HOSPITAL DATABASE MANAGEMENT SYSTEM

PROJECT BACKGROUND

The Hospital Management System (HMS) signifies a pivotal advancement in healthcare technology, addressing the urgent need for streamlined hospital administration and optimized patient care delivery.

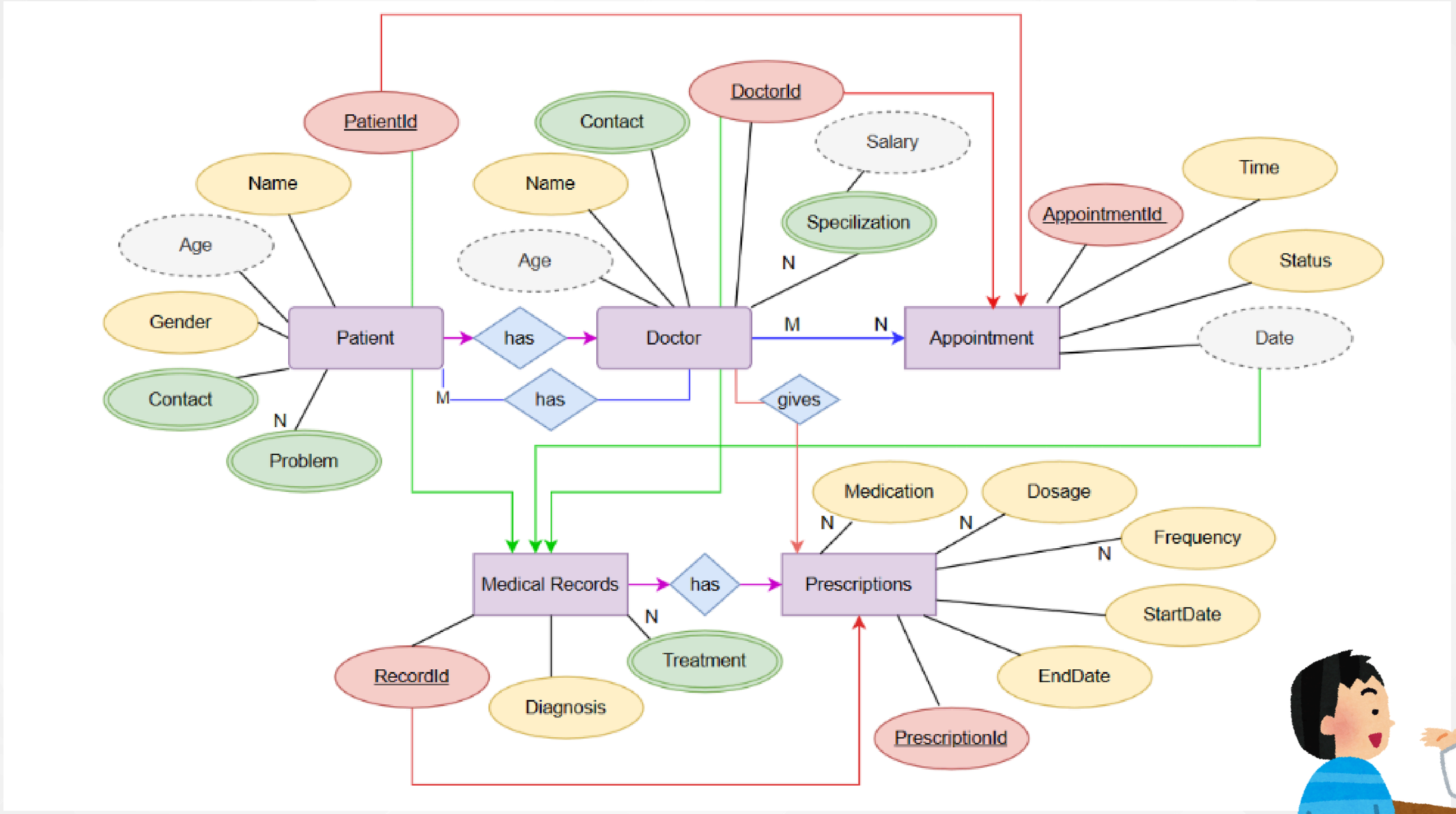
In response to the growing complexities of healthcare management, HMS emerges as a comprehensive software solution poised to revolutionize healthcare operations. With core functionalities encompassing patient and doctor management, appointment scheduling, medical records handling, and prescription management, the HMS offers a holistic approach to healthcare administration.

By providing a centralized platform for data management and communication, the HMS empowers healthcare professionals to streamline operations and improve clinical workflows. In an era defined by digital innovation and evolving healthcare demands, HMS progress further promising to enhance efficiency, productivity, and quality in hospital management, thereby transforming the healthcare landscape better.

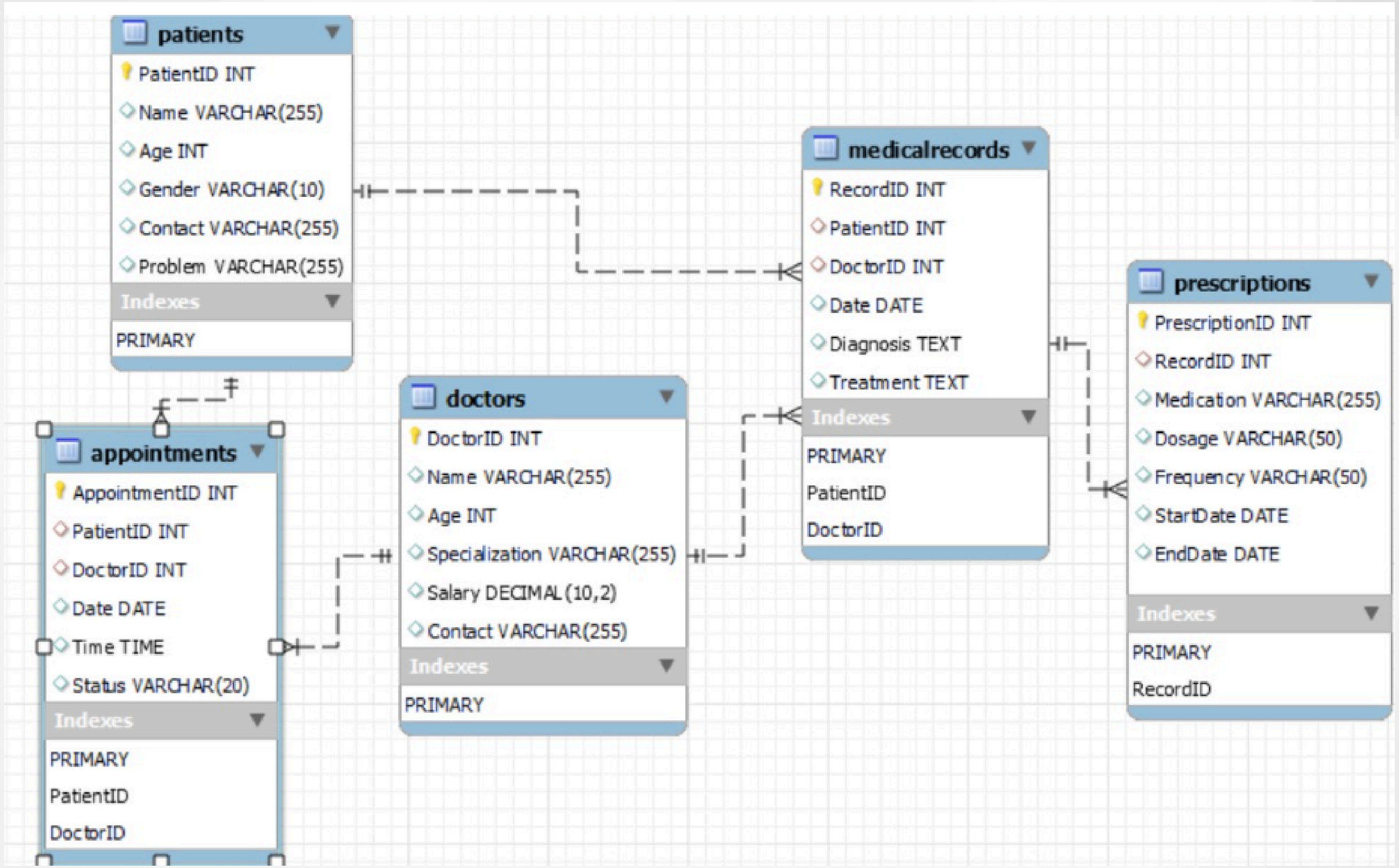
DESCRIPTION OF THE PROJECT

The Hospital Management System (HMS) utilizes a database management system (DBMS) to streamline various hospital operations. Patient management, appointment scheduling, inventory control, pharmacy management, laboratory information, and billing modules interact with the DBMS to efficiently handle tasks. For instance, patient records and medical histories are stored in the DBMS for easy access by healthcare providers. Appointment scheduling optimizes slots and doctor availability through the DBMS. Inventory and pharmacy modules track stock levels and medication orders, while laboratory information systems manage test orders and results. The billing module generates accurate invoices by integrating data from multiple modules. This integration enhances efficiency, patient care, and regulatory compliance within the hospital.

ER DIAGRAM


PatientId
Name
Age
Gender
Contact
Problem
Patient
has
M
N
DoctorId
Contact
Name
Age
Salary
Specilization
N
Doctor
has
gives
M
N
Appointment
AppointmentId
Time
Status
Date
Medication
Dosage
Frequency
N
N
N
Medical Records
has
Prescriptions
N
RecordId
Diagnosis
Treatment
StartDate
EndDate
PrescriptionId

SCHEMA REPRESENTATION



DESCRIPTION OF ER DIAGRAM

The ER diagram outlines the core entities and their relationships within the hospital management system. Patients are represented by a unique identifier (PatientID) and encompass details such as name, age, gender, contact information, and the health problem they're facing. Doctors are another vital entity, distinguished by their unique DoctorID, and include information like name, age, specialization, salary, and contact details.

Appointments serve as a bridge between patients and doctors, with each appointment uniquely identified by AppointmentID. This entity captures essential details such as the involved PatientID and DoctorID, appointment date and time, and status (whether scheduled, canceled, or completed). Medical Records provide comprehensive insights into a patient's medical history and treatment, featuring a unique RecordID. They encompass key elements such as PatientID, DoctorID, record creation date, diagnosis, and treatment details. Prescriptions complement medical records by detailing prescribed medications, each with a unique PrescriptionID linked to a specific medical record. Information includes the medication name, dosage, frequency, and start and end dates for the prescription duration.

CREATING TABLES

Table ‘Patients’:

- ‘PatientID’: Integer, Primary Key, Auto Increment
- ‘Name’: Variable Character (VARCHAR), Maximum Length 255
- ‘Age’: Integer
- ‘Gender’: Variable Character (VARCHAR), Maximum Length 10
- ‘Contact’: Variable Character (VARCHAR), Maximum Length 255
- ‘Problem’: Variable Character (VARCHAR), Maximum Length 255

Table ‘Doctors’:

- ‘DoctorID’: Integer, Primary Key, Auto Increment
- ‘Name’: Variable Character (VARCHAR), Maximum Length 255
- ‘Age’: Integer
- ‘Specialization’: Variable Character (VARCHAR), Maximum Length 255
- ‘Salary’: Decimal, Precision 10, Scale 2
- ‘Contact`: Variable Character (VARCHAR), Maximum Length 255

Table `Appointments`:

- `AppointmentID`: Integer, Primary Key, Auto Increment
- `PatientID`: Integer, Foreign Key referencing `Patients.PatientID`
- `DoctorID`: Integer, Foreign Key referencing `Doctors.DoctorID`
- `Date`: Date
- `Time`: Time
- `Status`: Variable Character (VARCHAR), Maximum Length 20

Table `MedicalRecords`:

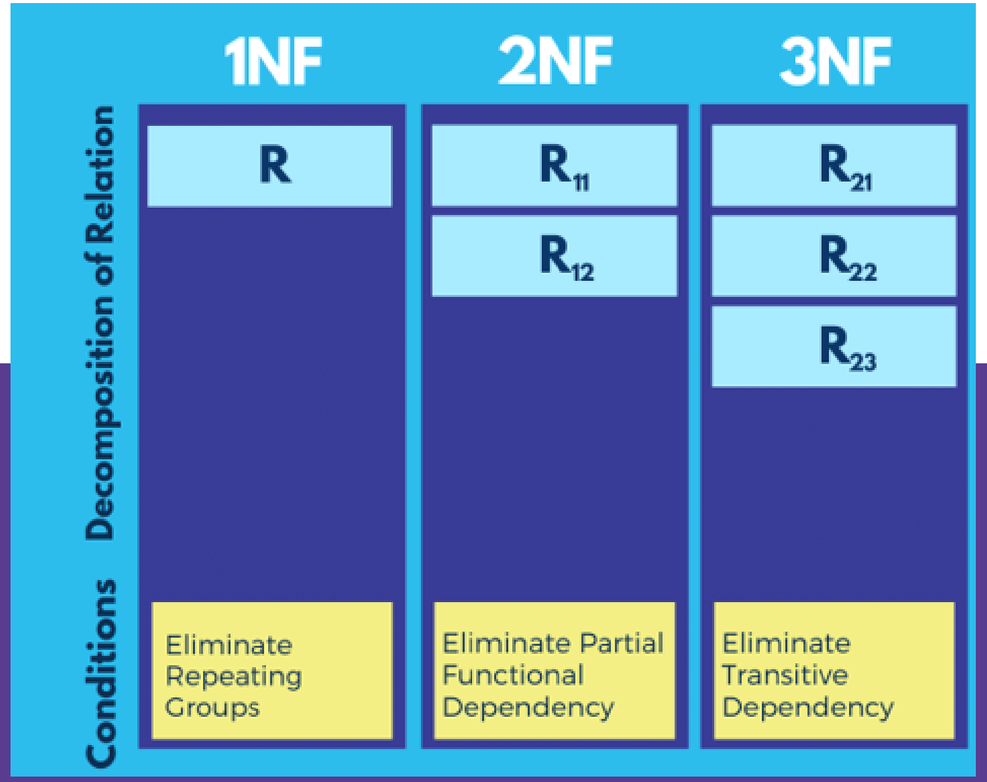
- `RecordID`: Integer, Primary Key, Auto Increment
- `PatientID`: Integer, Foreign Key referencing `Patients.PatientID`
- `DoctorID`: Integer, Foreign Key referencing `Doctors.DoctorID`
- `Date`: Date
- `Diagnosis`: Text
- `Treatment`: Text

Table `Prescriptions`:

- `PrescriptionID`: Integer, Primary Key, Auto Increment
- `RecordID`: Integer, Foreign Key referencing `MedicalRecords.RecordID`
- `Medication`: Variable Character (VARCHAR), Maximum Length 255
- `Dosage`: Variable Character (VARCHAR), Maximum Length 50
- `Frequency`: Variable Character (VARCHAR), Maximum Length 50
- `StartDate`: Date
- `EndDate`: Date

NORMALIZATION

Normalization in DBMS is a process of organizing data in a database efficiently. Its primary goal is to reduce data redundancy and dependency by dividing the database into smaller, manageable tables and defining relationships between them. There are several normal forms, each with specific rules that a database must meet to be considered normalized. The most commonly used normal forms are First Normal Form (1NF),Second Normal Form (2NF) ,Third Normal Form (3NF)



TYPES OF NORMAL FORM

First Normal Form (1NF): Ensures that each column contains atomic values and there are no repeating groups or arrays within a table.

Second Normal Form (2NF): Requires that the table is in 1NF and all non-key attributes are fully dependent on the primary key. This means that no partial dependencies exist.

Third Normal Form (3NF): Requires that the table is in 2NF and all attributes are dependent only on the primary key, not on other non-key attributes.

DATA RETRIEVAL QUERIES

Data retrieval queries are SQL statements used to extract specific information from a database. These queries enable users to retrieve data based on various criteria, such as specific conditions, sorting requirements, or aggregation functions. These queries are crucial for obtaining meaningful insights and information from the database.

EXAMPLES

- **SELECT ALL RECORDS FROM A TABLE**

```
SELECT *
FROM Medical Records;
```

- **SELECT SPECIFIC COLUMNS FROM A TABLE**

```
SELECT * Medication, Dosage, Frequency
FROM Prescriptions;
```

- **SELECT RECORDS BASED ON CONDITION**

```
SELECT*
FROM Appointments
WHERE DoctorId = 4;
```

DATA MANIPULATION QUERIES

Data manipulation queries are essential SQL statements used to modify the data stored in a database, enabling users to perform various operations such as insertion, updating, deletion, and modification of records within tables. These queries play a crucial role in managing the integrity and consistency of the database by allowing users to interact with the data effectively.

Data manipulation queries are fundamental tools in database management, facilitating the maintenance and modification of data to meet evolving requirements and ensure the accuracy and integrity of the database content.

EXAMPLES

ALTER RECORDS FROM A TABLE

```
ALTER TABLE Patients
ADD Address VARCHAR(255);
```

MODIFY RECORDS FROM A TABLE

```
UPDATE Patients
 SET Age = 35
 WHERE PatientID = 1001;
```

DELETE RECORDS FROM A TABLE

```
DELETE FROM Appointments
WHERE PatientID = '1002';
```

UPDATE EXISTING RECORDS

```
UPDATE Doctors
 SET Salary = 180000
WHERE DoctorID = 1;
```

AGGREGATE FUNCTIONS

Aggregate functions in SQL are powerful tools used to perform calculations on sets of data within a database. These functions enable users to obtain summarized information, such as totals, averages, counts, minimum values, and maximum values, from selected columns or rows in a table. Commonly used aggregate functions include COUNT(), which counts the number of rows that meet specified criteria, SUM(), which calculates the total of numeric values in a column, AVG(), which computes the average of numeric values, MIN(), which retrieves the minimum value from a column, and MAX(), which retrieves the maximum value from a column. By applying aggregate functions, users can gain valuable insights into their data, facilitating analysis and decision-making processes.

EXAMPLES

AVERAGE

```
SELECT AVG(Age) AS AverageAge
 FROM Patients;
```

SUM

```
SELECT SUM(Salary) AS TotalSalary
 FROM Doctors;
```

VIEWS

Views in database management systems (DBMS) serve as virtual tables derived from one or more underlying tables. Unlike physical tables, views don't store data themselves; rather, they offer a customized or restricted presentation of data from existing tables. Acting as predefined queries, views encapsulate complex logic or frequently used joins, streamlining data retrieval and manipulation. They offer numerous benefits such as enhanced data security by restricting access to sensitive information, simplified querying by abstracting away complex joins, and improved performance by precomputing and caching results. Additionally, views provide a layer of data abstraction, hiding underlying table complexities, and ensuring data consistency across users and applications. Overall, views are indispensable tools in DBMS for optimizing data management, enhancing query efficiency, and maintaining data integrity.

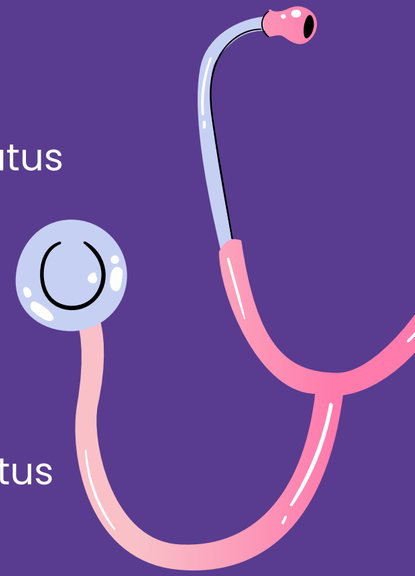
EXAMPLES

- **Patient Appointments View**

```
CREATE VIEW PatientAppointmentsView AS
SELECT p.Name AS PatientName, p.Age, p.Gender, a.Date AS AppointmentDate, a.Time AS AppointmentTime, a.Status
FROM Patients p
INNER JOIN Appointments a ON p.PatientID = a.PatientID;
```

- **Doctor Schedule View**

```
CREATE VIEW DoctorScheduleView AS
SELECT d.Name AS DoctorName, d.Specialization, a.Date AS AppointmentDate, a.Time AS AppointmentTime, a.Status
FROM Doctors d
INNER JOIN Appointments a ON d.DoctorID = a.DoctorID;
```



THANK YOU!!

SUBMITTED BY:

Bandhavi Kanyadhara-AP22110010079
Hema Kakarlapudi-AP22110010080
Swathi Chowdary-AP22110010084
Vaishnavi Kolasani-AP22110010126

Github link: https://github.com/Vaishnavik05/Hospital-dbms

Done by: K. Bandhavi