**WolfHospital Database Management System**

For a hospital in North Carolina

CSC 540 Database Systems

Project Report #2

Project Team Member:

Xiaoting Fu, Yifan Zhao, Xiao Ma, Xuejiao Fang

# Assumptions

Assumptions

1. One ward can only be assigned to one patient, no matter how many beds the ward has (similar to hotel booking schema, 1-bed-room and 2-beds rooms can only be booked by one customer).
2. Each patient needs to check-in and be assigned a ward no matter if the patient going to stay overnight or not
3. One nurse can be responsible for multiple wards and when assigning a nurse to a ward the system will select a nurse who has the least number ward that he/she is in charge of.
4. The ward can be used for patient to rest or doctor to give treatment
5. Staff can be subdivided into Doctors, Nurses and Operators
6. Only Doctors have professional title
7. One check-in can accept many treatments, each treatment generates a medical record.
8. When a patient arrives at the hospital, he will provide his requirement for a ward, if the ward exists, he will create an account otherwise his account is not created.
9. When creating a new patient account, his billing account created at the same time.
10. A billing record is created in the following three cases: when a new medical record is created, when a registration fee is generated and when the accommodation fee is generated.

# **1. Database Schema**

* Patients(PID, SSN, Name, DoB, Gender, Age, Status, Address, PhoneNum)
* Wards(WNum, Capacity, ChargePerDay, AvaliableBeds)
* Beds(BNum, WNum, Occupied)
* Staff(SID, Name, Age, Gender, JobTitle, ProfTitle, Department, Address, PhoneNum)
* BillingAccounts(PID, BillingAddr, CreatedDate, Balance)
* BillingRecords(RID, MID, CreatedDate)
* PaymentMethods(CardNum,Type, CardValidDate)
* MedicalRecord(MID, RecordType,StartDate, EndDate, Prescription, Diagnosis)
* MedRecordOf(PID,MID)
* ResponsibleFor(MID, SID)
* CheckIn(PID, WNum, BNum, StartDate, EndDate)
* AssignedTo(SID, WNum)
* Select(CardNum, PID)
* Add(RID, PID)

The schemas are all in 3NF because all entities have unique IDs associated with them which makes it easier to identify a specific tuple.

All functional dependencies are obvious because of the IDs associated with all entities.

*Patients*(PID, SSN, Name, DoB, Gender, Age, Status, Address, PhoneNum)

The only FD is **PID->SSN, Name, DoB, Gender, Age, Status, Address, PhoneNum** which holds because each individual patient, identified by a unique PID, has an SSN, name, date of birth, gender, age, status, address and phone number.

This relation is in BCNF because PID is a key.

*Wards*(WNum, Capacity, ChargePerDay,Occupied)

The only FD is **WNum->Capacity, ChargePerDay ,Occupied** holds because each ward, identified by a unique ward number, has capacity and the charge per day.

This relation is in BCNF because WNum is a key.

*Staff*(SID, Name, Age, Gender, JobTitle, ProfTitle, Department, Address, PhoneNum)

The only FD is **SID -> Name, Age, Gender, JobTitle, ProfTitle, Department, Address, PhoneNum** holds because each staff, identified by a unique SID, has a name, age, gender, job title, professional title, department, address, phone number, type.

JobTitle includes doctor, nurse, and operators.

This relation is in BCNF because SID is a key.

*BillingAccounts*(PID, BillingAddr, CreatedDate, Balance)

The only FD is **PID -> BillingAddr, CreatedDate, Balance** holds because each patient is identified by the unique PID, has a billingAddr, CreatedDate and balance.

This relation is in BCNF because PID is a key.

*BillingRecords*(RID, MID, CreatedDate)

The only FD is **RID->MID, CreatedDate** holds because each billing record is identified by the unique RID, and its corresponding MID and the date the record created.

This relation is in BCNF because RID is a key.

*PaymentMethods*(CardNum, Type, CardValidDate)

**CardNum -> Type, CardValidDate** holds because each payment method is identified by CardNum, CardNum, and CardValidDate.

This relation is in BCNF because CardNum is a key.

*MedicalRecord*(MID, RecordType, StartDate, EndDate, Prescription, Diagnosis)

**MID ->StartDate, EndDate, RecordType, Prescription, Diagnosis** holds because each medical record is identified by MID, and has StartDate, EndDate, Prescription, Diagnosis.

This relation is in BCNF because MID is a key.

*CheckIn*(PID, WNum, StartDate, EndDate, BedNum)

**PID, WNum -> StartDate, EndDate, BedNum**

This relation is in BCNF because PID and WNum are keys.

*AssignedTo*(SID, WNum)

**SID, WNum -> SID, WNum**

This relation is in BCNF (and thus 3NF) because it contains two key attributes.

*ResponsibleFor*(MID, SID)

**MID, SID -> MID, SID**

This relation is in BCNF (and thus 3NF) because it contains three key attributes.

*Select*(CardNum, PID)

**CardNum, PID-> CardNum, PID**

This relation is in BCNF (and thus 3NF) because it contains one key attribute.

*Add*(RID, PID)

**RID, PID -> RID, PID**

This relation is in BCNF (and thus 3NF) because it contains two key attributes.

*MedRecordOf*(PID,MID)

**PID, MID -> PID, MID**

This relation is in BCNF (and thus 3NF) because it contains two key attributes.

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# **2. Design Decisions**

1.The mechanical approach was used to create the global schema with a few exceptions.

* Each entity set was made into a relation with the same set of attributes
* Relationships were replaced by a relation whose attributes are the keys for the connected entity sets

2. All types of staff including operators, nurses, doctors are distinguished by the “type” attribute.

3. Many-to-one relationships were combined with other relations. Combining relations in this way makes it more efficient to answer queries that involve attributes of one relation than to answer queries involving attributes of several relations.

* combine tables *BillingRecords* and *Add* and add *PID* as an attribute in *BillingRecords.*
* combine tables *PaymentMethods* and *Select* and add *PID* as an attribute in *PaymentMethods.*
* combine tables *MedicalRecords* and *MedRecordOf* and add *PID* as an attribute in *MedicalRecords.*
* combine tables *MedicalRecords* and *ResponsibleFor* and add *SID* as an attribute in *MedicalRecords.*
* combine tables *Wards* and *AssignedTo* and add *SID* as an attribute in *Wards.*

***Patients*(PID, SSN, Name, DoB, Gender, Age, Status, Address, PhoneNum)**

* PID (Primary Key) – unique identifier
* SSN (Can be NULL) - not all patient has a SSN
* Name (NOT NULL) - identification and diagnosis purposes
* DoB (NOT NULL) - identification and diagnosis purposes
* Gender (NOT NULL) - identification and diagnosis purposes
* Age (NOT NULL) - identification and diagnosis purposes
* Status (NOT NULL) - must be one of the followings: “processing treatment plan”, “in ward” and “completing treatment”
* Address (NOT NULL) - contact purpose
* PhoneNum (NOT NULL) - contact purpose

***Wards*(WNum, SID, Capacity, ChargePerDay,Occupied)**

* WNum (Primary Key) – unique identifier
* SID (Referential Integrity) - refer to other entities within the database (Staff)
* Capacity (NOT NULL) - identification purposes
* ChargePerDay (NOT NULL) - identification purposes, has default value
* Occupied(NOT NULl) – must be ‘Y’ or ‘N’, the default is ‘N’

***Staff*(SID, Name, Age, Gender, JobTitle, ProfTitle, Department, Address, PhoneNum)**

* SID (Primary Key) – unique identifier
* Name (NOT NULL) - identification purpose
* Age (NOT NULL) - identification purpose
* Gender (NOT NULL) - identification purpose
* JobTitle (NOT NULL) - a staff must be one of the followings: “nurse”, “doctor”, “operator”
* ProfTitle (Can be NULL) – a staff such as a billing staff may not have a professional title
* Department (Can be NULL) - a staff may be not assigned a department
* Address (NOT NULL) - contact purpose
* PhoneNum (NOT NULL) - contact purpose

***BillingAccounts*(PID, BillingAddr, CreatedDate, Balance)**

* PID (Referential Integrity, Primary Key) – unique identifier that refer to other entities within the database (Patients)
* BillingAddr (NOT NULL) - contact purpose
* CreatedDate (NOT NULL) - assigned current date when an account is created
* Balance (NOT NULL) - has default 0.0

***BillingRecords*(RID, PID, MID, CreatedDate)**

* RID (Primary Key) – unique identifier
* PID (Referential Integrity) – refer to other entities within the database (Patients)
* MID (Referential Integrity) – refer to other entities within the database (MedicalRecord)
* CreatedDate (NOT NULL) - assigned current date when a record is created

***PaymentInformation*(CardNum, PID, PaymentMethod, CardValidDate)**

* CardNum (Primary Key) – unique identifier
* PID (Referential Integrity) – refer to other entities within the database (Patients)
* PaymentMethod (NOT NULL) - must be either ‘debit’ or ‘credit’
* CardValidDate (NOT NULL) - must be a valid card valid date for successful payments

***MedicalRecord*(MID, PID, SID, RecordType, StartDate, EndDate, Prescription, Diagnosis)**

* MID (Primary Key) – unique identifier
* PID (Referential Integrity) – refer to other entities within the database (Patients)
* SID (Referential Integrity) – refer to other entities within the database (Staff)
* StartDate (NOT NULL) - must be assigned a start date when created
* EndDate (NOT NULL) - must be assigned an end date when created
* RecordType(NOT NULL)- select from 'treatment','test', 'check-in', 'accommodation'
* Prescription (Can be NULL) - can be filled in later
* Diagnosis (Can be NULL) - can be filled in later

***CheckIn*(PID, WNum, StartDate, EndDate, BNum)**

* PID (Primary Key) – unique identifier
* WNum (Referential Integrity, Primary Key) – unique identifier that refer to other entities within the database (Wards)
* StartDate (Primary Key) - must be assigned a start date when created
* EndDate (NOT NULL) - must be assigned an end date when created
* BedNum(NOT NULL) - must be assigned the desired bed number

# **3. SQL Statements**

**3.1 Create Tables**

*Also seen codes in* [*[github]*](https://raw.githubusercontent.com/MaggieFang/HospitalDBMS-CourseDBMS/master/src/CreateTable.java)*:* [*https://raw.githubusercontent.com/MaggieFang/HospitalDBMS-CourseDBMS/master/src/CreateTable.java*](https://raw.githubusercontent.com/MaggieFang/HospitalDBMS-CourseDBMS/master/src/CreateTable.java) *,method Name: createTables();*

CREATE TABLE Patients (

PID INT AUTO\_INCREMENT,

SSN VARCHAR(9),

Name VARCHAR(40) NOT NULL,

DoB DATE NOT NULL,

Gender CHAR(1) check (Gender IN ('F','M')),

Age INT NOT NULL,

Status CHAR(64) NOT NULL,

Address VARCHAR(64),

PhoneNum VARCHAR(10) NOT NULL,

Primary Key(PID)

);  
  
CREATE TABLE Staff (

SID INT AUTO\_INCREMENT,

Name VARCHAR(40) NOT NULL,

Age INT NOT NULL,

Gender VARCHAR(1) check (Gender IN ('F','M')),

JobTitle VARCHAR(20) NOT NULL,

ProfTitle VARCHAR(20),

Department VARCHAR(20),

Address VARCHAR(64) NOT NULL,

PhoneNum VARCHAR(10) NOT NULL,

Type VARCHAR(10) check (Type IN ('Doctor','Nurse','Operator','Others')),

Primary Key(SID)

);  
  
CREATE TABLE Wards (

WNum INT AUTO\_INCREMENT,

SID INT,

Capacity INT NOT NULL,

ChargePerDay FLOAT NOT NULL,

Occupied CHAR(1) DEFAULT 'N' check (Occupied IN ('Y','N')),

Primary Key(WNum),

Foreign Key(SID) references Staff(SID)

);  
  
CREATE TABLE BillingAccounts (

PID INT,

BillingAddr VARCHAR(64) NOT NULL,

CreatedDate DATE NOT NULL,

Balance FLOAT NOT NULL DEFAULT 0.0,

Primary Key(PID),

Foreign Key(PID) references Patients(PID)

);  
  
CREATE TABLE BillingRecords (

RID INT AUTO\_INCREMENT,

PID INT,

MID INT,

CreatedDate DATE NOT NULL,

Primary Key(RID),

Foreign Key(PID) references Patients(PID)

Foreign Key(MID) references MedicalRecord(MID)

);  
  
CREATE TABLE PaymentMethods (

CardNum VARCHAR(16),

PID INT,

Type VARCHAR(20) check (Type IN ('debit','credit')),

CardValidDate VARCHAR(4) NOT NULL,

Primary Key(CardNum),

Foreign Key(PID) references Patients(PID)

);  
  
CREATE TABLE MedicalRecord (

MID INT AUTO\_INCREMENT,

PID INT,

SID INT,

RecordType VARCHAR(20) check (RecordType in ('treatment','test', 'check-in', 'accommodation'))

StartDate DATE NOT NULL,

EndDate DATE NOT NULL,

Prescription VARCHAR(256),

Diagnosis VARCHAR(256),

Primary Key(MID),

Foreign Key(PID) references Patients(PID),

Foreign Key(SID) references Staff(SID)

);  
  
CREATE TABLE CheckIn (

PID INT,

WNum INT,

StartDate DATE NOT NULL,

EndDate DATE NOT NULL,

Primary Key(PID, WNum),

Foreign Key(PID) references Patients(PID),

Foreign Key(WNum) references Wards(WNum)

);

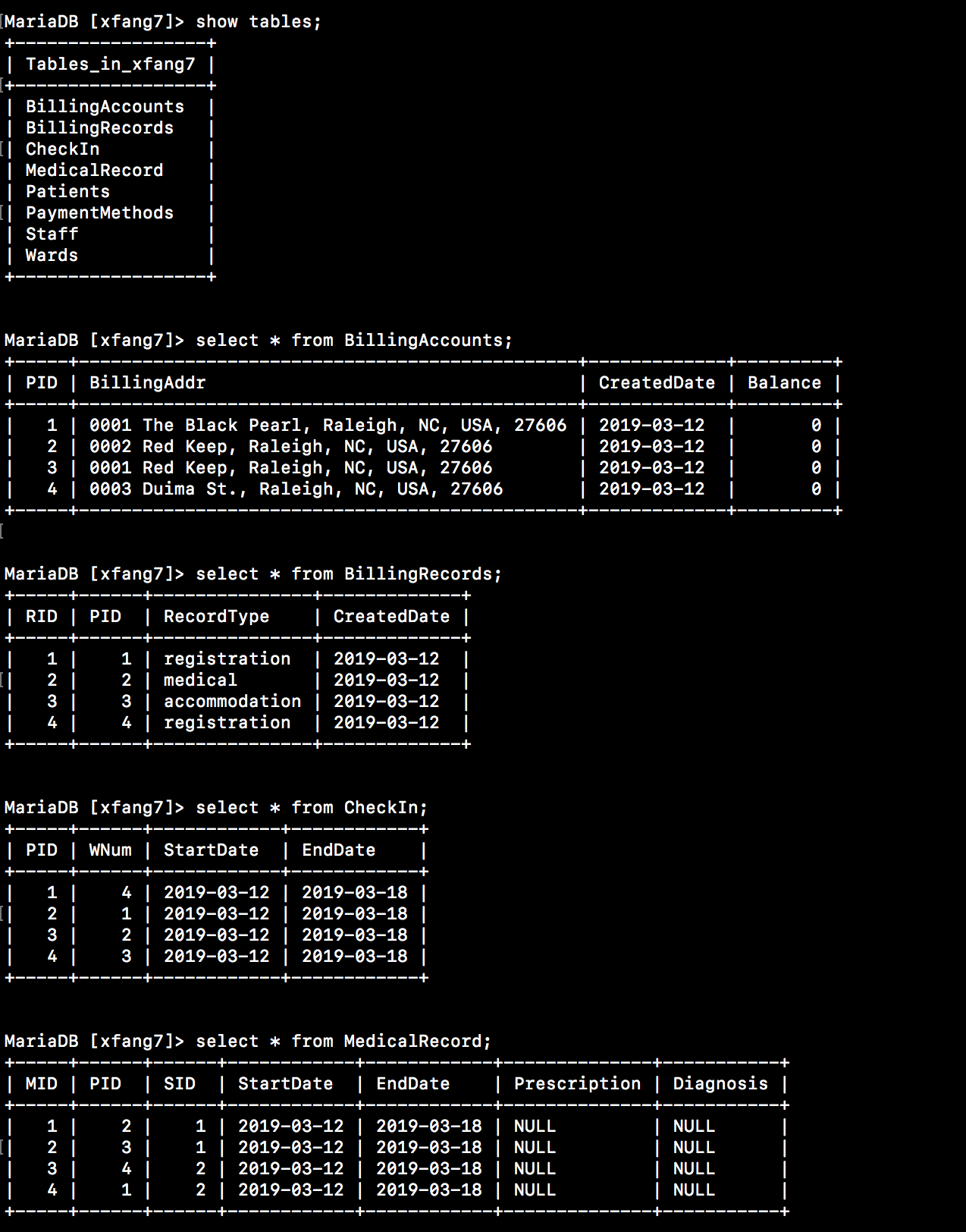
**3.2 Insert Data to Tables**

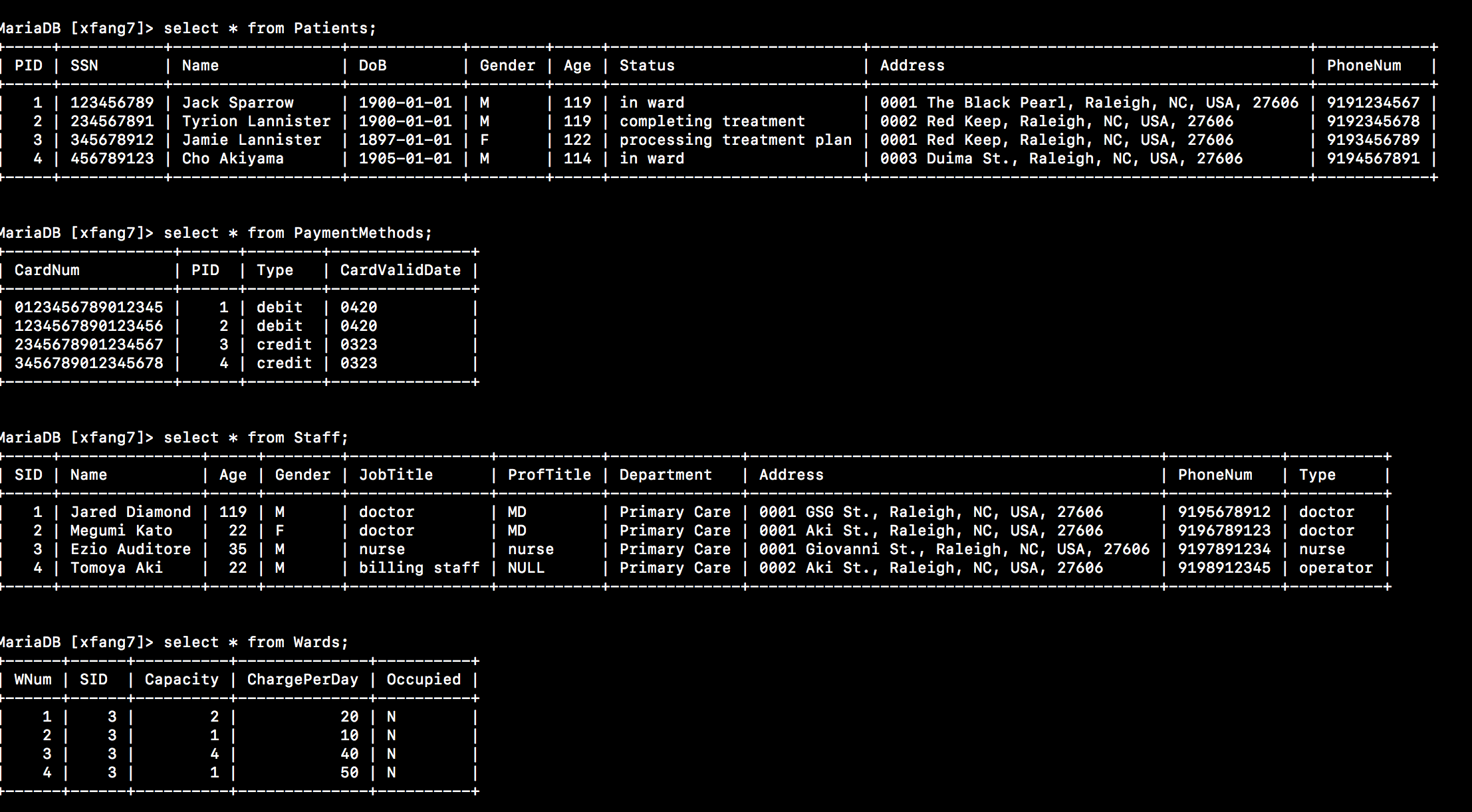
*Also seen codes in* [*[github]*](https://raw.githubusercontent.com/MaggieFang/HospitalDBMS-CourseDBMS/master/src/CreateTable.java)*:* [*https://raw.githubusercontent.com/MaggieFang/HospitalDBMS-CourseDBMS/master/src/CreateTable.java*](https://raw.githubusercontent.com/MaggieFang/HospitalDBMS-CourseDBMS/master/src/CreateTable.java) *,method Name: insertData();*

INSERT INTO Patients VALUES (NULL, '123456789', 'Jack Sparrow', '1900-01-01', 'M', 119, 'in ward', '0001 The Black Pearl, Raleigh, NC, USA, 27606', '9191234567');  
INSERT INTO Patients VALUES (NULL, '234567891', 'Tyrion Lannister', '1900-01-01', 'M', 119, 'completing treatment', '0002 Red Keep, Raleigh, NC, USA, 27606', '9192345678');  
INSERT INTO Patients VALUES (NULL, '345678912', 'Jamie Lannister', '1897-01-01', 'F', 122, 'processing treatment plan', '0001 Red Keep, Raleigh, NC, USA, 27606', '9193456789');  
INSERT INTO Patients VALUES (NULL, '456789123', 'Cho Akiyama', '1905-01-01', 'M', 114, 'in ward', '0003 Duima St., Raleigh, NC, USA, 27606', '9194567891');  
  
  
INSERT INTO Staff VALUES (NULL, 'Jared Diamond', 119, 'M', 'doctor', 'MD', 'Primary Care', '0001 GSG St., Raleigh, NC, USA, 27606', '9195678912', 'doctor');  
INSERT INTO Staff VALUES (NULL, 'Megumi Kato', 22, 'F', 'doctor', 'MD', 'Primary Care', '0001 Aki St., Raleigh, NC, USA, 27606', '9196789123', 'doctor')");  
INSERT INTO Staff VALUES (NULL, 'Ezio Auditore', 35, 'M', 'nurse', 'nurse', 'Primary Care', '0001 Giovanni St., Raleigh, NC, USA, 27606', '9197891234', 'nurse');  
INSERT INTO Staff VALUES (NULL, 'Tomoya Aki', 22, 'M', 'billing staff', NULL, 'Primary Care', '0002 Aki St., Raleigh, NC, USA, 27606', '9198912345', 'operator');  
  
  
INSERT INTO Wards VALUES (NULL, 3, 2, 20.0,'N');  
INSERT INTO Wards VALUES (NULL, 3, 1, 10.0,'N');  
INSERT INTO Wards VALUES (NULL, 3, 4, 40.0,'N');  
INSERT INTO Wards VALUES (NULL, 3, 1, 50.0,'N');  
  
  
INSERT INTO BillingAccounts VALUES (1, '0001 The Black Pearl, Raleigh, NC, USA, 27606', '2019-03-12', 0.0);  
INSERT INTO BillingAccounts VALUES (2, '0002 Red Keep, Raleigh, NC, USA, 27606', '2019-03-12', 0.0);  
INSERT INTO BillingAccounts VALUES (3, '0001 Red Keep, Raleigh, NC, USA, 27606', '2019-03-12', 0.0);  
INSERT INTO BillingAccounts VALUES (4, '0003 Duima St., Raleigh, NC, USA, 27606', '2019-03-12', 0.0);  
  
  
INSERT INTO BillingRecords VALUES (NULL, 1, 1, '2019-03-12');  
INSERT INTO BillingRecords VALUES (NULL, 2, 2, '2019-03-12');  
INSERT INTO BillingRecords VALUES (NULL, 3, 3, '2019-03-12');  
INSERT INTO BillingRecords VALUES (NULL, 4, 4, '2019-03-12');  
  
  
INSERT INTO PaymentMethods VALUES ('0123456789012345', 1, 'debit', '0420');  
INSERT INTO PaymentMethods VALUES ('1234567890123456', 2, 'debit', '0420');  
INSERT INTO PaymentMethods VALUES ('2345678901234567', 3, 'credit', '0323');  
INSERT INTO PaymentMethods VALUES ('3456789012345678', 4, 'credit', '0323');  
  
  
INSERT INTO MedicalRecord VALUES (NULL, 2, 1,'check-in' '2019-03-12', '2019-03-18', NULL, NULL);  
INSERT INTO MedicalRecord VALUES (NULL, 3, 1,'treatment' '2019-03-12', '2019-03-18', NULL, NULL);  
INSERT INTO MedicalRecord VALUES (NULL, 4, 2, 'test','2019-03-12', '2019-03-18', NULL, NULL);  
INSERT INTO MedicalRecord VALUES (NULL, 1, 2, 'treatment','2019-03-12', '2019-03-18', NULL, NULL);  
  
  
INSERT INTO CheckIn VALUES (2, 1, '2019-03-12', '2019-03-18');  
INSERT INTO CheckIn VALUES (3, 2, '2019-03-12', '2019-03-18');  
INSERT INTO CheckIn VALUES (4, 3, '2019-03-12', '2019-03-18');  
INSERT INTO CheckIn VALUES (1, 4, '2019-03-12', '2019-03-18');

**3.3 Select from tables**

**Select \* Queries & Results in MariaDB are shown in the figures below:**





# **4. Interactive SQL Queries**

Note: All results sets from select statements have been formatted for readability.

**4.1 Queries for tasks and operations**

*Also seen codes in* [*[github]*](https://raw.githubusercontent.com/MaggieFang/HospitalDBMS-CourseDBMS/master/src/CreateTable.java)*,method Name: infoProcess() and Origina pictures can be seen in the directory img of the project*

**4.1.1 Information processing**

**4.1.1.1 Insert & Update & Delete in Staff**

**A screenshot of a video game

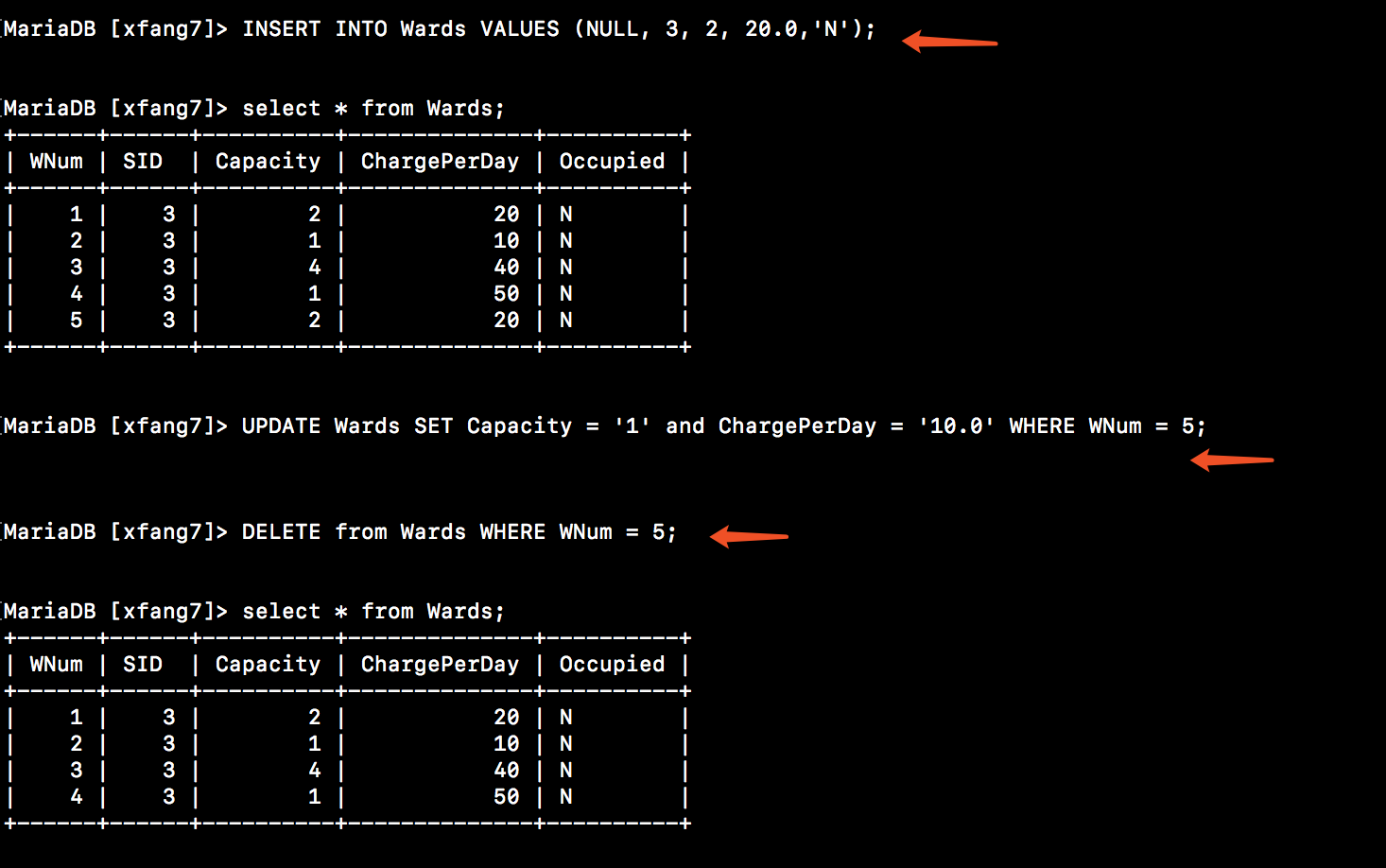
Description automatically generated**

**4.1.1.2 Insert & Update & Delete in Patients**

**A screenshot of a video game

Description automatically generated**

**4.1.1.3 Insert & Update & Delete in Wards**

****

**4.1.1.4 Check Wards Information, including the available wards.**

The queries below show information for available wards and wards with 1 or 2 or 4 beds;

**A close up of text on a black background

Description automatically generated**

**4.1.1.5 Assign wards/beds to patients according to their requests (1-bed, 2-bed, and 4-bed ward)**

Assume the request beds is 1 for Patient with PID = 1. Assign wards to him. It will include actions to find an available ward with Capacity = 1 and insert a row of CheckIn and Update the Wards Occupied field.

**A screenshot of a cell phone

Description automatically generated**

**4.1.1.6 Reserve wards with certain number of beds**

Assume the request beds is 2 for Patient with PID = 2. Assign wards to him. It will include actions to find an available ward with Capacity = 2 and insert a row of CheckIn and Update the Wards Occupied field;

**A screenshot of a cell phone

Description automatically generated**

Assume the request beds is 2 for Patient with PID = 3. Assign wards to him. It will include actions to find an available ward with Capacity = 4 and insert a row of CheckIn and Update the Wards Occupied field;

**A screenshot of a cell phone

Description automatically generated**

**4.1.1.7 Release a ward.**

It will include actions to delete the row from CheckIn and Update the Wards Occupied field;

**A screenshot of a cell phone

Description automatically generated**

**4.1.2** ***Maintaining*** ***medical records***

**4.1.2.1 Enter/update a new medical record for each treatment, test, and check-in.**

Enter three Medical Records for patient with PID = 2 for check-in, test and treatments respectively. And then update the three records. Modify the startDate or Diagnosis.

**A screenshot of a cell phone

Description automatically generated**

**4.1.3** ***Maintaining Billing Accounts***

**4.1.3.1 Generate/maintain billing accounts for every visit of every patient**

We assume that once a new patient is added. There will be a billing account created that is binding to him. We create Billing Accounts for patients with PID = 1,2,3,4 and update the address of PID = 1;

**A screenshot of a cell phone

Description automatically generated**

**4.1.4 *Reports***

**4.1.4.1 Report the medical history for a given patient**

The query below reports the medical records for patients with PID = 2;

**A screen shot of a keyboard

Description automatically generated**

**4.1.4.2 Report the medical history for a given patient for a certain time period (month/year)**

The query below reports the medical records for patients with PID=2 in March,2019.

**A screen shot of a computer

Description automatically generated**

**4.1.4.3 Report the usage of the wards that are Occupied, including details and Numbers.**

**4.1.4.3.1. The query below reports the number of *available* wards and details:**

**A picture containing text

Description automatically generated**

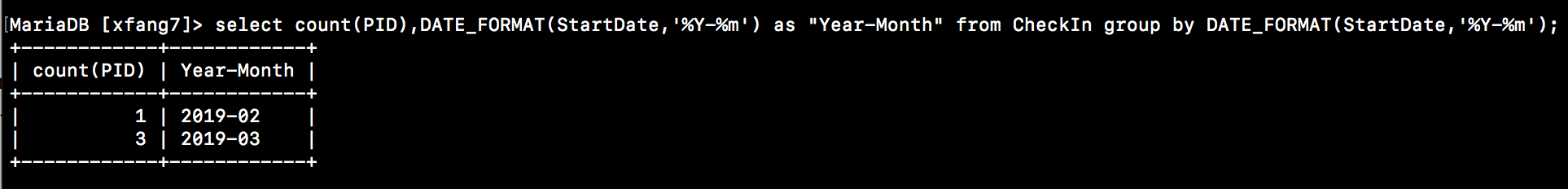
**4.1.4.3.2. The query below reports the number of *unavailable* wards and details:**

**A screenshot of a cell phone

Description automatically generated**

**4.1.4.4 Report the number of patients per month/year.**

When a patient visited. He has a checkIn information in table CheckIn. So just group by Year-Month and count the PID. Results are shown below:



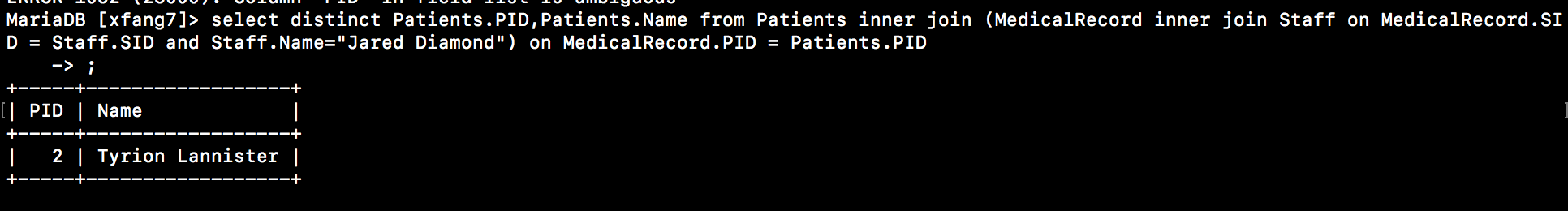
**4.1.4.5 Report the ward-usage percentage.**

A screenshot of a cell phone

Description automatically generated

**4.1.4.6 Report Information on all the patients a given doctor is currently responsible for.**

The query below find all the patients(PID,Name) that doctor with name “Jared Diamond” is responsible for.



**4.1.4.7 Report the number of staff, group by their roles and then show each group’s detail information.**

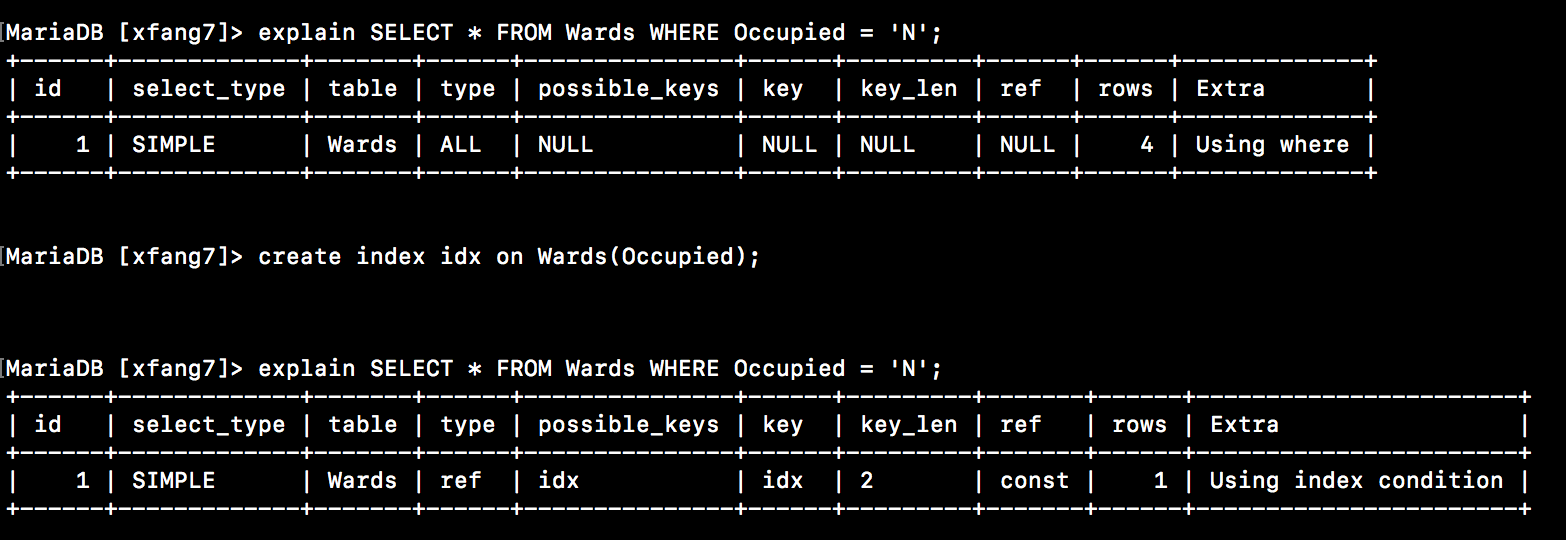
A screenshot of text

Description automatically generated

**4.2 Autotrace and Indexes for two tables**

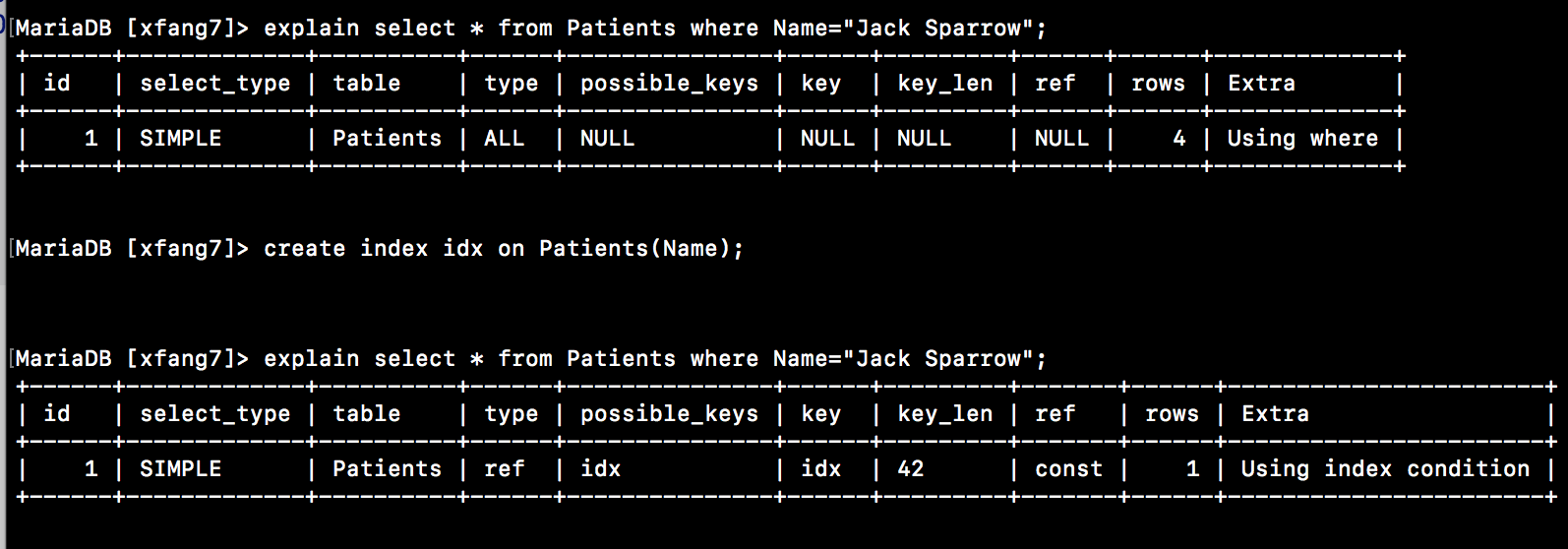
**4.2.1 The first index was added to Wards based on the "Occupied" column.**

Before the creation of index, 4 rows are examined in the Wards table; after the index is created, only one row is examined.



**4.2.2. The second index was added to Patients based on the "Name" column.**

Before the creation of index, we need to examine 4 rows in the Patients table, after that, we only need to exam 1 row.



**4.3 Query correctness proofs**

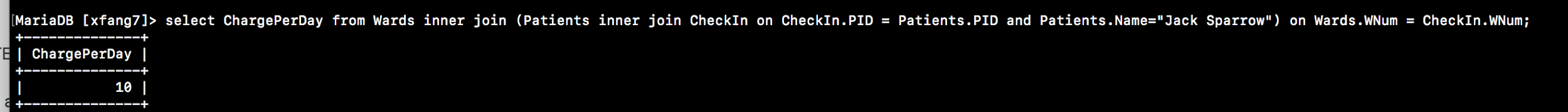
**Query 1**

1. **Specification:**

“find the charges per day of the ward which patient “Jack Sparrow” is living in”

1. **SQL query:**

select ChargePerDay from Wards inner join (Patients inner join CheckIn on CheckIn.PID = Patients.PID and Patients.Name="Jack Sparrow") on Wards.WNum = CheckIn.WNum;



1. **algebra expression:**

ΠChargePerDay (Wards⋈Wards.WNum=CheckIn.WNum(Patients⋈Patient.Name=”Jack Sparrow” AND Patient.PID=CheckIn.PIDCheckIn))

1. **correctness proof:**

Suppose p is any tuple in the *Patient* relation, c is any tuple in the *CheckIn* relation, such that the value p.PID is the same as the value c.PID and p.Name is equal to the given patient name. Each such combination of tuples (p, c) gives all information about the given patient and his/her check-in information. Then for such tuples (p, c) and w which is any tuples in the *Ward* table, such that the value c.WNum is the same as the value w.WNum. Each such combination of tuples (p, c, w) gives all the information about the given patient, his/her check-in information and the ward he/she is living in. Finally the query returns the charges per day (w.ChargePerDay) of the patient’s ward of in such combination of tuples (p, c, w).

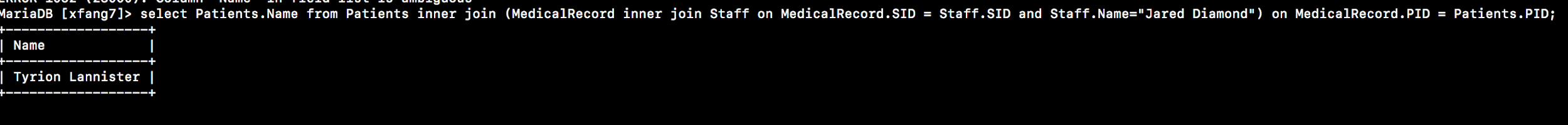
**Query 2**

1. **Specification:**

“find all the patients a given doctor “Jared Diamond” is responsible for”

1. **SQL query:**

select Patients.Name from Patients inner join (MedicalRecord inner join Staff on MedicalRecord.SID = Staff.SID and Staff.Name="Jared Diamond") on MedicalRecord.PID = Patients.PID;



1. **algebra expression:**

ΠPatient.Name (Patients⋈Patients.PID=MedicalRecord.PID(MedicalRecord⋈Staff.name=”Jared Diamond” AND Staff.SID=MedicalRecord.SIDStaff))

1. **correctness proof:**

Suppose s is any tuple in the *Staff* relation, m is any tuple in the *MedicalRecord* relation, such that the value s.SID is the same as the value m.SID and s.Name is equal to the given doctor name. Each such combination of tuples (s, m) gives all the information about the given doctor, together with the medical records that the doctor participates in. Then for such tuples (s, m) and p which is any tuples in the *Patient* table, such that the value p.PID is the same as the value m.PID. Each such combination of tuples (s, m, p) gives all the information about the given doctors, the medical record that the doctor participates in and the patients that the doctor is responsible for. Finally the query returns all the patient name (p.Name) of in such combination of tuples (s, m, p).