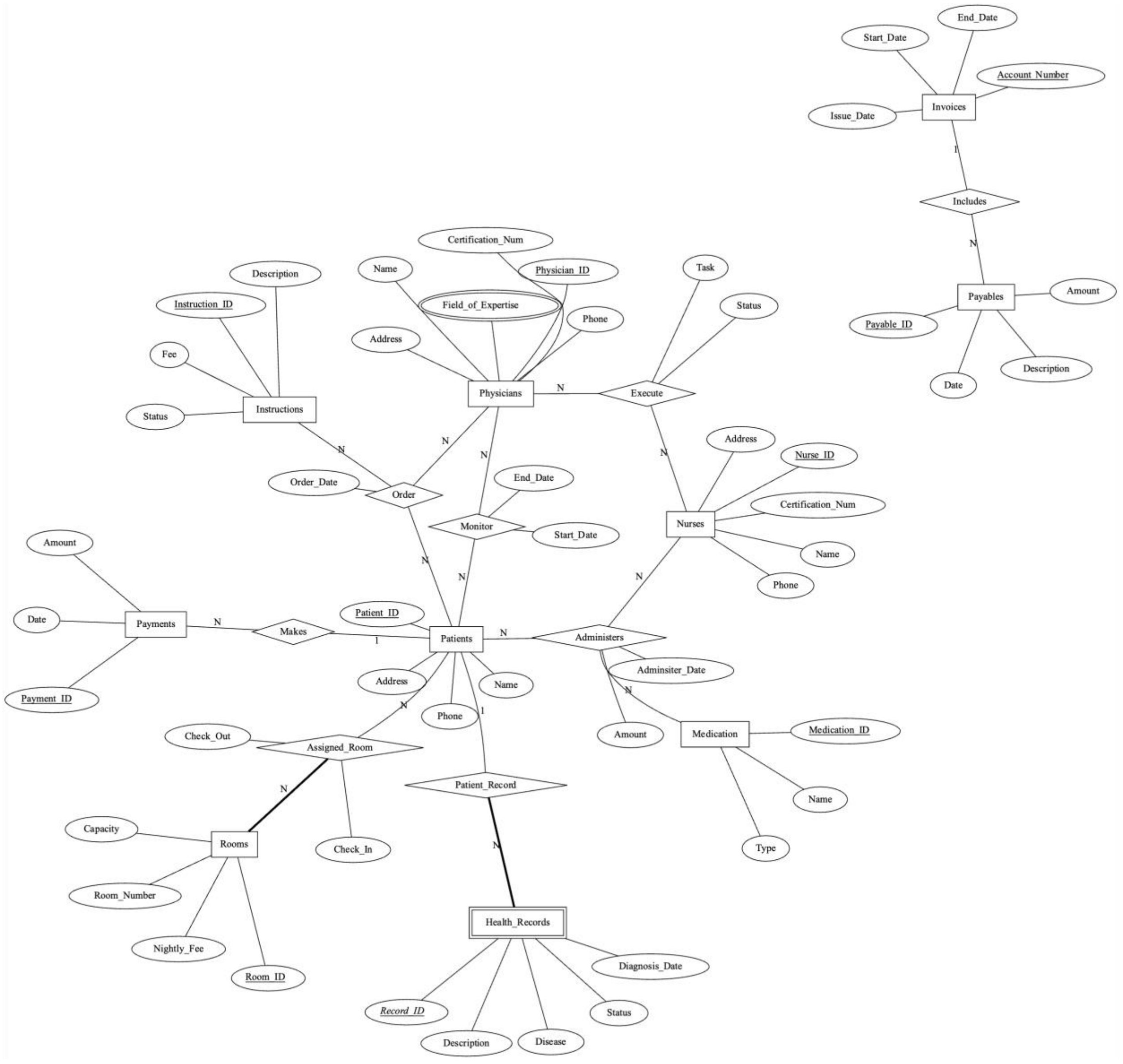
Name: Zaheer Safi

Group Members: Rizwan Mohammad, Sammy Dandu

Professor: Sara Riazi

Final Project Design Document

(E)ERD Diagram:



Assumptions:

Composite Key for Health Records:

- Assumption: Health_Records uses a composite key consisting of Patient_ID and Record_ID.
- Reason: While Record_ID is mentioned as the key, combining it with Patient_ID ensures unique records per patient and avoids potential conflicts.

Multivalued Field of Expertise:

- Assumption: Field_of_Expertise in Physicians can hold multiple values, potentially stored as a list or comma-separated values.
- Reason: Although not specified, this allows a physician to have expertise in multiple areas.

Tracking Multiple Medications and Tasks:

- Assumption: The Administers and Execute tables allow tracking of multiple instances of medication administration and task executions.
- Reason: This facilitates detailed records of each medication administered and each task executed, although the requirement does not specify multiple entries.

Detailed Tracking of Invoices and Payments:

- Assumption: Invoices can have multiple associated payables, and payments are recorded for each patient.
- Reason: Assumes flexibility in handling complex billing and payment scenarios, even though the requirements do not specify multiple payables or payments explicitly.

Non-overlapping Payables:

- Assumption: Each Payable in the Payables table is unique and associated with a specific invoice without overlap in description or amount.
- Reason: Ensures clarity and avoids duplication of payables for the same invoice, which is not explicitly mentioned in the requirements.

Order Dates for Instructions:

- Assumption: Each Order entry has an Order_Date to track when instructions were ordered.
- Reason: Provides chronological context for each order, though the requirement only specifies the need to track instructions.

Handling of Room Assignments:

- Assumption: The Assigned_Rooms table allows multiple assignments of the same room to different patients with precise check-in and check-out dates.
- Reason: Assumes flexibility in room assignments, even though the requirement does not specify handling overlapping room assignments.

Entities And Relationships:

Rooms(Room_ID, Room_Number, Capacity, Nightly_Fee)
primary key: {Room_ID}

Patients(Patient_ID, Name, Address, Phone)
primary key: {Patient_ID}

Health_Records(Record_ID, Patient_ID, Disease, Diagnosis_Date, Status, Description)
primary key: {Record_ID, Patient_ID}
foreign key: {Patient_ID references Patients(Patient_ID)}

Physicians(Physician_ID, Name, Certification_Num, Field_of_Expertise, Address, Phone)
primary key: {Physician_ID}

Nurses(Nurse_ID, Name, Certification_Num, Address, Phone)
primary key: {Nurse_ID}

Medication(Medication_ID, Name, Type)
primary key: {Medication_ID}

Instructions(Instruction_ID, Description, Status, Fee)
primary key: {Instruction_ID}

Invoices(Account_Number, Issue_Date, Start_Date, End_Date)
primary key: {Account_Number}

Payables(Payable_ID, Invoice_Account_Number, Amount, Date, Description)
primary key: {Payable_ID}
foreign key: {Invoice_Account_Number references Invoices(Account_Number)}

Payments(Payment_ID, Patient_ID, Amount, Date)
primary key: {Payment_ID}
foreign key: {Patient_ID references Patients(Patient_ID)}

Assigned_Rooms(Patient_ID, Room_ID, Check_In, Check_Out)
primary key: {Patient_ID, Room_ID, Check_In}
foreign key: {Patient_ID references Patients(Patient_ID), Room_ID references Rooms(Room_ID)}

Monitor(Physician_ID, Patient_ID, Start_Date, End_Date)

primary key: {Physician_ID, Patient_ID}
foreign key: {Physician_ID references Physicians(Physician_ID), Patient_ID references Patients(Patient_ID)}

Execute(Physician_ID, Nurse_ID, Task, Status)
primary key: {Physician_ID, Nurse_ID, Task}
foreign key: {Physician_ID references Physicians(Physician_ID), Nurse_ID references Nurses(Nurse_ID)}

Order(Physician_ID, Patient_ID, Instruction_ID, Order_Date)
primary key: {Physician_ID, Patient_ID, Instruction_ID}
foreign key: {Physician_ID references Physicians(Physician_ID), Patient_ID references Patients(Patient_ID), Instruction_ID references Instructions(Instruction_ID)}

Administers(Patient_ID, Nurse_ID, Medication_ID, Administer_Date, Amount)
primary key: {Patient_ID, Nurse_ID, Medication_ID, Administer_Date}
foreign key: {Patient_ID references Patients(Patient_ID), Nurse_ID references Nurses(Nurse_ID), Medication_ID references Medication(Medication_ID)}

Relationships and Their Cardinality:

[Relationship]
Patient_Record
--
Patients (1)
Health_Records (N) @total_participation

[Relationship]
Assigned_Room
--
Rooms (N) @total_participation
Patients (N)
--
Check_In [date]
Check_Out [date]

[Relationship]
Monitor
--
Physicians (N)
Patients (N)
--
Start_Date [date]
End_Date [date]

[Relationship]
Execute
--
Physicians (N)
Nurses (N)
--
Task
Status

[Relationship]
Order
--
Physicians (N)
Patients (N)
Instructions (N)
--
Order_Date

[Relationship]
Administers
--
Patients (N)
Nurses(N)
Medication(N)
--
Adminsiter_Date
Amount

[Relationship]
Includes
--
Invoices (1)
Payables (N)

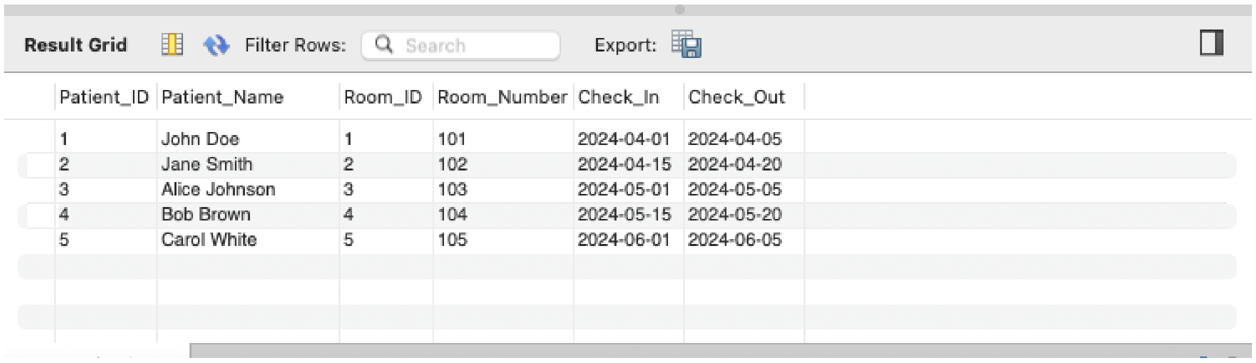
[Relationship]
Makes
--
Patients (1)
Payments (N)

Queries:

Query1: Retrieve all patients along with their assigned room information.

```
SELECT
  Patients.Patient_ID,
  Patients.Name AS Patient_Name,
  Rooms.Room_ID,
  Rooms.Room_Number,
  Assigned_Rooms.Check_In,
  Assigned_Rooms.Check_Out
FROM
  Patients
JOIN
  Assigned_Rooms ON Patients.Patient_ID = Assigned_Rooms.Patient_ID
JOIN
  Rooms ON Assigned_Rooms.Room_ID = Rooms.Room_ID;
```

Screenshot:

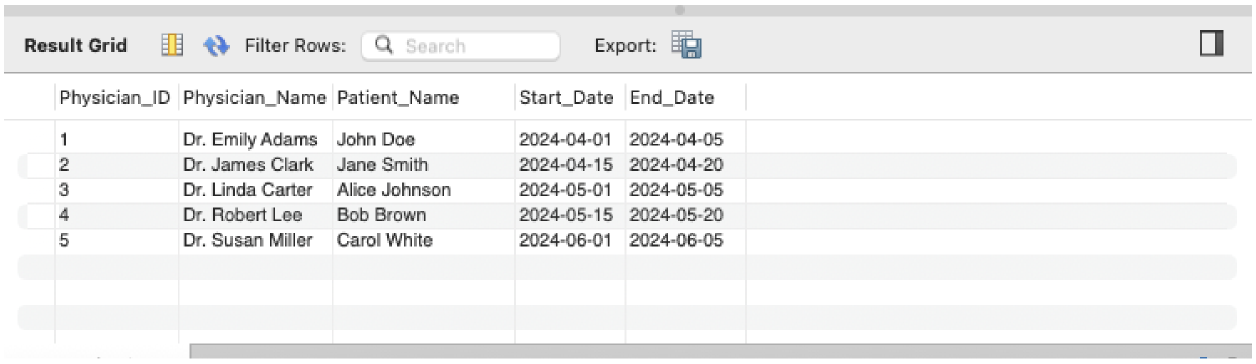


Patient_ID	Patient_Name	Room_ID	Room_Number	Check_In	Check_Out
1	John Doe	1	101	2024-04-01	2024-04-05
2	Jane Smith	2	102	2024-04-15	2024-04-20
3	Alice Johnson	3	103	2024-05-01	2024-05-05
4	Bob Brown	4	104	2024-05-15	2024-05-20
5	Carol White	5	105	2024-06-01	2024-06-05

Query2: List physicians and the patients they monitor along with the start and end dates of monitoring.

```
SELECT

  Physicians.Physician_ID,

  Physicians.Name AS Physician_Name,

  Patients.Name AS Patient_Name,

  Monitor.Start_Date,

  Monitor.End_Date

FROM

  Physicians

JOIN

  Monitor ON Physicians.Physician_ID = Monitor.Physician_ID

JOIN

  Patients ON Monitor.Patient_ID = Patients.Patient_ID;
```

Screenshot:

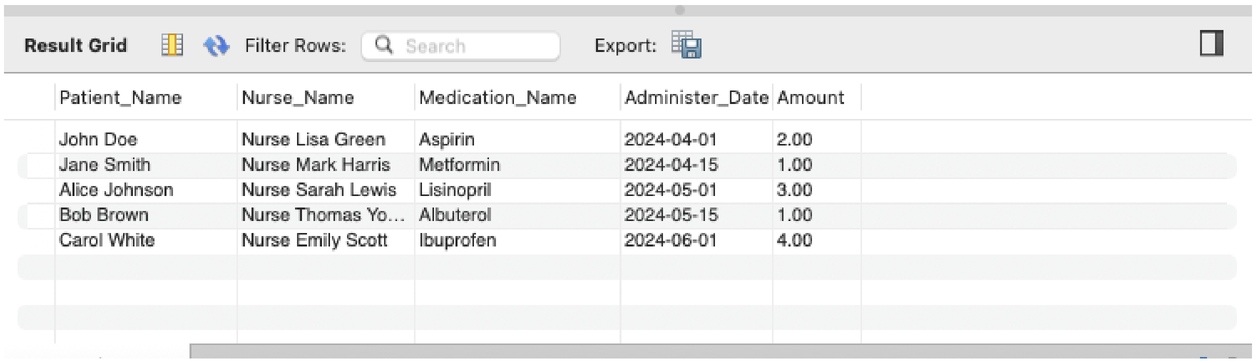


Physician_ID	Physician_Name	Patient_Name	Start_Date	End_Date
1	Dr. Emily Adams	John Doe	2024-04-01	2024-04-05
2	Dr. James Clark	Jane Smith	2024-04-15	2024-04-20
3	Dr. Linda Carter	Alice Johnson	2024-05-01	2024-05-05
4	Dr. Robert Lee	Bob Brown	2024-05-15	2024-05-20
5	Dr. Susan Miller	Carol White	2024-06-01	2024-06-05

Query3: Retrieve medication administrations along with patient and nurse details.

```
SELECT

    Patients.Name AS Patient_Name,

    Nurses.Name AS Nurse_Name,

    Medication.Name AS Medication_Name,

    Administers.Administer_Date,

    Administers.Amount

FROM

    Administers

JOIN

    Patients ON Administers.Patient_ID = Patients.Patient_ID

JOIN

    Nurses ON Administers.Nurse_ID = Nurses.Nurse_ID

JOIN

    Medication ON Administers.Medication_ID = Medication.Medication_ID;
```

Screenshot:



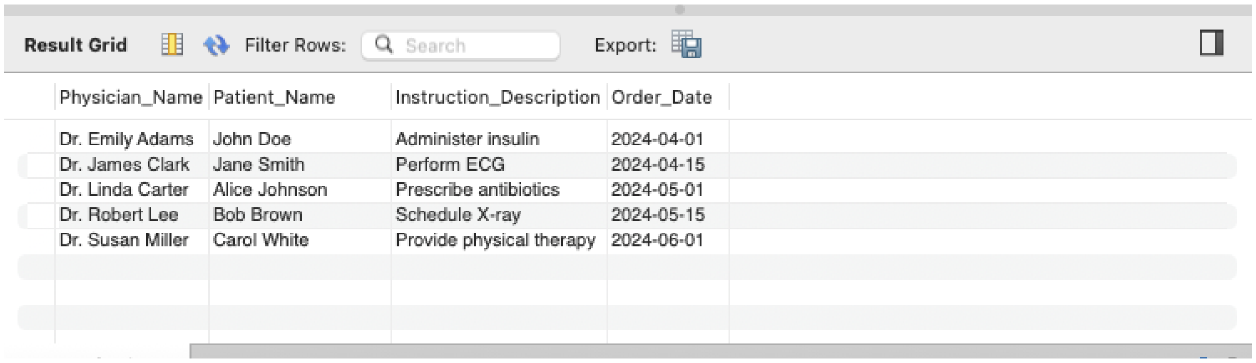
Patient_Name	Nurse_Name	Medication_Name	Administer_Date	Amount
John Doe	Nurse Lisa Green	Aspirin	2024-04-01	2.00
Jane Smith	Nurse Mark Harris	Metformin	2024-04-15	1.00
Alice Johnson	Nurse Sarah Lewis	Lisinopril	2024-05-01	3.00
Bob Brown	Nurse Thomas Yo...	Albuterol	2024-05-15	1.00
Carol White	Nurse Emily Scott	Ibuprofen	2024-06-01	4.00

Query 4: List the physicians and the instructions they have ordered for patients, including the order date.

```
SELECT
```

```
  Physicians.Name AS Physician_Name,
  Patients.Name AS Patient_Name,
  Instructions.Description AS Instruction_Description,
  `Order`.Order_Date
FROM
  `Order`
JOIN
  Physicians ON `Order`.Physician_ID = Physicians.Physician_ID
JOIN
  Patients ON `Order`.Patient_ID = Patients.Patient_ID
JOIN
  Instructions ON `Order`.Instruction_ID = Instructions.Instruction_ID;
```

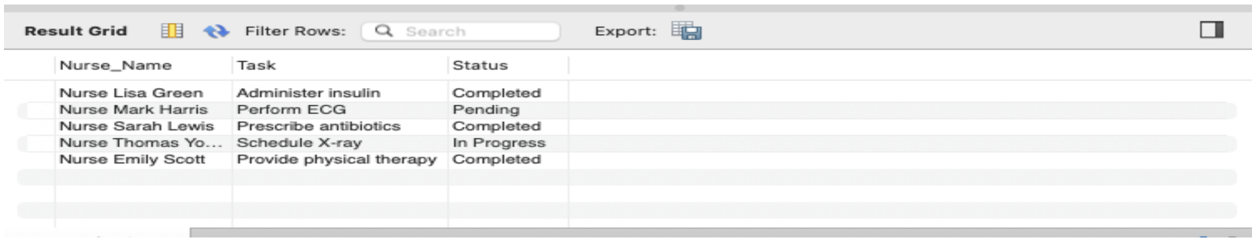
Screenshot:



Physician_Name	Patient_Name	Instruction_Description	Order_Date
Dr. Emily Adams	John Doe	Administer insulin	2024-04-01
Dr. James Clark	Jane Smith	Perform ECG	2024-04-15
Dr. Linda Carter	Alice Johnson	Prescribe antibiotics	2024-05-01
Dr. Robert Lee	Bob Brown	Schedule X-ray	2024-05-15
Dr. Susan Miller	Carol White	Provide physical therapy	2024-06-01

Query 5: Get the list of nurses and the tasks they have executed along with their status.

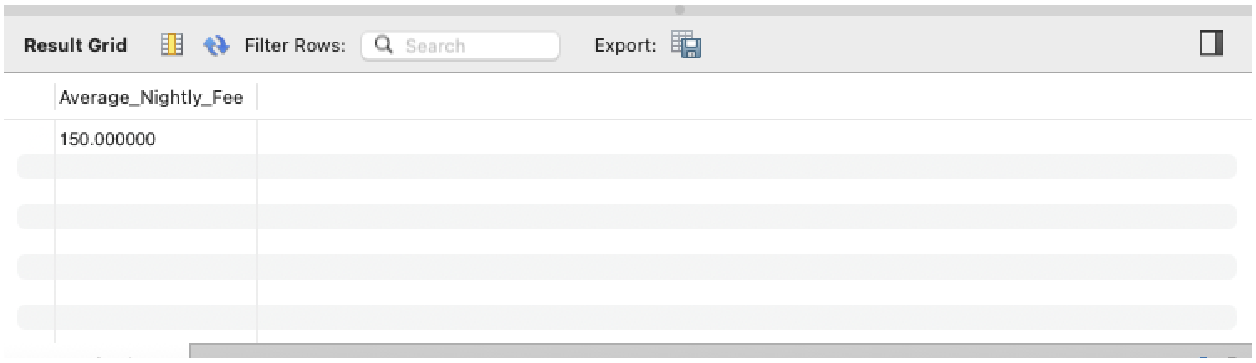
```
SELECT
  Nurses.Name AS Nurse_Name,
  Execute.Task,
  Execute.Status
FROM
  Execute
JOIN
  Nurses ON Execute.Nurse_ID = Nurses.Nurse_ID;
```



Nurse_Name	Task	Status
Nurse Lisa Green	Administer insulin	Completed
Nurse Mark Harris	Perform ECG	Pending
Nurse Sarah Lewis	Prescribe antibiotics	Completed
Nurse Thomas Yo...	Schedule X-ray	In Progress
Nurse Emily Scott	Provide physical therapy	Completed

Query 6: Retrieve the average nightly fee for rooms that are currently assigned to patients.

```
SELECT
  AVG(Rooms.Nightly_Fee) AS Average_Nightly_Fee
FROM
  Rooms
JOIN
  Assigned_Rooms ON Rooms.Room_ID = Assigned_Rooms.Room_ID
```



Average_Nightly_Fee
150.000000

Query 7: Calculate the total amount of payables for each invoice.

```
SELECT
  Invoices.Account_Number,
  SUM(Payables.Amount) AS Total_Payable
```

FROM

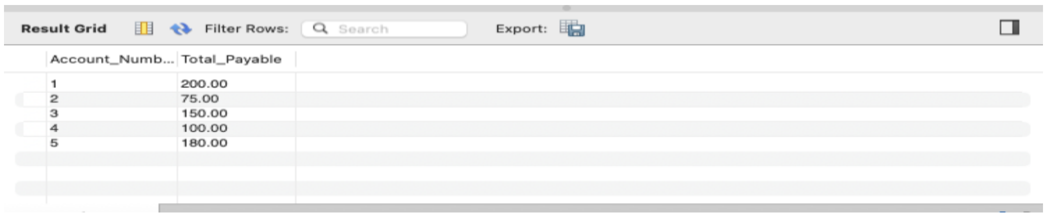
  Invoices

JOIN

  Payables ON Invoices.Account_Number = Payables.Invoice_Account_Number

GROUP BY

  Invoices.Account_Number;



```
Account_Numb... Total_Payable
1               200.00
2               75.00
3               150.00
4               100.00
5               180.00
```

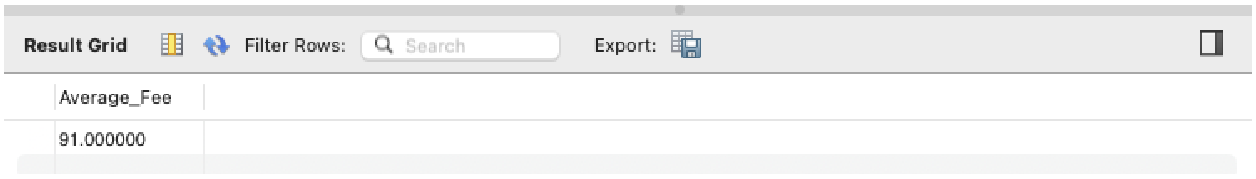
Query 8: Find the average fee of instructions.

SELECT

  AVG(Instructions.Fee) AS Average_Fee

FROM

  Instructions;



```
Average_Fee
91.000000
```

Query 9: Count the number of medications administered to each patient.

SELECT

  Patients.Name AS Patient_Name,

  COUNT(Administers.Medication_ID) AS Number_of_Medications

FROM

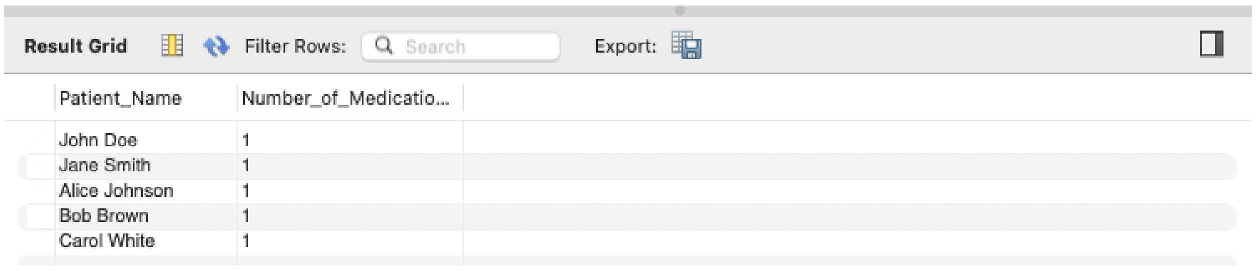
  Patients

JOIN

  Administers ON Patients.Patient_ID = Administers.Patient_ID

GROUP BY

  Patients.Name;



Result Grid | Filter Rows: | Export:

Patient_Name	Number_of_Medicatio...
John Doe	1
Jane Smith	1
Alice Johnson	1
Bob Brown	1
Carol White	1

Query10: Sum the total payments made by each patient.

SELECT

  Patients.Name AS Patient_Name,

  SUM(Payments.Amount) AS Total_Paid

FROM

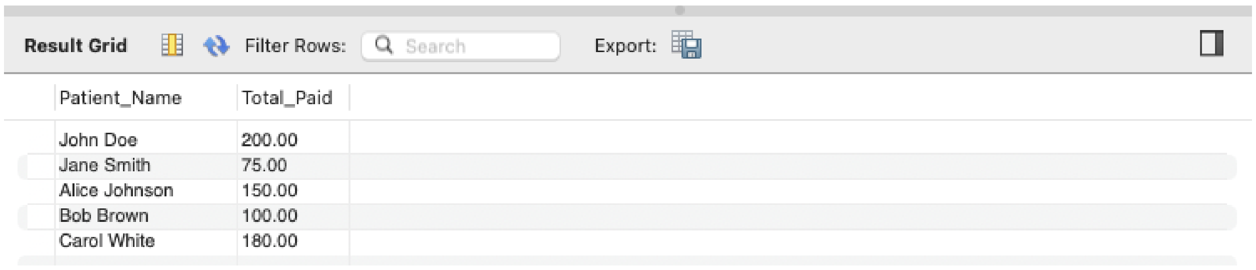
  Patients

JOIN

  Payments ON Patients.Patient_ID = Payments.Patient_ID

GROUP BY

  Patients.Name;



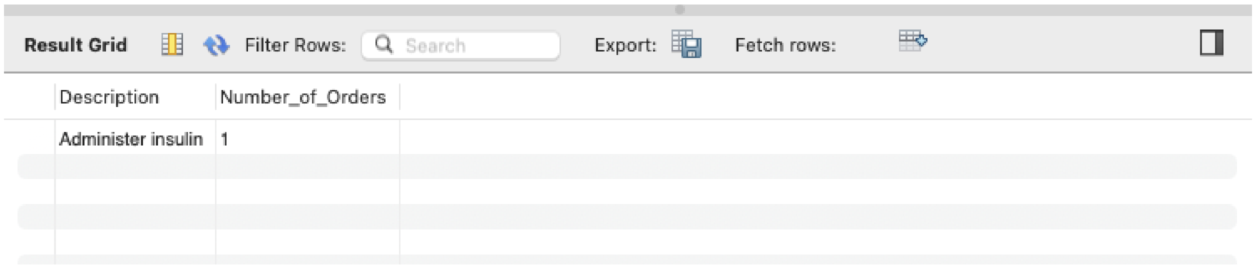
Result Grid | Filter Rows: | Export:

Patient_Name	Total_Paid
John Doe	200.00
Jane Smith	75.00
Alice Johnson	150.00
Bob Brown	100.00
Carol White	180.00

Query 11: Retrieve the details of instructions that have been ordered the most.

SELECT

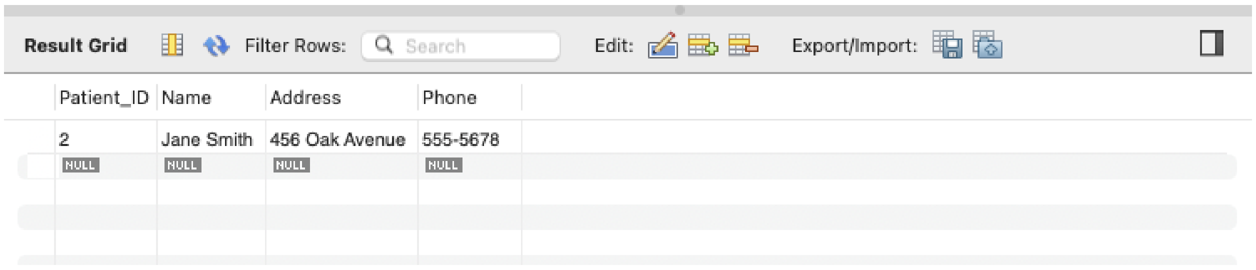
```
  Instructions.Description,
  COUNT(`Order`.Instruction_ID) AS Number_of_Orders
FROM
  Instructions
JOIN
  `Order` ON Instructions.Instruction_ID = `Order`.Instruction_ID
GROUP BY
  Instructions.Description
ORDER BY
  Number_of_Orders DESC
LIMIT 1;
```



Description	Number_of_Orders
Administer insulin	1

Query 12: Retrieve the details of all patients who have been prescribed an instruction by a physician whose field of expertise is 'Cardiology'

```
SELECT *
FROM Patients
WHERE Patient_ID IN (
  SELECT `Order`.Patient_ID
  FROM `Order`
  JOIN Physicians ON Physicians.Physician_ID = `Order`.Physician_ID
  WHERE Field_of_Expertise = 'Cardiology'
);
```

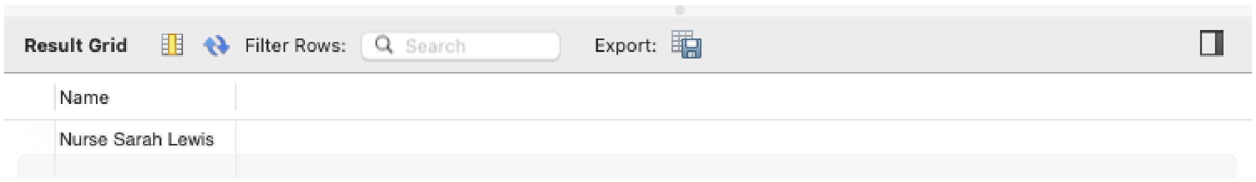

Result Grid Filter Rows: Edit: Export/Import:

Patient_ID	Name	Address	Phone
2	Jane Smith	456 Oak Avenue	555-5678
NULL	NULL	NULL	NULL

Query 13: Find the names of nurses who have administered medication to

patients diagnosed with 'Hypertension'

```
SELECT Nurses.Name
FROM Nurses
WHERE Nurse_ID IN (
  SELECT Administers.Nurse_ID
  FROM Administers
  JOIN Patients ON Administers.Patient_ID = Patients.Patient_ID
  JOIN Health_Records ON Patients.Patient_ID = Health_Records.Patient_ID
  WHERE Health_Records.Disease = 'Hypertension'
);
```


Result Grid Filter Rows: Export:

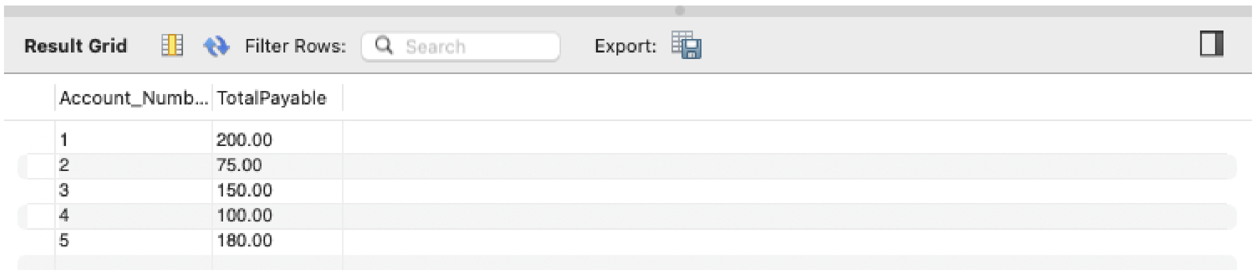
Name
Nurse Sarah Lewis

Query 14: Find the total amount payable for each invoice

```
SELECT Invoices.Account_Number, SUM(Payables.Amount) AS TotalPayable
FROM Invoices
JOIN Payables ON Invoices.Account_Number = Payables.Invoice_Account_Number
WHERE Invoices.Account_Number IN (
  SELECT Payables.Invoice_Account_Number
  FROM Payables
```

)

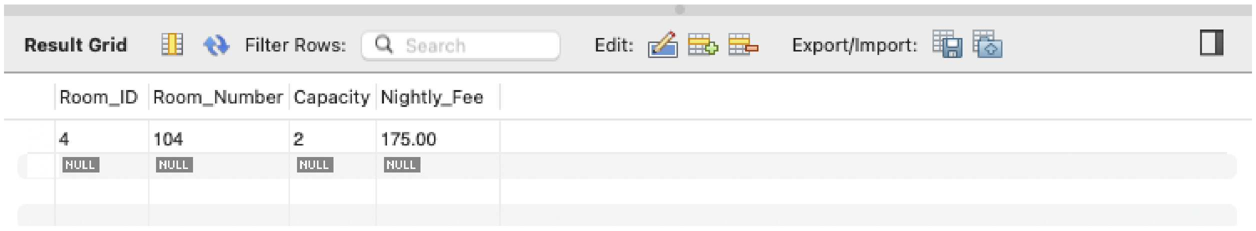
GROUP BY Invoices.Account_Number;



```
Account_Numb...  TotalPayable
1                200.00
2                75.00
3                150.00
4                100.00
5                180.00
```

Query 15: List the room details for rooms that are currently assigned to patients with the disease 'Asthma'

```
SELECT *

FROM Rooms

WHERE Room_ID IN (

  SELECT Assigned_Rooms.Room_ID

  FROM Assigned_Rooms

  JOIN Patients ON Assigned_Rooms.Patient_ID = Patients.Patient_ID

  JOIN Health_Records ON Patients.Patient_ID = Health_Records.Patient_ID

  WHERE Health_Records.Disease = 'Asthma'

);
```



```
Room_ID  Room_Number  Capacity  Nightly_Fee
4        104          2         175.00
NULL     NULL         NULL      NULL
```

Views:

View: PatientDetails

This view combines patient information with their health records.

```
CREATE VIEW PatientDetails AS
SELECT
   p.Patient_ID,
   p.Name AS Patient_Name,
   p.Address AS Patient_Address,
   p.Phone AS Patient_Phone,
   hr.Record_ID,
   hr.Disease,
   hr.Diagnosis_Date,
   hr.Status AS Health_Status,
   hr.Description AS Health_Description
FROM
   Patients p
JOIN
   Health_Records hr ON p.Patient_ID = hr.Patient_ID;
```

Why it's useful: This view simplifies the retrieval of comprehensive patient information along with their health records, making it easier for healthcare providers to get a complete overview of a patient's medical history without writing complex joins every time.

View: RoomOccupancy

This view provides information about room occupancy, including the patient assigned to each room and the check-in and check-out dates.

```
CREATE VIEW RoomOccupancy AS
SELECT
   r.Room_ID,
   r.Room_Number,
   r.Capacity,
   r.Nightly_Fee,
   ar.Patient_ID,
   p.Name AS Patient_Name,
   ar.Check_In,
   ar.Check_Out
FROM
   Rooms r
JOIN
```

```
    Assigned_Rooms ar ON r.Room_ID = ar.Room_ID
JOIN
    Patients p ON ar.Patient_ID = p.Patient_ID;
```

Why it's useful: This view helps in monitoring the occupancy status of each room, making it easier for administrative staff to manage room assignments and ensure efficient use of hospital resources.

View: PhysicianPatientMonitor

This view provides a list of patients monitored by each physician.

```
CREATE VIEW PhysicianPatientMonitor AS
SELECT
    ph.Physician_ID,
    ph.Name AS Physician_Name,
    ph.Field_of_Expertise,
    m.Patient_ID,
    p.Name AS Patient_Name,
    m.Start_Date,
    m.End_Date
FROM
    Physicians ph
JOIN
    Monitor m ON ph.Physician_ID = m.Physician_ID
JOIN
    Patients p ON m.Patient_ID = p.Patient_ID;
```

Why it's useful: This view simplifies the task of tracking which patients are under the care of specific physicians, providing quick access to monitoring data for both physicians and hospital administration.

View: NurseMedicationAdministration

This view provides details of medications administered by nurses to patients.

```
CREATE VIEW NurseMedicationAdministration AS
SELECT
   n.Nurse_ID,
   n.Name AS Nurse_Name,
   a.Patient_ID,
   p.Name AS Patient_Name,
   a.Medication_ID,
   m.Name AS Medication_Name,
   a.Administer_Date,
   a.Amount
FROM
   Nurses n
JOIN
   Administers a ON n.Nurse_ID = a.Nurse_ID
JOIN
   Patients p ON a.Patient_ID = p.Patient_ID
JOIN
   Medication m ON a.Medication_ID = m.Medication_ID;
```

Why it's useful: This view aids in the oversight of medication administration, making it easier to track which medications have been given to which patients by which nurses. It helps ensure accountability and proper medication management.

View: InvoiceDetails

This view consolidates invoice and payable details for easy financial tracking.

```
CREATE VIEW InvoiceDetails AS
SELECT
   i.Account_Number,
   i.Issue_Date,
   i.Start_Date,
   i.End_Date,
   p.Payable_ID,
   p.Amount AS Payable_Amount,
   p.Date AS Payable_Date,
   p.Description AS Payable_Description
```

```
FROM
   Invoices i
JOIN
   Payables p ON i.Account_Number = p.Invoice_Account_Number;
```

Why it's useful: This view provides a comprehensive look at invoice and payable information, which is useful for the financial department to manage billing and payments efficiently.

Triggers:

Trigger: UpdateRoomCapacity

This trigger updates the room capacity when a patient checks in or checks out.

```
CREATE TRIGGER UpdateRoomCapacity
AFTER INSERT ON Assigned_Rooms
FOR EACH ROW
BEGIN
   UPDATE Rooms
   SET Capacity = Capacity - 1
   WHERE Room_ID = NEW.Room_ID;
END;
```

Why it's useful: These triggers automatically update the room capacity when a patient checks in or out, ensuring that the capacity data is always accurate without manual intervention.

Trigger: UpdateInvoiceDate

This trigger updates the Issue_Date of an invoice when the Start_Date is modified.

```
CREATE TRIGGER UpdateInvoiceDate

BEFORE UPDATE ON Invoices

FOR EACH ROW

BEGIN

   IF NEW.Start_Date != OLD.Start_Date THEN
```

```
    SET NEW.Issue_Date = CURDATE();
  END IF;
END;
```

Why it's useful: This trigger ensures that the Issue_Date is updated to the current date whenever the Start_Date of an invoice is changed. This helps in maintaining accurate and current records of when invoices are issued.

Transactions:

Transaction: AssignRoomToPatient

This transaction assigns a room to a patient and updates the room capacity.

```
START TRANSACTION;

-- Step 1: Assign the room to the patient
INSERT INTO Assigned_Rooms (Patient_ID, Room_ID, Check_In, Check_Out)
VALUES (1, 101, '2024-07-25', NULL);

-- Step 2: Update the room capacity
UPDATE Rooms
SET Capacity = Capacity - 1
WHERE Room_ID = 101;

COMMIT;
```

Why it's useful: This transaction ensures that both the assignment of a room to a patient and the updating of the room capacity are completed successfully. If either operation fails, the entire transaction is rolled back, maintaining data integrity.

Transaction: ProcessPatientPayment

This transaction records a payment from a patient and updates the patient's invoice.

```
START TRANSACTION;


-- Step 1: Insert the payment record

INSERT INTO Payments (Payment_ID, Patient_ID, Amount, Date)

VALUES (6, 6, 200.00, '2024-07-25');


-- Step 2: Update the invoice payable amount

UPDATE Payables

SET Amount = Amount - 200.00

WHERE Invoice_Account_Number = (

    SELECT Account_Number

    FROM Invoices

    WHERE Patient_ID = 1

);


COMMIT;
```

Why it's useful: This transaction ensures that the payment from a patient is recorded correctly and that the corresponding invoice amount is updated. It maintains the consistency of financial records by ensuring that both steps succeed or fail together.

Transaction: OrderMedicationForPatient

This transaction orders medication for a patient and logs the administration of the medication.

```
START TRANSACTION;


-- Step 1: Insert the order
```

```
INSERT INTO `Order` (Physician_ID, Patient_ID, Instruction_ID, Order_Date)
VALUES (6, 6, 101, '2024-07-25');

-- Step 2: Log the medication administration
INSERT INTO Administers (Patient_ID, Nurse_ID, Medication_ID, Administer_Date, Amount)
VALUES (6, 7, 201, '2024-07-25', 2.5);

COMMIT;
```

Why it's useful: This transaction ensures that the ordering of medication for a patient and the logging of its administration are performed together. It maintains data consistency by ensuring both operations are completed successfully or rolled back if any part fails.