Data Manipulation**: Inserting, updating, and deleting data within tables.
- **Data Retrieval**: Querying the database to fetch specific information, often using SQL (Structured Query Language).
- **Data Control**: Managing access permissions and ensuring data security and integrity.

4. Data Integrity and Constraints

- **Primary Key**: A unique identifier for a record in a table.
- **Foreign Key**: A field in one table that links to the primary key of another table, establishing relationships between tables.
- **Unique Constraint**: Ensures that all values in a column are distinct.
- **Check Constraint**: Validates that values in a column satisfy certain conditions.

5. Data Storage

- **Data Files**: The actual files on disk where the data is physically stored.
- **Transactions**: Ensure that multiple data operations are performed in a reliable and consistent manner. Transactions follow the ACID (Atomicity, Consistency, Isolation, Durability) properties.

In summary, in a DBMS, data is organized in a structured way to facilitate efficient storage, retrieval, and management, with various components and operations designed to handle and manipulate this data effectively.

What is DBMS?

DBMS stands for "Database Management System." It is a software system designed to manage, store, organize, and retrieve data from a database. A database is a structured collection of data organized and stored to allow for efficient querying, manipulation, and analysis.

A DBMS provides an interface and tools for users and applications to interact with the database without worrying about the underlying complexities of data storage and retrieval. It offers various features and functions that facilitate data management.

Common examples of DBMSs include MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server, SQLite, and MongoDB. Each DBMS has its strengths, features, and use cases. The choice of a specific DBMS depends on factors such as the nature of the data, scalability requirements, performance considerations, and application needs

MySQL:

MySQL is an open-source relational database management system. Based in Structured Query Language (SQL), MySQL can run on most platforms and is used for web-based applications. It is written in C and C++.

What is a DBMS Project?

A DBMS project involves creating a database management system designed to store, manage, and facilitate access to data efficiently. These projects typically focus on designing schemas, integrating data input forms, and creating queries and reports to manage copious amounts of data effectively. DBMS projects are fundamental in industries like healthcare, finance, and education to handle complex data operations securely and reliably.

Real Estate Management System (RMS):

An RMS is a comprehensive software solution designed to streamline and centralize all aspects of property management. It acts as a central hub for data and operations, offering a robust platform to manage:

- **Properties:** Detailed information about each property, including square footage, amenities, location, photos, and floor plans.
- **Tenants:** Efficient tenant management with features like lease information, contact details, rent payment history, and maintenance requests.
- **Leases & Contracts:** Streamlined lease creation, tracking, and renewal processes with automated reminders and document storage.
- **Maintenance Management:** A dedicated system for tracking maintenance requests, assigning technicians, scheduling repairs, and managing work orders.
- **Financial Transactions:** Integrated accounting features for rent collection, expense tracking, generating reports, and managing financial health.

Benefits of an RMS:

- **Improved Efficiency:** Automates routine tasks, reduces paperwork, and saves time for property managers.
- **Enhanced Organization:** Provides a central location for all property-related data, ensuring easy access and retrieval of information.
- **Better Communication:** Facilitates communication between property managers, tenants, and maintenance personnel.
- **Stronger Decision Making:** Delivers data-driven insights for informed decisions about properties, tenants, and finances.
- **Increased Revenue:** Streamlines rent collection, reduces vacancies, and helps optimize rental income.

Overall, a well-implemented RMS empowers property managers to operate efficiently, deliver exceptional service to tenants, and maximize their overall investment returns.

Objectives

The primary objectives of this mini project are:

- Designing a relational database schema to store essential information related to properties, tenants, owners, leases, maintenance requests, and financial transactions.

- Implementing basic functionalities such as inserting, updating, querying, and deleting data to simulate real-world interactions within a property management environment.
- Demonstrating proficiency in SQL queries for retrieving specific information, generating reports, and analyzing data related to property occupancy, maintenance costs, and financial performance.

Database Schema

The database schema for this mini project includes the following tables:

1. **Properties**: Stores information about real estate properties, including property ID, name, type, location, size, and ownership details.
2. **Tenants**: Contains details about tenants, such as tenant ID, name, contact information, lease start and end dates, and rental terms.
3. **Leases**: Tracks records of lease agreements, including lease ID, property ID, tenant ID, rent amount, security deposit, and lease duration.
4. **Maintenance**: Stores details about maintenance activities, including maintenance ID, property ID, date, description, cost, and vendor details.
5. **Transactions**: Records financial transactions related to properties, such as transaction ID, date, amount, type, property ID, and tenant ID.

Functionalities

The mini project will implement the following functionalities using SQL queries:

- **Insertion**: Adding new records for vehicles, drivers, officers, traffic violations, and incidents.
- **Updating**: Modifying existing data, such as updating fine amounts for violations or marking vehicle registrations as expired.
- **Querying**: Retrieving specific information from the database, such as listing all violations issued by a particular officer or incidents of a certain type within a date range.
- **Deletion**: Removing outdated or erroneous records from the database, such as expired vehicle registrations.
- **Reporting**: Generating reports on various metrics, such as total fines collected, trends in violation types, or incident frequencies.

Implementation Details

The mini project will be implemented using a chosen DBMS (e.g., MySQL, PostgreSQL) and a suitable programming language (e.g., Python, Java) for database interactions. The emphasis will be on demonstrating proper database design principles, efficient query execution, and handling of relational data.

02. Project Code:

To design a Real Estate Management System, you would typically need a database schema that includes tables for entities like properties, tenants, leases, maintenance, and transactions. Here is a simplified example with SQL queries for creating some basic tables:

Tables:

1. Properties

- Stores information about properties:

```
CREATE TABLE Properties (
    property_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    type VARCHAR(50) NOT NULL,
    location VARCHAR(255) NOT NULL,
    size INT NOT NULL,
    owner_id INT NOT NULL,
    FOREIGN KEY (owner_id) REFERENCES Owners(owner_id)
);
```

OUTPUT:

```
SQL> connect
Enter user-name: system
Enter password:
Connected.
SQL> CREATE TABLE Drivers (    driver_id INT PRIMARY KEY,    driver_name VARCHAR(100),    license_number VARCHAR(20),
 contact_number VARCHAR(15));

Table created.

SQL> desc drivers
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DRIVER_ID                                 NOT NULL NUMBER(38)
 DRIVER_NAME                                        VARCHAR2(100)
 LICENSE_NUMBER                                     VARCHAR2(20)
 CONTACT_NUMBER                                     VARCHAR2(15)
```

This query creates a properties table with details about each property, including property_id as the unique identifier. It links to an Owners table through the owner_id foreign key, ensuring that each property has an owner listed in the Owners table.

Snapshot:

property_id	name	type	location	size	owner_id
1	Cozy Cottage	Residential	123 Elm St	1200	101
2	Downtown Office	Commercial	456 Market St	5000	102

2. Tenants:

- Stores information about tenants:

```
CREATE TABLE Tenants (
    tenant_id SERIAL PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    contact_info VARCHAR(255) NOT NULL,
    lease_start_date DATE,
    lease_end_date DATE
);
```

OUTPUT:

```
SQL> CREATE TABLE Vehicle (    vehicle_id INT PRIMARY KEY,    vehicle_number VARCHAR(20),    vehicle_type VARCHAR(50),
  owner_name VARCHAR(100));

Table created.

SQL> desc Vehicle
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 VEHICLE_ID                                NOT NULL NUMBER(38)
 VEHICLE_NUMBER                                     VARCHAR2(20)
 VEHICLE_TYPE                                       VARCHAR2(50)
 OWNER_NAME                                         VARCHAR2(100)
```

This query creates a `Tenants` table to store information about tenants, including `tenant_id` as the unique identifier. It captures personal details and lease dates, if applicable.

Snapshot:

tenant_id	first_name	last_name	contact_info	lease_start_date	lease_end_date
1	Alice	Johnson	alice@example.com	2024-01-01	2024-12-31
2	Bob	Smith	bob@example.com	2024-03-15	2025-03-14

3. Leases:

- Stores lease agreements:

```
CREATE TABLE Leases (
    lease_id SERIAL PRIMARY KEY,
    property_id INT NOT NULL,
    tenant_id INT NOT NULL,
    rent_amount DECIMAL(10, 2) NOT NULL,
    security_deposit DECIMAL(10, 2),
    lease_duration INT NOT NULL,
    FOREIGN KEY (property_id) REFERENCES Properties(property_id),
    FOREIGN KEY (tenant_id) REFERENCES Tenants(tenant_id)
);
```

OUTPUT:

```
SQL> CREATE TABLE Violation (    violation_id INT PRIMARY KEY,    violation_type VARCHAR(100),    violation_date DATE,
  vehicle_id INT,    driver_id INT,    FOREIGN KEY (vehicle_id) REFERENCES Vehicles(vehicle_id),    FOREIGN KEY (driver_
id) REFERENCES Drivers(driver_id));

Table created.

SQL> desc Violation
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 VIOLATION_ID                              NOT NULL NUMBER(38)
 VIOLATION_TYPE                                     VARCHAR2(100)
 VIOLATION_DATE                                     DATE
 VEHICLE_ID                                         NUMBER(38)
 DRIVER_ID                                          NUMBER(38)
```

This query creates a Leases table to record lease agreements, linking each lease to a property_id and tenant_id. It includes details like rent amount, security deposit, and lease duration.

Snapshot:

lease_id	property_id	tenant_id	rent_amount	security_deposit	lease_duration
1	1	1	1500.00	300.00	12
2	2	2	5000.00	1000.00	24

4. Maintenance:

- Stores maintenance activities:

```
CREATE TABLE Maintenance (
    maintenance_id SERIAL PRIMARY KEY,
    property_id INT NOT NULL,
    maintenance_date DATE NOT NULL,
    description TEXT,
    cost DECIMAL(10, 2),
    vendor_details VARCHAR(255),
    FOREIGN KEY (property_id) REFERENCES Properties(property_id)
);
```

OUTPUT:

```
SQL> drop table fine;

Table dropped.

SQL> CREATE TABLE fin (fine_id number(10),violation_id number(10),amount number(10),  payment_status number(10));

Table created.
```

This query creates a `Maintenance` table to track maintenance activities for properties. Each entry is linked to a `property_id`, with details about the maintenance date, description, cost, and vendor.

Snapshot:

maintenance_id	property_id	maintenance_date	description	cost	vendor_details
1	1	2024-07-15	HVAC Repair	250.00	AC Repair Co.
2	2	2024-08-01	Plumbing Check	150.00	Pipe Solutions

5.Transactions:

- Stores financial transactions:

```
CREATE TABLE Transactions (
    transaction_id SERIAL PRIMARY KEY,
    date DATE NOT NULL,
    amount DECIMAL(10, 2) NOT NULL,
    type VARCHAR(50) NOT NULL,
    property_id INT,
    tenant_id INT,
    FOREIGN KEY (property_id) REFERENCES Properties(property_id),
    FOREIGN KEY (tenant_id) REFERENCES Tenants(tenant_id)
);
```

OUTPUT:

Snapshot:

transaction_id	date	amount	type	property_id	tenant_id
1	2024-07-01	1500.00	Rent	1	1
2	2024-08-01	5000.0	Payment	2	2

This query creates a `Transactions` table to manage financial transactions related to properties and tenants. It includes details like transaction date, amount, type, and optional foreign keys to `Properties` and `Tenants`.

03. Inserted Statements:

- **Insert example properties:**

```
-- Insert example properties
INSERT INTO Properties (name, type, location, size, owner_id)
VALUES ('Sunset Apartments', 'Residential', '123 Main St, Anytown', 2000, 1);
```

  - Adds a new property named 'Sunset Apartments' with specified details to the Properties table, associating it with owner ID 1.

Output:

- **Insert example tenants and Insert example leases:**

```
-- Insert example tenants
INSERT INTO Tenants (first_name, last_name, contact_info, lease_start_date, lease_end_date
VALUES ('John', 'Doe', 'john.doe@example.com', '2024-01-01', '2024-12-31');

-- Insert example leases
INSERT INTO Leases (property_id, tenant_id, rent_amount, security_deposit, lease_duration)
VALUES (1, 1, 1200.00, 1200.00, 12);
```

- Adds a new tenant named John Doe with contact info and lease dates to the Tenants table.
- Create a lease for property ID 1 with tenant ID 1, specifying the rent amount, security deposit, and lease duration (12 months).

Output:

- **Insert example maintenance and Insert example transactions:**

```
-- Insert example maintenance
INSERT INTO Maintenance (property_id, maintenance_date, description, cost, vendor_details)
VALUES (1, '2024-06-15', 'Air conditioning repair', 150.00, 'Cool Air Services');

-- Insert example transactions
INSERT INTO Transactions (date, amount, type, property_id, tenant_id)
VALUES ('2024-07-01', 1200.00, 'Rent Payment', 1, 1);
```

- Logs a maintenance record for property ID 1 detailing an air conditioning repair done by 'Cool Air Services' on June 15, 2024.
- Records a transaction for a rent payment of $1200 made on July 1, 2024, for property ID 1 by tenant ID 1.

Output:

04. Additional Operations:(Updating records)

- **Update tenant information:**

  Modify the status of a property when it is sold or rented.

```
UPDATE Tenants
SET contact_info = 'new.email@example.com'
WHERE tenant_id = 1;
```

Output:

- **UPDATE:** This keyword indicates that you are modifying existing data in a table.
- **Properties:** This specifies the table name where the data will be updated.
- **SET Status = 'Sold':** This part changes the value of the 'Status' column to 'Sold' for the selected rows.
- **WHERE PropertyID = 1:** This condition specifies which rows to update. In this case, it is updating the property with the ID of 1.

  If you have a property with ID 1 that is currently listed for sale, running this command will change its status to 'Sold' in the 'Properties' table.

- **Update lease terms:**

Calculate the total sales amount for a given period.

```
UPDATE Leases
SET rent_amount = 1300.00
WHERE lease_id = 1;
```

Output:

Explanation:

- **SELECT:** This keyword is used to retrieve data from the database.
- **SUM(Amount):** This calculates the total sum of the 'Amount' column in the specified table.
- **FROM Transactions:** This specifies the table from which data will be retrieved.

- **WHERE Type = 'Sale':** This filter ensures that only transactions with the type 'Sale' are included in the calculation.
  Example: If you have a 'Transactions' table with columns 'Amount' and 'Type', running this command will give you the total amount of all sales transactions recorded in the table.

- **Generating Property Listings Report**

This operation allows you to generate a report of all properties available for sale or rent, including their details such as address, type, price, and status. This report can help agents and clients instantly view vacant properties.

- **SQL Query to Generate Property Listings Report:**

```
SELECT PropertyID, Address, Type, Price, Status
FROM Properties
WHERE Status = 'For Sale' OR Status = 'For Rent';
```

Explanation:

- This query retrieves all properties with a status of 'For Sale' or 'For Rent'.
- It selects the `PropertyID`, `Address`, `Type`, `Price`, and `Status` columns to provide a comprehensive overview of vacant properties.

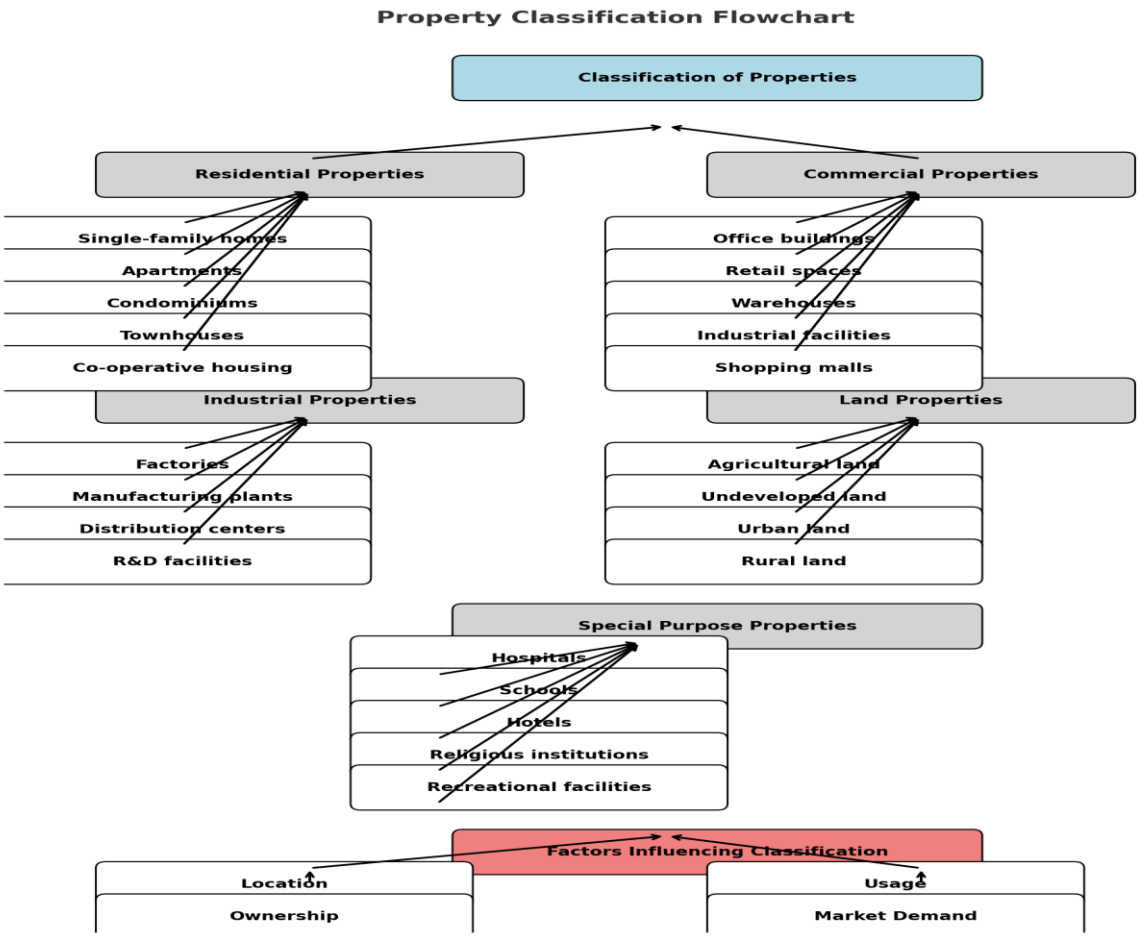
Output:

This query retrieves all properties from the Properties table that meet the following criteria:

- **Type:** The property type should be either "Apartment" or "Condo".
- **Price:** The price should be less than $300,000.
- **Status:** The status should be "For Sale".

05. Classification of Properties:

Properties can be categorized into distinct types based on their primary function and purpose. Residential properties are designed for habitation, encompassing houses, apartments, and townhouses. Commercial properties serve business operations, including offices, retail stores, and shopping centers. Industrial properties facilitate manufacturing and production processes, such as factories and warehouses. Land properties represent undeveloped plots available for various uses. Lastly, special purpose properties are dedicated to specific functions like education, healthcare, or government services. Factors such as location, intended use, ownership, and market demand significantly influence property classification



Property Classification Flowchart

06. Conclusion:

A Real Estate Management System (RMS) is a cornerstone of efficient property management, offering a robust platform to streamline operations, enhance decision-making, and optimize financial performance. By centralizing property, tenant, lease, and financial data, RMS empowers property managers to make informed choices, improve tenant satisfaction, and maximize investment returns.
This mini project has provided a foundational understanding of RMS development, encompassing database schema design, core functionalities, and SQL query implementation. Through this practical experience, the significance of a well-structured database and efficient data management in the real estate domain has been highlighted.

The real estate industry is undergoing rapid digital transformation. Adopting advanced RMS solutions is no longer an option but a necessity for businesses to thrive in this competitive landscape. By leveraging technology and data-driven insights, property managers can achieve operational excellence, enhance tenant experiences, and unlock new opportunities for growth and profitability.
This project serves as a steppingstone towards developing more comprehensive and sophisticated RMS solutions. As technology continues to evolve, there is immense potential to integrate advanced features such as artificial intelligence, machine learning, and predictive analytics to further revolutionize property management.

The successful implementation of an RMS can significantly contribute to the overall efficiency and profitability of real estate portfolios, making it an indispensable tool for property managers and investors alike.