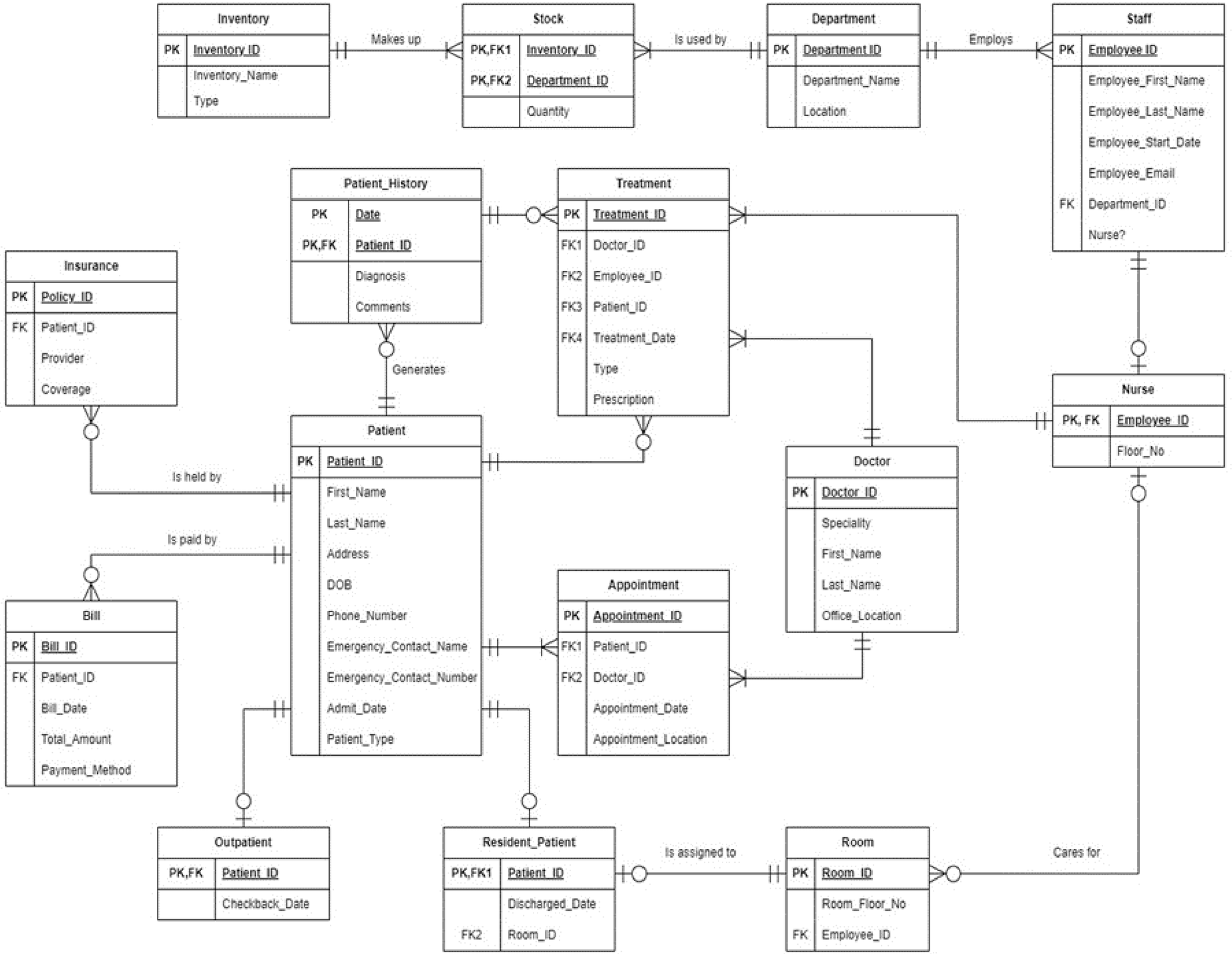
Group 8 - Hospital Management System

Overview

Our Hospital Management System is a comprehensive database management and design project aimed at revolutionizing healthcare administration. With meticulous planning and implementation, we have developed a robust database that streamlines various facets of hospital operations, including patient management, staff scheduling, inventory control, and billing processes. Our system leverages advanced technologies to enhance efficiency, accuracy, and patient care delivery, ultimately leading to improved outcomes and satisfaction. Join us as we unveil the power of data-driven decision-making in transforming the healthcare landscape.

Design


Inventory
PK Inventory ID
Inventory_Name
Type
Makes up
Stock
PK,FK1 Inventory_ID
PK,FK2 Department_ID
Quantity
Is used by
Department
PK Department ID
Department_Name
Location
Employs
Staff
PK Employee ID
Employee_First_Name
Employee_Last_Name
Employee_Start_Date
Employee_Email
FK Department_ID
Nurse?
Patient_History
PK Date
PK,FK Patient_ID
Diagnosis
Comments
Generates
Treatment
PK Treatment_ID
FK1 Doctor_ID
FK2 Employee_ID
FK3 Patient_ID
FK4 Treatment_Date
Type
Prescription
Insurance
PK Policy_ID
FK Patient_ID
Provider
Coverage
Is held by
Patient
PK Patient_ID
First_Name
Last_Name
Address
DOB
Phone_Number
Emergency_Contact_Name
Emergency_Contact_Number
Admit_Date
Patient_Type
Nurse
PK, FK Employee_ID
Floor_No
Doctor
PK Doctor_ID
Speciality
First_Name
Last_Name
Office_Location
Is paid by
Bill
PK Bill_ID
FK Patient_ID
Bill_Date
Total_Amount
Payment_Method
Appointment
PK Appointment_ID
FK1 Patient_ID
FK2 Doctor_ID
Appointment_Date
Appointment_Location
Outpatient
PK,FK Patient_ID
Checkback_Date
Resident_Patient
PK,FK1 Patient_ID
Discharged_Date
FK2 Room_ID
Is assigned to
Room
PK Room_ID
Room_Floor_No
FK Employee_ID
Cares for

VIEWS

```sql
CREATE OR ALTER VIEW UpcomingAppointmentsView AS
SELECT
    a.appointment_ID,
    a.appointment_date,
    a.appointment_location,
    p.first_name AS patient_first_name,
    p.last_name AS patient_last_name,
    d.first_name AS doctor_first_name,
    d.last_name AS doctor_last_name,
    d.specialty
FROM
    Appointment a
JOIN
    Patient p ON a.patient_ID = p.patient_ID
JOIN
    Doctor d ON a.doctor_ID = d.doctor_ID
WHERE
    a.appointment_date >= CAST(GETDATE() AS DATE)
```

```sql
CREATE OR ALTER VIEW DoctorScheduleView AS
SELECT TOP 100 PERCENT
    d.doctor_ID,
    d.first_name AS doctor_first_name,
    d.last_name AS doctor_last_name,
    d.specialty,
    a.appointment_date,
    a.appointment_location,
    p.first_name AS patient_first_name,
    p.last_name AS patient_last_name
FROM
    Doctor d
JOIN
    Appointment a ON d.doctor_ID = a.doctor_ID
JOIN
    Patient p ON a.patient_ID = p.patient_ID
ORDER BY
    a.appointment_date, d.doctor_ID;
GO
```

VIEWS

```sql
CREATE OR ALTER VIEW PatientOverviewView AS
SELECT
    p.patient_ID,
    p.first_name,
    p.last_name,
    p.street,
    p.city,
    p.[state],
    p.zip_code,
    p.date_of_birth,
    p.phone_number,
    d.first_name AS doctor_first_name,
    d.last_name AS doctor_last_name,
    d.specialty
FROM
    Patient p
LEFT JOIN
    Appointment a ON p.patient_ID = a.patient_ID
LEFT JOIN
    Doctor d ON a.doctor_ID = d.doctor_ID;
GO
```

```sql
CREATE OR ALTER VIEW PatientTreatmentHistoryView AS
SELECT
    t.patient_ID,
    t.[date],
    ph.diagnosis,
    ph.comments,
    t.[type] AS treatment_type,
    t.prescription,
    d.first_name AS doctor_first_name,
    d.last_name AS doctor_last_name
FROM
    Treatment t
JOIN
    PatientHistory ph ON t.patient_ID = ph.patient_ID AND t.[date] = ph.[date]
JOIN
    Doctor d ON t.doctor_ID = d.doctor_ID;
GO
```

STORED PROCEDURE

```sql
CREATE OR ALTER PROCEDURE ScheduleAppointmentWithPreCheck
    @PatientID INT,
    @DoctorID INT,
    @AppointmentDate DATE,
    @AppointmentLocation VARCHAR(10),
    @IsScheduled BIT OUTPUT
AS
BEGIN
    SET @IsScheduled = 0;

    BEGIN TRY
        BEGIN TRANSACTION;
        IF EXISTS (
            SELECT 1
            FROM Appointment
            WHERE doctor_ID = @DoctorID AND appointment_date = @AppointmentDate
        )
        BEGIN
            SET @IsScheduled = 0;
            GOTO EndProcedure;
        END
        INSERT INTO Appointment (patient_ID, doctor_ID, appointment_date, appointment_location)
        VALUES (@PatientID, @DoctorID, @AppointmentDate, @AppointmentLocation);
        SET @IsScheduled = 1;
        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION;
        SET @IsScheduled = 0;
    END CATCH
    EndProcedure:
END
GO
```

```sql
CREATE OR ALTER PROCEDURE ScheduleTreatmentAndCheckInventory
    @PatientID INT,
    @DoctorID INT,
    @TreatmentDate DATE,
    @TreatmentType VARCHAR(20),
    @InventoryItems InventoryItemTableType READONLY,
    @IsScheduled BIT OUTPUT,
    @InventoryShortage BIT OUTPUT
AS
BEGIN
    SET NOCOUNT ON;
    SET @IsScheduled = 0;
    SET @InventoryShortage = 0;
    DECLARE @ErrorMessage NVARCHAR(4000);
    BEGIN TRY
        BEGIN TRANSACTION;
        DECLARE @ItemID INT, @QuantityUsed INT, @AvailableQuantity INT;
        DECLARE inventory_cursor CURSOR FOR
            SELECT ItemID, QuantityUsed FROM @InventoryItems;
        OPEN inventory_cursor;
        FETCH NEXT FROM inventory_cursor INTO @ItemID, @QuantityUsed;
        WHILE @@FETCH_STATUS = 0
        BEGIN
            SELECT @AvailableQuantity = quantity FROM Inventory WHERE inventory_ID = @ItemID;
            IF @AvailableQuantity < @QuantityUsed
            BEGIN
                SET @InventoryShortage = 1;
                SET @ErrorMessage = 'Not enough inventory for item ID: ' + CAST(@ItemID AS VARCHAR(10));
                THROW 50000, @ErrorMessage, 1; -- Using the prepared message
            END
            UPDATE Inventory SET quantity = quantity - @QuantityUsed WHERE inventory_ID = @ItemID;
            FETCH NEXT FROM inventory_cursor INTO @ItemID, @QuantityUsed;
        END
        CLOSE inventory_cursor;
        DEALLOCATE inventory_cursor;
        IF @InventoryShortage = 0
        BEGIN
            INSERT INTO Treatment(patient_ID, doctor_ID, [date], [type])
            VALUES (@PatientID, @DoctorID, @TreatmentDate, @TreatmentType);
            SET @IsScheduled = 1;
        END
        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION;
        SET @IsScheduled = 0;
    END CATCH
END
GO
```

STORED PROCEDURE

```sql
CREATE OR ALTER PROCEDURE CompleteTreatmentAndUpdateInventory
    @TreatmentID INT,
    @Diagnosis VARCHAR(50),
    @Comments VARCHAR(100),
    @InventoryItems dbo.InventoryItemType READONLY
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @PatientID INT, @Date DATE;

    SELECT @PatientID = patient_ID, @Date = [date] FROM Treatment WHERE treatment_ID = @TreatmentID;

    BEGIN TRY
        BEGIN TRANSACTION;
        IF NOT EXISTS(SELECT 1 FROM PatientHistory WHERE patient_ID = @PatientID AND [date] = @Date)
        BEGIN
            INSERT INTO PatientHistory(patient_ID, [date], diagnosis, comments)
            VALUES (@PatientID, @Date, @Diagnosis, @Comments);
        END
        DECLARE @ItemID INT, @QuantityUsed INT;
        DECLARE cur CURSOR FOR SELECT ItemID, QuantityUsed FROM @InventoryItems;
        OPEN cur;
        FETCH NEXT FROM cur INTO @ItemID, @QuantityUsed;
        WHILE @@FETCH_STATUS = 0
        BEGIN
           FETCH NEXT FROM cur INTO @ItemID, @QuantityUsed;
        END
        CLOSE cur;
        DEALLOCATE cur;
        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        ROLLBACK TRANSACTION;
    END CATCH
END
GO
```

```sql
CREATE OR ALTER PROCEDURE GenerateDetailedMonthlyBillingReport
    @Year INT,
    @Month INT
AS
BEGIN
    SELECT
        p.patient_ID,
        p.first_name + ' ' + p.last_name AS PatientName,
        COUNT(DISTINCT a.appointment_ID) AS AppointmentCount,
        SUM(b.total_amount) AS TotalBilled
    FROM
        Bill b
        INNER JOIN Patient p ON b.patient_ID = p.patient_ID
        LEFT JOIN Appointment a ON p.patient_ID = a.patient_ID
                                AND YEAR(a.appointment_date) = @Year
                                AND MONTH(a.appointment_date) = @Month
    WHERE
        YEAR(b.bill_date) = @Year AND MONTH(b.bill_date) = @Month
    GROUP BY
        p.patient_ID, p.first_name, p.last_name;
END
GO
```

```sql
-- GenerateDoctorActivityReport
CREATE OR ALTER PROCEDURE GenerateDoctorActivityReport
    @DoctorID INT,
    @ReportMonth INT,
    @ReportYear INT,
    @TreatmentCount INT OUTPUT,
    @PatientCount INT OUTPUT,
    @TotalBilling DECIMAL(18,2) OUTPUT
AS
BEGIN
    SELECT
        @TreatmentCount = COUNT(DISTINCT t.treatment_ID),
        @PatientCount = COUNT(DISTINCT t.patient_ID),
        @TotalBilling = SUM(b.total_amount)
    FROM
        Treatment t
    JOIN
        Bill b ON t.patient_ID = b.patient_ID
    WHERE
        t.doctor_ID = @DoctorID
        AND MONTH(t.[date]) = @ReportMonth
        AND YEAR(t.[date]) = @ReportYear;
END
GO
```

TRIGGERS

```sql
-- Delete Trigger for Patient History
CREATE OR ALTER TRIGGER onDelete_PatientHistory
ON PatientHistory
FOR DELETE
AS
BEGIN
    INSERT INTO DeletedPatientHistory (patient_ID, [date], diagnosis, comments)
    SELECT patient_ID, [date], diagnosis, comments
    FROM deleted
END
GO
```

```sql
-- Insert trigger for Appointment
CREATE OR ALTER TRIGGER onInsert_Appointment
ON Appointment
FOR INSERT
AS
BEGIN
    INSERT INTO AppointmentLog (appointment_ID, patient_ID, doctor_ID, appointment_date, appointment_location)
    SELECT appointment_ID, patient_ID, doctor_ID, appointment_date, appointment_location
    FROM inserted
END
GO
```

USER DEFINED FUNCTIONS

```sql
CREATE FUNCTION CalculateAge (@dob date)
RETURNS int
AS
BEGIN
    DECLARE @age int;
    -- Calculate age based on current date and dob
    SET @age = YEAR(GETDATE()) - YEAR(@dob);
    RETURN @age
END
GO
-- Create column using UDF
ALTER TABLE Patient
ADD age int;

UPDATE Patient
SET age = dbo.CalculateAge(date_of_birth)
FROM Patient

select * from Patient
```

```sql
CREATE FUNCTION dbo.IsDoctorAvailable(@doctorID int)
RETURNS bit
AS
BEGIN
    DECLARE @isAvailable bit;

    IF EXISTS (SELECT 1
               FROM Appointment
               WHERE doctor_ID = @doctorID
                 AND CAST(appointment_date AS date) = CAST(GETDATE() AS date))
    BEGIN
        SET @isAvailable = 0; -- Not available
    END
    ELSE
    BEGIN
        SET @isAvailable = 1; -- Available
    END

    RETURN @isAvailable;
END
GO


-- Add a non-persisted computed column to Doctor table
ALTER TABLE Doctor
ADD is_available AS dbo.IsDoctorAvailable(doctor_ID);

-- Querying the Doctor table to see the computed value
SELECT doctor_ID, is_available FROM Doctor;
```

USER DEFINED FUNCTIONS

```sql
CREATE FUNCTION dbo.GetStockStatus(@inventoryID int, @threshold int)
RETURNS VARCHAR(10)
AS
BEGIN
    DECLARE @quantity int;
    DECLARE @status VARCHAR(10);

    SELECT @quantity = quantity FROM Inventory WHERE inventory_ID = @inventoryID;

    SET @status = CASE
                    WHEN @quantity > @threshold THEN 'high'
                    WHEN @quantity < @threshold THEN 'low'
                    ELSE 'normal' -- Optional: Add a 'normal' case if needed
                  END;

    RETURN @status;
END
GO

ALTER TABLE Inventory ADD stock_status varchar(10);

DECLARE @threshold INT = 50;

UPDATE Inventory
SET stock_status = dbo.GetStockStatus(inventory_ID, @threshold);
```

DASHBOARD

HOSPITAL MANAGEMENT SYSTEM

No of Doctors

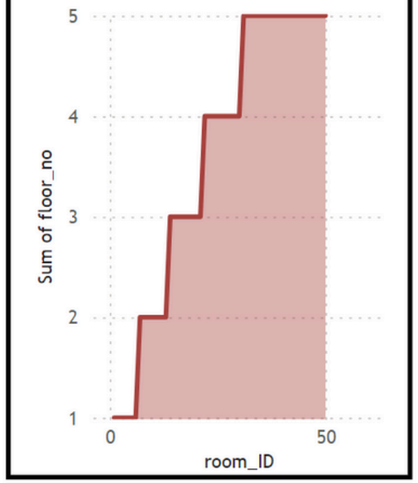
50

Types of Patients at Hospital

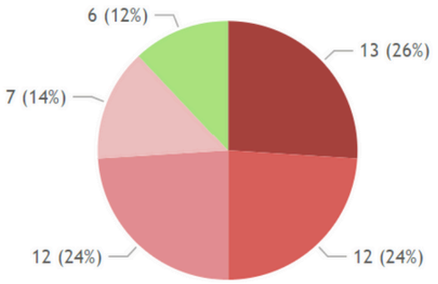
No of Patients

50

No of Rooms per Floor



Insurance Provider Distribution Among Patients



Departments

Categories
Cardiology
Dental
Dermatology
Endocrinology
ENT
ER
Gastroenterology
Neurology
Oncology
Ophthalmology
Orthopedics
Pathology

GUI


Hospital Management System
Patient ID
First Name
Last Name
Street
City
State
Zip Code
Date of Birth
Phone Number
Emergency Contact Name
Emergency Contact Number
Admit Date
Patient Type
Create Record
Read Records
Update Record
Delete Record

Thank you!