# Assumptions

State the set of all used assumptions that are not covered by the given requirements:

Patient total participations: {Hospitalized because every patient has to be hospitalized in a room}, {Make Payment because every patient has to make a payment either it be $0 or more}

Patient Cardinalities: {M:N Provide Medication: Multiple patients can receive multiple medications from multiple nurses}, {1:1 hospitalized: Only one patient can occupy one room at a time, a patient can not have multiple rooms}, {M:N Monitor: Every patient can have multiple physician monitoring them}

Nurse Cardinalities: {1:N Execute: two nurses can not execute the same instruction but one nurse can execute multiple unique instructions}

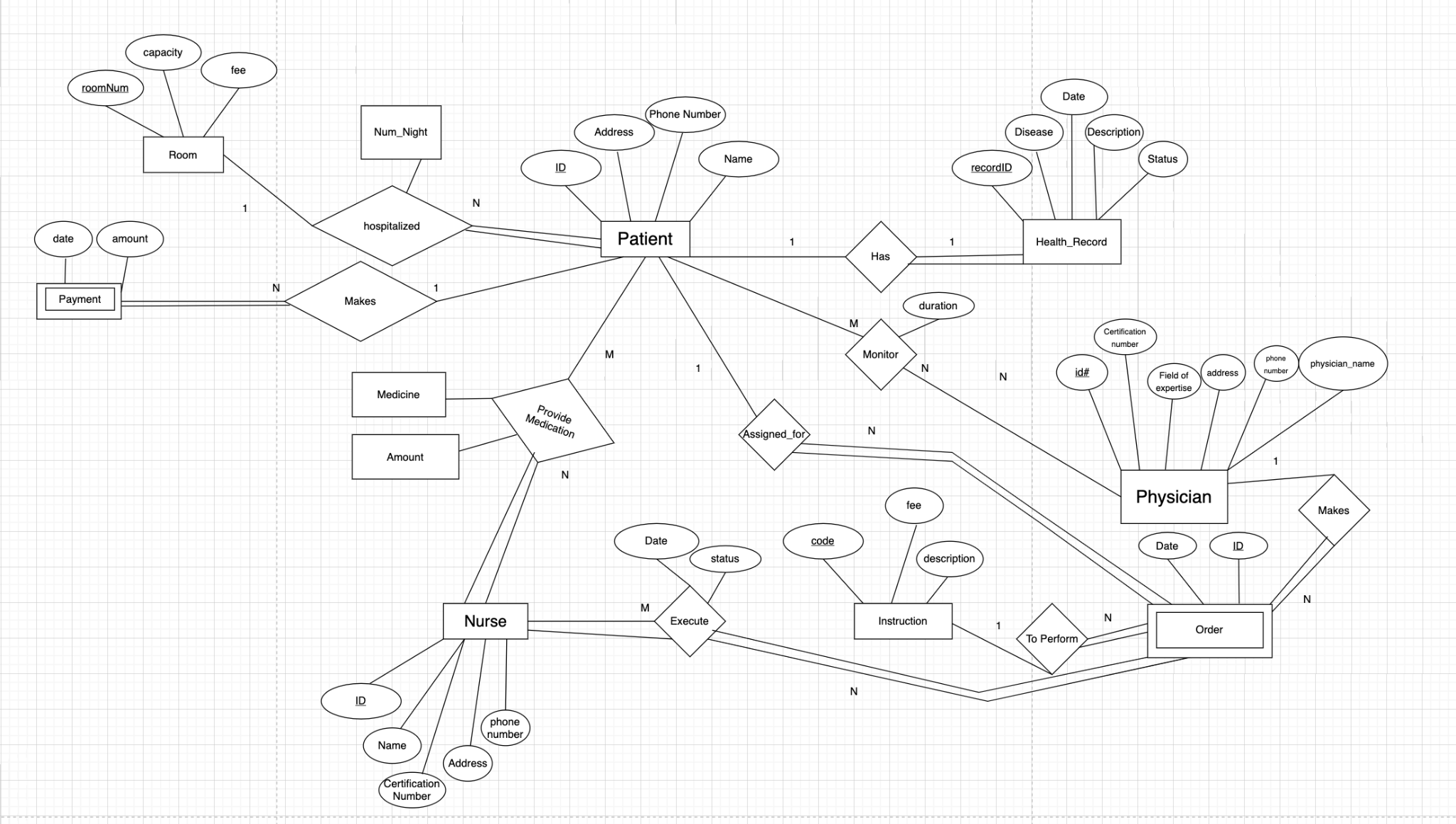
Instructions are predetermined list so multiple patient can be ordered the same instruction and multiple nurses can execute the same instruction

One physician makes multiple orders for one patient.

Order is an entity and has an ID because each order is unique and orders are made to perform one or more of the predestined instructions, each order is assigned for patients and they are to be executed by a number of nurses.

Payments cannot exist without a patient making them.

# (E)ERD: Data Modelling



# Set of Relations

Room (roomNum, capacity, fee)

Pk {roomNum}

Fk {}

Physician (ID, physician\_name, certification\_number, field\_of\_expertise, address, phone\_number )

Pk {ID}

Fk {}

Patient ( ID, patient\_name, phone\_number, address, num\_night, room\_number )

Pk {ID}

Fk {room\_number references room(roomNum)}

Nurse (ID, nurse\_name,certification\_number, address, phone\_number)

Pk {ID}

Fk {}

Payment ( patientID, payment\_date, amount)

Pk {patientID }

Fk {patientID references Patient(ID)}

Instruction (instr\_code, fee, instr\_description)

Pk {code }

Fk {}

Health\_Record (recordID, patientID ,diseases, hr\_date, hr\_description, hr\_status)

Pk {recordID, patientID}

Fk {patientID references patient(ID)}

Monitor (physicianID, patientID, duration)

Pk {}

Fk {physicianID references Physician(ID), patientID references Patient(ID)}

Provide\_medication (patientID, nurseID, medicine\_name, amount)

Pk {}

Fk{patientID references Patient(ID), nurseID references Nurse(ID)}

Order\_Instruction(ID,instruction\_code,physician\_ID, Patient\_ID,instr\_date)

Pk {ID}

Fk {instruction\_code references Instruction(instr\_code), patientID references Patient(ID), physicianID references Physician(ID)}

Executed\_Order (orderID, nurseID, eo\_date, eo\_status)  
Pk {}

Fk {orderID references Order\_Instruction(ID), nurseID references Nurse(ID)}

# Views and descriptions

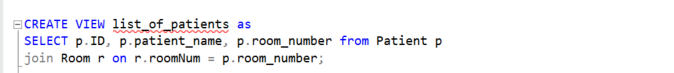
* The description and body of views. Also, include a discussion on why the provided view is useful for your database.

**First View**  
What it does: This view finds the Nurse and the status of an order instruction executed by that nurse.  
Why it’s useful to the database: This view is useful because it helps the hospital track the status of orders and the nurses who carry out those orders as it is essential for ensuring patient safety. A picture containing text, font, white, screenshot

Description automatically generated

**Second View**  
What it does: Finds the total amount of payments made by all patients in the hospital.  
Why it’s useful to the database: This view is useful because it can provide insights to the hospital’s financial health, and show patterns in the payment amounts made by patients.  
A picture containing text, font, screenshot, white

Description automatically generated

**Third View**  
What it does: Retrieves the list of patients currently occupying rooms in the hospital.  
Why it’s useful to the database: This view is useful because it can help the providers manage room availability and patient flow more effectively.  


# Triggers and descriptions

* The description and body of triggers. Also, include a discussion on why the provided trigger is useful for your database. –
* Update\_payment\_amount trigger: Whenever the room table updates, the fee of the room gets added to the amount that is to be paid by the patient in that room in the payment table as patientID is foreign key in the Payment table. This is useful as it will keep track of the total amount of payment that needs to be made by a particular patient.

A screenshot of a computer program

Description automatically generated with low confidence

* Trigger for execute order status: When an order is executed it results in a status. For our Executed\_order table when an entry is made to this table with status as null the status should be assigned as incomplete status on insert into the table as instruction should not be assumed to completed by a nurse by default.

A picture containing text, font, number, screenshot

Description automatically generated

* Trigger for default room capacity: if a new room is entered into the database without a capacity or null capacity, it should have the capacity of at least 1 as it is entered. This will help the hospital to keep track of vacant spaces rather than there being rooms but not being able to assign a patient to it as the capacity is null.

A screenshot of a computer program

Description automatically generated with low confidence

# Queries, descriptions, and results.

* The description, body, and execution result of your SQL queries

**SQL Query 1: (JOIN)**

**Retrieve the names of patients that have diabetesA screenshot of a computer

Description automatically generated with medium confidence**

**SQL Query 2: (JOIN)**

**Retrieve the names of all patients and their physicians.**

**A screenshot of a computer

Description automatically generated**

This query retrieves the patient names and their corresponding physicians by using the relationship of the Monitor table that links both the physicians and patients.

**SQL Query 3: (JOIN)**

**Retrieve the names of all patients and their physicians, AND their assigned nurses.**

A screenshot of a computer

Description automatically generated with medium confidence

This query makes use of the LEFT JOIN keyword, and retrieves the patient names, physician names, and nurse names. This joins the provide\_medication table, that relates the patients to the nurses that provide the medication.

**SQL Query 4: (AGGREGATE)**

**Retrieve the total amount of medications given to each patient.**

**A screenshot of a computer

Description automatically generated with medium confidence**

This query retrieves the sum of the amount of medication prescribed to each patient and uses the provide\_medication table that stores the information regarding the number of prescriptions.

**SQL Query 5: (Nested Query)**

**Retrieve the name and address of the nurse that administered the most medication**

* A screenshot of a computer program

  Description automatically generated with medium confidence
* This Query retrieves the name and address of the nurse that administers the most medication, but it gets the required NurseID from another select query. This is a nested query.

**SQL Query 6: (Nested Query)**

**Retrieve the name and address of the patient that has been administered the most amount of medication**

A screenshot of a computer program

Description automatically generated with medium confidence

This Query retrieves the name and address of the patient that got administered the most medication, but it gets the required patientID from another select query. This is a nested query.

**SQL Query 7: (Nested Query)**

**Retrieve the name, address and phone number of a patient that was prescibed Ibuprofen.**

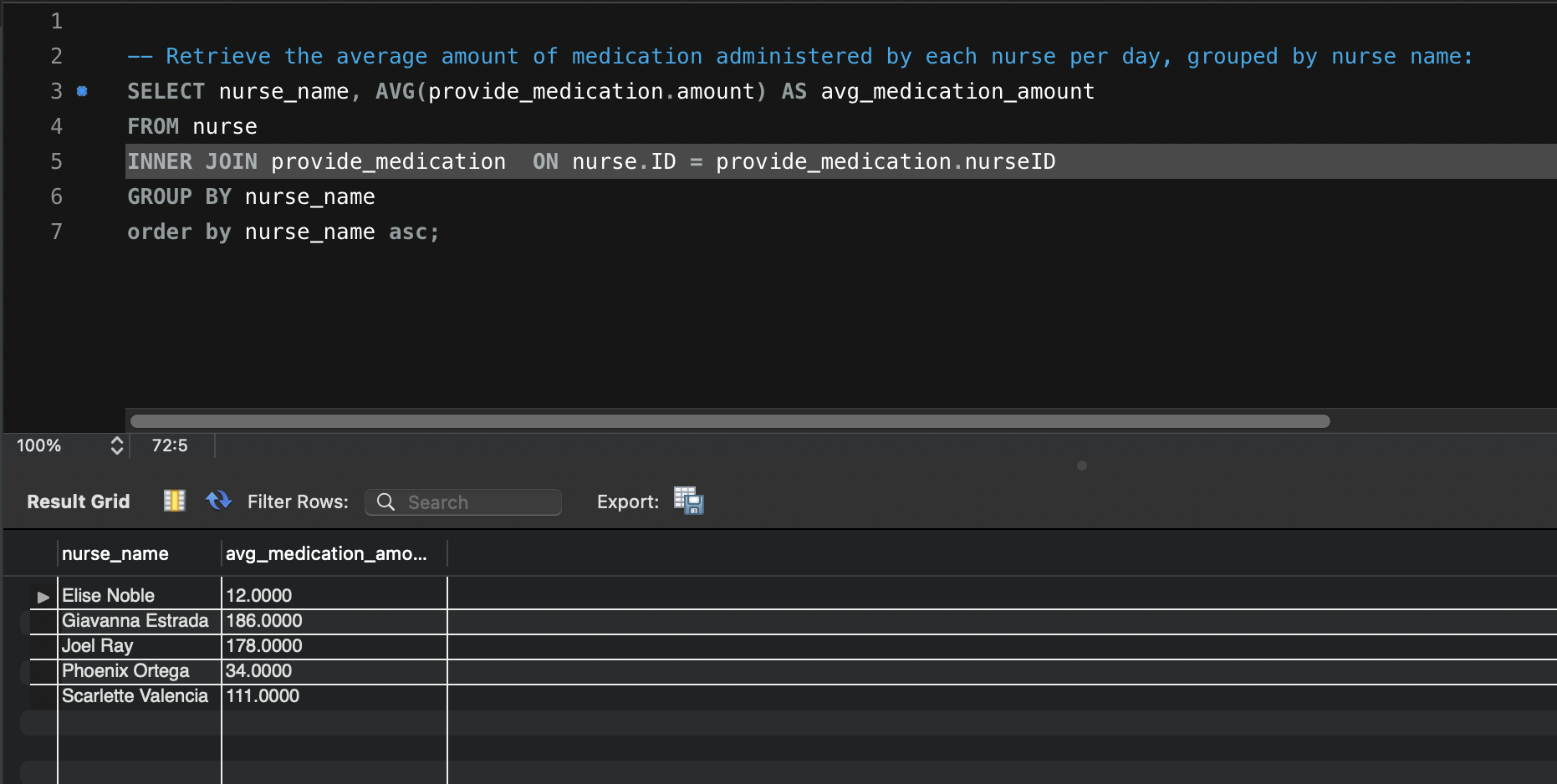
* A screenshot of a computer

  Description automatically generated with medium confidence

This query uses a nested query to avoid using joins on another table. It selects the patient name, address and phone number that was prescribed Ibuprofen.

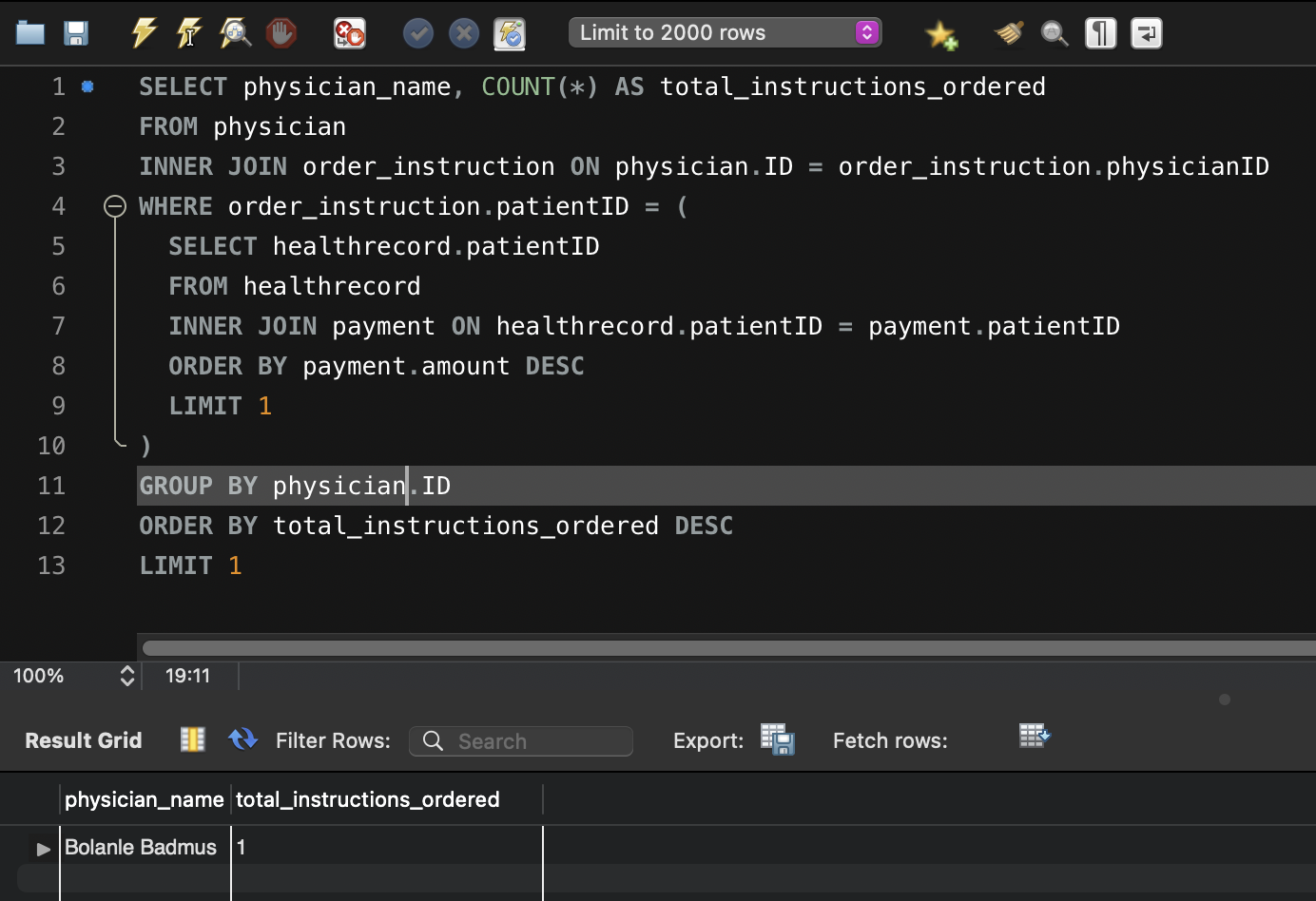
**SQL Query 8: (Aggregate Functions):**

**Retrieve the average amount of medication administered by each nurse per day, grouped by nurse name:**

****

**SQL Query 9: (AGGREGATE)**

**Retrieve the physician who has ordered the most instructions for the patient with the highest total fees charged to the hospital:**

****

This query retrieves the name of the physician that has the highest number of fees charged to the hospital. Aggregate functions were used to find the total number of charges, and a subquery was used to retrieve the ID of the physician that matched the requirement.

**SQL Query 10: (JOIN)**

**Retrieve the name and field of expertise of the physicians who have ordered instructions for patients with Epilepsy:**

# 

This query retrieves the name and physician that have ordered instructions for patients with Epilepsy. There are two join statements used to accomplish this.

**SQL Query 11: (AGGREGATE)**

**Retrieve the name of the patient that has a diagnosis of Asthma and the total amount they paid.**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**SQL Query 12: (NESTED)**

**Query to find the total number of patients who have been treated by a physician with a certification number that is between 20 and 40.**

**A screenshot of a computer

Description automatically generated**

This query is a combination of an aggregate query and nested query. And it finds the total number of patients that were treated by a certain physician with a certification number betweenb 20 and 40.

**SQL Query 13: (NESTED)**

**Retrieve the name, address, and total amount of medication administered to a patient**

**A screenshot of a computer

Description automatically generated**

**SQL Query 14:**

**Retrieve the name and total amount paid by each patient who has made a payment to the hospital. If a patient has not made any payment, display 0 as the amount paid.**

# A screenshot of a computer Description automatically generated

**SQL Query 15:**

**Retrieve the patient with the most medication received who also has a health record that includes a diagnosis of Anxiety.**

**A screenshot of a computer

Description automatically generated**

# Transactions and description

The description and body of transactions.

This transaction is used to enforce the integrity of the data while adding a new patient such as checking if the patient Is already in our database or checking if the room the patient is being assigned is already at full capacity before deciding to commit the transaction or roll it back based on those conditions. This helps the hospital add patients and also help them track what rooms might be full to assign patients

A screenshot of a computer program

Description automatically generated with medium confidence