Healthcare Management System

Group Number: 19

Members:

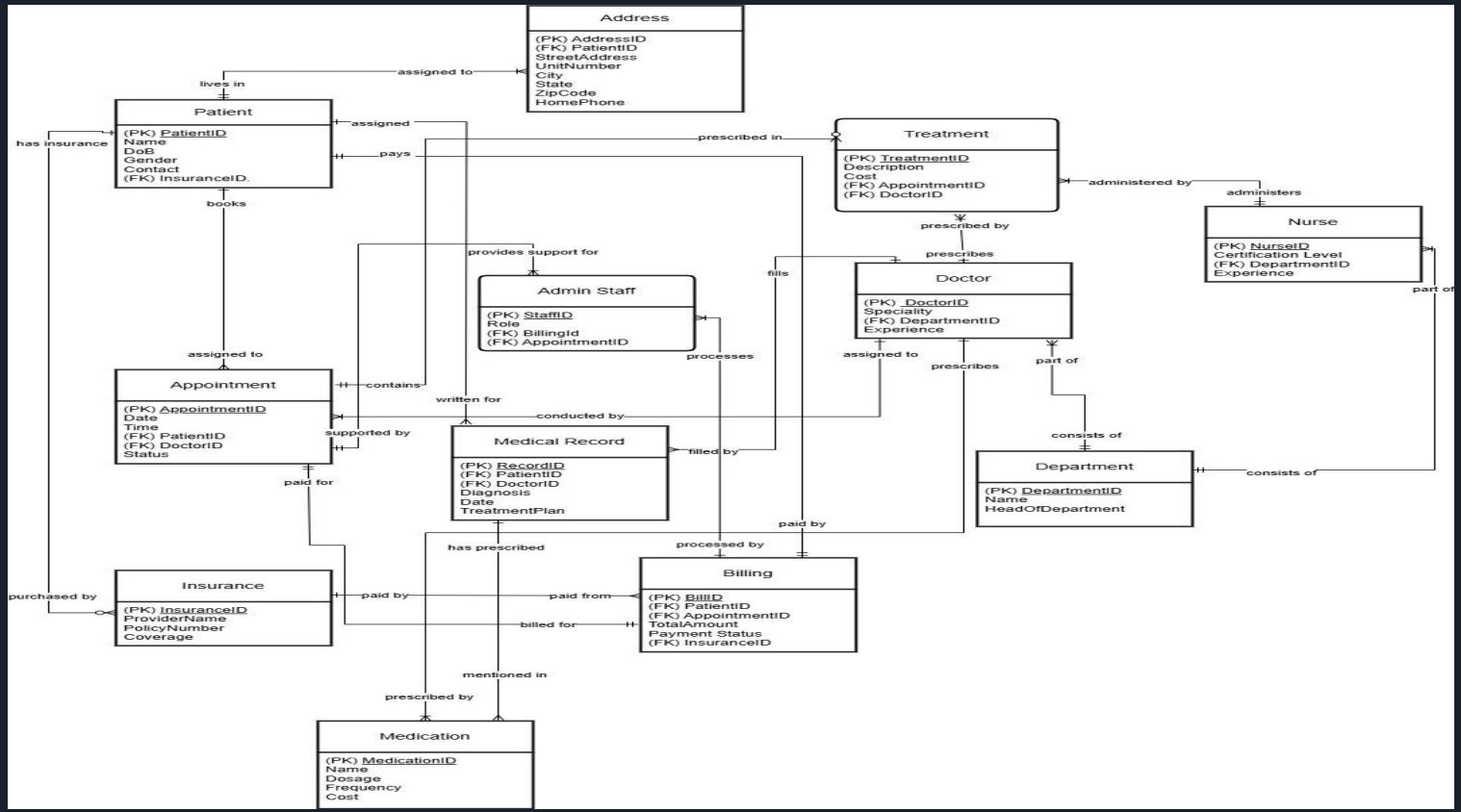
Aakruti Ghatole
Neha Mahangade
Suhas Kommuri
Anurag Kashyap

Project and its Purpose

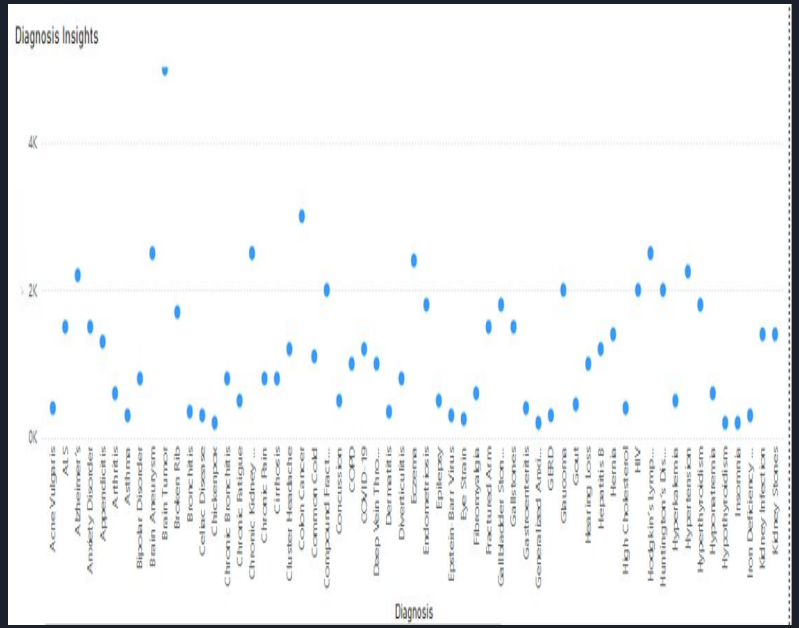
Efficiently managing patient information is critical in healthcare. Challenges like scattered files and poor communication hurt care quality and record maintenance. This project aims to address these by creating a secure, user-friendly platform that simplifies patient management, improves operational efficiency, and ensures robust data protection.

By focusing on seamless communication, task automation, and patient empowerment, the system promises scalability and adaptability to meet evolving industry needs.

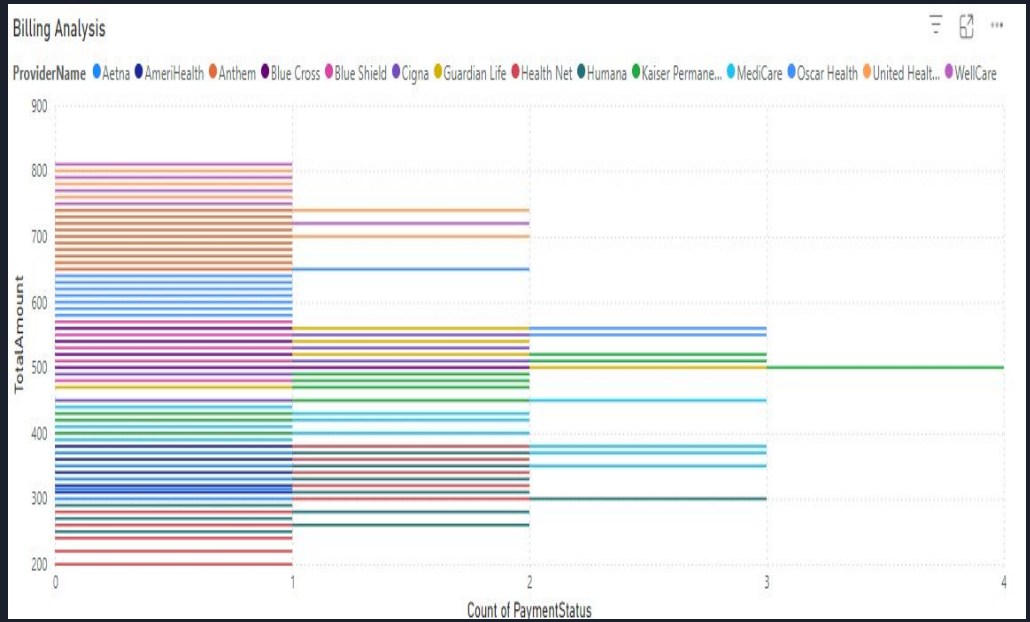
Entity Relationship Diagram



Data Visualisation - Diagnostics Count



Data Visualisation - Billing



Stored Procedures

Stored procedure to count all appointments for a specific date :

```sql
--Procedure 2 : count appointments for a specific date
CREATE PROCEDURE GetAppointmentsByDoctor
    @DoctorID NVARCHAR(50),
    @AppointmentCount INT OUTPUT
AS
BEGIN
    SELECT AppointmentID, AppointmentDate, PatientID, DoctorID
    FROM Appointment
    WHERE DoctorID = @DoctorID;

    SELECT @AppointmentCount = COUNT(*)
    FROM Appointment
    WHERE DoctorID = @DoctorID;
END;
```

```sql
--calling procedure2 to display results
DECLARE @AppointmentCount INT;
EXEC GetAppointmentsByDoctor @DoctorID = 'Doc001',
@AppointmentCount = @AppointmentCount OUTPUT;
SELECT @AppointmentCount AS AppointmentCount;
```

Results | Messages

	TotalBilling
1	650.00

Stored procedure to calculate total billing for a patient :

```sql
--Procedure 3 : calculate total billing for a patient
CREATE PROCEDURE CalculateTotalBilling
    @PatientID VARCHAR(50),
    @TotalAmount DECIMAL(10, 2) OUTPUT
AS
BEGIN
    SELECT @TotalAmount = SUM(TotalAmount)
    FROM Billing
    WHERE PatientID = @PatientID;
END;
```

```sql
--calling procedure3 to display results
DECLARE @TotalAmount DECIMAL(10, 2);
EXEC CalculateTotalBilling @PatientID = P008,
@TotalAmount = @TotalAmount OUTPUT;
SELECT @TotalAmount AS TotalBilling;
```

Results | Messages

	TotalBilling
1	350.00

Views

View for patient insurance details :

```sql
--View for Patient Insurance Details
CREATE VIEW vw_PatientInsurance AS
SELECT
P.PatientID,
P.Name AS PatientName,
P.Contact,
I.ProviderName,
I.PolicyNumber,
I.Coverage,
I.ExpirationDate
FROM Patient P
LEFT JOIN Insurance I ON P.InsuranceID = I.InsuranceID;

SELECT * FROM vw_PatientInsurance;
```

Results | Messages

	P.	Patie…	Co…	Pro…	P…	Co…	Expi…
1	P001	John Doe	111111…	Blue C…	POL001	Full C…	2025-05-…
2	P002	Jane Smith	123-45…	Aetna	POL002	Partia…	2025-11-…
3	P003	Alice Joh…	123-45…	Cigna	POL003	Full C…	2026-05-…
4	P004	Bob Brown	123-45…	United…	POL004	Emerge…	2026-11-…
5	P005	Charlie D…	123-45…	Humana	POL005	Full C…	2027-05-…
6	P006	Diana Gre…	123-45…	Anthem	POL006	Partia…	2027-11-…

View for billing summary :

```sql
--View for Billing Summary
CREATE VIEW vw_BillingSummary AS
SELECT
B.BillID,
P.Name AS PatientName,
A.AppointmentDate,
B.TotalAmount,
B.PaymentStatus,
I.ProviderName AS InsuranceProvider,
I.Coverage
FROM Billing B
JOIN Patient P ON B.PatientID = P.PatientID
JOIN Appointment A ON B.AppointmentID = A.AppointmentID
JOIN Insurance I ON B.InsuranceID = I.InsuranceID;

SELECT * FROM vw_BillingSummary WHERE PaymentStatus = 'Unpaid';
```

Results | Messages

	B	PatientName	Appoin…		P…	I…	Coverage
1	B002	Jane Smith	2024-02-2…	30…	Unp…	Aetna	Partial Coverage
2	B005	Charlie Davis	2024-05-1…	25…	Unp…	Huma…	Full Coverage
3	B007	Eve White	2024-07-1…	40…	Unp…	Kais…	Full Coverage
4	B009	Grace Blue	2024-09-0…	55…	Unp…	Osca…	Full Coverage

User defined Functions

Function to check insurance validity :

```
199  --Function to Check Insurance Validity
200  CREATE FUNCTION dbo.fn_IsInsuranceValid (@InsuranceID VARCHAR(50))
201  RETURNS BIT
202  AS
203  BEGIN
204  DECLARE @IsValid BIT;
205  SELECT @IsValid = CASE WHEN ExpirationDate >= GETDATE() THEN 1 ELSE 0 END
206  FROM Insurance
207  WHERE InsuranceID = @InsuranceID;
208  RETURN ISNULL(@IsValid, 0);
209  END;
210
211  SELECT dbo.fn_IsInsuranceValid('I007') AS IsValidInsurance;
212
```

Results | Messages

	IsValidInsurance
1	1

Function to get average treatment cost for a department :

```
225  --Function to Get Average Treatment Cost for a Department
226  CREATE FUNCTION dbo.fn_GetAvgTreatmentCost (@DepartmentID VARCHAR(50))
227  RETURNS DECIMAL(10, 2)
228  AS
229
230  BEGIN
231  RETURN (SELECT AVG(T.Cost)
232  FROM Treatment T
233  JOIN Doctor D ON T.DoctorID = D.DoctorID
234  WHERE D.DepartmentID = @DepartmentID);
235  END;
236
237  SELECT dbo.fn_GetAvgTreatmentCost('D007') AS AvgCost;
238
```

Results | Messages

	AvgCost
1	1500.00

Indexes :

Index on appointment date for date based searches

```sql
-- Index 2: Non-clustered index on AppointmentDate for date-based searches
CREATE NONCLUSTERED INDEX IDX_AppointmentDate
ON Appointment(AppointmentDate);

-- Test for IDX_AppointmentDate
SELECT * FROM Appointment
WHERE AppointmentDate BETWEEN '2024-01-01' AND '2024-12-31';
```

Results | Messages

	AppointmentID	AppointmentDate	PatientID	DoctorID
1	A001	2024-01-15 09:00:00.000	P001	Doc001
2	A002	2024-02-20 10:00:00.000	P002	Doc002
3	A003	2024-03-25 14:00:00.000	P003	Doc003
4	A004	2024-04-30 11:00:00.000	P004	Doc004
5	A005	2024-05-15 13:00:00.000	P005	Doc005
6	A006	2024-06-20 09:30:00.000	P006	Doc006
7	A007	2024-07-10 08:00:00.000	P007	Doc007
8	A008	2024-08-18 15:00:00.000	P008	Doc008
9	A009	2024-09-05 11:30:00.000	P009	Doc009
10	A010	2024-10-01 10:30:00.000	P010	Doc010

Index on patient for appointment history

```sql
-- Index 3: Non-clustered index on PatientID for patient appointment history
CREATE NONCLUSTERED INDEX IDX_PatientID_Appointments
ON Appointment (PatientID);

-- Test for IDX_PatientID_Appointments
SELECT * FROM Appointment
WHERE PatientID = 'P009';
```

Results | Messages

	AppointmentID	AppointmentDate	PatientID	DoctorID
1	A009	2024-09-05 11:30:00.000	P009	Doc009

Trigger : tracking all DML changes to the patient table

```sql
-- Create a log table for tracking changes
CREATE TABLE PatientLog (
    LogID INT IDENTITY PRIMARY KEY,
    PatientID VARCHAR(50),
    ChangeType VARCHAR(50),
    ChangedOn DATETIME DEFAULT GETDATE(),
    OldContact VARCHAR(15),
    NewContact VARCHAR(15),
    PerformedBy SYSNAME
);

-- Create the DML trigger
CREATE TRIGGER trg_PatientAudit
ON Patient
AFTER UPDATE, DELETE
AS BEGIN
    -- Log updates
    IF EXISTS (SELECT 1 FROM inserted) AND EXISTS (SELECT 1 FROM deleted)
    BEGIN INSERT INTO PatientLog (PatientID, ChangeType, OldContact, NewContact, PerformedBy)
        SELECT d.PatientID, 'UPDATE', d.Contact, i.Contact, SYSTEM_USER
        FROM deleted d INNER JOIN inserted i ON d.PatientID = i.PatientID
        WHERE d.Contact <> i.Contact; END

    -- Log deletions
    IF NOT EXISTS (SELECT 1 FROM inserted) AND EXISTS (SELECT 1 FROM deleted)
    BEGIN INSERT INTO PatientLog (PatientID, ChangeType, OldContact, PerformedBy)
        SELECT PatientID,'DELETE',Contact, SYSTEM_USER
        FROM deleted; END END;

--Test the trigger
UPDATE Patient SET Contact = '1111111111' WHERE PatientID = 'P001';
select * from PatientLog
```

Results | Messages

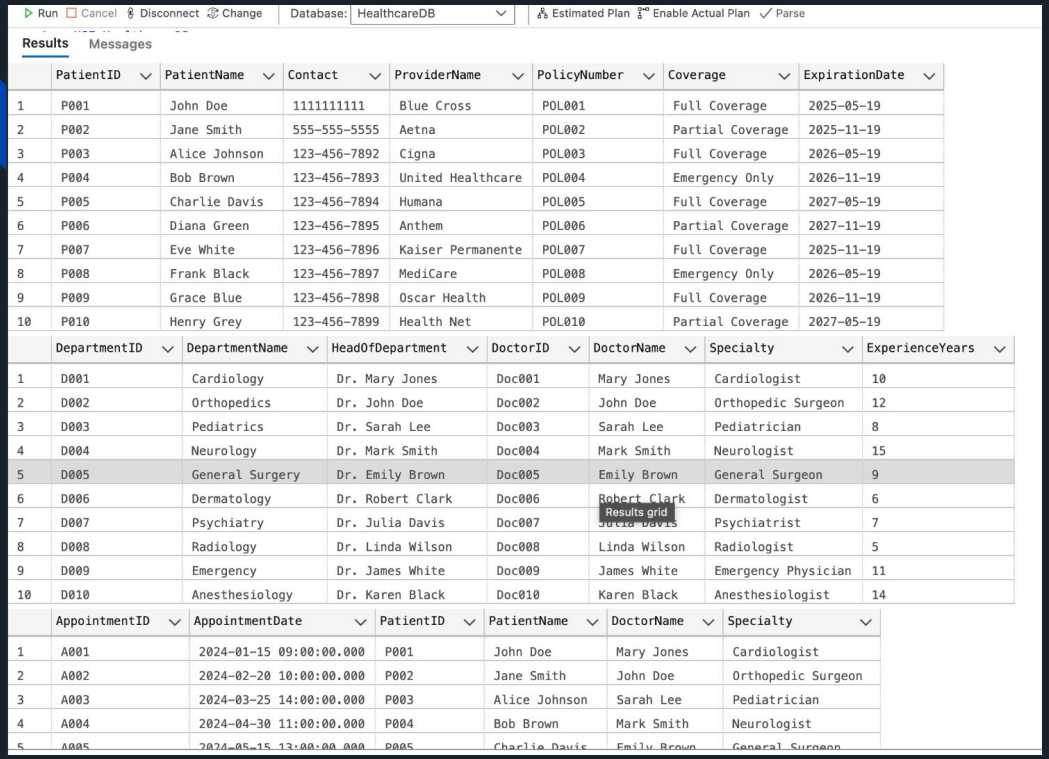
	LogID	PatientID	ChangeType	ChangedOn	OldContact	NewContact	PerformedBy
1	1	P001	UPDATE	2024-11-22 22:30:15.537	123-456-7890	1111111111	sa

Encryption on patient table for contact details column

```sql
-- Step 1: Create a Master Key for encryption
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'StrongPassword123!';

-- Step 2: Create a Certificate for encryption
CREATE CERTIFICATE PatientCert WITH SUBJECT = 'Encryption for Patient Contact';

-- Step 3: Create a Symmetric Key for column encryption
CREATE SYMMETRIC KEY PatientSymKey WITH ALGORITHM = AES_256 ENCRYPTION BY CERTIFICATE PatientCert;

-- Step 4: Add a new column for encrypted data
ALTER TABLE Patient ADD EncryptedContact VARBINARY(MAX);

-- Step 5: Open the Symmetric Key for encryption
OPEN SYMMETRIC KEY PatientSymKey DECRYPTION BY CERTIFICATE PatientCert;

-- Step 6: Encrypt data from the Contact column and store it in the new EncryptedContact column
UPDATE Patient SET EncryptedContact = ENCRYPTBYKEY(KEY_GUID('PatientSymKey'), CAST(Contact AS NVARCHAR(MAX)));

-- Step 7: Close the Symmetric Key after encryption
CLOSE SYMMETRIC KEY PatientSymKey;

-- Step 8: Query to decrypt and view Contact data from the EncryptedContact column
OPEN SYMMETRIC KEY PatientSymKey DECRYPTION BY CERTIFICATE PatientCert;

SELECT PatientID, Name,CONVERT(VARCHAR(15), DECRYPTBYKEY(EncryptedContact)) AS DecryptedContact FROM Patient;

CLOSE SYMMETRIC KEY PatientSymKey;
```

Results | Messages

	PatientID	Name	DecryptedContact
1	P001	John Doe	NULL
2	P002	Jane Smith	NULL
3	P003	Alice Johnson	NULL
4	P004	Bob Brown	NULL
5	P005	Charlie Davis	NULL
6	P006	Diana Green	NULL



	PatientID	PatientName	Contact	ProviderName	PolicyNumber	Coverage	ExpirationDate
1	P001	John Doe	1111111111	Blue Cross	POL001	Full Coverage	2025-05-19
2	P002	Jane Smith	555-555-5555	Aetna	POL002	Partial Coverage	2025-11-19
3	P003	Alice Johnson	123-456-7892	Cigna	POL003	Full Coverage	2026-05-19
4	P004	Bob Brown	123-456-7893	United Healthcare	POL004	Emergency Only	2026-11-19
5	P005	Charlie Davis	123-456-7894	Humana	POL005	Full Coverage	2027-05-19
6	P006	Diana Green	123-456-7895	Anthem	POL006	Partial Coverage	2027-11-19
7	P007	Eve White	123-456-7896	Kaiser Permanente	POL007	Full Coverage	2025-11-19
8	P008	Frank Black	123-456-7897	MediCare	POL008	Emergency Only	2026-05-19
9	P009	Grace Blue	123-456-7898	Oscar Health	POL009	Full Coverage	2026-11-19
10	P010	Henry Grey	123-456-7899	Health Net	POL010	Partial Coverage	2027-05-19

	DepartmentID	DepartmentName	HeadOfDepartment	DoctorID	DoctorName	Specialty	ExperienceYears
1	D001	Cardiology	Dr. Mary Jones	Doc001	Mary Jones	Cardiologist	10
2	D002	Orthopedics	Dr. John Doe	Doc002	John Doe	Orthopedic Surgeon	12
3	D003	Pediatrics	Dr. Sarah Lee	Doc003	Sarah Lee	Pediatrician	8
4	D004	Neurology	Dr. Mark Smith	Doc004	Mark Smith	Neurologist	15
5	D005	General Surgery	Dr. Emily Brown	Doc005	Emily Brown	General Surgeon	9
6	D006	Dermatology	Dr. Robert Clark	Doc006	Robert Clark	Dermatologist	6
7	D007	Psychiatry	Dr. Julia Davis	Doc007	Julia Davis	Psychiatrist	7
8	D008	Radiology	Dr. Linda Wilson	Doc008	Linda Wilson	Radiologist	5
9	D009	Emergency	Dr. James White	Doc009	James White	Emergency Physician	11
10	D010	Anesthesiology	Dr. Karen Black	Doc010	Karen Black	Anesthesiologist	14

Results grid

	AppointmentID	AppointmentDate	PatientID	PatientName	DoctorName	Specialty
1	A001	2024-01-15 09:00:00.000	P001	John Doe	Mary Jones	Cardiologist
2	A002	2024-02-20 10:00:00.000	P002	Jane Smith	John Doe	Orthopedic Surgeon
3	A003	2024-03-25 14:00:00.000	P003	Alice Johnson	Sarah Lee	Pediatrician
4	A004	2024-04-30 11:00:00.000	P004	Bob Brown	Mark Smith	Neurologist
5	A005	2024-05-15 13:00:00.000	P005	Charlie Davis	Emily Brown	General Surgeon

Results grid

	ntID	Name	DoB	Gender	Contact	InsuranceID	EncryptedContact
1	P001	John Doe	1980-02-20	M	1111111111	I001	0x003C34B15CE26947924BCC2E2318D38F020000004835B7A794…

Sample Execution of GUI