Sif Islam

Database Systems CSci 126 (Fresno State)

Prof. David Ruby

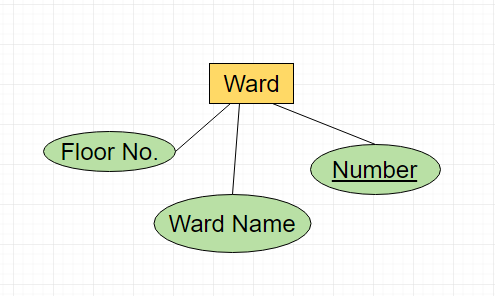
Date: 05/05/2017

*A database management system for Fresno Hospital*

This database is built to work with any hospital system. Currently we will work with a Fresno hospital as a part of initial implementation. Please remember, that this is not a final database system, it has the potential to grow into a vast database system. Thus, more and more features can easily be added to it, and it is not limited to its current capabilities. This database will currently take care of the basic requirement of a general hospital system. I will be able to keep records of wards, doctors, nurses, patients, diagnosis that patient receives, payment made by patients, appointments by patients and many others.

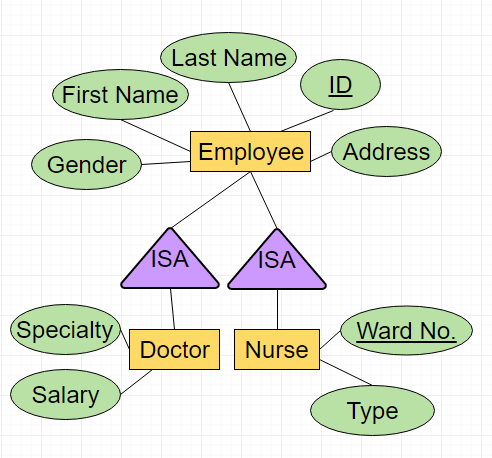
**Part 1. Entities in Fresno Hospital Domain Description**

* Ward
* Employee
  + Doctor
  + Nurse
* Research Lab
* Patient
* Admitted patient
* Appointment
* Diagnosis
* Payment



Ward Domain

* Entity Set: Wards
* Entity: Ward
* Attributes: A ward has…
  + Number (Primary key)
  + Floor No.
  + Ward name

Employee Domain

* Entity Set: Employees
* Entity: Employee
* Attributes: An Employee has…
  + ID (Primary key)
  + First Name
  + Last Name
  + Gender
  + Address

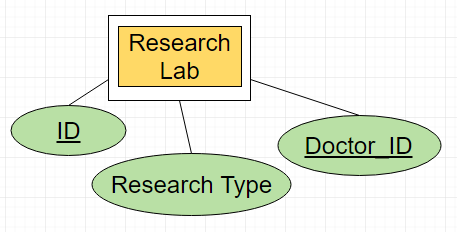
Doctor Domain

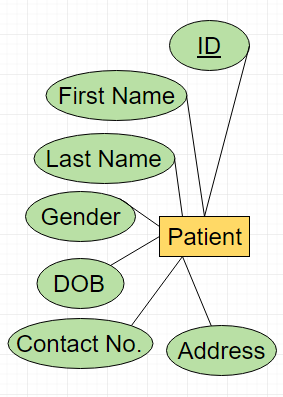
* Attributes: A Doctor has all employee attributes plus
  + Specialty
  + Salary

Nurse Domain

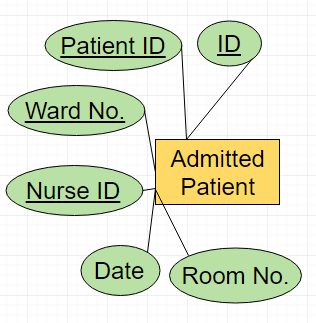
* Attributes: A Nurse has all employee attributes plus
  + Specialty
  + Ward No. (Foreign key)

Research Lab Domain

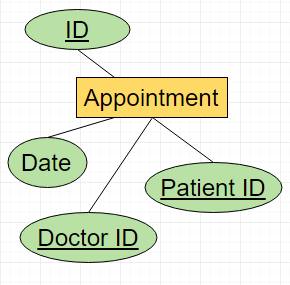
* Entity Set: Research labs
* Entity: Research lab
* Attributes: A Research lab has…
  + ID (Primary key)
  + Research Type
  + Doctor\_ID (Foreign key)

Patient Domain

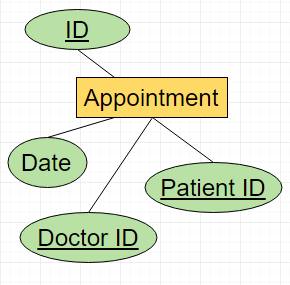
* Entity Set: Patients
* Entity: Patient
* Attributes: A Patient has…
  + ID (Primary key)
  + First Name
  + Last Name
  + Gender
  + DOB
  + Contact No.
  + Address

Admitted Patient Domain

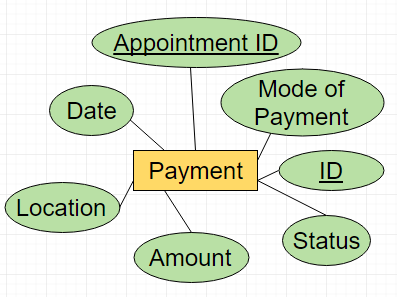
* Entity Set: Admitted Patients
* Entity: Admitted Patient
* Attributes: An Admitted has…
  + ID (Primary key)
  + Patient ID (Foreign Key)
  + Nurse ID (Foreign Key)
  + Date
  + Room No.

Appointment Domain

* Entity Set: Appointments
* Entity: Appointment
* Attributes: An Appointment has…
  + ID (Primary key)
  + Patient ID (Foreign Key)
  + Doctor ID (Foreign Key)
  + Date

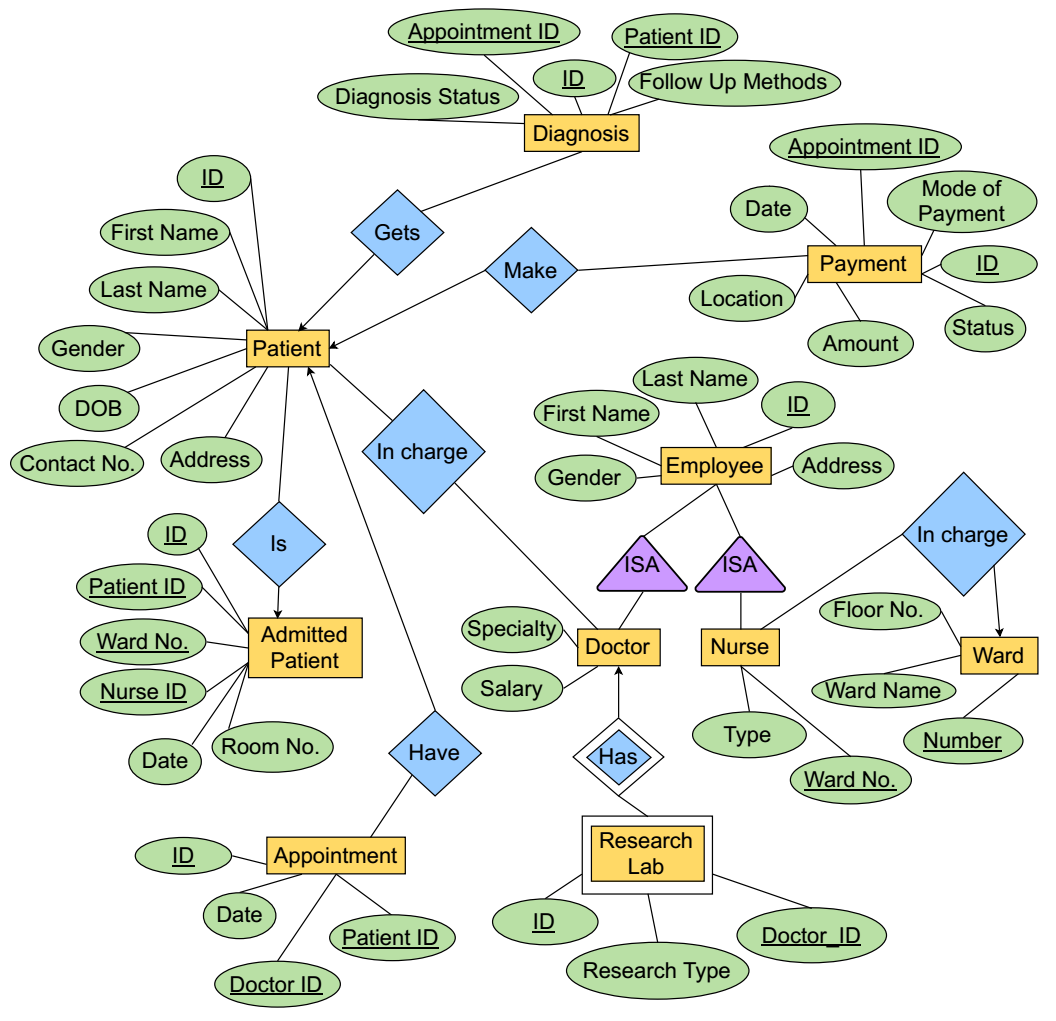
Diagnosis Domain

* Entity Set: Diagnosis
* Entity: Diagnosis
* Attributes: A Diagnosis has…
  + ID (Primary key)
  + Patient ID (Foreign Key)
  + Doctor ID (Foreign Key)
  + Date

Payment Domain

* Entity Set: Payments
* Entity: Payment
* Attributes: A Payment has…
  + ID (Primary key)
  + Appointment ID (Foreign Key)
  + Mode of Payment
  + Status
  + Amount
  + Location
  + Date

**Part 2. Database Design**



Relationships:

|  |  |  |  |
| --- | --- | --- | --- |
| **Domain 1** | **Domain 2** | **Relationship** | **Note** |
| Nurse | Ward | One-to-many | One ward can have multiple nurse, but one nurse can only belong to a single ward |
| Patient | Diagnosis | One-to-many | One patient can have multiple diagnosis |
| Patient | Payment | One-to-many | One patient can have multiple payments |
| Patient | Admitted Patient | One-to-one | One patient cannot have multiple admission at a time |
| Patient | Appointment | One-to-many | One patient can have multiple appointments |
| Doctor | Patient | Many-to-many | One patient can have multiple doctors, and one doctor can also have multiple patients |
| Doctor | Research Lab | Weak-Entity Set | Doctors may belong to a research lab |

**Part 3: Database Schema/Normal Forms**

Ward Domain will have the following SQL INSERT statement

CREATE TABLE Ward(

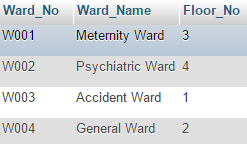
Ward\_No VARCHAR(8) NOT NULL PRIMARY KEY,

Ward\_Name VARCHAR(20) NOT NULL,

Floor\_No VARCHAR(1) NOT NULL

);

Sample looks like this:



The schema is **free** from violations for:

– 3rd Normal form (all column depends on Ward\_No column)

– Boyce-Codd Normal Form (All function dependencies have a key on the left-hand side)

– 4th Normal Form (there are no multi value dependencies)

Doctor Domain will have the following SQL INSERT statement

CREATE TABLE Doctor(

Doctor\_ID VARCHAR(8) NOT NULL PRIMARY KEY,

Doctor\_First\_Name VARCHAR(20) NOT NULL,

Doctor\_Last\_Name VARCHAR(20) NOT NULL,

Doctor\_Gender VARCHAR(1) NOT NULL,

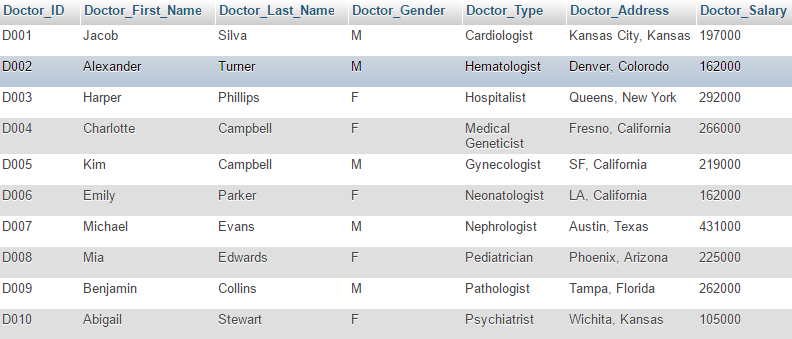
Doctor\_Specialty VARCHAR(20) NOT NULL,

Doctor\_Address VARCHAR(20) NOT NULL,

Doctor\_Salary VARCHAR(10) NOT NULL

);

Sample looks like this:



The schema is **free** from violations for:

– 3rd Normal form (all column depends on Doctor\_ID column)

– Boyce-Codd Normal Form (All function dependencies have a key on the left-hand side)

– 4th Normal Form (there are no multi value dependencies)

Nurse Domain will have the following SQL INSERT statement

CREATE TABLE Nurse(

Nurse\_ID VARCHAR(8) NOT NULL PRIMARY KEY,

Nurse\_First\_Name VARCHAR(20) NOT NULL,

Nurse\_Last\_Name VARCHAR(20) NOT NULL,

Nurse\_Gender VARCHAR(1) NOT NULL,

Ward\_No VARCHAR(8) NOT NULL,

Nurse\_Type Varchar(20),

FOREIGN KEY (Ward\_No) REFERENCES Ward(Ward\_No)

)

Sample looks like this:



The schema is **free** from violations for:

– 3rd Normal form (all column depends on Doctor\_ID column)

– Boyce-Codd Normal Form (All function dependencies have a key on the left-hand side)

The schema is a violation for:

– 4th Normal Form (Column Nurse\_Type has multi value dependencies)

Patient Domain will have the following SQL INSERT statement

CREATE TABLE Patient(

Patient\_ID VARCHAR(8) NOT NULL PRIMARY KEY,

Patient\_First\_Name VARCHAR(20) NOT NULL,

Patient\_Last\_Name VARCHAR(20) NOT NULL,

Patient\_Gender VARCHAR(1) NOT NULL,

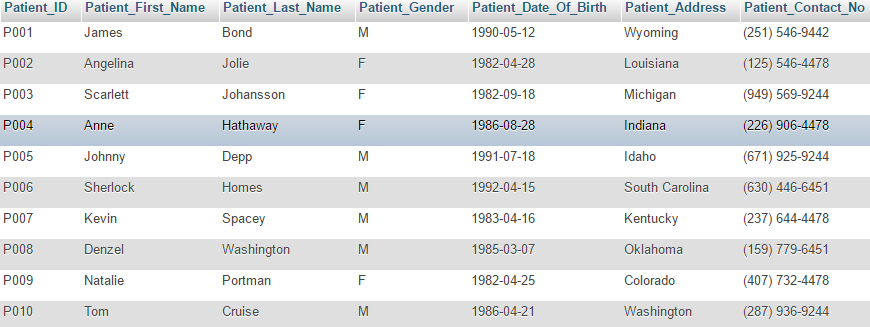
Patient\_Date\_Of\_Birth DATE NOT NULL,

Patient\_Address VARCHAR(20) NOT NULL,

Patient\_Contact\_No VARCHAR(24) NOT NULL

);

Sample looks like this:



The schema is **free** from violations for:

– 3rd Normal form (all column depends on Patient\_ID column)

– Boyce-Codd Normal Form (All function dependencies have a key on the left-hand side)

– 4th Normal Form (there are no multi value dependencies)

Admitted Patient Domain will have the following SQL INSERT statement

CREATE TABLE Admitted\_Patient(

Admitted\_ID VARCHAR(8) NOT NULL PRIMARY KEY,

Patient\_ID VARCHAR(8) NOT NULL,

Ward\_No VARCHAR(8) NOT NULL,

Nurse\_ID VARCHAR(8) NOT NULL,

Admitted\_Date\_Time DATETIME NOT NULL,

Room\_No VARCHAR(20) NOT NULL,

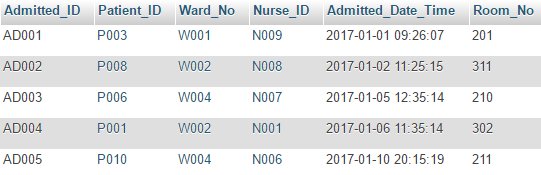
FOREIGN KEY (Patient\_ID) REFERENCES Patient(Patient\_ID),

FOREIGN KEY (Ward\_No) REFERENCES Ward(Ward\_No),

FOREIGN KEY (Nurse\_ID) REFERENCES Nurse(Nurse\_ID)

)

Sample looks like this:



The schema is **free** from violations for:

– 3rd Normal form (all column depends on Admitted\_ID column)

– Boyce-Codd Normal Form (All function dependencies have a key on the left-hand side)

– 4th Normal Form (there are no multi value dependencies)

Appointment Domain will have the following SQL INSERT statement

CREATE TABLE Appointment(

Appointment\_ID VARCHAR(8) NOT NULL PRIMARY KEY,

Patient\_ID VARCHAR(8) NOT NULL,

Doctor\_ID VARCHAR(8) NOT NULL,

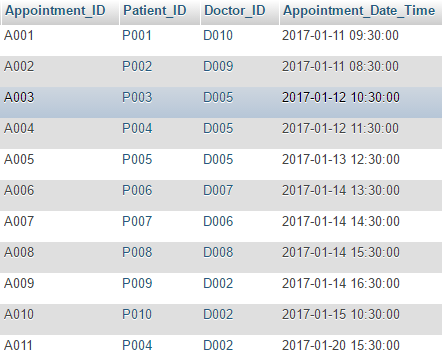
Appointment\_Date\_Time DATETIME NOT NULL,

FOREIGN KEY (Doctor\_ID) REFERENCES Doctor(Doctor\_ID),

FOREIGN KEY (Patient\_ID) REFERENCES Patient(Patient\_ID)

);

Sample looks like this:



The schema is **free** from violations for:

– 3rd Normal form (all column depends on Appointment\_ID column)

– Boyce-Codd Normal Form (All function dependencies have a key on the left-hand side)

– 4th Normal Form (there are no multi value dependencies)

Diagnosis Domain will have the following SQL INSERT statement

CREATE TABLE Diagnosis(

Diagnosis\_ID VARCHAR(8) NOT NULL PRIMARY KEY,

Appointment\_ID VARCHAR(8) NOT NULL,

Patient\_ID VARCHAR(8) NOT NULL,

Diagnosis\_Status VARCHAR(50) NOT NULL,

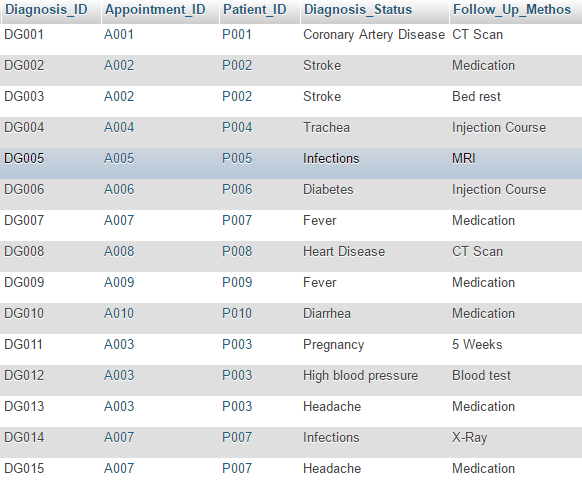
Follow\_Up\_Methods VARCHAR(50) NOT NULL,

FOREIGN KEY (Appointment\_ID) REFERENCES Appointment(Appointment\_ID),

FOREIGN KEY (Patient\_ID) REFERENCES Patient(Patient\_ID)

);

Sample looks like this:



The schema is **free** from violations for:

– 3rd Normal form (all column depends on Diagnosis\_ID column)

– Boyce-Codd Normal Form (All function dependencies have a key on the left-hand side)

– 4th Normal Form (there are no multi value dependencies)

NOTE: Though Diagnosis\_Status column and Follow\_Up\_Methodes column have some data that are common, but these fields are text (comment) fields, they can contain anything, nothing (null value) or duplicates.

Payment Domain will have the following SQL INSERT statement

CREATE TABLE Payment(

Payment\_ID VARCHAR(8) NOT NULL PRIMARY KEY,

Appointment\_ID VARCHAR(8) NOT NULL,

Mode\_Of\_Payment VARCHAR(20) NOT NULL,

Payment\_Date\_Time DATETIME NOT NULL,

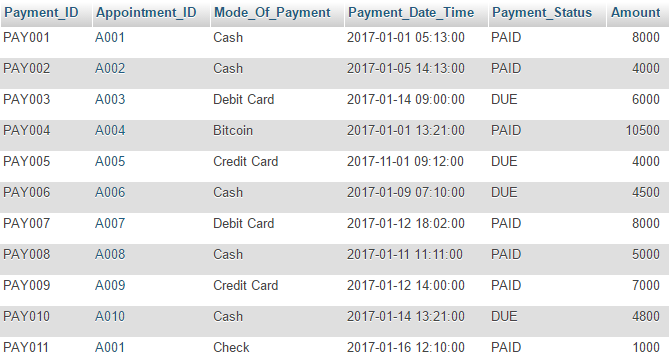
Payment\_Status VARCHAR(20) NOT NULL,

Payment\_Location VARCHAR(20) NOT NULL,

Amount INT NOT NULL,

FOREIGN KEY (Appoinment\_ID) REFERENCES Appointment(Appointment\_ID)

);



The schema is **free** from violations for:

– 3rd Normal form (all column depends on Payment\_ID column)

– Boyce-Codd Normal Form (All function dependencies have a key on the left-hand side)

The schema is a violation for:

– 4th Normal Form (Column Mode\_Of\_Payment and Payment\_Status has multi value dependencies)

Research Lab Domain will have the following SQL INSERT statement

CREATE TABLE Research\_Lab(

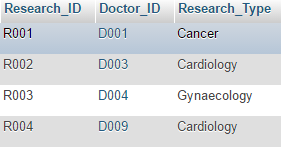
Research\_ID VARCHAR(8) NOT NULL PRIMARY KEY,

Doctor\_ID VARCHAR(8) NOT NULL,

Research\_Type VARCHAR(20) NOT NULL,

FOREIGN KEY (Doctor\_ID) REFERENCES Doctor(Doctor\_ID)

);



The schema is **free** from violations for:

– 3rd Normal form (all column depends on Research\_ID column)

– Boyce-Codd Normal Form (All function dependencies have a key on the left-hand side)

– 4th Normal Form (there are no multi value dependencies)

**Part 4: Queries**

1. A sample select query

**- find a patient by address**

SELECT \*

FROM Patient

WHERE Patient\_Address='South Carolina';

Result:



2. A sample select query with AND clause

**- find a specific doctor**

SELECT \*

FROM Doctor

WHERE Doctor\_Gender='M' AND Doctor\_Last\_Name='Campbell';



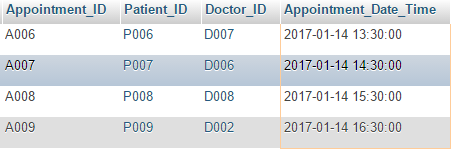
3. A sample select query with build in SQL function “DATEDIFF”

**- find appointment in a given date**

SELECT \*

FROM Appointment

WHERE DATEDIFF(Appointment\_Date\_Time, '2017-01-14')=0



4. A sample Query with subquery

**- find the phone number of female patients who are suffering from strokes**

SELECT Patient\_Contact\_No

FROM Patient

WHERE Patient\_ID IN

( SELECT Patient\_ID

FROM Diagnosis

WHERE Diagnosis\_Status like('Stroke')

)

AND Patient\_Gender like('F');

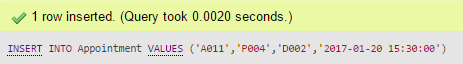


5. A sample Insert Query

**- create a new appointment**

INSERT INTO Appointment VALUES

('A011','P004','D002','2017-01-20 15:30:00');



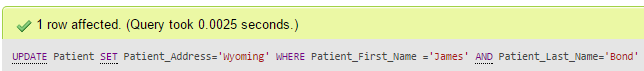
6. A sample Update Query

**- update the address of a given patient**

UPDATE Patient

SET Patient\_Address='Wyoming'

WHERE Patient\_First\_Name ='James' AND Patient\_Last\_Name='Bond';



7. A more advanced select Query

**- find assigned doctors for a selected patient, and his salary**

SELECT Doctor\_First\_Name, Doctor\_Last\_Name, Doctor\_Salary

FROM Appointment a, Doctor d

WHERE

d.Doctor\_ID = a.Doctor\_ID

AND

a.Patient\_ID =

(

SELECT Patient\_ID FROM Patient

WHERE Patient\_First\_Name= 'Scarlett'

AND Patient\_Last\_Name= 'Johansson'

)

;



8. Another more advanced select Query

**- find number of upcoming appointments between dates**

SELECT count(Appointment\_ID) AS Appointments\_Upcomming

FROM Appointment

WHERE Appointment\_Date\_Time

BETWEEN '2017-01-14 00:00:00'

AND '2017-01-15 23:59:00';



9. Another more advanced select Query with Aggregation

**- find total amount paid by a single patient**

SELECT SUM(Amount) AS Total\_Amount\_Paid\_By\_James\_Bond

FROM Payment p, Appointment a

WHERE

a.Appointment\_ID = p.Appointment\_ID

AND

a.Patient\_ID = (

SELECT Patient\_ID FROM Patient

WHERE Patient\_First\_Name = 'James'

AND Patient\_Last\_Name = 'Bond'

);



10. Another more advanced select Query with Subquery and Order by clause

**- find patient with appointment that listed ‘fever’ as their illness**

SELECT Patient\_First\_Name, Patient\_Last\_Name

FROM Patient

WHERE Patient\_ID IN

(SELECT Patient\_ID FROM Appointment WHERE Appointment\_ID IN

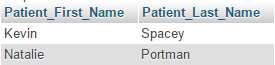
(SELECT Appointment\_ID

FROM Diagnosis

WHERE Diagnosis\_Status='Fever')

)

ORDER BY Patient\_ID;



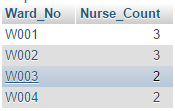
11. Query with Count clause

**- find number of nurse assigned for each ward**

SELECT Ward\_No, COUNT(\*) AS Nurse\_Count

FROM Nurse

GROUP BY Ward\_No



12. Query with Subquery

**- find nurses in a certain ward with patients in it**

SELECT Nurse\_First\_Name, Nurse\_Last\_Name

FROM Nurse

WHERE Nurse\_ID IN (

SELECT Nurse\_ID FROM Admitted\_Patient

WHERE Ward\_No='W001'

);



13. Another Query with Aggregation

**- find maximum payment received by the hospital by a single patient**

SELECT Mode\_Of\_Payment, Amount AS Max\_Payment\_Made\_By\_Patient

FROM Payment

WHERE Amount=(SELECT MAX(Amount) FROM Payment);



14. Query with Subquery and Aggregation

**- count the number of mail patient that are currently admitted in the hospital**

SELECT COUNT(Admitted\_ID) AS Male\_Patient\_COUNT

FROM Admitted\_Patient

WHERE Patient\_ID IN(

SELECT Patient\_ID FROM Patient

WHERE Patient\_Gender='M'

);



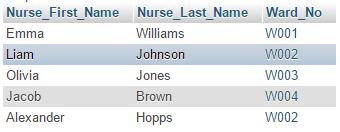
15. Query with NOT IN clause

**- Find free nurses**

SELECT Nurse\_First\_Name, Nurse\_Last\_Name, Ward\_No

FROM Nurse

WHERE Nurse\_ID NOT IN (SELECT Nurse\_ID FROM Admitted\_Patient);



16. An advanced DISTINCT Query with Order by clause

**- Find doctors who are treating ‘Diabetes’**

SELECT DISTINCT doc.Doctor\_ID, Doctor\_First\_Name, Doctor\_Last\_Name

FROM Doctor doc, Appointment a, Diagnosis dig

WHERE doc.Doctor\_ID = a.Doctor\_ID

AND dig.Diagnosis\_Status = 'Diabetes'

AND dig.Appointment\_ID = a.Appointment\_ID

ORDER BY Doctor\_ID;

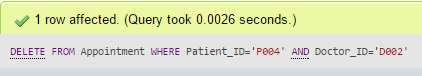


17. Delete Query

**- remove an appointment**

DELETE FROM Appointment

WHERE Patient\_ID='P004' AND Doctor\_ID='D002';



18. Insert Query (add multiple appointments)

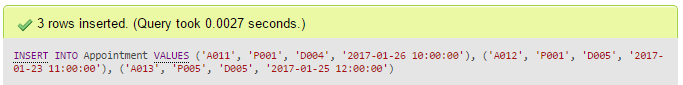
**- create multiple appointments at the same time**

INSERT INTO Appointment VALUES

('A011', 'P001', 'D004', '2017-01-26 10:00:00'),

('A012', 'P001', 'D005', '2017-01-23 11:00:00'),

('A013', 'P005', 'D005', '2017-01-25 12:00:00');



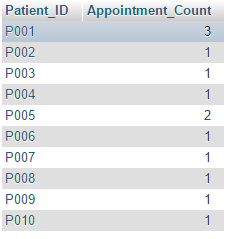
19. Verifying Insert Query

**[Appointment count for Patient P001 is now 3]**

SELECT Patient\_ID, COUNT(\*) AS Appointment\_Count

FROM Appointment

GROUP BY Patient\_ID;



20. Select Query with HAVING keyword

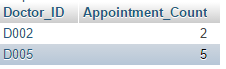
**- find Doctors who have multiple appointments**

SELECT Doctor\_ID,COUNT(Appointment\_ID) AS Appointment\_Count

FROM Appointment

GROUP BY Doctor\_ID

HAVING COUNT(Appointment\_ID) >= 2;



**Part 5: Application/Use Cases**

This database management system is designed to work with any hospital system. It has all the basic requirements to manage Doctors, Nurse, Patients. Including assigning nurses to specific wards and assigning doctors to patients. Doctors are also allowed to have their own (optional) research labs. This system also has the appointment scheduling and payment system build into it. Once a patient is admitted to the hospital, doctors can assign appropriate diagnosis methods that suits the patient.

As mentioned earlier, this is an open-end system, modifications are welcome. Thus, it has the room to grow big and bigger. And once this database grows to its potential, it can not only be used by a hospital system but in variety of other systems. People can perform research based on this database. For example, they can calculate the average cost of a patient visiting a hospital in a certain area, thus enabling them to compare and contrast this data to make a table/graph for some project or paper. This is just one small example; this database is not limited to calculating payments. Not only humans, but AI like IBM Watson can use this database to gather knowledge about the diagnosis of patients and increase its knowledge base allowing it to be more aware. One example for Watson would be to calculate the main type of illness that people get in a certain area. Later, Watson can use this data to determine diagnosis not only based on a person’s health but also taking the factor of patient’s location, which a real doctor might overlook. These few example proves that this system can be used for border range of tasks, including maintaining a hospital. It will grow bigger, and help may in the process.

Thank you