PATIENT HEALTH MANAGEMENT SYSTEM

TEAM 2

Team Members

- Shashank Siripragada
- Mayannk Kumaar
- Akanksha Telagam Setty
- Ghanshyam Vibhandik
- Sumit Patil

PURPOSE

The purpose of the database is to provide a patient-centered healthcare system to the hospitals that facilitate monitoring and improvement of their services. Through this database we

- Provide accurate, up-to-date, and complete information about patients.
- Enable quick access to patient records for more coordinated, efficient care.
- Help healthcare providers more effectively diagnose patients, reduce medical errors, and provide safer care.
- Enable healthcare providers to improve efficiency and meet their business goals.
- Reduce costs through decreased paperwork, improved safety, reduced duplication of testing, and improved health.

Tools:

- Data Model: MS-SQL
- Visualization Tool: Tableau
- Others: SQL server, ERStudio

Entities

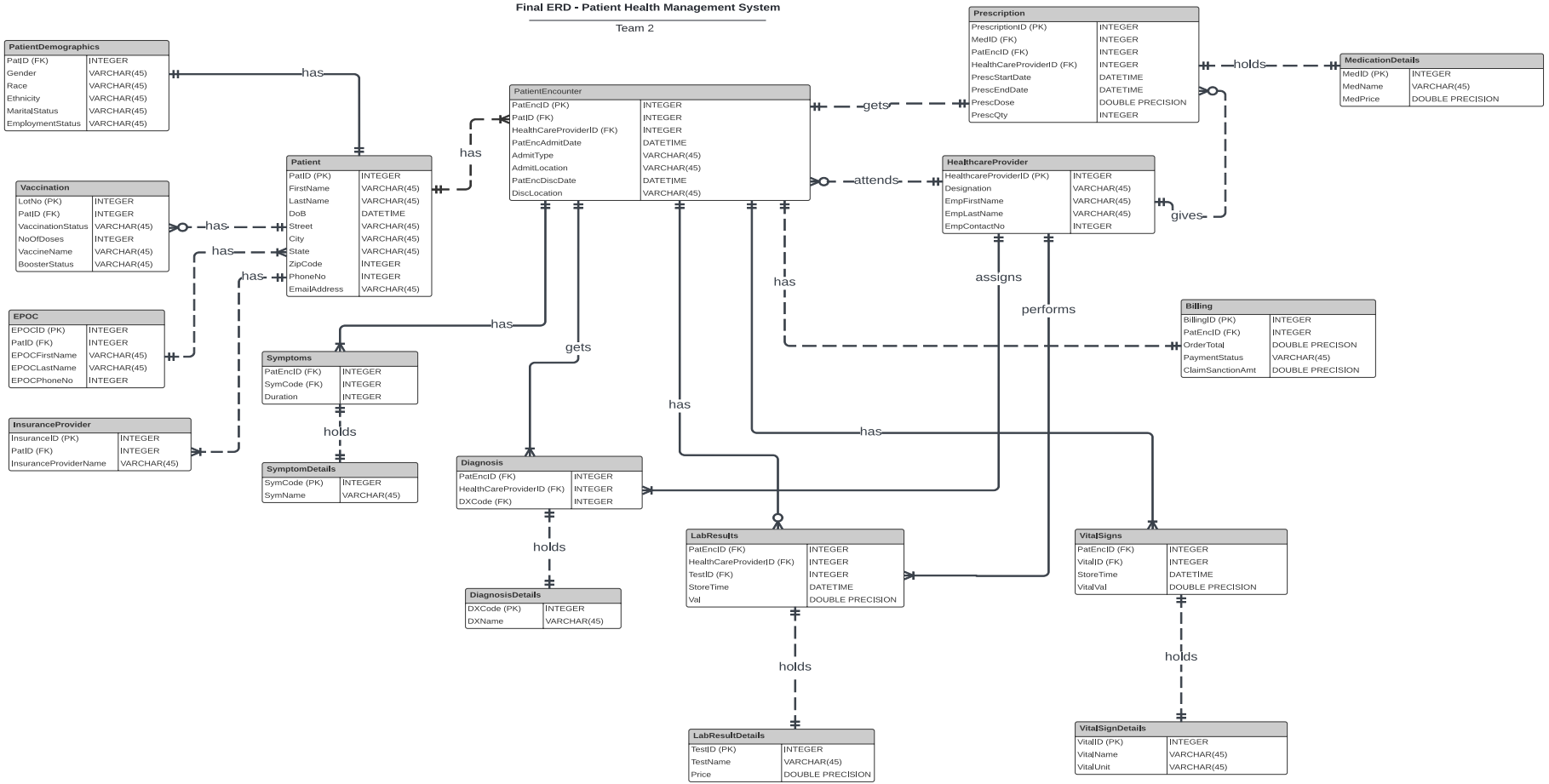
Patient Level

Entity	Overview
Patient	Details of patient
Patient Demographics	Details for patient demographics
Insurance Provider	Lookup table for all insurance providers
Vaccination	Details about patients vaccination records
EPOC	Emergency point of contact details

Patient Encounter Level

Entity	Overview
Patient Encounter	Details of patient visits
Symptoms	Symptoms experienced by Patients
Vital Signs	Vital signs observed in a patient visit
Prescription	Meds prescribed in a patient visit
Diagnosis	Diagnosis assigned to patient.
Lab Results	Details of Lab tests done during a visit.
Billing	Consolidated billing information for a visit

ER Diagram


Final ERD - Patient Health Management System
Team 2
PatientDemographics
PatID (FK) INTEGER
Gender VARCHAR(45)
Race VARCHAR(45)
Ethnicity VARCHAR(45)
MaritalStatus VARCHAR(45)
EmploymentStatus VARCHAR(45)
Patient
PatID (PK) INTEGER
FirstName VARCHAR(45)
LastName VARCHAR(45)
DoB DATETIME
Street VARCHAR(45)
City VARCHAR(45)
State VARCHAR(45)
ZipCode INTEGER
PhoneNo INTEGER
EmailAddress VARCHAR(45)
PatientEncounter
PatEncID (PK) INTEGER
PatID (FK) INTEGER
HealthCareProviderID (FK) INTEGER
PatEncAdmitDate DATETIME
AdmitType VARCHAR(45)
AdmitLocation VARCHAR(45)
PatEncDiscDate DATETIME
DiscLocation VARCHAR(45)
Prescription
PrescriptionID (PK) INTEGER
MedID (FK) INTEGER
PatEncID (FK) INTEGER
HealthCareProviderID (FK) INTEGER
PrescStartDate DATETIME
PrescEndDate DATETIME
PrescDose DOUBLE PRECISION
PrescQty INTEGER
MedicationDetails
MedID (PK) INTEGER
MedName VARCHAR(45)
MedPrice DOUBLE PRECISION
HealthcareProvider
HealthcareProviderID (PK) INTEGER
Designation VARCHAR(45)
EmpFirstName VARCHAR(45)
EmpLastName VARCHAR(45)
EmpContactNo INTEGER
Vaccination
LotNo (PK) INTEGER
PatID (FK) INTEGER
VaccinationStatus VARCHAR(45)
NoOfDoses INTEGER
VaccineName VARCHAR(45)
BoosterStatus VARCHAR(45)
EPOC
EPOCID (PK) INTEGER
PatID (FK) INTEGER
EPOCFirstName VARCHAR(45)
EPOCLastName VARCHAR(45)
EPOCPhoneNo INTEGER
InsuranceProvider
InsuranceID (PK) INTEGER
PatID (FK) INTEGER
InsuranceProviderName VARCHAR(45)
Symptoms
PatEncID (FK) INTEGER
SymCode (FK) INTEGER
Duration INTEGER
SymptomDetails
SymCode (PK) INTEGER
SymName VARCHAR(45)
Diagnosis
PatEncID (FK) INTEGER
HealthCareProviderID (FK) INTEGER
DXCode (FK) INTEGER
DiagnosisDetails
DXCode (PK) INTEGER
DXName VARCHAR(45)
LabResults
PatEncID (FK) INTEGER
HealthCareProviderID (FK) INTEGER
TestID (FK) INTEGER
StoreTime DATETIME
Val DOUBLE PRECISION
LabResultDetails
TestID (PK) INTEGER
TestName VARCHAR(45)
Price DOUBLE PRECISION
VitalSigns
PatEncID (FK) INTEGER
VitalID (FK) INTEGER
StoreTime DATETIME
VitalVal DOUBLE PRECISION
VitalSignDetails
VitalID (PK) INTEGER
VitalName VARCHAR(45)
VitalUnit VARCHAR(45)
Billing
BillingID (PK) INTEGER
PatEncID (FK) INTEGER
OrderTotal DOUBLE PRECISON
PaymentStatus VARCHAR(45)
ClaimSanctionAmt DOUBLE PRECISION
has
gets
attends
holds
gives
assigns
performs

SQL DDL CREATE

```
CREATE TABLE Patient
(
PatID INT PRIMARY KEY,
FirstName VARCHAR(45),
LastName VARCHAR(45),
DoB DateTime,
Street VARCHAR(45),
City VARCHAR(45),
State VARCHAR(45),
ZipCode INT,
PhoneNo BIGINT,
EmailAddress VARCHAR(45)
);
```

```
CREATE TABLE PatientEncounter
(
PatEncID INT PRIMARY KEY,
PatID INT NOT NULL REFERENCES Patient(PatID),
HealthCareProviderID INT NOT NULL
REFERENCES HealthCareProvider(HealthCareProviderID),
PatEncAdmitDate DateTime,
AdmitType VARCHAR(45),
AdmitLocation VARCHAR(45),
PatEncDiscDate DATETIME,
DiscLocation VARCHAR(45)
);
```

```
CREATE TABLE PatientDemographics
(
PatID INT NOT NULL REFERENCES Patient(PatID),
Gender VARCHAR(45),
Race VARCHAR(45),
Ethnicity VARCHAR(45),
MaritalStatus VARCHAR(45),
EmploymentStatus VARCHAR(45)
);

Create table Vaccination
(
LotNo int Primary key,
PatID INT NOT NULL REFERENCES Patient(PatID),
VaccinationStatus VARCHAR(45),
NoOfDoses int,
VaccineName  VARCHAR(45),
BoosterStatus VARCHAR(45)
);


CREATE TABLE EPOC
(
EPOCID int Primary key,
PatID INT NOT NULL REFERENCES Patient(PatID),
EPOCFirstName VARCHAR(45),
EPOCLastName VARCHAR(45),
EPOCPhoneNo BIGINT
);

create table InsuranceProvider
(
InsuranceID int primary key,
PatID INT NOT NULL REFERENCES Patient(PatID),
InsuranceProviderName VARCHAR(45)
);
```

SQL DDL CREATE

```
CREATE TABLE VitalSignDetails
(
VitalID INT PRIMARY KEY ,
VitalName VARCHAR(45),
VitalUnit VARCHAR(45)
);

CREATE TABLE VitalSigns
(
PatEncID INT  NOT NULL REFERENCES PatientEncounter(PatEncID),
VitalID INT NOT NULL REFERENCES VitalSignDetails(VitalID),
StoreTime DATETIME,
VitalVal DOUBLE PRECISION
);
```

```
CREATE TABLE LabResultDetails
(
TestID INT PRIMARY KEY ,
TestName VARCHAR(45),
Price DOUBLE PRECISION
);

CREATE TABLE LabResults
(
PatEncID INT NOT NULL REFERENCES PatientEncounter(PatEncID),
HealthCareProviderID INT NOT NULL
REFERENCES HealthCareProvider(HealthCareProviderID),
TestID INT NOT NULL REFERENCES LabResultDetails(TestID),
StoreTime Date,
Val double precision
);
```

```
CREATE TABLE MedicationDetails
(
MedID INT PRIMARY KEY,
MedName VARCHAR(45),
MedPrice DOUBLE PRECISION
);

 CREATE TABLE Prescription
(
PrescriptionID INT PRIMARY KEY,
MedID INT NOT NULL REFERENCES MedicationDetails(MedID),
PatEncID INT NOT NULL REFERENCES PatientEncounter(PatEncID),
HealthCareProviderID INT NOT NULL
REFERENCES HealthCareProvider(HealthCareProviderID),
PrescStartDate DATETIME,
PrescEndDate DATETIME,
PrescDose FLOAT,
PrescQty FLOAT
);
```

```
create table DiagnosisDetails
(
DxCode int primary key,
DxName varchar(45)
);

create table Diagnosis
(
PatEncID int not null references PatientEncounter(PatEncID),
HealthCareProviderID int not null
references HealthCareProvider(HealthCareProviderID),
DxCode int not null references DiagnosisDetails(DxCode)
);
```

```
CREATE TABLE Billing
(
BillingID INT PRIMARY KEY,
PatEncID INT NOT NULL
REFERENCES PatientEncounter(PatEncID),
PaymentStatus VARCHAR(45),
ClaimSanctionAmt DOUBLE PRECISION
);
```

COMPUTED COLUMNS BASED ON FUNCTIONS

Compute Order Total (Labs + Prescription amt)

```
drop function fn_CalculateOrderTotal

create function fn_CalculateOrderTotal(@PatEncID int)
returns double precision
as
    begin
        Declare @OrderAmt double precision =
            (
            Select
            Price from
            (
            Select
            patenc.PatEncID,
            isnull(sum(lrd.Price), 0)+ sum(meds.MedPrice) as Price
            from PatientEncounter patenc
            left join LabResults lr
                on  patenc.PatEncID = lr.PatEncID
            left join LabResultDetails lrd
                on lr.TestID = lrd.TestID
            left join Prescription ps
                on patenc.PatEncID = ps.PatEncID
            left join MedicationDetails meds
                on ps.MedID = meds.MedID
            group by patenc.PatEncID
            )a
            where a.PatEncID = @PatEncID
            );
        return @OrderAmt;
    end

alter table dbo.Billing
Add OrderTotal as (dbo.fn_CalculateOrderTotal(PatEncID));

Select * from Billing
```

Compute Age

```
drop function fn_CalculateAge;

CREATE FUNCTION fn_CalculateAge(@PatID int)
RETURNS int AS
begin
    Declare @age int =
        (
            SELECT
            DATEDIFF(hour, pat.DOB, GETDATE())/8766 AS Age
            from PHMS.dbo.Patient pat
            WHERE pat.PatID = @PatID
        );
    RETURN @age;
end

alter table dbo.Patient
Add Age as (dbo.fn_CalculateAge(PatID));

select * from Patient
```

Compute Length of Stay of Patient Encounter

```
drop function fn_CalculateLengthOfStay

CREATE FUNCTION fn_CalculateLengthOfStay(@PatEncID INT)
RETURNS INT
AS
   BEGIN
      DECLARE @los int =
         (
          SELECT isnull(DATEDIFF(day, PatEncAdmitDate, PatEncDiscDate),
          DATEDIFF(day, PatEncAdmitDate, GETDATE())) as LengthOfStay
          FROM PHMS.dbo.PatientEncounter patenc
          WHERE PatEncID = @PatEncID
         );
      RETURN @los;
END

ALTER TABLE dbo.PatientEncounter
ADD LengthOfStay AS (dbo.fn_CalculateLengthOfStay(PatEncID));

select * from dbo.PatientEncounter
```

VIEWS

View to obtain all the details of the Patient

```
drop view PatientDetails

CREATE VIEW PatientDetails AS SELECT
pat.PatID, pat.FirstName, pat.LastName, pat.DoB, pat.Street,
pat.City, pat.State, pat.ZipCode, pat.PhoneNo, pat.EmailAddress,
patdemo.Gender, patdemo.Ethnicity, patdemo.MaritalStatus, patdemo.EmploymentStatus,
ins.InsuranceProviderName, epoc.EPOCFirstName, epoc.EPOCLastName, epoc.EPOCPhoneNo
FROM PHMS.dbo.Patient pat
JOIN PHMS.dbo.PatientDemographics patdemo
    ON pat.PatID = patdemo.PatID
JOIN PHMS.dbo.EPOC epoc
    on pat.PatID = epoc.PatID
Join PHMS.dbo.InsuranceProvider ins
    on pat.PatID = ins.PatID

SELECT * FROM PatientDetails;
```

View to obtain all the PatientEncounter level LabResults and VitalSigns Details

```
drop view PatientEncounterLabVitals

CREATE VIEW PatientEncounterLabVitals AS SELECT
patenc.PatEncID,
patenc.AdmitType,
patenc.AdmitLocation
, isnull(lrd.TestName, 'N/A') as TestName
, isnull(lr.val, 0) as LabValue
, vsd.VitalName
, ( cast(vs.VitalVal as varchar) + ' ' + vsd.VitalUnit) as VitalValue
FROM PHMS.dbo.PatientEncounter patenc
left JOIN PHMS.dbo.LabResults lr
    ON patenc.PatEncID = lr.PatEncID
left JOIN PHMS.dbo.LabResultDetails lrd
    on lr.TestID = lrd.TestID
left JOIN PHMS.dbo.VitalSigns vs
    ON patenc.PatEncID = vs.PatEncID
left JOIN PHMS.dbo.VitalSignDetails vsd
    on vs.VitalID = vsd.VitalID

Select * from PatientEncounterLabVitals
```

VIEWS

View to obtain all the PatientEncounter level Billing Details

```
drop view PatientEncounterBilling

CREATE VIEW PatientEncounterBilling AS SELECT
patenc.PatEncID,
isnull(sum(lrd.Price), 0) as LabOrderAmt,
isnull(sum(meds.MedPrice), 0) as MedOrderAmt,
isnull(sum(lrd.Price), 0)+sum(meds.MedPrice) as OrderAmt
from PatientEncounter patenc
left join LabResults lr
    on  patenc.PatEncID = lr.PatEncID
left join LabResultDetails lrd
    on lr.TestID = lrd.TestID
left join Prescription ps
    on patenc.PatEncID = ps.PatEncID
left join MedicationDetails meds
    on ps.MedID = meds.MedID
group by patenc.PatEncID


Select * from PatientEncounterBilling
```

View to obtain all the PatientEncounter level Symptom and Diagnosis Details

```
drop view PatientEncounterSymDiagDetails

CREATE VIEW PatientEncounterSymDiagDetails AS SELECT
patenc.PatEncID,
patenc.PatID,
patenc.HealthCareProviderID,
patenc.PatEncAdmitDate,
patenc.AdmitType,
patenc.AdmitLocation,
patenc.PatEncDiscDate
, patenc.DiscLocation
, symp.SymCode
, sympd.SymName
, diag.DxCode
, diagd.DxName
FROM PHMS.dbo.PatientEncounter patenc
left JOIN PHMS.dbo.Symptoms symp
    ON patenc.PatEncID = symp.PatEncID
left JOIN PHMS.dbo.SymptomDetails sympd
    on symp.SymCode = sympd.SymCode
left JOIN PHMS.dbo.Diagnosis diag
    ON patenc.PatEncID = diag.PatEncID
left JOIN PHMS.dbo.DiagnosisDetails diagd
    on diag.DxCode = diagd.DxCode

Select * from PatientEncounterSymDiagDetails
```

CHECK CONSTRAINTS

Table Level Check on Admitting Physician

```
drop function checkAdmitPhysc

create function checkAdmitPhysc (@HealthcareProviderID int)
returns BIT
begin
   declare @flag BIT;
   declare @des varchar(40);
   if exists (select Designation from HealthCareProvider
            where HealthCareProviderID=@HealthcareProviderID AND
            Designation in ('Attending physician','Emergency physician',
                            'Surgeon','Resident Doctor'))
   begin
       set @flag = 1
   end
   else
       begin
           set @flag = 0
       end
return @flag
end


alter table PatientEncounter
drop constraint ckAdmit

alter table PatientEncounter add CONSTRAINT ckAdmit CHECK (dbo.checkAdmitPhysc (HealthCareProviderID) =1);
```

- Here, the function Accepts the HealthCareProviderID and checks its corresponding designation.
- If the designation is a valid admitting physician, the function sets the flag as 1 indicating that the physician can admit the Patient.

Constraint output

```
Messages
Msg 547, Level 16, State 0, Line 1668
The INSERT statement conflicted with the CHECK constraint "ckAdmit". The conflict occurred in database "PHMS", table "dbo.PatientEncounter", column 'HealthCareProviderID'.
The statement has been terminated.

Completion time: 2022-04-21T23:04:15.3227525-04:00
```

CHECK CONSTRAINTS

Table Level Check on Diagnosing Physician

```
drop function checkDiagnosingPhysc

create function checkDiagnosingPhysc(@HealthcareProviderID int)
returns BIT
begin
   declare @flag BIT;
   declare @des varchar(40);
   if exists (select Designation from HealthCareProvider
   where HealthCareProviderID=@HealthcareProviderID AND Designation in
   (
        'Hospital pharmacist',
        'Chief Medical Officer',
        'Social worker',
        'Physical therapist',
        'Clinical Assitant',
        'Anesthesiologist',
        'Pathologist',
        'Chief executive officer'
   ))
   begin
       set @flag = 0
   end
   else
       begin
           set @flag = 1
       end
return @flag
end


alter table Diagnosis
drop constraint ckDiagnosis

alter table Diagnosis add CONSTRAINT ckDiagnosis CHECK (dbo.checkDiagnosingPhysc (HealthCareProviderID) =1);
```

- Here, the function accepts the HealthCareProviderID and checks its corresponding designation.
- If the designation is valid diagnosing physician, the function sets the flag as 1 indicating that the Physician can diagnose the Patient.

Constraint output

```
Messages
Msg 547, Level 16, State 0, Line 1673
The INSERT statement conflicted with the CHECK constraint "ckDiagnosis". The conflict occurred in database "PHMS", table "dbo.Diagnosis", column 'HealthCareProviderID'.
The statement has been terminated.

Completion time: 2022-04-21T23:14:50.0986226-04:00
```

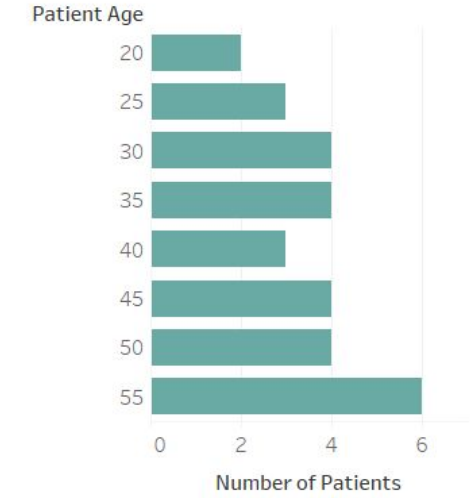
TRIGGERS

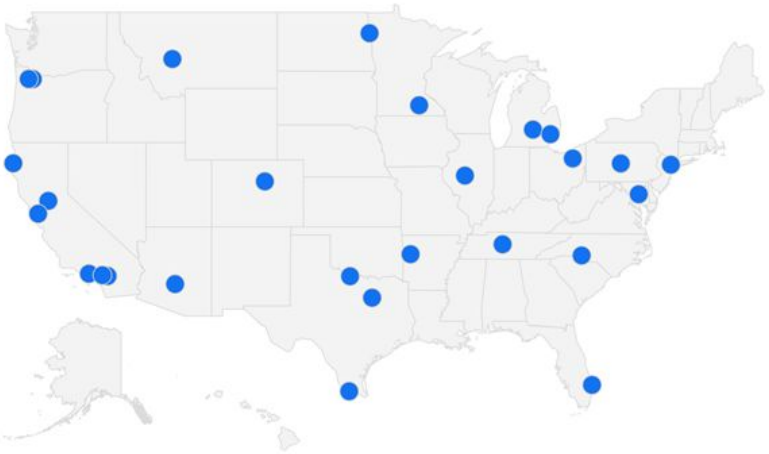
Trigger to check and Update PaymentStatus based on OrderTotal and ClaimSanctionAmt

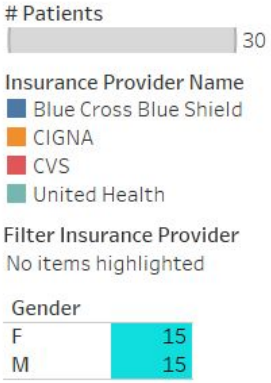
```
Create trigger tr_UpdatePaymentStatus
on Billing
after INSERT, UPDATE, DELETE
As begin
declare @OrderAmt money = 0;
declare @PatEncID varchar(20);
declare @ClaimAmt money = 0;
declare @status varchar(40);
select @PatEncID = isnull (i.PatEncID, d.PatEncID)
    from inserted i full join deleted d
    on i.PatEncID = d.PatEncID;
select @OrderAmt = OrderTotal,
        @ClaimAmt=ClaimSanctionAmt
    from Billing
        where PatEncID = @PatEncID;
    if @ClaimAmt <= (@OrderAmt*0.7)
        begin
            set @status = 'Follow up required'
            PRINT 'PaymentStatus set as - Follow up required'
        end

    else if @ClaimAmt > (@OrderAmt*0.7) AND @ClaimAmt < (@OrderAmt)
        begin
            set @status = 'Partial Payment Received'
            PRINT 'PaymentStatus set as - Partial Payment Received'
        end
    else
        begin
            set @status = 'Complete Payment Received'
            PRINT 'PaymentStatus set as - Complete Payment Received'
        end
    update Billing
    set PaymentStatus = @status
    where PatEncID = @PatEncID
end
```

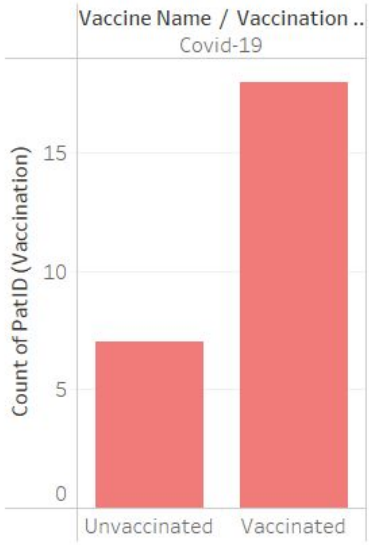
Output

```
(1 row affected)
PaymentStatus set as - Complete Payment Received

(1 row affected)

(1 row affected)
PaymentStatus set as - Partial Payment Received

(1 row affected)

(1 row affected)
PaymentStatus set as - Partial Payment Received

(1 row affected)

(1 row affected)
PaymentStatus set as - Follow up required
```



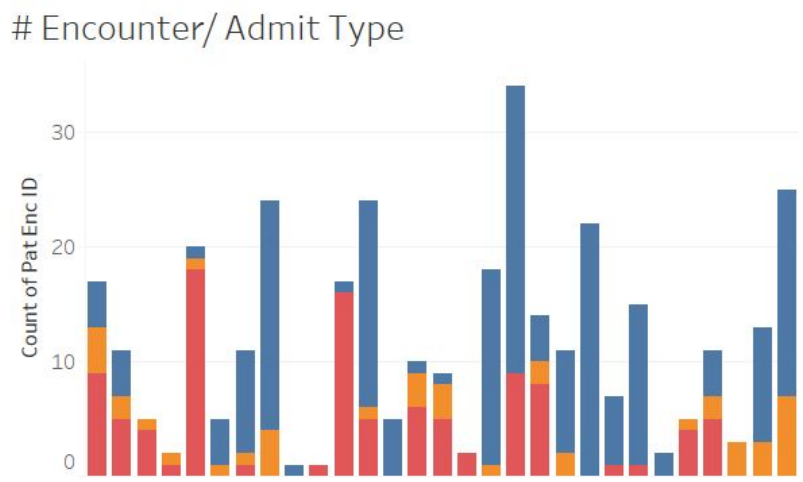


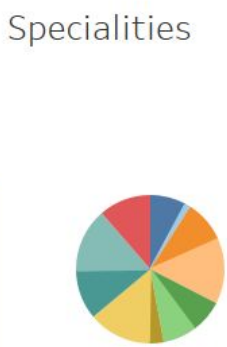


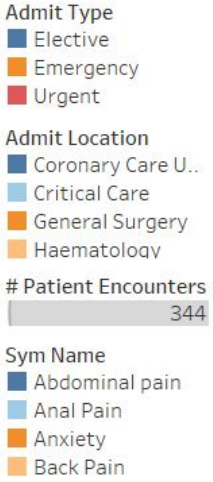


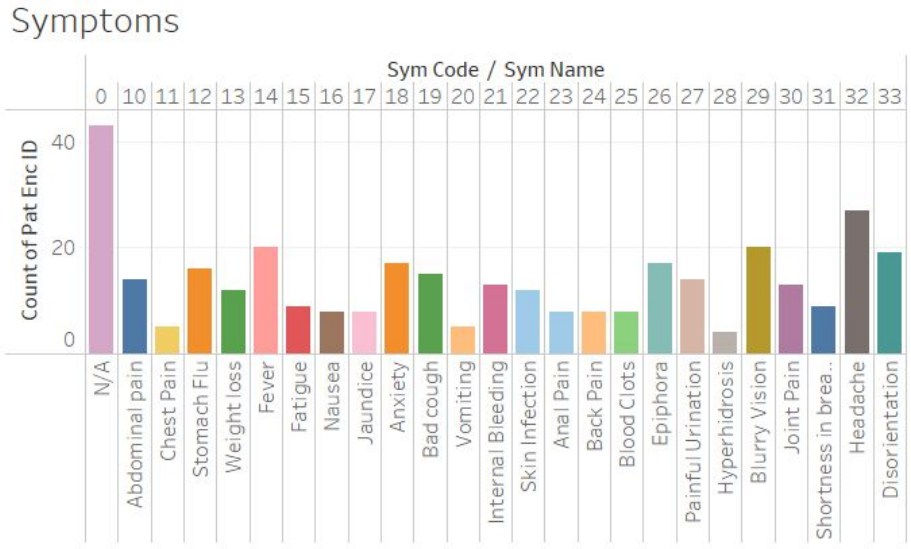
PATIENT LEVEL REPORT

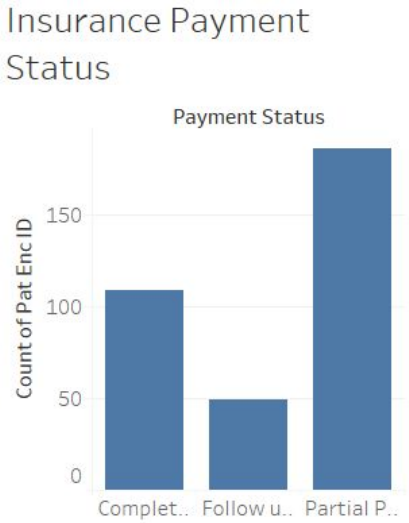
- Pie chart indicating the distribution of insurance providers.
- Bar graph indicating COVID-19 vaccination status among patients.
- Geographical distribution of Patients.







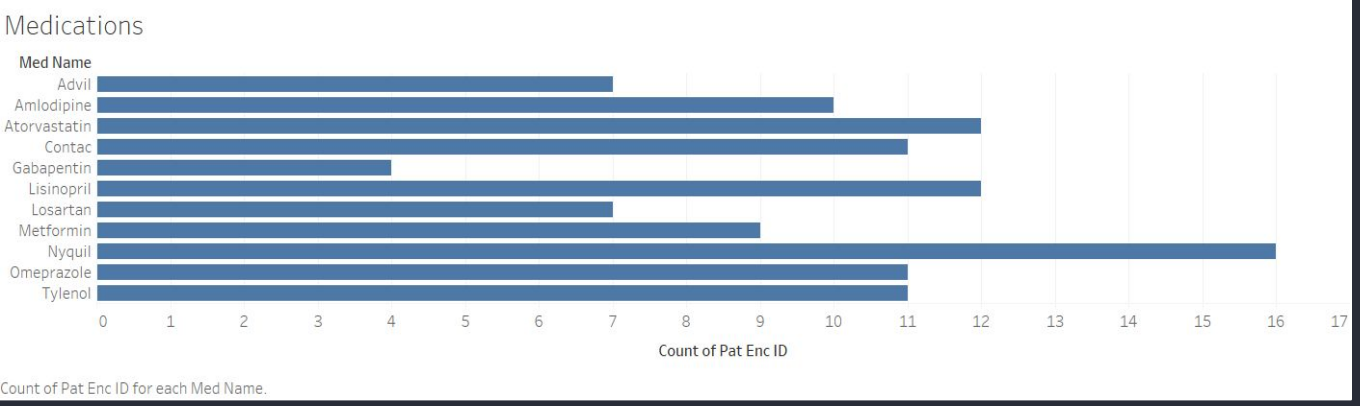


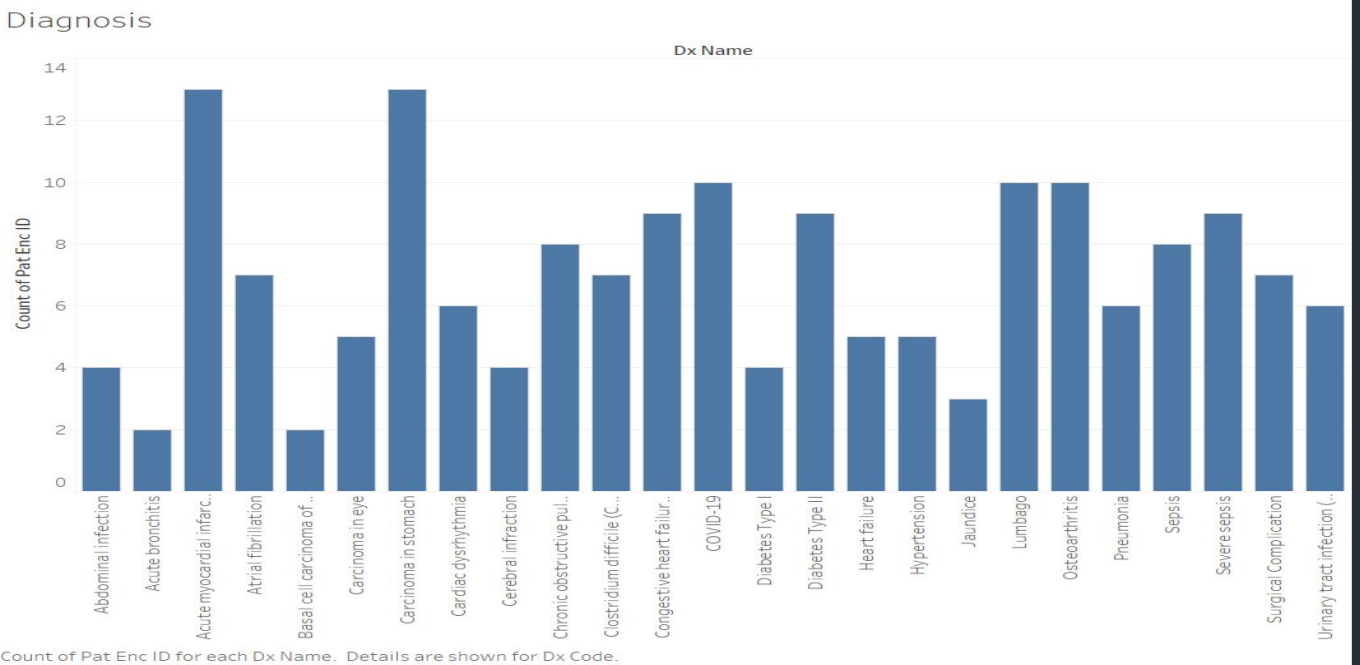


PATIENT ENCOUNTER LEVEL REPORT

- Stacked bar chart indicating # Encounters for an Admit type of a patient.
- Pie chart indicating the % of Patient encounters per speciality in the hospital.
- Bar chart indicating distribution of Symptoms.
- Payment Status bar graph indicates remittance status.

Note: Symptom N/A represents emergency admits





PATIENT ENCOUNTER LEVEL REPORT

- Medications Bar chart indicating distribution of meds prescribed.
- Bar chart indicating distribution of Diagnosis received among the Patient encounters.

CONCLUSION

1. Our database helps the healthcare providers seamless access patient healthcare records.
2. Our database also streamlines assignment of diagnoses and prescriptions to patients thereby reducing potential medical errors and providing safer care.
3. Our database also helps hospital administration with Insurance Claims by accurately calculating patient order totals and by keeping track of the claim sanction amount to provide timely notifications to staff about payment status
4. Our database provides comprehensive reporting on patients which further enables the hospital staff to understand large volumes of patients' data and gain insights

THANK YOU!