Term Project
Ahmed Ashah
Pawel Rozanski

1- Assumptions

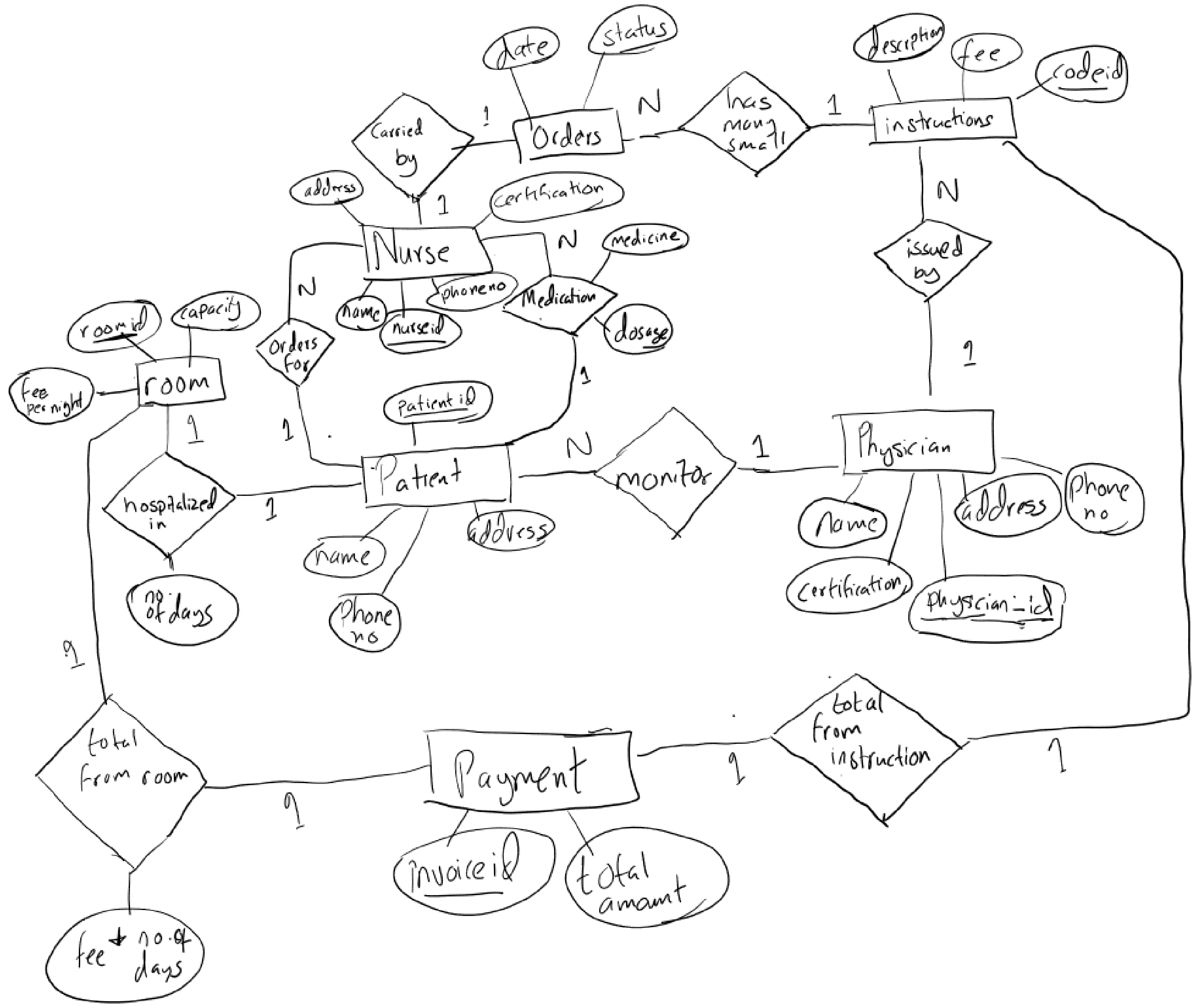
The Orders that the nurses execute are a part of the Instructions concerned with the physician. That is why the physician table and the orders table are connected with each instruction having 1 or many orders carried by 1 or many nurses.

Since the medicine is all taken from outside, it is not covered in the total payment.

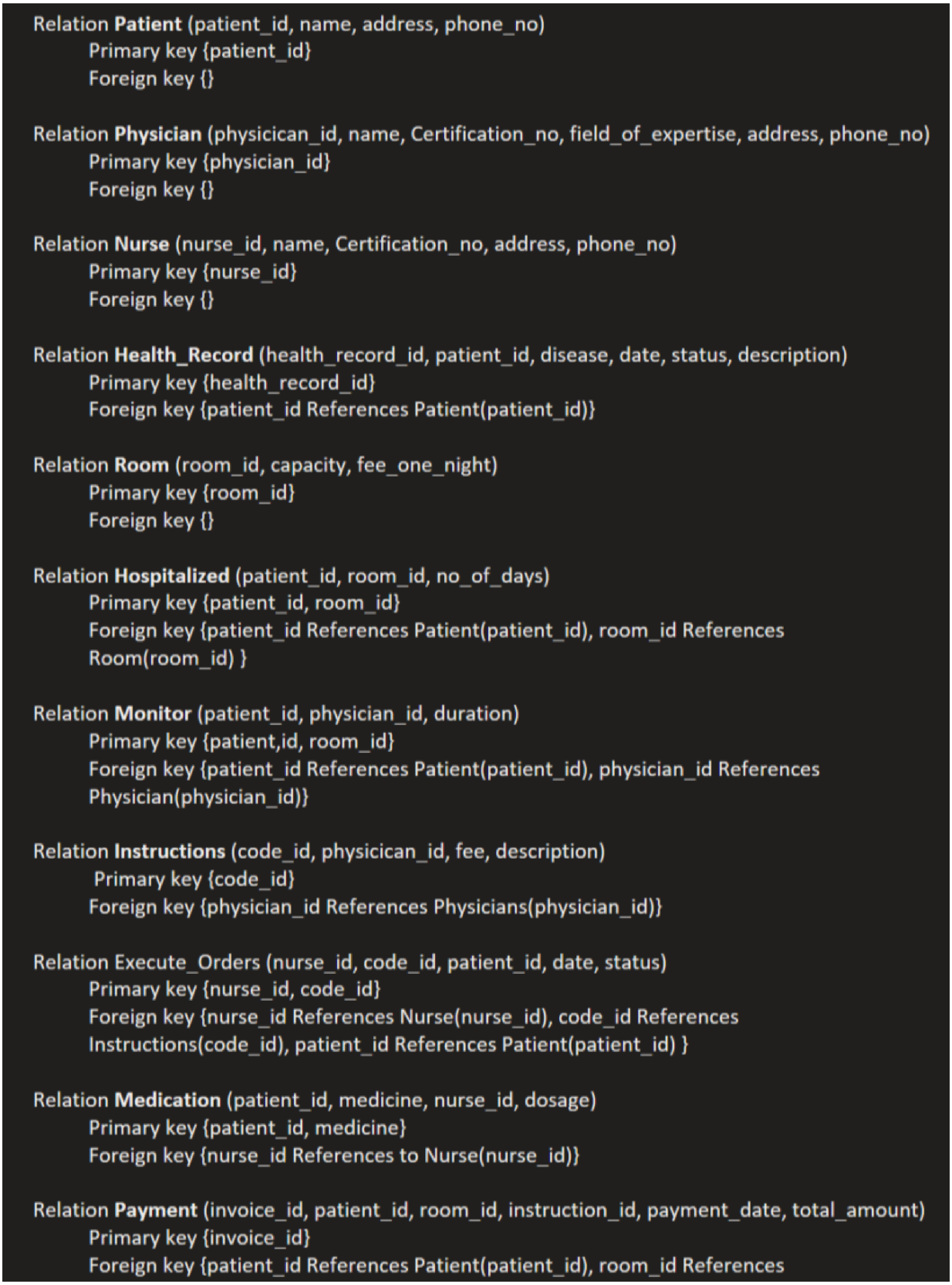
A patient can only have one physician monitoring them but can have multiple nurses executing orders for the patient.

Only one instrunction can be happen per patient.

2- EER Diagram



3 - Relations and Keys



Relation **Patient** (patient_id, name, address, phone_no)
    Primary key {patient_id}
    Foreign key {}

Relation **Physician** (physicican_id, name, Certification_no, field_of_expertise, address, phone_no)
    Primary key {physician_id}
    Foreign key {}

Relation **Nurse** (nurse_id, name, Certification_no, address, phone_no)
    Primary key {nurse_id}
    Foreign key {}

Relation **Health_Record** (health_record_id, patient_id, disease, date, status, description)
    Primary key {health_record_id}
    Foreign key {patient_id References Patient(patient_id)}

Relation **Room** (room_id, capacity, fee_one_night)
    Primary key {room_id}
    Foreign key {}

Relation **Hospitalized** (patient_id, room_id, no_of_days)
    Primary key {patient_id, room_id}
    Foreign key {patient_id References Patient(patient_id), room_id References Room(room_id) }

Relation **Monitor** (patient_id, physician_id, duration)
    Primary key {patient,id, room_id}
    Foreign key {patient_id References Patient(patient_id), physician_id References Physician(physician_id)}

Relation **Instructions** (code_id, physicican_id, fee, description)
    Primary key {code_id}
    Foreign key {physician_id References Physicians(physician_id)}

Relation Execute_Orders (nurse_id, code_id, patient_id, date, status)
    Primary key {nurse_id, code_id}
    Foreign key {nurse_id References Nurse(nurse_id), code_id References Instructions(code_id), patient_id References Patient(patient_id) }

Relation **Medication** (patient_id, medicine, nurse_id, dosage)
    Primary key {patient_id, medicine}
    Foreign key {nurse_id References to Nurse(nurse_id)}

Relation **Payment** (invoice_id, patient_id, room_id, instruction_id, payment_date, total_amount)
    Primary key {invoice_id}
    Foreign key {patient_id References Patient(patient_id), room_id References

4 - Views and Descriptions

View 1:
This view shows us the name of the patient and what room they are residing in

```
CREATE VIEW room_check AS
SELECT p.name AS 'Patient Name', room_id as 'Room number', no_of_days as 'Days In' FROM
hospitalized
JOIN patient p ON p.patient_id = hospitalized.patient_id;
```

This can be useful to quickly find out where a patient is located if they have a visitor or if a doctor/nurse needs to find them quickly

View 2:
This view shows us the patient's bill information

```
CREATE VIEW price_check AS
SELECT p.name as 'Patient Name', total_amount as 'Total Due', payment_date as 'Due Date'
FROM payment
JOIN patient p ON p.patient_id = payment.patient_id;
```

This can be used when the patient wants to see how much they owe the hospital and when the bill is due

View 3:
This view shows which patient is taking what medicine and the nurse that is in charge of that medication

```
CREATE VIEW medication_information AS
SELECT n.name AS 'Nurse in charge', p.name AS 'Recieving Patient', medicine, dosage
FROM medication
JOIN patient p ON p.patient_id = medication.patient_id
JOIN nurse n ON n.nurse_id = medication.nurse_id;
```

This can be useful when a doctor needs to check what medication the patient is currently taking and which nurse to contact for more information

5- Triggers and Descriptions

Trigger 1:
This trigger auto increments the patient id's whenever a patient without a valid ID is entered

```
mysql> delimiter //
CREATE  TRIGGER  auto_assign_patient
BEFORE insert
ON patient FOR EACH ROW
BEGIN
IF new.patient_id IS NULL THEN
SET new.patient_id = (SELECT MAX(patient_id) FROM patient) + 1;
END IF;
END//
mysql> delimiter ;
```

This can be useful for the person entering the patients information into the system they don't have to remember the id of the previous patient

Trigger 2:
This trigger automatically adds up the total cost that the patient owes

```
mysql> delimiter //
CREATE  TRIGGER  auto_add_total
BEFORE insert
ON payment FOR EACH ROW
BEGIN
IF new.total_amount IS NULL THEN
SET new.total_amount = new.amount_room + new.amount_instruction;
END IF;
END//
mysql> delimiter ;
```

This can be useful because the person entering the information wont have to
manually add up the costs of the procedures and rooms together

Trigger 3:
This trigger finds the cost of a procedure that's in the bill

```
mysql> delimiter //
CREATE  TRIGGER  auto_add_instruction_fee
BEFORE insert
ON payment FOR EACH ROW
BEGIN
IF new.amount_instruction IS NULL THEN
SET new.amount_instruction = (SELECT fee FROM instructions WHERE instructions.code_id =
new.instruction_id);
END IF;
END//
mysql> delimiter ;
```

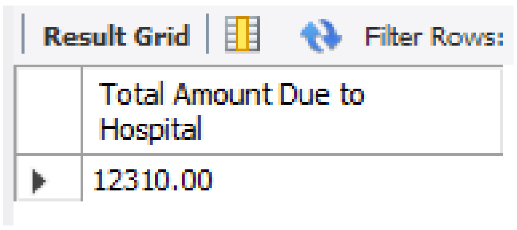
This can be useful because the program will automatically find the cost of the procedure and input it so the costs of procedures don't have to be known by the person entering them

6- Queries, descriptions, and results

Query 1: This query returns the total amount due to the hospital

```
SELECT SUM(total_amount) AS 'Total Amount Due to Hospital'
FROM payment;
```

Results:

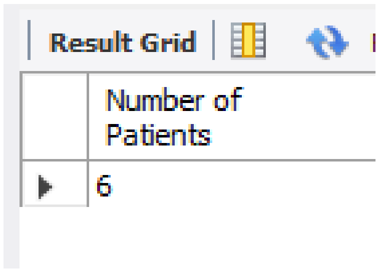


```
Total Amount Due to Hospital
12310.00
```

Query 2: This Query returns the number of patients

```
SELECT COUNT(patient_id) AS 'Number of Patients' FROM patient;
```

Results:

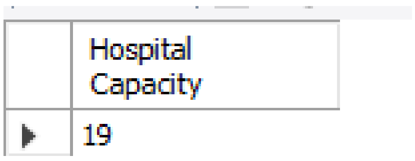


```
Number of Patients
6
```

Query 3: This query returns the total capacity the hospital has

```
SELECT SUM(capacity) AS 'Hospital Capacity' FROM room;
```

Results:

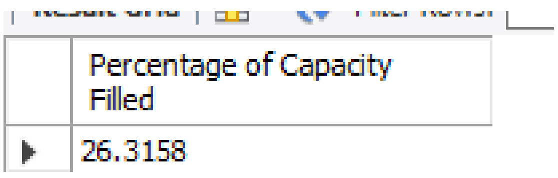


```
Hospital Capacity
19
```

Query 4: This query returns the percentage of the capacity filled

```
SELECT (COUNT(patient.patient_id) / SUM(capacity)) * 100
AS 'Percentage of Capacity Filled' FROM patient, room;
```

Results:

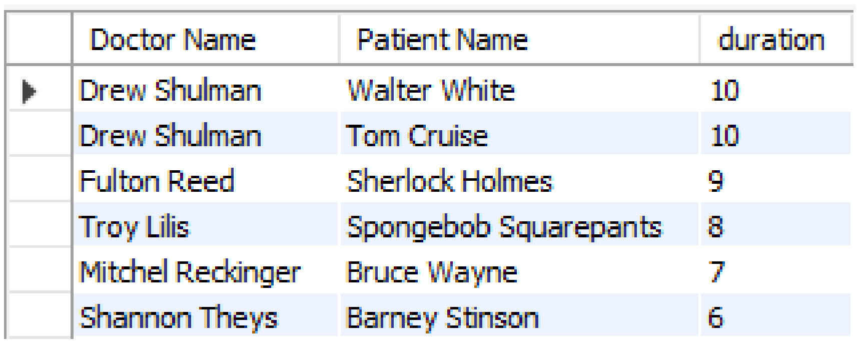


```
Percentage of Capacity Filled
26.3158
```

Query 5: This query returns which doctor is monitoring what patient and for how long

```
SELECT d.name AS 'Doctor Name', p.name AS 'Patient Name', duration FROM monitor
JOIN patient p ON p.patient_id = monitor.patient_id
JOIN physician d ON d.physician_id = monitor.physician_id;
```

Results:

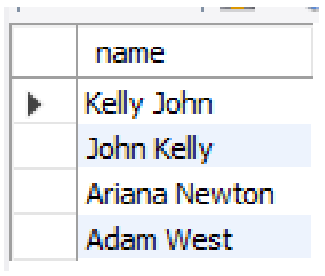


Doctor Name	Patient Name	duration
Drew Shulman	Walter White	10
Drew Shulman	Tom Cruise	10
Fulton Reed	Sherlock Holmes	9
Troy Lilis	Spongebob Squarepants	8
Mitchel Reckinger	Bruce Wayne	7
Shannon Theys	Barney Stinson	6

Query 6: Returns names of nurses working on orders

```
SELECT name FROM NURSE
WHERE nurse.nurse_id IN
(SELECT nurse_id FROM Execute_Orders
WHERE patient_id = 1);
```

Results:

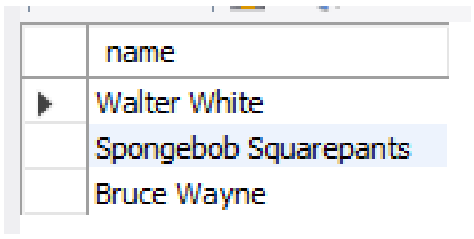


name
Kelly John
John Kelly
Ariana Newton
Adam West

Query 7: Returns names of patients with an active disease

```
SELECT name FROM Patient
WHERE patient.patient_id IN (SELECT patient_id FROM Health_record
WHERE status = 'active');
```

Results:

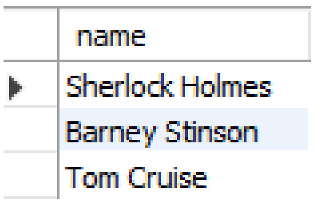


name
Walter White
Spongebob Squarepants
Bruce Wayne

Query 8: Returns list of patients without an active disease

```
SELECT name FROM Patient
WHERE patient.patient_id NOT IN (SELECT patient_id FROM Health_record
WHERE status = 'active');
```

Results:

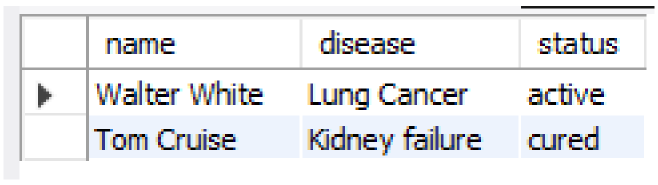


name
Sherlock Holmes
Barney Stinson
Tom Cruise

Query 9: Returns the health record of any patient that is being monitored by Drew Shulman

```
SELECT Patient.name, Health_record.disease, Health_record.status FROM Patient
JOIN Health_record ON Health_record.patient_id = Patient.patient_id
JOIN Monitor ON Monitor.patient_id = Patient.Patient_id
WHERE Monitor.Physician_id IN (SELECT physician_id FROM Physician WHERE Physician.name =
'Drew Shulman');
```

Results:

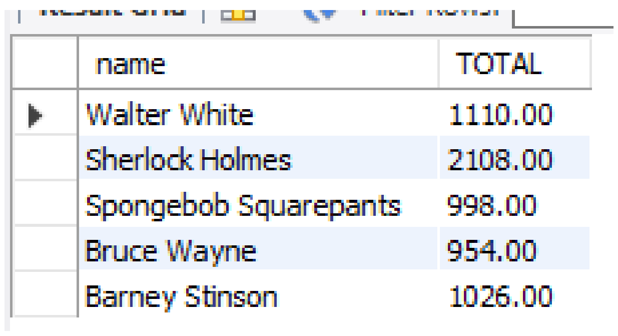


name	disease	status
Walter White	Lung Cancer	active
Tom Cruise	Kidney failure	cured

Query 10: Returns the total amount of money each patient owes

```
SELECT name, ((Hospitalized.no_of_days * Room.room_id) + Instructions.fee) AS TOTAL FROM
Patient
JOIN Hospitalized ON Patient.patient_id = Hospitalized.patient_id
JOIN Payment ON Patient.Patient_id = payment.patient_id
JOIN room ON Room.room_id = Payment.room_id
JOIN instructions ON instructions.code_id = payment.instruction_id;
```

Results:

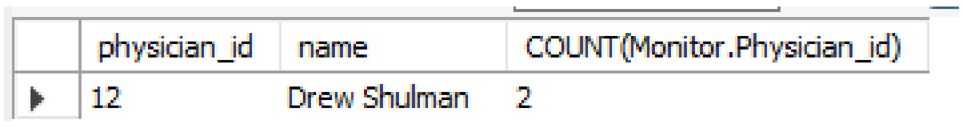


name	TOTAL
Walter White	1110.00
Sherlock Holmes	2108.00
Spongebob Squarepants	998.00
Bruce Wayne	954.00
Barney Stinson	1026.00

Query 11: Returns the physicians that monitor more than 1 patient and how many they monitor

```
SELECT Physician.physician_id, Physician.name , COUNT(Monitor.Physician_id)
FROM Monitor
JOIN Physician ON Monitor.physician_id = Physician.physician_id
GROUP BY Monitor.Physician_id
Having COUNT(Monitor.Physician_id) >1;
```

Results:

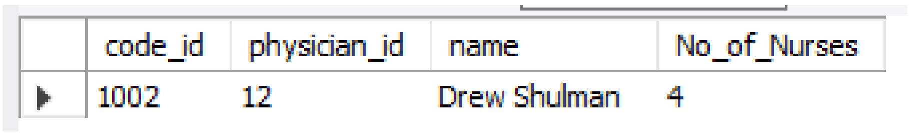


physician_id	name	COUNT(Monitor.Physician_id)
12	Drew Shulman	2

Query 12: Returns the instruction with the most nurses working on it and how many

```
SELECT instructions.code_id, instructions.physician_id, physician.name, COUNT(nurse_id) AS
No_of_Nurses FROM execute_orders
JOIN instructions ON execute_orders.code_id = instructions.code_id
JOIN physician ON physician.physician_id = instructions.physician_id
GROUP BY instructions.code_id, instructions.physician_id, physician.name
HAVING COUNT(execute_orders.nurse_id) = (
SELECT MAX(nurse_count)
FROM (SELECT code_id, COUNT(nurse_id) as nurse_count
FROM execute_orders
GROUP BY code_id
) AS nurse_counts
);
```

Results:

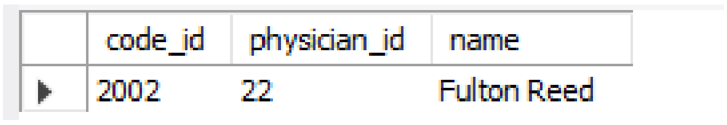


code_id	physician_id	name	No_of_Nurses
1002	12	Drew Shulman	4

Query 13: Returns any instruction that doesn't have any nurses working on it

```
SELECT code_id, Instructions.physician_id, Physician.name
FROM instructions
JOIN Physician ON physician.physician_id = instructions.physician_id
WHERE instructions.code_id NOT IN (SELECT code_id FROM execute_orders);
```

Results:

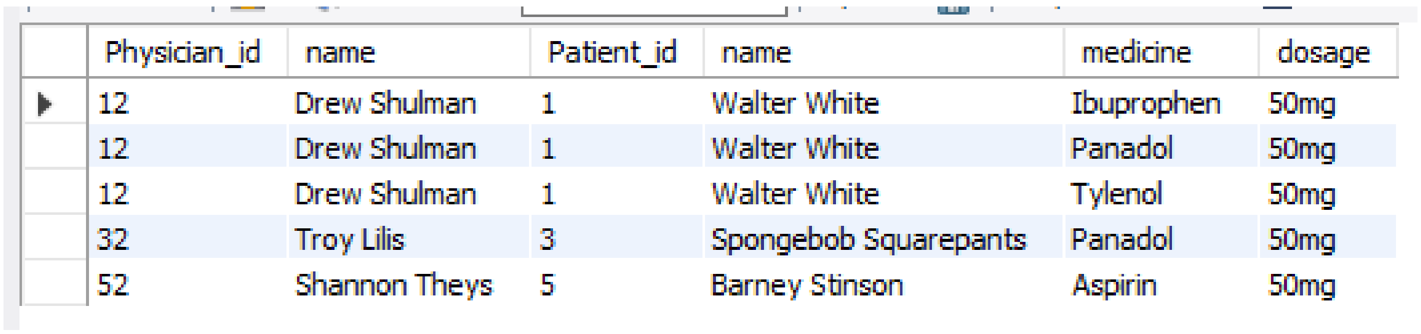


code_id	physician_id	name
2002	22	Fulton Reed

Query 14: Returns what medicine was prescribed to each patient by which doctor

```
SELECT Monitor.Physician_id, Physician.name, Monitor.Patient_id, Patient.name,
Medication.medicine, Medication.dosage
FROM Monitor
JOIN Physician ON Monitor.Physician_id = Physician.Physician_id
JOIN Patient ON Monitor.Patient_id = Patient.Patient_id
JOIN Medication On Medication.Patient_id = Patient.Patient_id ;
```

Results:

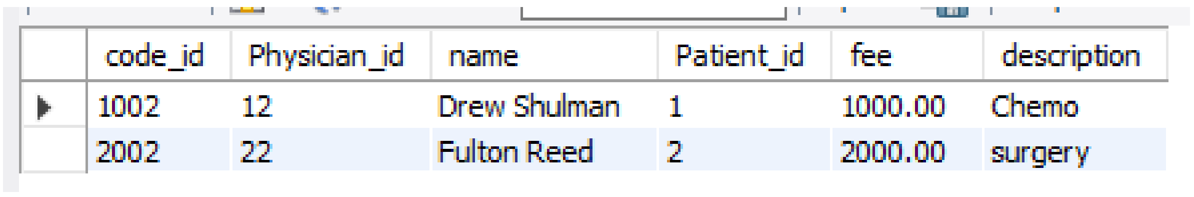


Physician_id	name	Patient_id	name	medicine	dosage
12	Drew Shulman	1	Walter White	Ibuprophen	50mg
12	Drew Shulman	1	Walter White	Panadol	50mg
12	Drew Shulman	1	Walter White	Tylenol	50mg
32	Troy Lilis	3	Spongebob Squarepants	Panadol	50mg
52	Shannon Theys	5	Barney Stinson	Aspirin	50mg

Query 15: Returns any instructions that cost more than 1000 dollars

```
SELECT Instructions.code_id ,Instructions.Physician_id, Physician.name, Patient.Patient_id, fee,
description
FROM Instructions
JOIN Physician On Physician.Physician_id = Instructions.physician_id
JOIN Payment On Instructions.code_id = Payment.instruction_id
JOIN Patient ON Patient.patient_id =  Payment.patient_id
WHERE fee >= 1000
ORDER BY Instructions.code_id  ;
```

Results:



code_id	Physician_id	name	Patient_id	fee	description
1002	12	Drew Shulman	1	1000.00	Chemo
2002	22	Fulton Reed	2	2000.00	surgery

7- Transactions and description

Transaction 1:
This transaction admits a patient into the hospital with an incremented id and all information filled out. Also returns the new patient's id

```
START TRANSACTION;
select @patient_id:= Max(patient_id) + 1 AS 'New Patient ID' from patient;
insert into patient values (@patient_id,"Ahmed","UIC",12345678);
```

This transaction is useful to quickly admit new patients and provide them with their id

Transaction 2:
This transaction sets a patients health record to a criminal to be able to search and find the criminal

```
START TRANSACTION;
UPDATE Health_record
SET Health_record.description = 'Is a criminal now'
WHERE Health_record.patient_id = 1;
```

This can be useful if the hospital is in a high crime area and needs to find criminals in the hospital