MIST4610 Group Project I

Daniel Palomino, Shaan Patel, Zack Kendrick, Jack Burke, John Bell

You:

Pretend you are the owner/operator of a emergency healthcare clinic needing to build a relational database. You hired some students from the MIST 4610 class at the University of Georgia to create the database for you. They need to know more about your organization to identify which entities, attributes, and relationships are important for you. Start by describing your business as a real client

ChatGPT:

Absolutely, I can help craft a comprehensive overview of the emergency healthcare clinic, which will be instrumental for the MIST 4610 class at the University of Georgia in developing a relational database tailored to our needs.

Business Overview

Name: QuickCare Emergency Clinic

Mission: To provide immediate, high-quality medical care for urgent and emergency health situations, with a strong focus on patient satisfaction and efficient healthcare delivery.

Services: Our clinic offers a wide range of emergency healthcare services, including but not limited to:

- Immediate care for injuries and illnesses
- Diagnostic services (e.g., X-rays, blood tests, EKGs)
- Minor surgical procedures
- Vaccinations and preventive care
- Referrals to specialists when necessary

Operating Hours: 24/7, 365 days a year, to ensure healthcare is available whenever it's needed.

Data Dictionary

Table: Patients

Column Name	Description	Data Type	Size	Format	Key?
patientID	Unique number that Identifies the patient with the database.	INT	20		PK
firstName	First name of the patient	VARCHAR	45		
lastName	Last name of the patient	VARCHAR	45		
dateOfBirth	The date of birth of the patient	DATE	8	YYYY-MM-DD	
Sex	The gender of the patient	VARCHAR	10		
Address	The address of the patient	VARCHAR	100		
phoneNumber	The contact information of the patient	VARCHAR	45	999-999-9999	
Medical_History	The patient's existing medical history	VARCHAR	255		

Table: Visits

Column Name	Description	Data Type	Size	Format	Key?
visitID	Unique number identifying the visit with the database	INT	20		PK
idPatients	ID of the patient associated with the visit	INT	20		FK
idInvoices	ID of the invoice associated with the visit	INT	20		FK
date	Date of the visit	DATETIME	14	YYYY-MM-DD HH:MI:SS	
reasonForVisit	A description for the visit	VARCHAR	255		

Table: Treatments

Column Name	Description	Data Type	Size	Format	Key?
treatmentID	Unique ID linking each treatment to the database	INT	20		PK
diagnosisID	ID of the diagnosis to be treated	INT	20		FK
treatmentDescription	Description of the provided treatment	VARCHAR	255		
treatmentName	The name of the treatment	VARCHAR	255		
follow-ups	The date of follow-up appointments	DATETIME	14	YYYY-MM-DD HH:MI:SS	

Table: Rooms

Column Name	Description	Data Type	Size	Format	Key?
roomNum	Unique number for each room	INT			PK
roomCapacity	Number of occupants allowed	INT			
cleaningStatus	Status indicating whether the room is clean or not	VARCHAR	45		

Table: Nurses

Column Name	Description	Data Type	Size	Format	Key?
nurseID	Unique number that Identifies the nurse with the database.	INT	20		PK
firstName	First name of the nurse	VARCHAR	45		
lastName	Last name of the nurse	VARCHAR	45		
phoneNumber	The phone number of the nurse	VARCHAR	45	999-999-9999	

certifications	Certifications held by the nurse	VARCHAR	45		
salary	The address of the patient	VARCHAR	45		

Table: Janitors

Column Name	Description	Data Type	Size	Format	Key?
janitorID	Unique number linking the janitor to the database	INT			PK
firstName	First name of the janitor	VARCHAR	45		
lastName	Last name of the janitor	VARCHAR	45		
phoneNumber	The phone number associated with a janitor	VARCHAR	45	999-999-9999	
salary	The salary of each janitor	VARCHAR	45		

Table: Invoices

Column Name	Description	Data Type	Size	Format	Key?
invoiceID	Unique number linking the invoice to the database	INT			PK
dateIssued	The issue date of the invoice	DATETIME	14	YYYY-MM-DD HH:MI:SS	
totalCharges	Last name of the janitor	VARCHAR	45		
itemsUsed	The phone number associated with a janitor	VARCHAR	255		
paymentStatus	Status of the payment	tinyint	1	0/1	

Table: Insurances

Column Name	Description	Data Type	Size	Format	Key?
insuranceID	Unique number linking insurance to the database	INT	20		PK
invoiceID	The invoice corresponding to the provided insurance	INT	20		FK
providerName	Name of the insurance provider	VARCHAR	45		
policyNumber	The policy number on the insurance card	INT	20		
coverageDetails	Details of what is covered by the insurance	VARCHAR	255		

Table: Equipments

Column Name	Description	Data Type	Size	Format	Key?
equipmentID	Unique number linking equipment to the database	INT	20		PK
roomNum	The room number of the room the equipment is in	INT	20		FK
model	Name of the equipment model	VARCHAR	45		
modelNumber	Model number of the equipment	VARCHAR	45		
manufacturer	Name of equipment manufacturer	VARCHAR	45		
status	Operational status of the equipment	VARCHAR	45		

Table: Doctors

Column Name	Description	Data Type	Size	Format	Key?
doctorID	Unique number linking each doctor to the database	INT	20		PK
firstName	First name of the doctor	VARCHAR	45		
lastName	Last name of the doctor	VARCHAR	45		
docSalary	Salary of the doctor	DECIMAL	7,0	9999999	

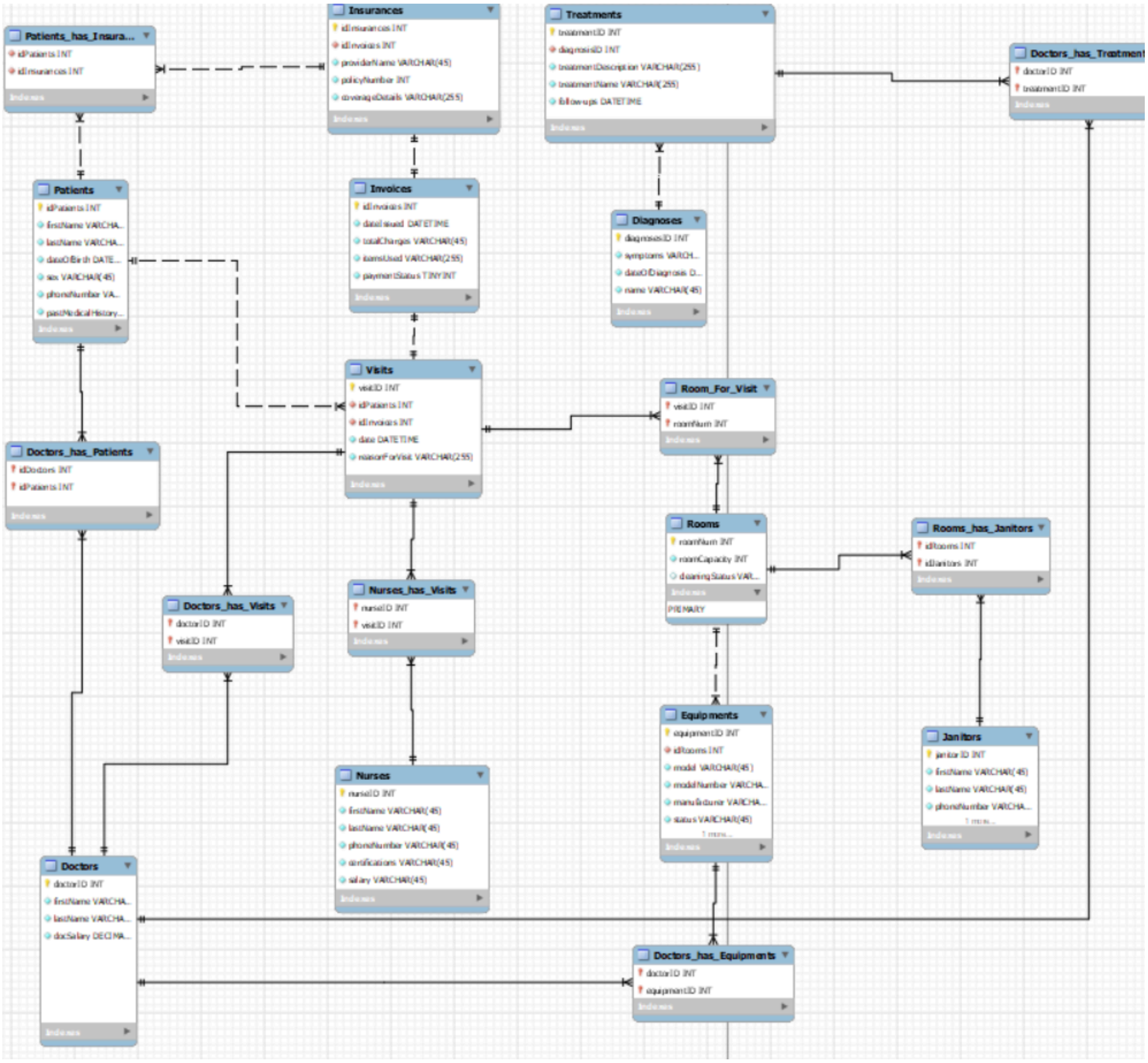
Table: Diagnoses

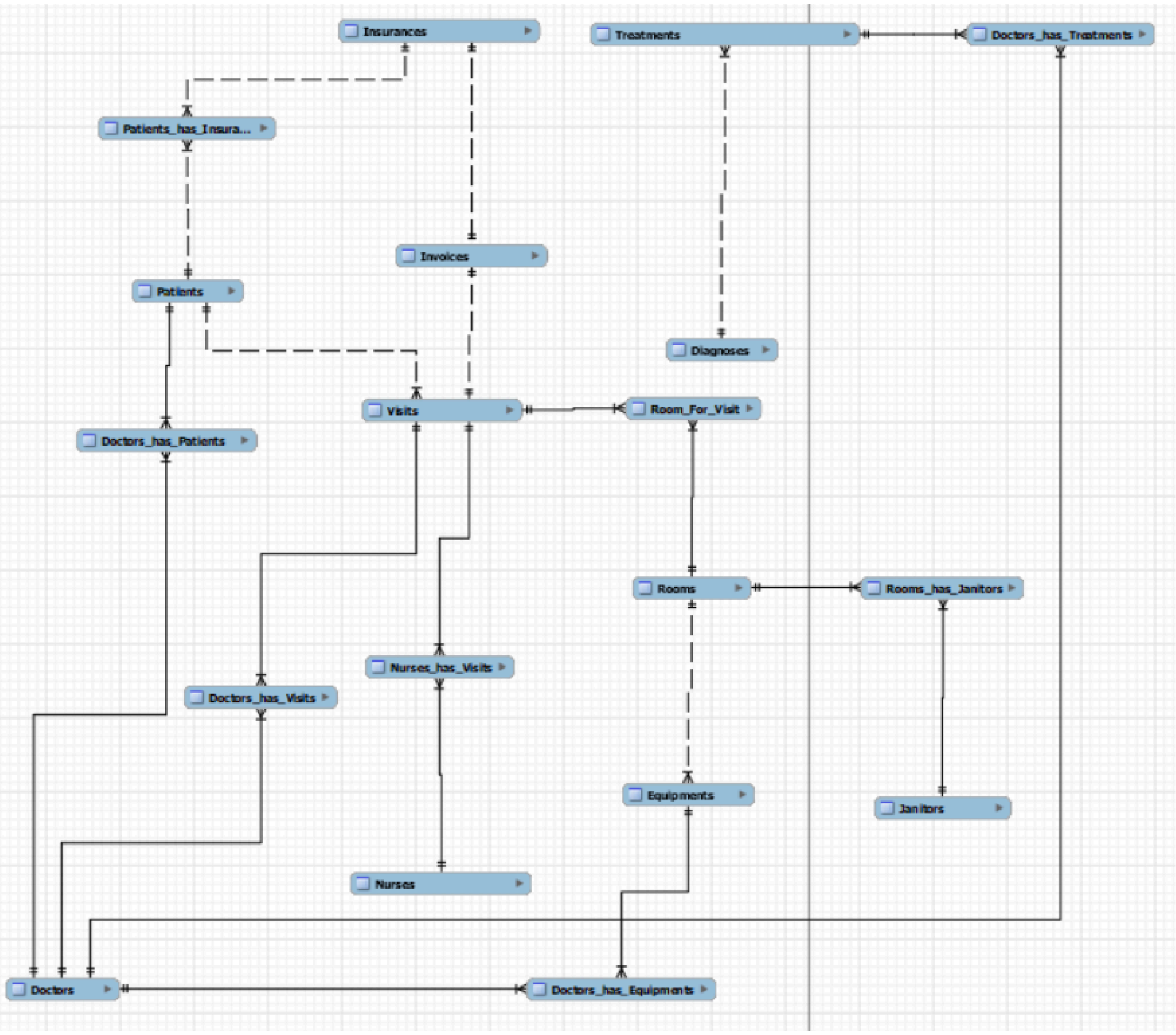
Column Name	Description	Data Type	Size	Format	Key?
diagnosisID	Unique number for each diagnosis	INT	20		PK
symptoms	Description of the symptoms associated with the diagnosis	VARCHAR	255		
dateOfDiagnosis	The date of the diagnosis	DATE	8	YYYY-MM-DD	
name	The name of the diagnosis	VARCHAR	45		

Additional Considerations

- Security and Compliance: Given the sensitive nature of healthcare data, ensuring the security and privacy of patient information in compliance with healthcare regulations such as HIPAA is paramount.
- Scalability: The database should be scalable to accommodate growth in patient volume and the potential addition of new services or locations.
- Interoperability: The ability to exchange data with other healthcare systems and providers, such as hospitals and specialist clinics, is essential for referrals and comprehensive patient care.

This overview should give the students a solid foundation to start with. It's crucial that they understand the complexity and sensitivity of healthcare data management and the importance of designing a database that is both robust and compliant with healthcare standards.




Insurances
Treatments
Doctors_has_Treatments
Patients_has_Insura...
Invoices
Patients
Diagnoses
Visits
Room_For_Visit
Doctors_has_Patients
Rooms
Rooms_has_Janitors
Nurses_has_Visits
Doctors_has_Visits
Equipments
Janitors
Nurses
Doctors
Doctors_has_Equipments

Queries

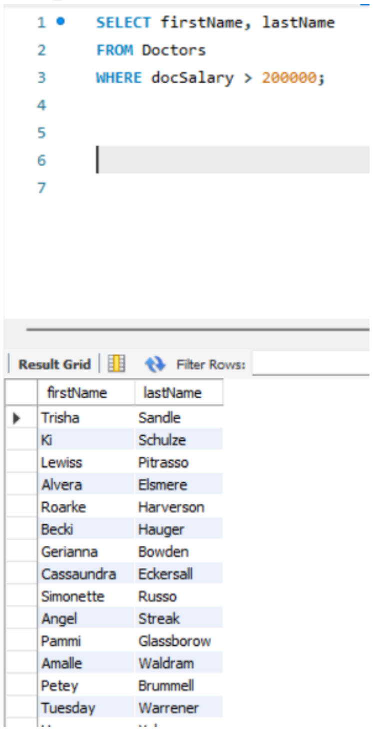
Simple:

1. **<u>List of all the doctor's Names that have a salary above $200,000</u>**

```
SELECT firstName, lastName
FROM Doctors
WHERE docSalary > 200000;
```

This SQL query retrieves the first names and last names of doctors whose salaries are above $200,000. It selects data from the "Doctors" table and filters it to include only those records where the doctor's salary is greater than $200,000.

Output:



```
SELECT firstName, lastName
FROM Doctors
WHERE docSalary > 200000;
```

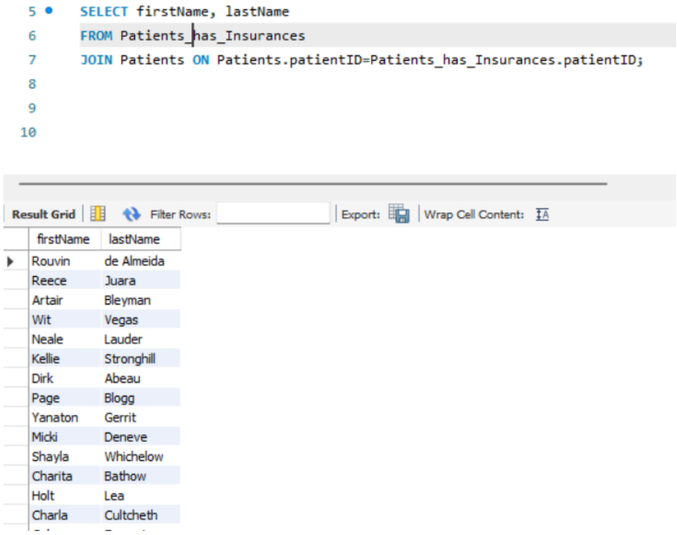
firstName	lastName
Trisha	Sandle
Ki	Schulze
Lewiss	Pitrasso
Alvera	Elsmere
Roarke	Harverson
Becki	Hauger
Gerianna	Bowden
Cassaundra	Eckersall
Simonette	Russo
Angel	Streak
Pammi	Glassborow
Amalle	Waldram
Petey	Brummell
Tuesday	Warrener

2. All the patients who have insurance.

```
SELECT firstName, lastName
FROM Patients_has_Insurances
JOIN Patients ON Patients.patientID=Patients_has_Insurances.patientID;
```

This query will list the first name and last name of all patients who have insurance.

Output:



```
SELECT firstName, lastName
FROM Patients_has_Insurances
JOIN Patients ON Patients.patientID=Patients_has_Insurances.patientID;
```

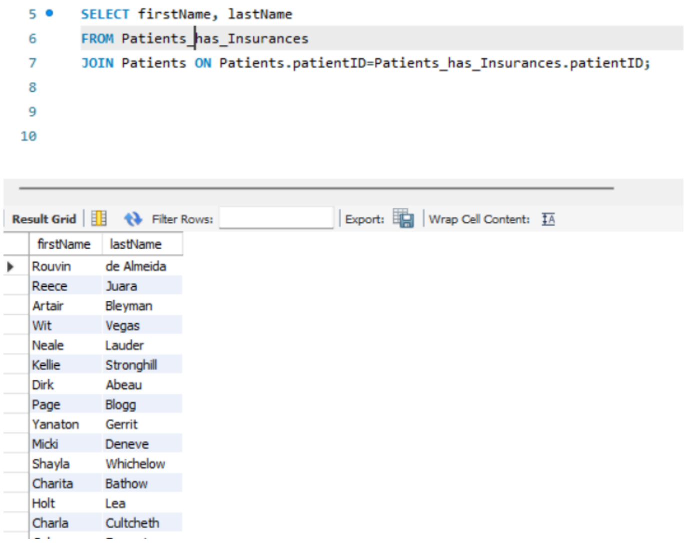
firstName	lastName
Rouvin	de Almeida
Reece	Juara
Artair	Bleyman
Wit	Vegas
Neale	Lauder
Kellie	Stronghill
Dirk	Abeau
Page	Blogg
Yanaton	Gerrit
Micki	Deneve
Shayla	Whichelow
Charita	Bathow
Holt	Lea
Charla	Cultcheth

3. The number of visits each patient has had.

```
SELECT firstName, lastName, COUNT(visitID) AS VisitCount
FROM Patients
JOIN Visits ON Patients.patientID = Visits.patientID
GROUP BY firstName, lastName;
```

This query will list each patient's first name, last name, and the count of visits they have had.

Output:



```
SELECT firstName, lastName
FROM Patients_has_Insurances
JOIN Patients ON Patients.patientID=Patients_has_Insurances.patientID;
```

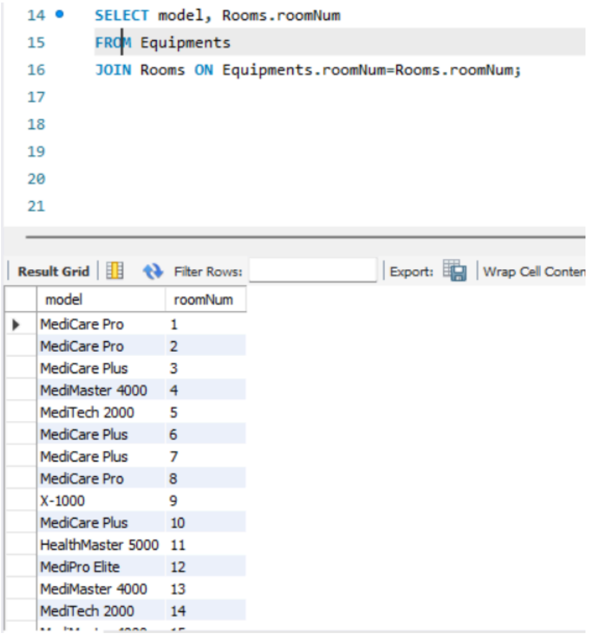
firstName	lastName
Rouvin	de Almeida
Reece	Juara
Artair	Bleyman
Wit	Vegas
Neale	Lauder
Kellie	Stronghill
Dirk	Abeau
Page	Blogg
Yanaton	Gerrit
Micki	Deneve
Shayla	Whichelow
Charita	Bathow
Holt	Lea
Charla	Cultcheth

4. List equipment model and which room it in

```
SELECT model, Rooms.roomNum
FROM Equipments
JOIN Rooms ON Equipments.roomNum=Rooms.roomNum;
```

This query will list each equipment model and its corresponding room number. If there are multiple equipment items in the same room, they will each be listed separately alongside the room number.

Output:



```
SELECT model, Rooms.roomNum
FROM Equipments
JOIN Rooms ON Equipments.roomNum=Rooms.roomNum;
```

model	roomNum
MediCare Pro	1
MediCare Pro	2
MediCare Plus	3
MediMaster 4000	4
MediTech 2000	5
MediCare Plus	6
MediCare Plus	7
MediCare Pro	8
X-1000	9
MediCare Plus	10
HealthMaster 5000	11
MediPro Elite	12
MediMaster 4000	13
MediTech 2000	14

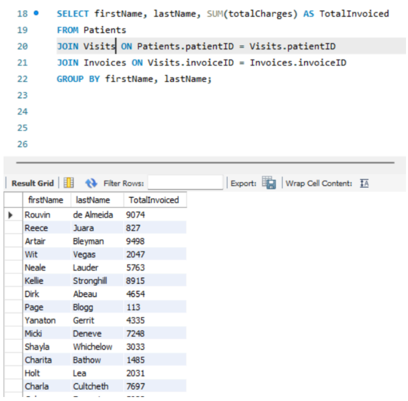
Advanced:

5. Get the total amount invoiced per patient

```
SELECT firstName, lastName, SUM(totalCharges) AS TotalInvoiced
FROM Patients
JOIN Visits ON Patients.patientID = Visits.patientID
JOIN Invoices ON Visits.invoiceID = Invoices.invoiceID
GROUP BY firstName, lastName;
```

This SQL query retrieves the total amount invoiced per patient. It joins the 'Patients', 'Visits', and 'Invoices' tables based on their respective IDs, calculates the sum of total charges for each patient, and then groups the results by the patient's first name and last name.

Output:



```
SELECT firstName, lastName, SUM(totalCharges) AS TotalInvoiced
FROM Patients
JOIN Visits ON Patients.patientID = Visits.patientID
JOIN Invoices ON Visits.invoiceID = Invoices.invoiceID
GROUP BY firstName, lastName;
```

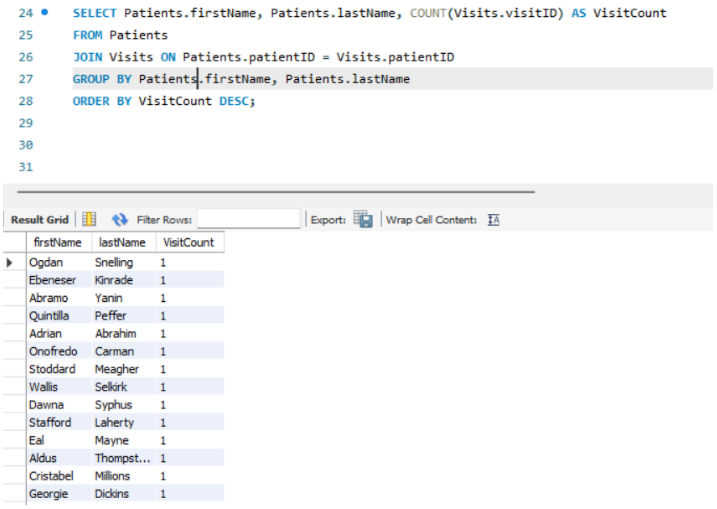
firstName	lastName	TotalInvoiced
Rouvin	de Almeida	9074
Reece	Juara	827
Artair	Bleyman	9498
Wit	Vegas	2047
Neale	Lauder	5763
Kellie	Stronghill	8915
Dirk	Abeau	4654
Page	Blogg	113
Yanaton	Gerrit	4335
Micki	Deneve	7248
Shayla	Whichelow	3033
Charita	Bathow	1485
Holt	Lea	2031
Charla	Cultcheth	7697

6. Determine which patient has the highest number of visits.

```
SELECT Patients.firstName, Patients.lastName, COUNT(Visits.visitID) AS VisitCount
FROM Patients
JOIN Visits ON Patients.patientID = Visits.patientID
GROUP BY Patients.firstName, Patients.lastName
ORDER BY VisitCount DESC;
```

This SQL query determines which patient has the highest number of visits. It achieves this by joining the 'Patients' and 'Visits' tables based on the patient ID, then counts the number of visits for each patient. The results are grouped by patient first name and last name, ordered by the visit count in descending order, showing the patient with the highest number of visits first.

Output:



```
SELECT Patients.firstName, Patients.lastName, COUNT(Visits.visitID) AS VisitCount
FROM Patients
JOIN Visits ON Patients.patientID = Visits.patientID
GROUP BY Patients.firstName, Patients.lastName
ORDER BY VisitCount DESC;
```

firstName	lastName	VisitCount
Ogdan	Snelling	1
Ebeneser	Kinrade	1
Abramo	Yanin	1
Quintilla	Peffer	1
Adrian	Abrahim	1
Onofredo	Carman	1
Stoddard	Meagher	1
Wallis	Selkirk	1
Dawna	Syphus	1
Stafford	Laherty	1
Eal	Mayne	1
Aldus	Thompst...	1
Cristabel	Millions	1
Georgie	Dickins	1

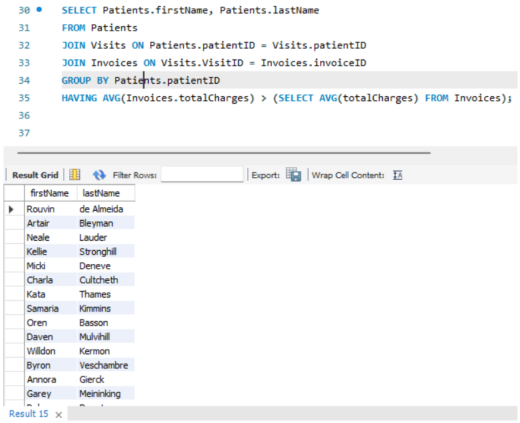
7. All patients who have paid more than the average amount

```
SELECT Patients.firstName, Patients.lastName
FROM Patients
JOIN Visits ON Patients.patientID = Visits.patientID
JOIN Invoices ON Visits.VisitID = Invoices.invoiceID
```

```
GROUP BY Patients.patientID
HAVING AVG(Invoices.totalCharges) > (SELECT AVG(totalCharges) FROM Invoices);
```

This SQL query retrieves all patients who have paid more than the average amount across all invoices. It achieves this by joining the 'Patients', 'Visits', and 'Invoices' tables, grouping the results by patient ID, and then comparing the average of the total charges for each patient with the overall average of total charges from all invoices.

Output:



```
SELECT Patients.firstName, Patients.lastName
FROM Patients
JOIN Visits ON Patients.patientID = Visits.patientID
JOIN Invoices ON Visits.VisitID = Invoices.invoiceID
GROUP BY Patients.patientID
HAVING AVG(Invoices.totalCharges) > (SELECT AVG(totalCharges) FROM Invoices);
```

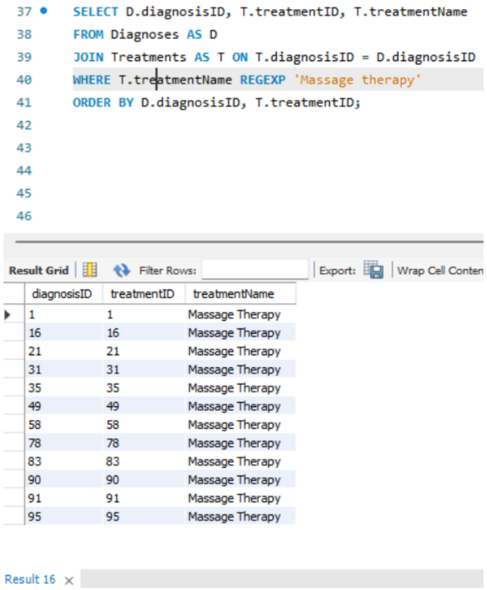
firstName	lastName
Rouvin	de Almeida
Artair	Bleyman
Neale	Lauder
Kellie	Stronghill
Micki	Deneve
Charla	Cultcheth
Kata	Thames
Samaria	Kimmins
Oren	Basson
Daven	Mulvihill
Willdon	Kermon
Byron	Veschambre
Annora	Gierck
Garey	Meininking

8. **<u>List the treatment for each diagnosis where treatment is ‘Massage Therapy’</u>**

```
SELECT D.diagnosisID, T.treatmentID, T.treatmentName
FROM Diagnoses AS D
JOIN Treatments AS T ON T.diagnosisID = D.diagnosisID
WHERE T.treatmentName REGEXP 'Massage therapy'
ORDER BY D.diagnosisID, T.treatmentID;
```

This SQL query retrieves the treatment for each diagnosis where the treatment is specifically 'Massage Therapy'. It does this by joining the 'Diagnoses' and 'Treatments' tables based on the diagnosis ID, filtering for treatments with the name 'Massage Therapy' using a regular expression match, and finally ordering the results by diagnosis ID and treatment ID.

Output:



```
SELECT D.diagnosisID, T.treatmentID, T.treatmentName
FROM Diagnoses AS D
JOIN Treatments AS T ON T.diagnosisID = D.diagnosisID
WHERE T.treatmentName REGEXP 'Massage therapy'
ORDER BY D.diagnosisID, T.treatmentID;
```

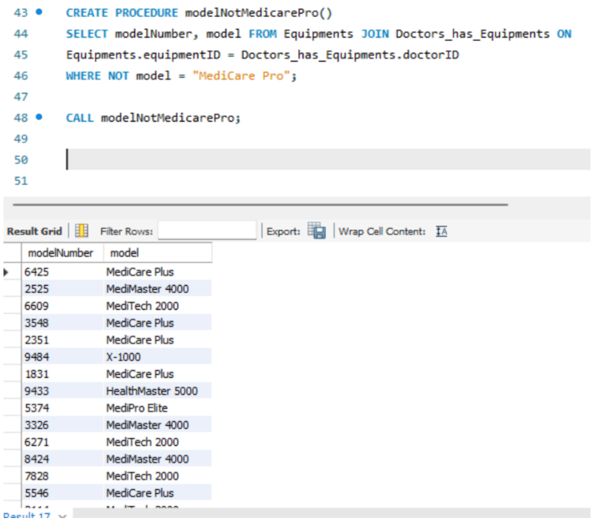
diagnosisID	treatmentID	treatmentName
1	1	Massage Therapy
16	16	Massage Therapy
21	21	Massage Therapy
31	31	Massage Therapy
35	35	Massage Therapy
49	49	Massage Therapy
58	58	Massage Therapy
78	78	Massage Therapy
83	83	Massage Therapy
90	90	Massage Therapy
91	91	Massage Therapy
95	95	Massage Therapy

9. List model that is not “Medicare Pro” within a procedure

```
CREATE PROCEDURE modelNotMedicarePro()
SELECT modelNumber, model FROM Equipments JOIN Doctors_has_Equipments ON
Equipments.equipmentID = Doctors_has_Equipments.doctorID
WHERE NOT model = "MediCare Pro";
```

This SQL stored procedure named "modelNotMedicarePro" lists the model numbers and names of equipment that are not categorized as "MediCare Pro" within a procedure. It achieves this by joining the "Equipments" and "Doctors_has_Equipments" tables based on the equipment ID, then filters the results to exclude equipment models labeled as "MediCare Pro" using the NOT operator. Finally, it selects the model number and name of the equipment meeting these criteria.

Output:



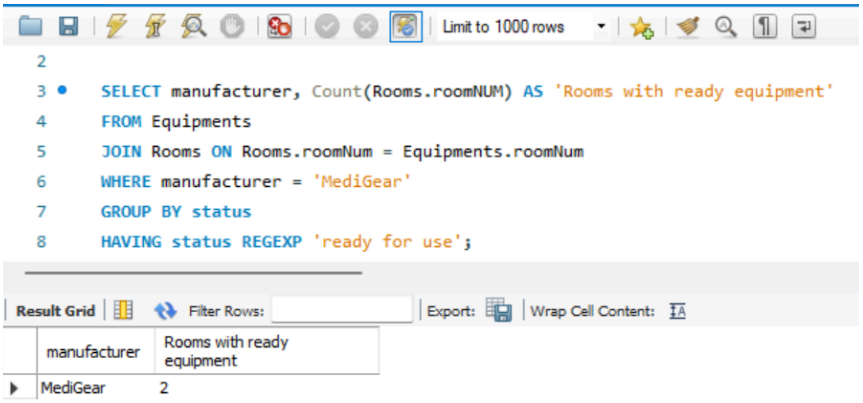
```
CREATE PROCEDURE modelNotMedicarePro()
SELECT modelNumber, model FROM Equipments JOIN Doctors_has_Equipments ON
Equipments.equipmentID = Doctors_has_Equipments.doctorID
WHERE NOT model = "MediCare Pro";

CALL modelNotMedicarePro;
```

modelNumber	model
6425	MediCare Plus
2525	MediMaster 4000
6609	MediTech 2000
3548	MediCare Plus
2351	MediCare Plus
9484	X-1000
1831	MediCare Plus
9433	HealthMaster 5000
5374	MediPro Elite
3326	MediMaster 4000
6271	MediTech 2000
8424	MediMaster 4000
7828	MediTech 2000
5546	MediCare Plus

10. **Count all rooms that have equipment made by MediGear and are ready for use**

SELECT manufacturer, Count(Rooms.roomNUM) AS ‘Rooms with ready equipment’
FROM Equipments
JOIN Rooms ON Rooms.roomNum = Equipments.roomNum
WHERE manufacturer = 'MediGear'
GROUP BY status
HAVING status REGEXP 'ready for use';

This SQL query retrieves the count of rooms that have equipment manufactured by "MediGear" and are ready for use. It accomplishes this by joining the "Equipments" and "Rooms" tables based on the room number, filtering for equipment made by "MediGear," and then grouping the results by the status of the rooms. Finally, it selects the count of rooms with the status "ready for use."



```

SELECT manufacturer, Count(Rooms.roomNUM) AS 'Rooms with ready equipment'
FROM Equipments
JOIN Rooms ON Rooms.roomNum = Equipments.roomNum
WHERE manufacturer = 'MediGear'
GROUP BY status
HAVING status REGEXP 'ready for use';
```

manufacturer	Rooms with ready equipment
MediGear	2

Query Matrix

Feature	Query 1	Query 2	Query 3	Query 4	Query 5	Query 6	Query 7	Query 8	Query 9	Query 10
Multiple Table Join					**X**					
Subquery							**X**			
GROUP BY			**X**		**X**	**X**	**X**			**X**
HAVING							**X**			**X**
JOIN		**X**	**X**	**X**	**X**		**X**	**X**	**X**	**X**
ORDER						**X**		**X**		
Procedure									**X**	
REGEXP								**X**		**X**
WHERE	**X**							**X**	**X**	**X**