Intro To DBMS Project Part -2
Name: - Anoop Reddy Yeddula(Anoop2@pdx.edu)
Siddhartha Challa (Sichalla@pdx.edu)

1)Patients Table:

The patients table would store all the patient information, including:
Patient ID (Primary Key)
Patient Name
Patient Address
Patient Phone Number
Patient Email
Date of Birth
Sex
Blood Group
Insurance Details

Query:

```
CREATE TABLE Patients (
  PatientID INT PRIMARY KEY,
  PatientName VARCHAR(50) NOT NULL,
  PatientAddress VARCHAR(100) NOT NULL,
  PatientPhoneNumber VARCHAR(20) NOT NULL,
  PatientEmail VARCHAR(50) NOT NULL,
  DateOfBirth DATE NOT NULL,
  Sex VARCHAR(10) NOT NULL,
  BloodGroup VARCHAR(5) NOT NULL,
  InsuranceDetails VARCHAR(100)
);
```

PatientID serves as the primary key in this database, and all other fields are necessary (NOT NULL). Because the DateOfBirth field is defined as a DATE data type, you can store dates in a way that makes them simple to query and sort. A VARCHAR(100) data type is used to define the InsuranceDetails field, which can store up to 100 characters of insurance-related data. These fields' dimensions can be changed to suit your requirements.

```
INSERT INTO Patients (PatientID, PatientName, PatientAddress, PatientPhoneNumber,
PatientEmail, DateOfBirth, Sex, BloodGroup, InsuranceDetails)
VALUES
(1, 'John Doe', '123 Main St.', '555-1234', 'johndoe@email.com', '1990-01-01', 'Male', 'O+', 'ABC
Insurance'),
(2, 'Jane Smith', '456 Maple Ave.', '555-5678', 'janesmith@email.com', '1985-05-15', 'Female',
'A-', 'XYZ Insurance');
```

```
select * from patients
```

patientid	patientname	patientaddress	patientphonenumber	patientemail	dateofbirth	sex	bloodgroup	insurancedetails
1	John Doe	123 Main St.	555-1234	johndoe@email.com	1990-01-01	Male	O+	ABC Insurance
2	Jane Smith	456 Maple Ave.	555-5678	janesmith@email.com	1985-05-15	Female	A-	XYZ Insurance

2 rows (0.001 s) Edit, Explain, Export

```
select * from patients;
```

2) Doctors Table

The doctors table would store all the doctor information, including:
Doctor ID (Primary Key)
Doctor Name
Doctor Address
Doctor Phone Number
Doctor Email
Department
Specialization

Query:

```
CREATE TABLE Doctors (
  DoctorID INT PRIMARY KEY,
  DoctorName VARCHAR(50) NOT NULL,
  DoctorAddress VARCHAR(100) NOT NULL,
  DoctorPhoneNumber VARCHAR(20) NOT NULL,
  DoctorEmail VARCHAR(50) NOT NULL,
  Department VARCHAR(50) NOT NULL,
  Specialization VARCHAR(50) NOT NULL
);
```

The primary key in this table is DoctorID, and all other fields are necessary (NOT NULL). The Department and Specialty fields have the data type VARCHAR(50), which allows them to store up to 50 characters of information on the department and specialization of the doctor. These fields' dimensions can be changed to suit your requirements.

```
INSERT INTO Doctors (DoctorID, DoctorName, DoctorAddress, DoctorPhoneNumber,
DoctorEmail, Department, Specialization)
VALUES
(1, 'Dr. John Smith', '123 Main St.', '555-1234', 'johnsmith@email.com', 'Cardiology',
'Cardiologist'),
```

(2, 'Dr. Jane Doe', '456 Maple Ave.', '555-5678', 'janedoe@email.com', 'Pediatrics', 'Pediatrician');

```
select * from doctors
```

doctorid	doctorname	doctoraddress	doctorphonenumber	doctoremail	department	specialization
1	Dr. John Smith	123 Main St.	555-1234	johnsmith@email.com	Cardiology	Cardiologist
2	Dr. Jane Doe	456 Maple Ave.	555-5678	janedoe@email.com	Pediatrics	Pediatrician

2 rows (0.001 s) Edit, Explain, Export

```
select * from doctors;
```

3)Appointments Table

The appointments table would store all the appointment information, including:
Appointment ID (Primary Key)
Patient ID (Foreign Key)
Doctor ID (Foreign Key)
Appointment Date
Appointment Time
Reason for Visit

Query:

```
CREATE TABLE Appointments (
  AppointmentID INT PRIMARY KEY,
  PatientID INT NOT NULL,
  DoctorID INT NOT NULL,
  AppointmentDate DATE NOT NULL,
  AppointmentTime TIME NOT NULL,
  ReasonForVisit VARCHAR(200),
  FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
  FOREIGN KEY (DoctorID) REFERENCES Doctors(DoctorID)
);
```

The primary key in this table is AppointmentID, while the foreign keys PatientID and DoctorID relate to the relevant PatientID and DoctorID entries in the Patients and Doctors databases. It is possible to store the appointment time in a way that can be readily queried and sorted since the AppointmentDate column is defined as a DATE data type and the AppointmentTime field is defined as a TIME data type. The VARCHAR(200) data type that the ReasonForVisit field is defined as can store up to 200 characters of information about the reason for the visit. These fields' dimensions can be changed to suit your requirements.

INSERT INTO Appointments (AppointmentID, PatientID, DoctorID, AppointmentDate, AppointmentTime, ReasonForVisit)
VALUES
(1, 1, 1, '2023-03-06', '10:00:00', 'Chest Pain'),
(2, 2, 2, '2023-03-07', '14:30:00', 'Annual Checkup');

```
select * from appointments
```

appointmentid	patientid	doctorid	appointmentdate	appointmenttime	reasonforvisit
1	1	1	2023-03-06	10:00:00	Chest Pain
2	2	2	2023-03-07	14:30:00	Annual Checkup

2 rows (0.001 s) Edit, Explain, Export

```
select * from appointments;
```

4)Medical Tests Table

The medical tests table would store all the medical test information, including:
Test ID (Primary Key)
Patient ID (Foreign Key)
Doctor ID (Foreign Key)
Test Name
Test Date
Test Results

Query:

```
CREATE TABLE Medical_Tests (
  TestID INT PRIMARY KEY,
  PatientID INT NOT NULL,
  DoctorID INT NOT NULL,
  TestName VARCHAR(50) NOT NULL,
  TestDate DATE NOT NULL,
  TestResults VARCHAR(200),
  FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
  FOREIGN KEY (DoctorID) REFERENCES Doctors(DoctorID)
);
```

TestID serves as the table's main key, while PatientID and DoctorID serve as foreign keys that point to the PatientID and DoctorID entries in the relevant Patients and Doctors databases. The TestName field has the data type VARCHAR(50), meaning it may store up to 50 characters of

test name information. Because the TestDate column is defined as a DATE data type, you can store the test date in a way that makes it simple to query and sort. The VARCHAR(200) data type is used to define the TestResults field, which can store up to 200 characters of data regarding the test results. These fields' dimensions can be changed to suit your requirements.

INSERT INTO Medical_Tests (TestID, PatientID, DoctorID, TestName, TestDate, TestResults)
VALUES
(1, 1, 1, 'Blood Test', '2023-03-08', 'Normal'),
(2, 2, 2, 'X-Ray', '2023-03-09', 'Negative');

```
select * from medical_tests
```

testid	patientid	doctorid	testname	testdate	testresults
1	1	1	Blood Test	2023-03-08	Normal
2	2	2	X-Ray	2023-03-09	Negative

2 rows (0.001 s) Edit, Explain, Export

```
select * from medical_tests;
```

5)Prescriptions Table

The prescriptions table would store all the prescription information, including:
Prescription ID (Primary Key)
Patient ID (Foreign Key)
Doctor ID (Foreign Key)
Medication Name
Dosage
Frequency
Start Date
End Date
These are just some example tables that could be used in a healthcare industry database. The exact schema of the database would depend on the specific needs and requirements of the healthcare organization.

Query:

```
CREATE TABLE Prescriptions (
  PrescriptionID INT PRIMARY KEY,
  PatientID INT NOT NULL,
  DoctorID INT NOT NULL,
```

```
  MedicationName VARCHAR(50) NOT NULL,
  Dosage VARCHAR(20) NOT NULL,
  Frequency VARCHAR(20) NOT NULL,
  StartDate DATE NOT NULL,
  EndDate DATE NOT NULL,
  FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
  FOREIGN KEY (DoctorID) REFERENCES Doctors(DoctorID)
);
```

PrescriptionID serves as the table's main key, while PatientID and DoctorID serve as foreign keys that refer to the relevant PatientID and DoctorID entries in the Patients and Doctors databases. With a maximum character limit of 50, 20, and 20 for the MedicationName, Dose, and Frequency columns, respectively, VARCHAR data types are used by definition. The start and end dates of the prescription can be stored in a way that is simple to query and sort because the StartDate and EndDate fields are defined as DATE data types. These fields' dimensions can be changed to suit your requirements.

```
INSERT INTO Prescriptions (PrescriptionID, PatientID, DoctorID, MedicationName, Dosage,
Frequency, StartDate, EndDate)
VALUES
(1, 1, 1, 'Amoxicillin', '500mg', '3 times a day', '2023-03-10', '2023-03-15'),
(2, 2, 2, 'Ibuprofen', '200mg', 'as needed', '2023-03-11', '2023-03-14');
```

```
select * from Prescriptions
```

prescriptionid	patientid	doctorid	medicationname	dosage	frequency	startdate	enddate
1	1	1	Amoxicillin	500mg	3 times a day	2023-03-10	2023-03-15
2	2	2	Ibuprofen	200mg	as needed	2023-03-11	2023-03-14

2 rows (0.001 s) Edit, Explain, Export

```
select * from Prescriptions;
```

6)Inpatient Details Table:

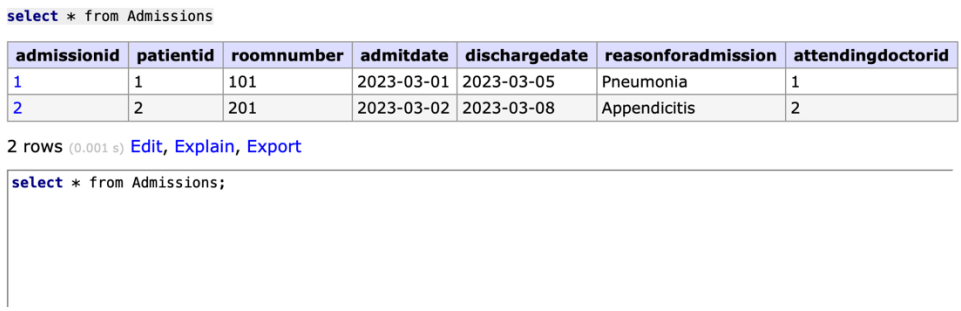
The inpatient details table would store all the information related to patients who have been admitted to the hospital, including:
Admission ID (Primary Key)
Patient ID (Foreign Key)
Room Number
Admit Date
Discharge Date
Reason for Admission
Attending Doctor ID (Foreign Key)

Query:

```
CREATE TABLE Admissions (
  AdmissionID INT PRIMARY KEY,
  PatientID INT NOT NULL,
  RoomNumber INT NOT NULL,
  AdmitDate DATE NOT NULL,
  DischargeDate DATE,
  ReasonForAdmission VARCHAR(200) NOT NULL,
  AttendingDoctorID INT NOT NULL,
  FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
  FOREIGN KEY (AttendingDoctorID) REFERENCES Doctors(DoctorID)
);
```

PatientID and AttendingDoctorID are foreign keys that correspond to the PatientID and DoctorID fields in the Patients and Doctors tables, respectively. AdmissionID serves as the table's main key. The room number can be stored as a whole number because the RoomNumber field is defined as an INT data type. The admit and discharge dates can be stored in a style that is simple to query and sort since the AdmitDate and DischargeDate fields are defined as DATE data types. A VARCHAR(200) data type is used to construct the ReasonForAdmission field, which can store up to 200 characters of information concerning the reason for admission. These fields' dimensions can be changed to suit your requirements.

```
INSERT INTO Admissions (AdmissionID, PatientID, RoomNumber, AdmitDate, DischargeDate,
ReasonForAdmission, AttendingDoctorID)
VALUES
(1, 1, 101, '2023-03-01', '2023-03-05', 'Pneumonia', 1),
(2, 2, 201, '2023-03-02', '2023-03-08', 'Appendicitis', 2),
(3, 3, 301, '2023-03-03', NULL, 'Fractured leg', 3);
```



```
select * from Admissions
```

admissionid	patientid	roomnumber	admitdate	dischargedate	reasonforadmission	attendingdoctorid
1	1	101	2023-03-01	2023-03-05	Pneumonia	1
2	2	201	2023-03-02	2023-03-08	Appendicitis	2

2 rows (0.001 s) Edit, Explain, Export

```
select * from Admissions;
```

7)Insurance Companies Table:

The insurance companies table would store information about the insurance companies that are affiliated with the hospital, including:
Insurance Company ID (Primary Key)
Insurance Company Name

Address
Phone Number
Email
Contact Person

Query:

```
CREATE TABLE InsuranceCompanies (
  InsuranceCompanyID INT PRIMARY KEY,
  InsuranceCompanyName VARCHAR(50) NOT NULL,
  Address VARCHAR(100) NOT NULL,
  PhoneNumber VARCHAR(20) NOT NULL,
  Email VARCHAR(50),
  ContactPerson VARCHAR(50)
);
```

The main key in this table is InsuranceCompanyID. InsuranceCompanyName, Address, PhoneNumber, Email, and ContactPerson are all declared as VARCHAR data types. These data types can each carry up to 50, 100, 20, 50, and 50 characters, respectively. These fields' dimensions can be changed to suit your requirements.

```
INSERT INTO Insurance_Companies (InsuranceCompanyID, InsuranceCompanyName, Address, PhoneNumber, Email, ContactPerson)
VALUES
(1, 'ABC Insurance', '123 Main St, Anytown, USA', '555-1234', 'info@abcinsurance.com', 'Jane Smith'),
(2, 'XYZ Insurance', '456 Elm St, Anytown, USA', '555-5678', 'info@xyzinsurance.com', 'John Doe')
```

```
select * from Insurance_Companies
```

insurancecompanyid	insurancecompanyname	address	phonenumber	email	contactperson
1	ABC Insurance	123 Main St, Anytown, USA	555-1234	info@abcinsurance.com	Jane Smith
2	XYZ Insurance	456 Elm St, Anytown, USA	555-5678	info@xyzinsurance.com	John Doe

2 rows (0.001 s) Edit, Explain, Export

```
select * from Insurance_Companies;
```

8)Billing Table:

The billing table would store all the billing information for patients, including:
Bill ID (Primary Key)
Patient ID (Foreign Key)
Admit ID (Foreign Key)
Test ID (Foreign Key)
Doctor ID (Foreign Key)

Amount Charged
Payment Status

Query:

```
CREATE TABLE Billing (
  BillID INT PRIMARY KEY,
  PatientID INT NOT NULL,
  AdmitID INT,
  TestID INT,
  DoctorID INT NOT NULL,
  AmountCharged DECIMAL(10, 2) NOT NULL,
  PaymentStatus VARCHAR(20) NOT NULL,
  FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
  FOREIGN KEY (AdmitID) REFERENCES Admissions(AdmissionID),
  FOREIGN KEY (TestID) REFERENCES Medical_Tests(TestID),
  FOREIGN KEY (DoctorID) REFERENCES Doctors(DoctorID)
);
```

PatientID, AdmitID, TestID, and DoctorID are foreign keys in this table that relate to the PatientID, AdmissionID, TestID, and DoctorID fields in their respective tables. BillID is the table's main key. The DECIMAL(10, 2) data type, which is how the AmountCharged field is defined, enables you to store decimal numbers up to 10 digits long with 2 digits to the right of the decimal point. The PaymentStatus field is a VARCHAR(20) data type that can store up to 20 characters of payment status information. These fields' dimensions can be changed to suit your requirements. Not all bills will necessarily be connected to an admission or a test, hence the AdmitID and TestID columns allow for NULL values.

```
INSERT INTO Billing (BillID, PatientID, AdmitID, TestID, DoctorID, AmountCharged,
PaymentStatus)
VALUES
  (3, 101, NULL, NULL, 1, 100.00, 'Paid'),
  (4, 1, 2, NULL, 2, 300.00, 'Pending');
```

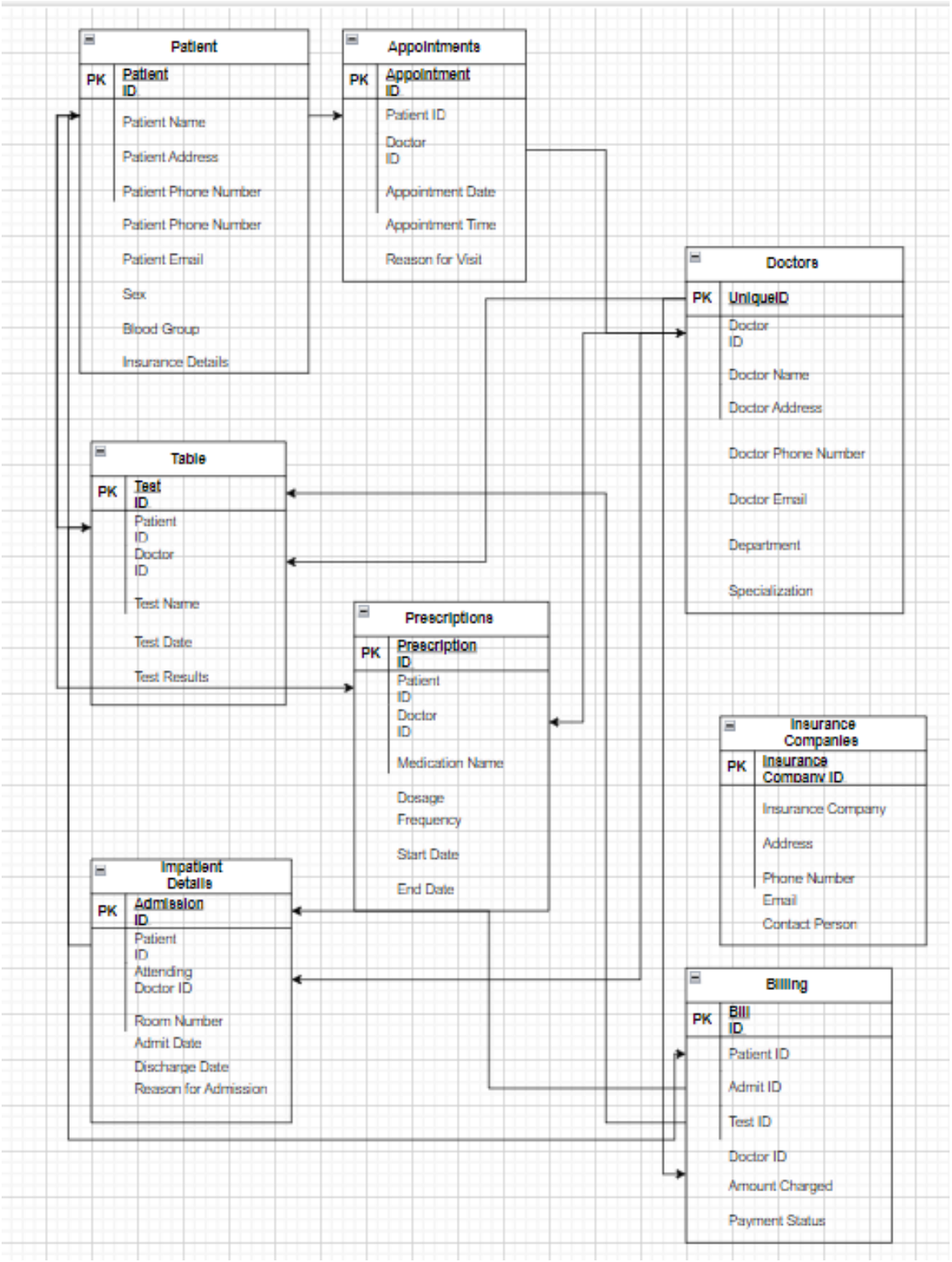
```
select * from billing
```

billid	patientid	admitid	testid	doctorid	amountcharged	paymentstatus
3	101	*NULL*	*NULL*	1	100.00	Paid
4	1	2	*NULL*	2	300.00	Pending

2 rows (0.001 s) Edit, Explain, Export

```
select * from billing;
```

ER Diagram


Patient
PK
Patient ID
Patient Name
Patient Address
Patient Phone Number
Patient Phone Number
Patient Email
Sex
Blood Group
Insurance Details
Appointments
PK
Appointment ID
Patient ID
Doctor ID
Appointment Date
Appointment Time
Reason for Visit
Doctors
PK
UniqueID
Doctor ID
Doctor Name
Doctor Address
Doctor Phone Number
Doctor Email
Department
Specialization
Table
PK
Test ID
Patient ID
Doctor ID
Test Name
Test Date
Test Results
Prescriptions
PK
Prescription ID
Patient ID
Doctor ID
Medication Name
Dosage
Frequency
Start Date
End Date
Insurance Companies
PK
Insurance Company ID
Insurance Company
Address
Phone Number
Email
Contact Person
Inpatient Details
PK
Admission ID
Patient ID
Attending Doctor ID
Room Number
Admit Date
Discharge Date
Reason for Admission
Billing
PK
Bill ID
Patient ID
Admit ID
Test ID
Doctor ID
Amount Charged
Payment Status