

12/1/2020

# EMERGENCY MEDICAL RESPONSE RECORD MANAGEMENT SYSTEM

Final Documentation

Jules Gammad  
Muhammad Khan  
CPS510 – SECTION 08

# Contents

Introduction	2
Enhanced Entity-Relationship Diagram	2
Schema Design	4
Normalization to 3NF	5
Examples	6
Employee Table	6
Doctors Table	6
Patients and Date Admitted Table	7
Normalization to BCNF by algorithm	9
Examples	9
Medicine Table	9
Logistics Table	10
Bed Table	10
Date Admitted Table	10
Patients Table	10
Medicine prescription Table	10
Emergency Contact Table	10
Queries/Views and Relational Algebra	10
Simple Queries	10
Advanced Queries	13
Views	16
Unix Implementation	16
Graphical User Interface Implementation	20
Set-up	20
Conclusion	27
Appendix A	28

## Introduction

Through the pandemic we have witnessed the importance of effective medical response and data collection to deliver immediate care for those in need. For example, during the peak of the Covid-19 pandemic multiple field hospitals were built across the globe to handle the influx of new infected patients. These field hospitals represent a logistical issue in that these platforms need to maintain fast and clean collection and management of information to be able to run efficiently and effectively.

Through this GUI-based application, there will be 2 distinct portals available to individuals based on their role. These individuals would be able to read, write and remove specific data and given their credentials. Operations staff would be able to read, write and remove any data from logistics and employee entities in the database while doctors and nurses would be able to read, write and remove the medicine, medicine prescription, and patient entities. In other words, operations staff would be able to see and perform operations on logistics and schedules of employees. Doctors and nurses would be able to see how much medicine is left as well as to give out medicine to patients.

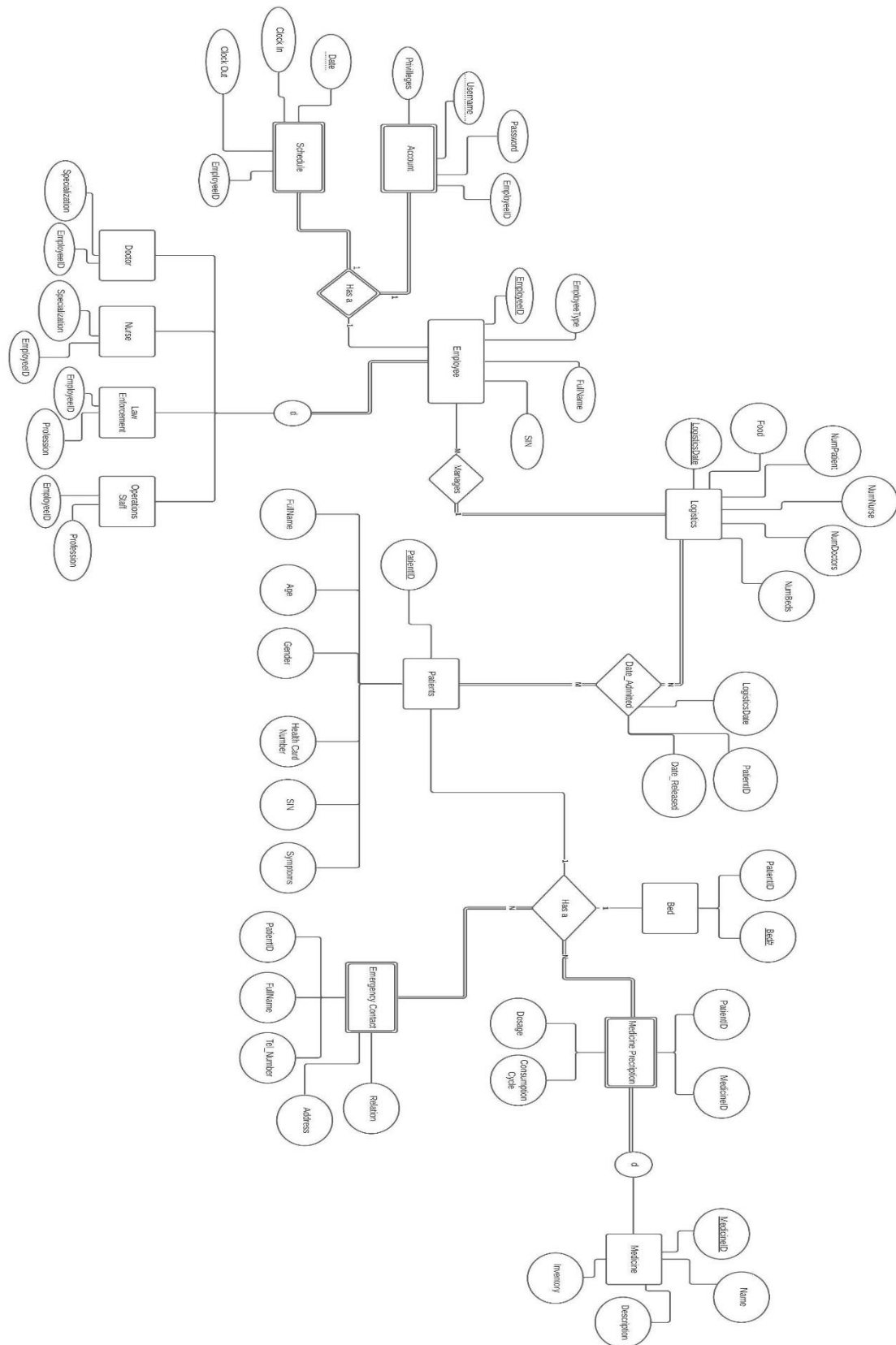
The database will consist of a wide variety of interconnected entity objects as well as standalone elements as shown on the EER diagram in **Figure 1**. Ultimately, having a centralized relation database management system will help streamline processes, allowing professional medical personnel to focus on their job and help save more people in times of tragedy.

## Enhanced Entity-Relationship Diagram

**Figure 1** below displays the enhanced entity-relationship diagram (EER) for the Emergency Medical Response Record Management System. The following points highlight the key aspects of understanding the diagram:

- The Employee entity establishes a disjoint and total specialization relationship with Doctor, Nurses, Law Enforcement and Operations Staff entity.
- Account and Schedule entity are weak entities of the Employee entity since they cannot exist on their own and as such, they have a 1:1 multiplicity.
- The Employee entity has a partial participation with the Logistics table since not all employees need to access logistics and as such, they have an N:1 multiplicity.
- The Date\_Admitted relationship entity connects Logistics entity and the rest of the database mainly through transitivity from the Patients entity. It has an M: N multiplicity because there can be multiple admissions on different dates of logistics.
- Emergency Contact and Medicine Prescription are weak entities of Patients as they cannot exist on their own. As such, they have a total participation compared to Patients which has a partial participation since patients do not necessarily have prescriptions or emergency contacts. Because of this, the relationship exhibits a 1:M multiplicity.
- The Bed entity has a 1:1 relationship with the Patients entity because each patient must only be assigned one bed. However, the Bed entity is not a weak entity of Patients because it can exist on its own.
- Medicine exhibits a disjoint and partial specialization with Medicine Prescription. This is because medicine prescription represents is-a relationship to medicine.

**Figure 1.** Enhanced Entity-Relationship Diagram for the Emergency Medical Response Record Management System.



## Schema Design

**Figure 2a, Figure 2b and Figure 3c** display the segmented code snippet for the Emergency Medical Record Response Management System. The code was written in PSQL using the Oracle SQL Developer as an IDE which is connected to the school's Oracle database. The code snippet clarifies and dives deeper into the actual implementation of mapping abstract concepts to real life examples.

**Figure 2a.** Code snippet of the first segment of the schema design implementation.

```

/*
Logistics
*/
CREATE TABLE LOGISTICS (
  NUMPATIENT NUMBER(10) NOT NULL,
  NUMDOCTOR NUMBER(10) NOT NULL,
  NUMNURSE NUMBER(10) NOT NULL,
  NUMBEDS NUMBER(10) NOT NULL,
  LOGISTICSDATE DATE NOT NULL,
  PRIMARY KEY (LOGISTICSDATE)
);

/*
Employee table
*/
CREATE TABLE EMPLOYEE (
  FULLNAME VARCHAR2(25) NOT NULL,
  SIN NUMBER(7,0),
  EMPLOYEEID NUMBER(10) NOT NULL,
  EMPLOYEEYPE VARCHAR2(20) NOT NULL CHECK (EMPLOYEEYPE IN ('doctor', 'nurses', 'law enforcement', 'operations staff')),
  PRIMARY KEY (EMPLOYEEID)
);

/*
Account
*/
CREATE TABLE ACCOUNT (
  USERNAME VARCHAR2(20) PRIMARY KEY,
  PASSWORD VARCHAR2(20) NOT NULL,
  EMP_PRIVILEGE CHAR(1) NOT NULL,
  EMPLOYEEID NUMBER(10),
  FOREIGN KEY (EMPLOYEEID) REFERENCES EMPLOYEE (EMPLOYEEID)
);

/*
Schedule table
*/
CREATE TABLE SCHEDULE (
  WORKDATE DATE,
  CLOCK_IN TIMESTAMP NOT NULL,
  CLOCK_OUT TIMESTAMP NOT NULL,
  EMPLOYEEID NUMBER(10),
  PRIMARY KEY (WORKDATE),
  FOREIGN KEY (EMPLOYEEID) REFERENCES EMPLOYEE (EMPLOYEEID)
);

/*
Doctor table
*/
CREATE TABLE DOCTORS (
  SPECIALIZATION VARCHAR2(20) NOT NULL,
  EMPLOYEEID NUMBER(10),
  FOREIGN KEY (EMPLOYEEID) REFERENCES EMPLOYEE (EMPLOYEEID)
);

```

**Figure 2b.** Code snippet of the second segment of the schema design implementation.

```

/*
Nurses table
*/
CREATE TABLE NURSES (
  EMPLOYEEID NUMBER(10),
  SPECIALIZATION VARCHAR2(40) NOT NULL,
  FOREIGN KEY (EMPLOYEEID) REFERENCES EMPLOYEE (EMPLOYEEID)
);

/*
Law Enforcement table
*/
CREATE TABLE LAW_ENFORCEMENT (
  PROFESSION VARCHAR2(20) NOT NULL,
  EMPLOYEEID NUMBER(10),
  FOREIGN KEY (EMPLOYEEID) REFERENCES EMPLOYEE (EMPLOYEEID)
);

/*
Operations Staff table
*/
CREATE TABLE OPERATIONS_STAFF (
  JOB_TITLE VARCHAR2(40) NOT NULL,
  EMPLOYEEID NUMBER(10),
  FOREIGN KEY (EMPLOYEEID) REFERENCES EMPLOYEE (EMPLOYEEID)
);

/*
Patients
*/
CREATE TABLE PATIENTS (
  PATIENTID NUMBER(10) NOT NULL,
  FULLNAME VARCHAR2(200) NOT NULL,
  AGE NUMBER(5) NOT NULL,
  GENDER CHAR NOT NULL,
  SIN NUMBER(5) NOT NULL,
  HEALTH_CN NUMBER(14,0) NOT NULL,
  SYMPTOMS VARCHAR2(200) NOT NULL,
  PRIMARY KEY (PATIENTID)
);

```

**Figure 2c.** Code snippet of the second segment of the schema design implementation.

```

CREATE TABLE DATE_ADMITTED (
    DATE_RELEASED DATE DEFAULT NULL,
    LOGISTICSDATE DATE NOT NULL,
    PATIENTID NUMBER(10) NOT NULL,
    FOREIGN KEY (LOGISTICSDATE) REFERENCES LOGISTICS(LOGISTICSDATE),
    FOREIGN KEY (PATIENTID) REFERENCES PATIENTS(PATIENTID)
);

/*
Bed
*/
CREATE TABLE BED(
    BED_ID NUMBER(5),
    PATIENTID NUMBER(10) NOT NULL,
    OCCUPIED CHAR DEFAULT 'N',
    PRIMARY KEY (BED_ID),
    FOREIGN KEY (PATIENTID) REFERENCES PATIENTS(PATIENTID)
);

/*
Medicine
*/
CREATE TABLE MEDICINE(
    MEDICINE_ID NUMBER(20,0) PRIMARY KEY,
    NAME VARCHAR2(20) NOT NULL,
    DESCRIPTION VARCHAR2(100) NOT NULL,
    INVENTORY NUMBER(20,0) NOT NULL
);

/*
Emergency Contact
*/
CREATE TABLE EMERGENCY_CONTACT(
    FULLNAME VARCHAR2(200) NOT NULL,
    RELATION VARCHAR2(200) NOT NULL,
    TEL_NUMBER NUMBER(10) NOT NULL,
    ADDRESS VARCHAR2(50) NOT NULL,
    PATIENTID NUMBER(10),
    FOREIGN KEY (PATIENTID) REFERENCES PATIENTS(PATIENTID)
);

/*
Medicine Prescription
*/
CREATE TABLE MEDICINE_PRESCRIPTION(
    MEDICINE_ID NUMBER(20,0) NOT NULL,
    PATIENTID NUMBER(10) NOT NULL,
    DOSAGE VARCHAR2(20) NOT NULL,
    CONSUMPTION_CYCLE VARCHAR2(100) NOT NULL,
    FOREIGN KEY (MEDICINE_ID) REFERENCES MEDICINE(MEDICINE_ID),
    FOREIGN KEY (PATIENTID) REFERENCES PATIENTS(PATIENTID)
);

```

## Normalization to 3NF

To normalize a database to 3NF, each table in the database must go through each normalization stage in sequence starting from 1NF then unto 2NF and finally, 3NF.

1. 1NF states that a table must contain all attributes be atomic. In other words, no table must contain multi-valued or composite attributes.
2. Once in 1NF, 2NF states that a table must establish a full functional dependency status. Having full functional dependency means that all non-prime attributes must not be derivable from any subset of composite key. Consequently, any table that has a primary key composing of just one attribute is automatically in 2NF. However, it applies that if there any composite keys that proper caution must be taken to make sure that the table is converted to 2NF either by making a new table or removing attributes/prime keys.
3. Once in 2NF, 3NF states that non-key attributes must not exhibit transitive dependence. In other words, non-key attributes must not be derivable from other non-key attributes.

## Examples

### Employee Table

Note that most of the tables of the database are already normalized to 3NF. One such example of a normalized table is the employee table below shown in **Figure 3**.

**Figure 3.** Code snippet of the employee entity.

```

/*
Employee table
*/
CREATE TABLE EMPLOYEE (
  FULLNAME VARCHAR2(25) NOT NULL,
  SIN NUMBER(7,0),
  EMPLOYEEID NUMBER(10) NOT NULL,
  EMPLOYEETYPE VARCHAR2(20) NOT NULL CHECK (EMPLOYEETYPE IN ('doctor', 'nurses', 'law enforcement', 'operations staff')),
  PRIMARY KEY (EMPLOYEEID)
);

```

1. 1NF: There are no multivalued or composite attributes.
2. 2NF: The table exhibits full functional dependency to EMPLOYEEID.
3. 3NF: The table exhibits no transitive dependence between its non-key attributes FULLNAME, SIN, AND EMPLOYEETYPE.

### Doctors Table

However, there were changes that were not reflected in the earlier part of this report to prevent clutter and confusion. Nurses, Law Enforcement, Doctors and Operations Staff entities were originally not normalized to 3NF. **Figure 4a** show the original code snippet for the Doctors entity. Note that one example is only shown because the other entities are identical to this example.

**Figure 4a.** Code snippet of the original Doctors entity.

```

/*
CREATE TABLE DOCTORS (
  SPECIALIZATION VARCHAR2(20) NOT NULL,
  EMPLOYEETYPE DEFAULT 'doctor'
  EMPLOYEEID NUMBER(10),
  FOREIGN KEY (EMPLOYEEID) REFERENCES EMPLOYEE(EMPLOYEEID)
);

```

The table is 2NF because all the attributes are atomic, and the table exhibits full functional dependency to EMPLOYEEID. However, it is not in 3NF because its non-key attributes exhibit transitive dependence in particular, the specialization attribute. Because specialization also exhibits functional dependency on EMPLOYEETYPE, this table is not in 3NF. And so, to normalize the table to 3NF, EMPLOYEETYPE was removed so no non-key attributes will exhibit transitive dependence. **Figure 4b** displays the new table for the Doctors entity.

**Figure 4b.** Code snippet of the new Doctors entity.

```
CREATE TABLE DOCTORS (
  SPECIALIZATION VARCHAR2(20) NOT NULL,
  EMPLOYEEID NUMBER(10),
  FOREIGN KEY (EMPLOYEEID) REFERENCES EMPLOYEE(EMPLOYEEID)
);
```

### Patients and Date Admitted Table

Another great example of converting one a table to 3NF is the Patients entity. Initially, the Patients entity was only in 1NF as shown in **Figure 5a**.

**Figure 5a.** Code snippet of the original Patients entity.

```
CREATE TABLE PATIENTS (
  FullName VARCHAR(200) NOT NULL,
  PatientID NUMBER(20) NOT NULL,
  Age NUMBER(5) NOT NULL,
  Gender CHAR NOT NULL,
  SIN NUMBER(5) NOT NULL,
  Health_CN NUMBER(14) NOT NULL,
  Symptoms VARCHAR(200) NOT NULL,
  Record VARCHAR(200) NOT NULL,
  PRIMARY KEY (PatientID),
  FOREIGN KEY (Logisticsdate) REFERENCES Logistics(Logisticsdate),
  FOREIGN KEY (Medicine_ID) REFERENCES Medicine_Prescription(Medicine_ID),
  FOREIGN KEY (Bed_ID) REFERENCES Beds(Bed_ID),
  FOREIGN KEY (Relation) REFERENCES Emergency_Contact(Relation)
);
```

This table did not represent 2NF because it exhibited partial dependency. As shown, the table consisted of a composite key which is composed of PatientID and several foreign keys. This caused an issue where the non-key attributes were not fully dependent on the composite key. To solve this issue, several redundant foreign keys were removed such as the Medicine\_ID, Bed\_ID and Relation foreign keys. To handle the LogisticsDate foreign key, an intersection table called Date\_Admitted was created to relate Logistics and Patients as shown in **Figure 5b**.

**Figure 5b.** Code snippet of the Date\_Admitted table.

```
CREATE TABLE DATE_ADMITTED (
  DATE_RELEASED DATE DEFAULT NULL,
  LOGISTICSDATE DATE NOT NULL,
  PATIENTID NUMBER(10) NOT NULL,
  FOREIGN KEY (LOGISTICSDATE) REFERENCES LOGISTICS(LOGISTICSDATE),
  FOREIGN KEY (PATIENTID) REFERENCES PATIENTS(PATIENTID)
);
```



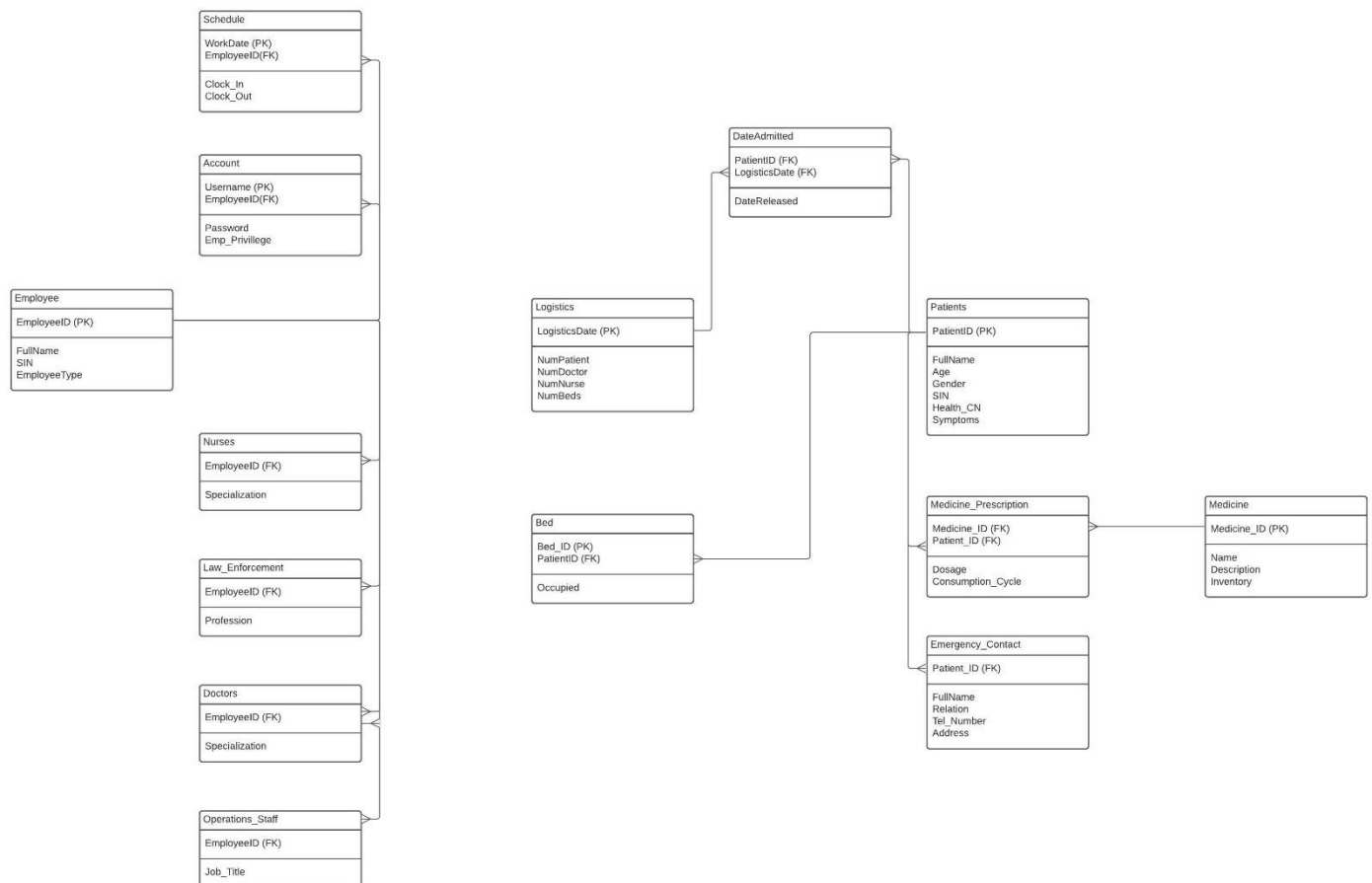
By making a new table and removing several redundant foreign keys, the Patients entity was able to be converted into 3NF. **Figure 5c**, shows the new code snippet for the Patients entity. It follows that all of the attributes are atomic, that it exhibits full functional dependency to PatientID and that the non-key attributes do not display transitive dependence.

**Figure 5c.** Code snippet of the new Patients entity.

```
CREATE TABLE PATIENTS (
PATIENTID NUMBER(10) NOT NULL,
FULLNAME VARCHAR2(200) NOT NULL,
AGE NUMBER(5) NOT NULL,
GENDER CHAR NOT NULL,
SIN NUMBER(5) NOT NULL,
HEALTH_CN NUMBER(14,0) NOT NULL,
SYMPTOMS VARCHAR2(200) NOT NULL,
PRIMARY KEY (PATIENTID)
);
```

**Figure 6** displays the full scope of the normalization to 3NF of the Emergency Medical Response Record Management System. Note that the fork to a table means that the table uses a foreign key from the end of the line.

**Figure 6.** Diagram of the 3NF tables.



## Normalization to BCNF by algorithm

BCNF is a special case of 3NF. It is accurate to say that most tables that are in 3NF are also in BCNF. The only differentiator between 3NF and BCNF lies within its functional dependencies. Consider a hypothetical table in 3NF where A is an attribute that is functionally dependent on B, that is to say  $A \rightarrow B$ . It would satisfy the 3NF definition if A is not a prime attribute and B is a prime attribute whereas BCNF insists that A must be a prime attribute.

### Examples

#### Medicine Table

The Bernstein's Algorithm is a series of steps that derives 3NF schema that is lossless and dependency preserving which are the two important properties when decomposing tables during the algorithm. Lossless join decomposition is segmenting a table into smaller parts and when joined back together produces the original table. Dependency preserving is when the all the functional dependency of a table is derivable from the closures of its decomposition. An example shown below is using the Bernstein algorithm to derive a 3NF schema that is lossless and dependency preserving for the Medicine Table:

1. Determine functional dependencies of Medicine table. Medicine can be represented as  $R \{ \text{Medicine\_ID, Name, Description, Inventory} \}$ . Let  $R = \{A, B, C, D\}$  where the letters correspond to the attributes accordingly. The functional dependency will then be  $FD = \{A \rightarrow BCD\}$ .
2. Expand the functional dependency. By using the decomposition rule, it is possible to expand the functional dependencies such that  $FD = \{A \rightarrow B, A \rightarrow C, A \rightarrow D\}$ .
3. Remove redundancies by finding the closures of each functional dependency.

$A \rightarrow B: A^+ = \{A, C, D\}$ , not

$A \rightarrow C: A^+ = \{A, C, D, B\}$ , we get C so redundant.

$A \rightarrow D: A^+ = \{A, C, D, B\}$ , we get D so redundant.

$B \rightarrow A: B^+ = \{B, C, D, A\}$ , we get A so redundant.

$B \rightarrow C: B^+ = \{B, A, D, C\}$ , we get B so redundant.

$AB \rightarrow C: AB^+ = \{A, B, C, D\}$ , we get C so redundant.

$AB \rightarrow D: AB^+ = \{A, B, C, D\}$ , we get C so redundant.

$CB \rightarrow A: CB^+ = \{A, B, C, D\}$ , we get C so redundant.

$CB \rightarrow D: CB^+ = \{A, B, C, D\}$ , we get D so redundant.

4. Finding the keys. To find the keys, there are three simple steps to follow. If an attribute is not in LHS and not in RHS, it is part of the key. If an attribute is only in LHS, it is part of the key. If an attribute is only in RHS, it is not part of the key.

Since  $A \rightarrow B$  are the only attributes that are not redundant,

It follows that A is a key and B is not a key.

5. Make tables. Therefore, it can be concluded that:

$R1(A, B)$  with  $FD: A \rightarrow B$

$R2(C, D)$  with no FD

Since the tables are already in 3NF, it is only needed to double check that the left-hand side contains super keys. The following tables show their functional dependency and the reasoning behind why the table is in BCNF.

#### Logistics Table

FD: {LogisticsDate --> NumPatient, NumDoctor, NumNurse, NumBeds}

This table is in BCNF because all attributes are dependent on the primary (candidate) key LogisticsDate.

#### Bed Table

FD: {Bed\_Id, Patient\_Id --> occupied}

This table is in BCNF because all attributes are dependent on the primary key (composite key) consisting of Bed\_Id and Patient\_Id.

#### Date Admitted Table

FD: {Patient\_Id, LogisticsDate --> DateReleased}

This table is in BCNF because all attributes are dependent on the primary key (composite key) consisting of Patient\_Id and LogisticsDate.

#### Patients Table

FD: {Patient\_Id --> FullName, Age Gender, SIN, Health\_CN, Symptoms}

This table is in BCNF because all attributes are dependent on the primary (candidate) key Patient\_Id.

#### Medicine prescription Table

FD: {Medicine\_ID, Patient\_Id --> Dosage, Consumption\_Cycle}

This table is in BCNF because all attributes are dependent on the primary key (composite key) consisting of Medicine\_ID and Patient\_Id.

#### Emergency Contact Table

FD: {Patient\_ID --> FullName, Relation, Tel\_Number, Address}

This table is in BCNF because all attributes are dependent on the primary (candidate) key Patient\_ID.

## Queries/Views and Relational Algebra

### Simple Queries

**Figure 7a and 7b** show the code snippet and the results of the query 1 to list the details of young patients who are younger than 21 years old and their emergency contacts.

**Figure 7a.** SQL code snippet for a simple query 1.

```
SELECT p.PatientID, p.FullName, p.Symptoms, ec.fullname AS Emergency_Contact_Name, ec.relation, ec.tel_number
FROM Patients p, Emergency_Contact ec
WHERE p.Age <= 21 AND p.PatientID = ec.PatientID;
```

**Figure 7b.** Table result for simple query 1.

	PATIENTID	FULLNAME	SYMPTOMS	EMERGENCY_CONTACT_NAME	RELATION	TEL_NUMBER
1	6	Brook Lopez	Flu	Rafael Lopez	Brother	3123244500
2	5	Monica Mercedes	Flu	Melina Mercedes	Mother	415900213
3	1	Dominique Rodriguez	Nausea	Dwayne Rodriguez	Father	516321021

This query will use the theta join operator where theta would be the condition that age is less than 21. This is useful because it joins only the tuples where PATIENTID is equal in both PATIENTID and EMERGENCY\_CONTACT tables where the patients age is less than 21. Then the projection operator is used to specifically output the columns. Therefore, its relational algebra is as follows:

$\pi_{\text{PATIENTID, FULLNAME, SYMPTOMS, EMERGENCY\_CONTACT\_NAME, RELATION, TEL\_NUMBER}} (\text{PATIENTS} \bowtie_{\text{AGE} < 21} \text{EMERGENCY\_CONTACT})$

**Figure 8a and 8b** show the code snippet and the results of the query 2 to list the details of the patients and the medicine prescription they are taking.

**Figure 8a.** SQL code snippet for simple query 2.

```
SELECT p.fullname, p.patientId, m.name, m.medicine_ID, m.description, mp.dosage, mp.consumption_cycle
FROM Patients p, Medicine_prescription mp, Medicine m
WHERE p.PatientID = mp.PatientID AND mp.Medicine_ID = m.Medicine_ID
```

**Figure 8b.** Table result for simple query 2.

	FULLNAME	PATIENTID	NAME	MEDICINE_ID	DESCRIPTION	DOSAGE	CONSUMPTION_CYCLE
1	Anderson Silva	4	Ibuprofen	1	Painkiller	3 times a day	3 months
2	Dominique Rodriguez	1	Zofran	2	Anti-nausea	1 times a day	1 week
3	Merci Tefioux	3	Zanamivir	3	Flu medicine	2 times a day	1 month

This query will again take advantage of the natural join operator. This is so that combining 3 tables together will eliminate tuples that do not have the same values to enforce equality. This means that when using the natural join operator on PATIENTS, MEDICINE AND MEDICINE\_PRESCRIPTION, it is the same as the WHERE statement above. From there, there is only the need to project the columns that is specified. As such the relational algebra for this query is:

$\pi_{\text{FULLNAME, PATIENTID, NAME, MEDICINE\_ID, DESCRIPTION, DOSAGE, CONSUMPTION\_CYCLE}} (\text{PATIENTS} \bowtie (\text{MEDICINE} \bowtie \text{MEDICINE\_PRESCRIPTION}))$

**Figure 9a and 9b** show the code snippet and the results of the query 3 to list any employees who have an account and its details.

**Figure 9a.** SQL code snippet for simple query 3.

```
SELECT DISTINCT emp.EmployeeType, emp.fullname, acc.username, acc.password, acc.EMP_privilege
FROM Employee emp, Account acc
WHERE acc.employeeID = emp.employeeID
ORDER BY emp_privilege DESC;
```

**Figure 9b.** Table result for simple query 3.

	EMPLOYEETYPE	FULLNAME	USERNAME	PASSWORD	EMP_PRIVILEGE
1	doctor	Ahmed Khan	akhan	admin	B
2	doctor	Jules Gammad	jgammad	admin	B
3	operations staff	Christopher Jielo	admin	cusadmin	A
4	operations staff	Melani Tot	admin2	cusadmin2	A

For this query's relational algebra, the natural join operator is used again. From there, the projection operator is used to specify the columns shown. Its relational algebra then follows that:

$$\Pi_{\text{EMPLOYEETYPE, FULLNAME, USERNAME, PASSWORD, EMP\_PRIVILEGE}} (\text{EMPLOYEE} \bowtie \text{ACCOUNT})$$

**Figure 10a and 10b** show the code snippet and the results of the query 4 to list all the details of the different types of medicines available.

**Figure 10a.** SQL code snippet for simple query 4.

```
SELECT DISTINCT *
FROM Medicine;
```

**Figure 10b.** Table result for simple query 4.

	MEDICINE_ID	NAME	DESCRIPTION	INVENTORY
1	1	Ibuprofen	Painkiller	200
2	2	Zofran	Anti-nausea	50
3	3	Zanamivir	Flu medicine	2000

Since the entire table is being outputted, this query will use the projection operator over the entire attribute space. Its relational algebra then follows:

$$\Pi_{\text{MEDICINE\_ID, NAME, DESCRIPTION, INVENTORY}}$$

## Advanced Queries

**Figures 11a and 11b** show the code snippet and the results of query 1 which uses the UNION operator. This query lists all the doctors and nurses' details.

**Figure 11a.** SQL code snippet for advanced query 1.

```
SELECT e.fullName, e.employeeID, e.employeeType, n.specialization
FROM Nurses n, Employee e
WHERE e.EmployeeID = n.EmployeeID
UNION
SELECT e.fullName, e.employeeID, e.employeeType, d.specialization
FROM Doctors d, Employee e
WHERE e.EmployeeID = d.EmployeeID
ORDER BY fullName ASC;
```

**Figure 11b.** Table result for advanced query 1.

	FULLNAME	EMPLOYEEID	EMPLOYEE TYPE	SPECIALIZATION
1	Ahmed Khan	1	doctor	General Surgery
2	Constantino Mello	6	nurses	Nursing Assistant
3	Jane Doe	3	nurses	Registered Nurse
4	John Doe	4	nurses	Medical-surgical Nurse
5	Jules Gammad	2	doctor	Neurosurgery
6	Tatiana Mesau	5	nurses	Intensive Care Unit Registered Nurse

For this advanced query, a combination of natural join and union operator will be used to produce the relational algebra. The natural join will be used between NURSES and EMPLOYEE and between DOCTORS and EMPLOYEE so that 2 tables are produced where one table is employees who are doctors and the other, nurses. The UNION operator will then bring the two tables together to form one table. The relational algebra is as follows:

$$[\pi_{\text{FULLNAME, EMPLOYEEID, EMPLOYEE TYPE, SPECIALIZATION}} (\text{NURSES} \bowtie \text{EMPLOYEE})] \cup [\pi_{\text{FULLNAME, EMPLOYEEID, EMPLOYEE TYPE, SPECIALIZATION}} (\text{DOCTORS} \bowtie \text{EMPLOYEE})]$$

**Figures 12a and 12b** show the code snippet and the results of query 2 which uses the MINUS operator. This query lists all the details of the patients who does not have nausea as a symptom.

**Figure 12a.** SQL code snippet for advanced query 2.

```
SELECT *
FROM Patients
MINUS
SELECT *
FROM PATIENTS
WHERE Symptoms = 'Nausea'
```

**Figure 12b.** Table result for advanced query 2.

	PATIENTID	FULLNAME	AGE	GENDER	SIN	HEALTH_CN	SYMPTOMS
1	3	Merci Tefioux	26	M	12332	18542	Flu
2	4	Anderson Silva	29	M	13222	53902	Internal Bleeding
3	5	Monica Mercedes	16	F	12312	98012	Flu
4	6	Brook Lopez	20	M	15512	23012	Flu

For advanced query 2, the difference operator is used over PATIENTS such that the resulting table is a subset of PATIENTS. Another, simpler way, of getting to the same result is using the selection operator over PATIENTS and selecting the condition where symptoms is not equal to nausea. In other words, the relational algebra can be expressed as:

$$(\Pi_{\text{PATIENTID, FULLNAME, AGE, GENDER, SIN, HEALTH\_CN, SYMPTOMS}}) - (\Pi_{\text{PATIENTID, FULLNAME, AGE, GENDER, SIN, HEALTH\_CN, SYMPTOMS}} (\sigma_{\text{SYMPTOMS} \neq \text{'NAUSEA'}} (\text{PATIENTS})))$$

Or

$$\sigma_{\text{SYMPTOMS} \neq \text{'NAUSEA'}} (\text{PATIENTS})$$

**Figures 13a and 13b** show the code snippet and the results of query 3 which uses multiple UNION operator and the GROUP BY operator to sort the table. This query lists all the details of all the employees.

**Figure 13a.** SQL code snippet for advanced query 3.

```
SELECT e.fullName, e.employeeID, e.employeeType, n.specialization
FROM Nurses n, Employee e
WHERE e.EmployeeID = n.EmployeeID
UNION
SELECT e.fullName, e.employeeID, e.employeeType, d.specialization
FROM Doctors d, Employee e
WHERE e.EmployeeID = d.EmployeeID
UNION
SELECT e.fullName, e.employeeID, e.employeeType, os.job_title
FROM Operations_Staff os, Employee e
WHERE e.EmployeeID = os.EmployeeID
UNION
SELECT e.fullName, e.employeeID, e.employeeType, le.profession
FROM Law_enforcement le, Employee e
WHERE e.EmployeeID = le.EmployeeID
ORDER BY employeeType ASC;
```

**Figure 13b.** Table result for advanced query 3.

	FULLNAME	EMPLOYEEID	EMPLOYEE TYPE	SPECIALIZATION
1	Ahmed Khan	1	doctor	General Surgery
2	Jules Gammad	2	doctor	Neurosurgery
3	Ian Refut	10	law enforcement	Fireman
4	Teo Mani	9	law enforcement	Police
5	Constantino Mello	6	nurses	Nursing Assistant
6	Jane Doe	3	nurses	Registered Nurse
7	John Doe	4	nurses	Medical-surgical Nurse
8	Tatiana Mesau	5	nurses	Intensive Care Unit Registered Nurse
9	Christopher Jielo	7	operations staff	Front-desk Specialist
10	Melani Tot	8	operations staff	Operations Manager

$[\pi_{\text{FULLNAME, EMPLOYEEID, EMPLOYEE TYPE, SPECIALIZATION}} (\text{NURSES} \bowtie \text{EMPLOYEE})] \cup [\pi_{\text{FULLNAME, EMPLOYEEID, EMPLOYEE TYPE, SPECIALIZATION}} (\text{DOCTORS} \bowtie \text{EMPLOYEE})] \cup [\pi_{\text{FULLNAME, EMPLOYEEID, EMPLOYEE TYPE, SPECIALIZATION}} (\text{LAW\_ENFORCEMENT} \bowtie \text{EMPLOYEE})] \cup [\pi_{\text{FULLNAME, EMPLOYEEID, EMPLOYEE TYPE, SPECIALIZATION}} (\text{OPERATIONS\_STAFF} \bowtie \text{EMPLOYEE})]$

For the relational algebra of this advanced query, a combination multiple natural joins and unions will be used to create the table. The union joins will be used to create tables of employee types and the union operator will merge all the tables into one. As such, the relational algebra is as follows:

**Figures 14a and 14b** show the code snippet and the results of query 4 which performs a subtraction operator in the SELECT statement and the GROUP BY operator to sort the table. This query lists all the patients who have used beds and their details.

**Figure 14a.** SQL code snippet for advanced query 4.

```
SELECT DISTINCT p.fullName, b.Bed_ID, b.Occupied, l.LogisticsDate, da.date_released, (da.date_released - l.logisticsDate) AS NumDays
FROM Bed b, Patients p, Logistics l, Date_admitted da
WHERE p.patientID = b.patientID AND p.patientID = da.patientID AND da.LOGISTICSDATE = l.LOGISTICSDATE
ORDER BY occupied ASC;
```

**Figure 14b.** Table result for advanced query 4.

	FULLNAME	BED_ID	OCCUPIED	LOGISTICSDATE	DATE_RELEASED	NUMDAYS
1	Dominique Rodriguez	1	N	20-10-26	20-11-26	31
2	Allyson Teofima	2	N	20-10-28	20-11-15	18
3	Monica Mercedes	5	N	20-11-10	20-12-12	32
4	Merci Tefioux	3	Y	20-10-28	(null)	(null)
5	Anderson Silva	4	Y	20-10-28	(null)	(null)
6	Brook Lopez	6	Y	20-11-10	(null)	(null)

The last advanced query is the most complex relational algebra to derive because the last column, NUMDAYS, is a derived attribute. To do this, two tables must be created and then joined. The first table will be using relational algebra to get the first 5 columns above. Working from the inside-out, the second



table is formed by first using the uniform join operators to get the appropriate tuples. Next, a projection operator over that relation is applied to only get DATE\_RELEASED and LOGISTICSDATE column. Then, a projection operator is used to project a column where the absolute value of the difference between LOGISTICSDATE and DATE\_RELEASED is taken to find the days lapsed in between. A rename operator is then used to rename this column appropriately. To join the two tables together, a cross-product is used between the table because they have different degrees. Because of the nature of cross product having tuples (M x N) long, a final select statement is used to make sure that there are no repetitions in the tuples. As such, the relational algebra for this query is:

$\sigma_{\text{PATIENTS.PATIENTID} = \text{BED.PATIENTID AND PATIENT.PATIENTID} = \text{DATE\_ADMITTED.PATIENTID AND DATE\_ADMITTED.LOGISTICSDATE} = \text{LOGISTICS.LOGISTICSDATE}} [(\pi_{\text{FULLNAME, BED\_ID, OCCUPIED, LOGISTICSDATE, DATE\_RELEASED}} ((\text{PATIENTS} \bowtie \text{BED}) \bowtie (\text{LOGISTICSDATE} \bowtie \text{DATE\_ADMITTED}))) \times (\rho_{\text{NUMDAYS}} (\pi_{|\text{LOGISTICSDATE} - \text{DATE\_ADMITTED}|} ((\text{PATIENTS} \bowtie \text{BED}) \bowtie (\text{LOGISTICSDATE} \bowtie \text{DATE\_ADMITTED}))))]$

## Views

**Figures 15a and 15b** show the code snippet and the results of a sample view. This view uses the COUNT operator and considers the total number of employees and the total number of employees in each type of employee.

**Figure 15a.** SQL code snippet for the sample view.

```
CREATE VIEW checkStaff (Total_Staff,Doctors, Nurses, Operation_Staff, Law_Enforcement) AS
(SELECT DISTINCT (SELECT COUNT(*) from Employee), (SELECT COUNT(*) from Doctors), (SELECT COUNT(*) from Nurses),
(SELECT COUNT(*) from Operations_Staff), (SELECT COUNT(*) from Operations_Staff)
FROM DOCTORS, NURSES, OPERATIONS_STAFF, LAW_ENFORCEMENT);

SELECT * FROM checkStaff
```

**Figure 15b.** Table result for the sample view.

	TOTAL_STAFF	DOCTORS	NURSES	OPERATION_STAFF	LAW_ENFORCEMENT
1	10	2	4	2	2

For the relational algebra of this query, the aggregate functions are used. As such, the relational algebra will consist of using the union operator and the aggregate function operator to derive the equation. As such its relational algebra is as follows:

$(\text{Total\_Staff } F_{\text{Count}(*)} (\text{Employee})) \cup (\text{Doctors } F_{\text{Count}(*)} (\text{Doctors})) \cup (\text{Nurses } F_{\text{Count}(*)} (\text{Nurses})) \cup (\text{Operations\_Staff } F_{\text{Count}(*)} (\text{Operations\_Staff})) \cup (\text{Law\_enforcement } F_{\text{Count}(*)} (\text{Law\_Enforcement}))$

## Unix Implementation

This implementation is to connect through the school's oracle database using Unix Shell menu commands. Features like creating, dropping, and populating tables were implemented with Shell scripts. In addition, basic and advanced queries were also added as an option to choose. **Figure 16a and 16b** show the code snippets of some of the Shell script which is shown below:

Figure 16a. Code snippet for main.sh

```

1  #!/bin/sh
2  MainMenu()
3  {
4      while [ "$CHOICE" != "START" ]
5      do
6          clear
7          echo "=====
8          echo "|                Oracle All Inclusive Tool                |"
9          echo "|      Main Menu -Select Desired Operation(s):              |"
10         echo "|      <CTRL-Z Anytime to Enter Interactive CMD Prompt>      |"
11         echo "=====
12         echo " $IS_SELECTEDM M)  View Manual"
13         echo " "
14         echo " $IS_SELECTED1 1)  Drop Tables"
15         echo " $IS_SELECTED2 2)  Create Tables"
16         echo " $IS_SELECTED3 3)  Populate Tables"
17         echo " $IS_SELECTED4 4)  Query Tables"
18         echo " "
19         echo " $IS_SELECTEDX X)  Force/Stop/Kill Oracle DB"
20         echo " "
21         echo " $IS_SELECTEDE E)  End/Exit"
22         echo "Choose: "
23         read CHOICE
24
25         if [ "$CHOICE" == "0" ]
26         then
27             echo "Nothing Here"
28         elif [ "$CHOICE" == "1" ]
29         then
30             bash drop_tables.sh
31             read CHOICE
32             Pause
33         elif [ "$CHOICE" == "2" ]
34         then
35             bash create_tables.sh
36             read CHOICE
37             Pause
38         elif [ "$CHOICE" == "3" ]
39         then
40             bash populate_tables.sh
41             read CHOICE
42             Pause
43         elif [ "$CHOICE" == "4" ]
44         then
45             clear

```

```

46 echo "=====
47 echo "|           Oracle All Inclusive Tool           |"
48 echo "|   Query Menu -Select Desired Operation(s):   |"
49 echo "|   <CTRL-Z Anytime to Enter Interactive CMD Prompt>   |"
50 echo "=====
51 echo " $IS_SELECTED1 1)  Advanced Query 1"
52 echo " $IS_SELECTED2 2)  Advanced Query 2"
53 echo " $IS_SELECTED3 3)  Advanced Query 3"
54 echo " $IS_SELECTED4 4)  Advanced Query 4"
55 echo " $IS_SELECTED4 5)  Advanced Query 5"
56 echo " "
57 echo " $IS_SELECTEDX X)  Force/Stop/Kill Oracle DB"
58 echo " "
59 echo " $IS_SELECTEDE E)  End/Exit"
60 echo "Choose: "
61 read CHOICE
62
63 if [ "$CHOICE" == "0" ]
64 then
65 echo "Nothing Here"
66 elif [ "$CHOICE" == "1" ]
67 then
68     bash q1.sh
69     read CHOICE
70 elif [ "$CHOICE" == "2" ]
71 then
72     bash q2.sh
73     read CHOICE
74 elif [ "$CHOICE" == "3" ]
75 then
76     bash q3.sh
77     read CHOICE
78 elif [ "$CHOICE" == "4" ]
79 then
80     bash q4.sh
81     read CHOICE
82 elif [ "$CHOICE" == "5" ]
83 then
84     bash q5.sh
85     read CHOICE
86 elif [ "$CHOICE" == "E" ]
87 then
88     exit
89 fi
90 Pause
91 elif [ "$CHOICE" == "E" ]
92 then
93     exit
94 fi
95
96 done
97 }
98 #--COMMENTS BLOCK--

```

Figure 16b. Code snippet for drop\_tables.sh

```

1  #!/bin/sh
2  #export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
3  sqlplus64 "Xx/Xx@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF
4  DROP TABLE Employee CASCADE CONSTRAINTS;
5  DROP TABLE account CASCADE CONSTRAINTS;
6  DROP TABLE schedule CASCADE CONSTRAINTS;
7  DROP TABLE doctors CASCADE CONSTRAINTS;
8  DROP TABLE nurses CASCADE CONSTRAINTS;
9  DROP TABLE Law_Enforcement CASCADE CONSTRAINTS;
10 DROP TABLE Operations_Staff CASCADE CONSTRAINTS;
11 DROP TABLE Medicine_Prescription CASCADE CONSTRAINTS;
12 DROP TABLE Logistics CASCADE CONSTRAINTS;
13 DROP TABLE Patients CASCADE CONSTRAINTS;
14 DROP TABLE Bed CASCADE CONSTRAINTS;
15 DROP TABLE Medicine CASCADE CONSTRAINTS;
16 DROP TABLE Emergency_Contact CASCADE CONSTRAINTS;
17 exit;
18 EOF

```

The resulting UI for the Unix Shell implementation is showing in **Figure 17a and 17b**. This is important because it is important that people who must use the database do not need a technical background to do work. This concept is explained further in the following GUI implementation section.

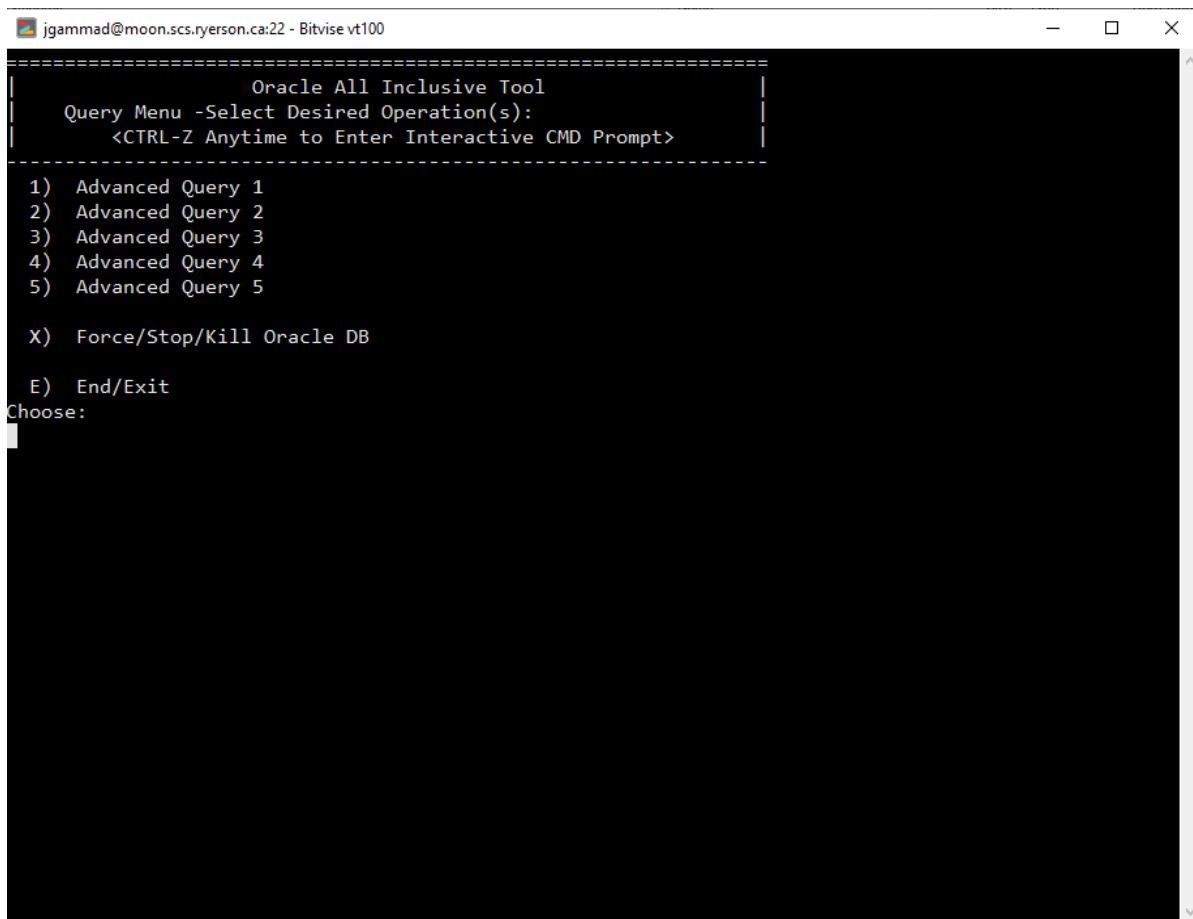
Figure 17a. Unix Shell Menu for the Emergency Medical Response Record Management System.

```

jgammad@moon.scs.ryerson.ca:22 - Bitvise vt100
===== Oracle All Inclusive Tool |
Main Menu -Select Desired Operation(s): |
<CTRL-Z Anytime to Enter Interactive CMD Prompt> |
-----
M) View Manual
1) Drop Tables
2) Create Tables
3) Populate Tables
4) Query Tables
X) Force/Stop/Kill Oracle DB
E) End/Exit
Choose: 

```

**Figure 17b.** Query Menu for the Emergency Medical Response Record Management System.



```

jgammad@moon.scs.ryerson.ca:22 - Bitvise vt100
=====
Oracle All Inclusive Tool
Query Menu -Select Desired Operation(s):
<CTRL-Z Anytime to Enter Interactive CMD Prompt>
=====
1) Advanced Query 1
2) Advanced Query 2
3) Advanced Query 3
4) Advanced Query 4
5) Advanced Query 5

X) Force/Stop/Kill Oracle DB

E) End/Exit
Choose:

```

To prevent the report from getting too long, please refer to the Shell script files attached in the submission for the rest of the code implementation.

## Graphical User Interface Implementation

All databases are useless without a proper bandwidth of interaction between its intended user and the database. For example, it would be highly inefficient and difficult to both hire and train a bank-teller to use a database and do their job at the same time. As such a graphical user interface (GUI) is a core aspect in being able to effectively use and work with databases.

### Set-up

The graphical implementation of the Emergency Response Record Management System was made using Python and Python libraries connected to the school database. To run the project, several steps must first be taken. The first step is to download Python as it is the language that the application is primarily using. The second step is to install the necessary applications and packages to make the program work. To edit the program, an IDE is needed that can run Python. Certain packages must also be installed as shown in **Figure 18**.

**Figure 18.** Requirements of installation of packages using pip install.

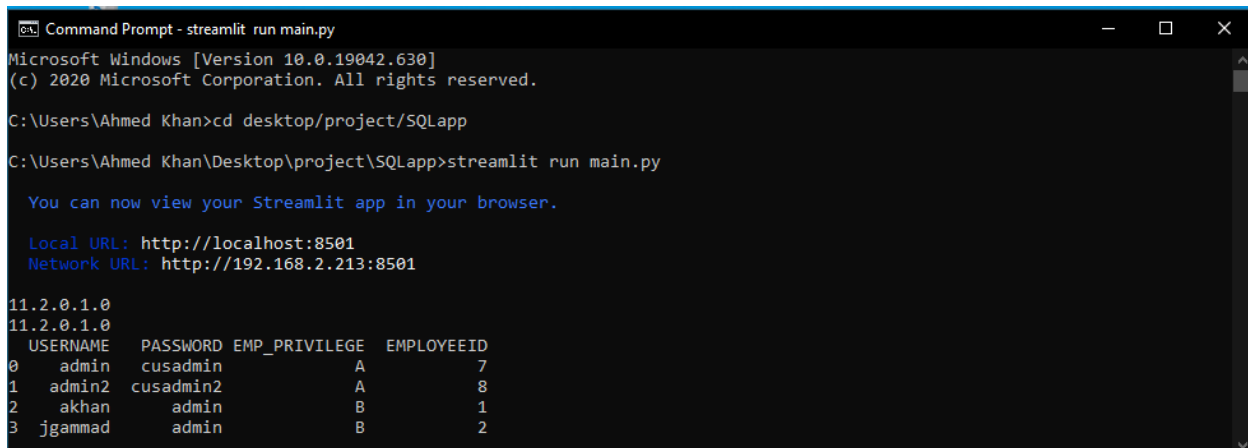
```

1  #REQUIREMENTS
2  #Run commands to install and initialize dependencies
3  #pip install streamlit
4  #streamlit hello
5  #pip install python-dotenv
6  #pip install pandas
7  #pip install cx-Oracle
8

```

After installing the requirements, the program must be running from the command prompt by following the code in **Figure 19**. After typing the code, the program should appear in the browser.

**Figure 19.** Code snippet of set-up in the command prompt.



```

C:\Users\Ahmed Khan>cd desktop/project/SQLapp
C:\Users\Ahmed Khan\Desktop\project\SQLapp>streamlit run main.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.2.213:8501

11.2.0.1.0
11.2.0.1.0
  USERNAME  PASSWORD EMP_PRIVILEGE  EMPLOYEEID
0      admin   cusadmin           A             7
1    admin2  cusadmin2          A             8
2      akhan    admin            B             1
3    jgammad   admin            B             2

```


The following GUI represents a real-life imitation of what an Emergency Medical Response Record Management System could look like. Before logging in, there is an option to create, delete and populate tables as per requirement in the project description. Otherwise, choosing the GUI DEMO option will initialize the login portal as shown in **Figure 20**. The username and password correspond to the accounts specified in the ACCOUNTS table. Doctors and nurses who have accounts have different portals than operations staff who have accounts. As shown in **Figure 21**, doctors can access and interact with the appropriate tables in the database. **Figure 22a**, **Figure 22b**, **Figure 22c** and **Figure 22d** show some of the queries that is possible to view in a doctor/nurse account. Similarly, operations staff have different access and interact with different tables in the database as show in **Figure 23**. Moreover, **Figure 24a**, **Figure 24b**, **Figure 24c** and **Figure 24d** show some of the queries that is possible to view under an operations staff account. In addition, all tables update automatically when adding or deleting tuples as shown in **Figure 25** and **Figure 26** respectively.

**Figure 20.** Login portal of the web application.

**Emergency Management System**

Select Demo Function

GUI DEMO



**Login**

Username:

Password:

☐ Log In

**Figure 21.** Doctor/nurse account portal of the web application.

Select Table to display

Bed Information

Bed Information

Date of Admittance

Emergency Contacts

Medicine Inventory

Medicine Prescriptions

Patients

Patient Prescriptions

	BED_ID	PATIENTID	OCCUPIED
0	1	1	N
1	2	2	N
2	3	3	Y
3	4	4	Y
4	5	5	N
5	6	6	Y

**Figure 22a.** Patient prescription query for doctors/nurses.

Select Table to display

Patient Prescriptions

☒ Enable Editing

Select

☐ Add Information

☒ Delete Information

Submit

	FULL NAME	PATIENTID	MEDICINE NAME	MEDICINE ID	DESCRIPTION	USAGE	CONSUMPTION CYCLE
0	Anderson Silva	4	Ibuprofen	1	Painkiller	3 times a day	3 months
1	Dominique Rodriguez	1	Zofran	2	Anti-nausea	1 times a day	1 week
2	Merci Tefioux	3	Zanamivir	3	Flu medicine	2 times a day	1 month

**Figure 22b.** Patients without nausea query for doctors/nurses.

Select Table to display

Patients Without Nausea

☒ Enable Editing

Select

☐ Add Information

☒ Delete Information

Submit

	PATIENT ID	FULLNAME	AGE	GENDER	SIN	HEALTH CARD	SYMPTOMS
0	3	Merci Tefioux	26	M	12332	18542	Flu
1	4	Anderson Silva	29	M	13222	53902	Internal Bleeding
2	5	Monica Mercedes	16	F	12312	98012	Flu
3	6	Brook Lopez	20	M	15512	23012	Flu



**Figure 22c.** Medicine Information query for doctors/nurses.

Select Table to display

Medicine Information

☒ Enable Editing

Select

☐ Add Information

☒ Delete Information

Submit

	MEDICINE ID	MEDICINE NAME	DESCRIPTION	INVENTORY
0	1	Ibuprofen	Painkiller	200
1	2	Zofran	Anti-nausea	50
2	3	Zanamivir	Flu medicine	2000

**Figure 22d.** Young patients query for doctors/nurses.

Select Table to display

Young Patients

☒ Enable Editing

Select

☐ Add Information

☒ Delete Information

Submit

	PATIENT ID	FULL NAME	SYMPTOMS	EMERGENCY CONTACT	RELATION	TELEPHONE NUMBER
0	6	Brook Lopez	Flu	Rafael Lopez	Brother	3123244500
1	5	Monica Mercedes	Flu	Melina Mercedes	Mother	415900213
2	1	Dominique Rodriguez	Nausea	Dwayne Rodriguez	Father	516321021

**Figure 23.** Operations staff account portal of the application.

Select Table to display

Date of Admittance

☒ Enable Editing

Select

☐ Add Information

☒ Delete Information

Patient Id

1

Submit

	DATE_RELEASED	LOGISTICSDATE	PATIENTID
0	Nov 26, 2020	Oct 26, 2020	1
1	Nov 15, 2020	Oct 28, 2020	2
2	Sep 20, 1677 6:55 PM	Oct 28, 2020	3
3	Sep 20, 1677 6:55 PM	Oct 28, 2020	4
4	Dec 12, 2020	Nov 10, 2020	5

**Figure 24a.** View Doctors and Nurses query for operations staff.

## Login

Username:

admin

Password:

.....

☒ Log In

Select Table to display

View Doctors and Nurses

☐ Enable Editing

	FULLNAME	EMPLOYEE ID	EMPLOYEE TYPE	SPECIALIZATION
0	Ahmed Khan	1	doctor	General Surgery
1	Constantino Mello	6	nurses	Nursing Assistant
2	Jane Doe	3	nurses	Registered Nurse
3	John Doe	4	nurses	Medical-surgical Nurse
4	Jules Gammad	2	doctor	Neurosurgery
5	Tatiana Mesau	5	nurses	Intensive Care Unit Registered Nurse

**Figure 24b.** Occupied Beds query for the operations staff.

Select Table to display

Occupied Beds ▼

☐ Enable Editing

	FULL NAME	BED ID	OCCUPIED	LOGISTIC DATE	DATE RELEASED	DAYS ADMITTED
0	Dominique Rodriguez	1	N	Oct 26, 2020	Nov 26, 2020	31
1	Allyson Teofima	2	N	Oct 28, 2020	Nov 15, 2020	18
2	Monica Mercedes	5	N	Nov 10, 2020	Dec 12, 2020	32
3	Merci Tefioux	3	Y	Oct 28, 2020	Sep 20, 1677 6:55 PM	NaN
4	Anderson Silva	4	Y	Oct 28, 2020	Sep 20, 1677 6:55 PM	NaN

**Figure 24c.** View employee accounts query for the operations staff.

Select Table to display

Employee Accounts ▼

☐ Enable Editing

	EMPLOYEE TYPE	FULL NAME	USERNAME	PASSWORD	PRIVILEGE
0	doctor	Ahmed Khan	akhan	admin	B
1	doctor	Jules Gammad	jgammad	admin	B
2	operations staff	Christopher Jielo	admin	cusadmin	A
3	operations staff	Melani Tot	admin2	cusadmin2	A

**Figure 24d.** Employee table query for the operations staff.

Select Table to display

View Employees ▼

☐ Enable Editing

	FULL NAME	EMPLOYEE ID	EMPLOYEE TYPE	SPECIALIZATION
0	Ahmed Khan	1	doctor	General Surgery
1	Jules Gammad	2	doctor	Neurosurgery
2	Ian Refut	10	law enforcement	Fireman
3	Constantino Mello	6	nurses	Nursing Assistant
4	Jane Doe	3	nurses	Registered Nurse
5	John Doe	4	nurses	Medical-surgical Nurse
6	Tatiana Mesau	5	nurses	Intensive Care Unit Registered Nurse
7	Christopher Jielo	7	operations staff	Front-desk Specialist
8	Melani Tot	8	operations staff	Operations Manager

The application uses cx.Oracle, Pandas and Streamlit libraries interconnectedly in order to produce a coherent and responsive webpage. The cx.Oracle library is in charge of the connection between the application and the school database. It also is in charge of sending query commands via command prompt and outputting the data. The Pandas library formats the data into tables that can be worked with. Finally, the Streamlit library takes the formatted Pandas table and converts it into HTML output. The Streamlit library is mainly in charge of hosting the webpage to local host and outputting any HTML content.

**Note: Please refer to Appendix A for the source code of the application.**

## Conclusion

The overall experience of this project was enlightening. The project has changed our perspective moving forward with respect to any software application in our day-to-day basis. The amount of effort and thought poured into the databases to keep trillions of data properly handled and usable is an extraordinary opportunity to peek into an area of software that we have not previously thought about. From database system architecture, normalization, database concepts and terms and ER diagrams the design experience have been phenomenal.

## Appendix A

### #REQUIREMENTS

#Run commands to install and initialize dependencies

#pip install streamlit

#streamlit hello

#pip install python-dotenv

#pip install pandas

#pip install cx-Oracle

import streamlit as st

import cx\_Oracle

import pandas as pd

from PIL import Image

import getpass

import os

from dotenv import load\_dotenv

load\_dotenv()

ORCL\_USER = os.getenv('ORCL\_USER')

ORCL\_PASSWD = os.getenv('ORCL\_PASSWD')

dsnStr = cx\_Oracle.makedsn("oracle.scs.ryerson.ca", "1521", "orcl")

conn = cx\_Oracle.connect(ORCL\_USER,ORCL\_PASSWD, dsn=dsnStr)

print (conn.version)

loggedin= False

st.title("Emergency Management System")

cur = conn.cursor()

def employee():

    cur.execute("""

    select \* from EMPLOYEE

    """)

    pull = cur.fetchall()

    Employeeedf = pd.DataFrame(pull, columns = ['FULLNAME','SIN','EMPLOYEEID','EMPLOYEEETYPE'])

    print(Employeeedf)

    return(Employeeedf)

def logic():

    cur.execute("""

    select \* from LOGISTICS

    """)

    pull = cur.fetchall()

    Logicdf = pd.DataFrame(pull, columns = ['NUMPATIENT', 'NUMDOCTOR', 'NUMNURSE', 'NUMBEDS',  
'LOGISTICSDATE'])

    print(Logicdf)

    return (Logicdf)

```

def account():
    cur.execute("""
    select * from ACCOUNT
    """)
    pull = cur.fetchall()

    Accountdf = pd.DataFrame(pull, columns = ['USERNAME', 'PASSWORD', 'EMP_PRIVILEGE', 'EMPLOYEEID'])
    print(Accountdf)
    return(Accountdf)

def bed():
    cur.execute("""
    select * from BED
    """)
    pull = cur.fetchall()

    Beddf = pd.DataFrame(pull, columns = ['BED_ID', 'PATIENTID', 'OCCUPIED'])
    print(Beddf)
    return(Beddf)

def date_admit():
    cur.execute("""
    select * from DATE_ADMITTED
    """)
    pull = cur.fetchall()

    Date_admitdf = pd.DataFrame(pull, columns = ['DATE_RELEASED', 'LOGISTICSDATE', 'PATIENTID'])
    print(Date_admitdf)
    return(Date_admitdf)

def doctor():
    cur.execute("""
    select * from DOCTORS
    """)
    pull = cur.fetchall()

    Doctorsdf = pd.DataFrame(pull, columns = ['SPECIALIZATION', 'EMPLOYEEID'])
    print(Doctorsdf)
    return (Doctorsdf)

def emergency():
    cur.execute("""
    select * from EMERGENCY_CONTACT
    """)
    pull = cur.fetchall()

    Emergencydf = pd.DataFrame(pull, columns = ['FULLNAME', 'RELATION', 'TEL_NUMBER', 'ADDRESS',
    'PATIENTID'])
    print(Emergencydf)
    return(Emergencydf)

def law():
    cur.execute("""

```

```

select * from LAW_ENFORCEMENT
"""
)
pull = cur.fetchall()

Lawdf = pd.DataFrame(pull, columns = ['PROFESSION', 'EMPLOYEEID'])
print(Lawdf)
return(Lawdf)

def medicine():
    cur.execute("""
select * from MEDICINE
"""
)
    pull = cur.fetchall()

    Medicinedf = pd.DataFrame(pull, columns = ['MEDICINE_ID', 'NAME', 'DESCRIPTION', 'INVENTORY'])
    print(Medicinedf)
    return(Medicinedf)

def medicinepre():
    cur.execute("""
select * from MEDICINE_PRESCRIPTION
"""
)
    pull = cur.fetchall()

    MedicinePredf = pd.DataFrame(pull, columns = ['MEDICINE_ID', 'PATIENTID', 'DOSAGE',
'CONSUMPTION_CYCLE'])
    print(MedicinePredf)
    return(MedicinePredf)

def nurse():
    cur.execute("""
select * from NURSES
"""
)
    pull = cur.fetchall()

    Nursesdf = pd.DataFrame(pull, columns = ['EMPLOYEEID', 'SPECIALIZATION'])
    print(Nursesdf)
    return(Nursesdf)

def op_staff():
    cur.execute("""
select * from OPERATIONS_STAFF
"""
)
    pull = cur.fetchall()

    OpStaffdf = pd.DataFrame(pull, columns = ['JOB_TITLE', 'EMPLOYEEID'])
    print(OpStaffdf)
    return(OpStaffdf)

def patients():
    cur.execute("""
select * from PATIENTS
"""
)

```

```

pull = cur.fetchall()

Patientsdf = pd.DataFrame(pull, columns = ['PATIENTID', 'FULLNAME', 'AGE', 'GENDER', 'SIN', 'HEALTH_CN',
'SYMPTOMS'])
print(Patientsdf)
return(Patientsdf)

def schedule():
    cur.execute("""
    select * from SCHEDULE
    """)
    pull = cur.fetchall()

    Scheduledf = pd.DataFrame(pull, columns = ['WORKDATE', 'CLOCK_IN', 'CLOCK_OUT', 'EMPLOYEEID'])
    print(Scheduledf)
    return(Scheduledf)

def Query1():
    cur.execute("""
    SELECT e.fullName, e.employeeID, e.employeeType, n.specialization
    FROM Nurses n, Employee e
    WHERE e.EmployeeID = n.EmployeeID
    UNION
    SELECT e.fullName, e.employeeID, e.employeeType, d.specialization
    FROM Doctors d, Employee e
    WHERE e.EmployeeID = d.EmployeeID
    ORDER BY fullName ASC
    """)
    pull = cur.fetchall()

    query1 = pd.DataFrame(pull, columns = ['FULLNAME', 'EMPLOYEE ID', 'EMPLOYEE TYPE', 'SPECIALIZATION'])
    print(query1)
    return(query1)

def Query2():
    cur.execute("""
    SELECT p.fullName, p.patientId, m.name, m.medicine_ID, m.description, mp.dosage, mp.consumption_cycle
    FROM Patients p, Medicine_prescription mp, Medicine m
    WHERE p.PatientID = mp.PatientID AND mp.Medicine_ID = m.Medicine_ID
    """)
    pull = cur.fetchall()

    query2 = pd.DataFrame(pull, columns = ['FULL NAME', 'PATIENTID', 'MEDICINE NAME', 'MEDICINE
ID', 'DESCRIPTION', 'USAGE', 'CONSUMPTION CYCLE'])
    print(query2)
    return(query2)

def Query3():
    cur.execute("""
    SELECT DISTINCT p.fullName, b.Bed_ID, b.Occupied, l.LogisticsDate, da.date_released, (da.date_released -
l.logisticsDate) AS NumDays
    FROM Bed b, Patients p, Logistics l, Date_admitted da
    WHERE p.patientID = b.patientID AND p.patientID = da.patientID AND da.LOGISTICSDATE = l.LOGISTICSDATE

```



ORDER BY occupied ASC

""")

pull = cur.fetchall()

query3 = pd.DataFrame(pull, columns = ['FULL NAME', 'BED ID', 'OCCUPIED', 'LOGISTIC DATE', 'DATE  
RELEASED', 'DAYS ADMITTED'])

print(query3)

return(query3)

def Query4():

cur.execute("""

SELECT DISTINCT emp.EmployeeType, emp.fullName, acc.username, acc.password, acc.EMP\_privilege  
FROM Employee emp, Account acc

WHERE acc.employeeID = emp.employeeID

ORDER BY emp\_privilege DESC

""")

pull = cur.fetchall()

query4 = pd.DataFrame(pull, columns = ['EMPLOYEE TYPE', 'FULL NAME', 'USERNAME', 'PASSWORD', 'PRIVILEGE'])

print(query4)

return(query4)

def Query5():

cur.execute("""SELECT \*

FROM Patients

MINUS

SELECT \*

FROM PATIENTS

WHERE Symptoms = 'Nausea'

""")

pull = cur.fetchall()

query5 = pd.DataFrame(pull, columns = ['PATIENT ID', 'FULLNAME', 'AGE', 'GENDER', 'SIN', 'HEALTH  
CARD', 'SYMPTOMS'])

print(query5)

return(query5)

def Query6():

cur.execute("""SELECT DISTINCT \*

FROM Medicine

""")

pull = cur.fetchall()

query6 = pd.DataFrame(pull, columns = ['MEDICINE ID', 'MEDICINE NAME', 'DESCRIPTION', 'INVENTORY'])

print(query6)

return(query6)

def Query7():

cur.execute("""SELECT p.PatientID, p.FullName, p.Symptoms, ec.fullName AS Emergency\_Contact\_Name,  
ec.relation, ec.tel\_number

From Patients p, Emergency\_Contact ec

WHERE p.Age <= 21 AND p.PatientID = ec.PatientID

""")

```

pull = cur.fetchall()

query7 = pd.DataFrame(pull, columns =['PATIENT ID','FULL NAME','SYMPTOMS','EMERGENCY
CONTACT','RELATION','TELEPHONE NUMBER'] )
print(query7)
return(query7)

def Query8():
    cur.execute("""
SELECT e.fullName, e.employeeID, e.employeeType, n.specialization
FROM Nurses n, Employee e
WHERE e.EmployeeID = n.EmployeeID
UNION
SELECT e.fullName, e.employeeID, e.employeeType, d.specialization
FROM Doctors d, Employee e
WHERE e.EmployeeID = d.EmployeeID
UNION
SELECT e.fullName, e.employeeID, e.employeeType, os.job_title
FROM Operations_Staff os, Employee e
WHERE e.EmployeeID = os.EmployeeID
UNION
SELECT e.fullName, e.employeeID, e.employeeType, le.profession
FROM Law_enforcement le, Employee e
WHERE e.EmployeeID = le.EmployeeID
ORDER BY employeeType ASC
""")
    pull = cur.fetchall()

    query8 = pd.DataFrame(pull, columns =['FULL NAME','EMPLOYEE ID','EMPLOYEE TYPE','SPECIALIZATION'] )
    print(query8)
    return(query8)

#SchoolDb      m534khan@//oracle.scs.ryerson.ca:1521/orcl
base = st.selectbox('Select Demo Function',('Create All Tables','Drop all Tables','Fill All Tables','GUI DEMO'))

if base == 'Create All Tables':
    if st.button('Submit'):
        cur.execute("""
CREATE TABLE Logistics (
NumPatient NUMBER(10) NOT NULL,
NumDoctor NUMBER(10) NOT NULL,
NumNurse NUMBER(10) NOT NULL,
NumBeds NUMBER(10) NOT NULL ,
LogisticsDate DATE NOT NULL,
PRIMARY KEY(LogisticsDate)
)""")

        cur.execute("""
CREATE TABLE Employee(
FullName VARCHAR2(25) NOT NULL,
SIN NUMBER(7,0),
EmployeeID NUMBER(10) NOT NULL,
EmployeeType VARCHAR2(20) NOT NULL CHECK (EmployeeType IN ('doctor', 'nurses', 'law enforcement',
'operations staff')),

```

```
PRIMARY KEY(EmployeeID)
)""")
```

```
cur.execute("""
CREATE TABLE account (
Username VARCHAR2(20) PRIMARY KEY,
Password VARCHAR2(20) NOT NULL,
Emp_Privilege CHAR(1) NOT NULL,
EmployeeID NUMBER(10),
FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
)""")
```

```
cur.execute("""
CREATE TABLE schedule (
WorkDate DATE,
Clock_In TIMESTAMP NOT NULL,
Clock_Out TIMESTAMP NOT NULL,
EmployeeID NUMBER(10),
PRIMARY KEY (WorkDate),
FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
)""")
```

```
cur.execute("""
CREATE TABLE doctors (
Specialization VARCHAR2(20) NOT NULL,
EmployeeID Number(10),
FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
)""")
```

```
cur.execute("""
CREATE TABLE nurses (
EmployeeID NUMBER(10),
Specialization VARCHAR2(40) NOT NULL,
FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
)""")
```

```
cur.execute("""
CREATE TABLE Law_Enforcement (
Profession VARCHAR2(20) NOT NULL,
EmployeeID Number(10),
FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
)""")
```

```
cur.execute("""
CREATE TABLE Operations_Staff (
Job_title VARCHAR2(40) NOT NULL,
EmployeeID NUMBER(10),
FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
)""")
```

```
cur.execute("""
CREATE TABLE Patients (
PatientID NUMBER(10) NOT NULL,
```

```

FullName VARCHAR2(200) NOT NULL,
Age NUMBER(5) NOT NULL,
Gender CHAR NOT NULL,
SIN NUMBER(5) NOT NULL,
Health_CN NUMBER(14,0) NOT NULL,
Symptoms VARCHAR2(200) NOT NULL,
PRIMARY KEY (PatientID)
)""")

```

```

cur.execute("""
CREATE TABLE Date_Admitted (
Date_released DATE DEFAULT NULL,
LogisticsDate DATE NOT NULL,
PatientID NUMBER(10) NOT NULL,
FOREIGN KEY (LogisticsDate) REFERENCES Logistics(LogisticsDate),
FOREIGN KEY (PatientID) REFERENCES Patients(PatientID)
)""")

```

```

cur.execute("""
CREATE TABLE Bed(
Bed_ID NUMBER(5) ,
PatientID NUMBER(10) NOT NULL,
Occupied CHAR DEFAULT 'N',
PRIMARY KEY (Bed_ID),
FOREIGN KEY (PatientID) REFERENCES Patients(PatientID)
)""")

```

```

cur.execute("""
CREATE TABLE Medicine(
Medicine_ID NUMBER (20,0) PRIMARY KEY,
Name VARCHAR2(20) NOT NULL,
Description VARCHAR2(100) NOT NULL,
Inventory NUMBER(20,0) NOT NULL
)""")

```

```

cur.execute("""
CREATE TABLE Emergency_Contact(
FullName VARCHAR2(200) NOT NULL,
Relation VARCHAR2(200) NOT NULL,
Tel_Number NUMBER(10) NOT NULL,
Address VARCHAR2 (50) NOT NULL,
PatientID NUMBER(10),
FOREIGN KEY (PatientID) REFERENCES Patients(PatientID)
)""")

```

```

cur.execute("""
CREATE TABLE Medicine_Prescription(
Medicine_ID NUMBER (20,0) NOT NULL,
PatientID NUMBER(10) NOT NULL,
Dosage VARCHAR2(20) NOT NULL,
Consumption_Cycle VARCHAR2 (100) NOT NULL,
FOREIGN KEY (Medicine_ID) REFERENCES Medicine(Medicine_ID),
FOREIGN KEY (PatientID) REFERENCES Patients(PatientID)
)""")

```

```

)
)

if base == 'Drop all Tables':
    if st.button('Submit'):
        cur.execute("DROP TABLE Employee CASCADE CONSTRAINTS")
        cur.execute("DROP TABLE account CASCADE CONSTRAINTS")
        cur.execute("DROP TABLE schedule CASCADE CONSTRAINTS")
        cur.execute("DROP TABLE doctors CASCADE CONSTRAINTS")
        cur.execute("DROP TABLE nurses CASCADE CONSTRAINTS")
        cur.execute("DROP TABLE Date_Admitted CASCADE CONSTRAINTS")
        cur.execute("DROP TABLE Law_Enforcement CASCADE CONSTRAINTS")
        cur.execute("DROP TABLE Operations_Staff CASCADE CONSTRAINTS")
        cur.execute("DROP TABLE Logistics CASCADE CONSTRAINTS")
        cur.execute("DROP TABLE Patients CASCADE CONSTRAINTS")
        cur.execute("DROP TABLE Bed CASCADE CONSTRAINTS")
        cur.execute("DROP TABLE Medicine CASCADE CONSTRAINTS")
        cur.execute("DROP TABLE Emergency_Contact CASCADE CONSTRAINTS")
        cur.execute("DROP TABLE Medicine_Prescription CASCADE CONSTRAINTS")

if base == 'Fill All Tables':
    if st.button('Submit'):
        cur.execute("""
INSERT INTO LOGISTICS
VALUES(1,2,4,100,TO_DATE('26/10/2020','DD/MM/YYYY'))""")

        cur.execute("""
INSERT INTO LOGISTICS
VALUES(3,2,4,100,TO_DATE('28/10/2020','DD/MM/YYYY'))""")

        cur.execute("""
INSERT INTO LOGISTICS
VALUES(2,2,4,100,TO_DATE('10/11/2020','DD/MM/YYYY'))""")

        cur.execute("""
INSERT INTO EMPLOYEE
VALUES('Ahmed Khan', '12345','00001','doctor')""")

        cur.execute("""
INSERT INTO EMPLOYEE
VALUES('Jules Gammad', '01234','00002','doctor')""")

        cur.execute("""
INSERT INTO EMPLOYEE
VALUES('Jane Doe', '12334','00003','nurses')""")

        cur.execute("""
INSERT INTO EMPLOYEE
VALUES('John Doe', '12445','00004','nurses')""")

        cur.execute("""
INSERT INTO EMPLOYEE
VALUES('Tatiana Mesau', '12355','00005','nurses')""")

```

```
cur.execute("""
INSERT INTO EMPLOYEE
VALUES('Constantino Mello', '11345','00006','nurses')""")
```

```
cur.execute("""
INSERT INTO EMPLOYEE
VALUES('Christopher Jiello', '11340','00007','operations staff')""")
```

```
cur.execute("""
INSERT INTO EMPLOYEE
VALUES('Melani Tot', '10045','00008','operations staff')""")
```

```
cur.execute("""
INSERT INTO EMPLOYEE
VALUES('Teo Mani', '00345','00009','law enforcement')""")
```

```
cur.execute("""
INSERT INTO EMPLOYEE
VALUES('Ian Refut', '11305','00010','law enforcement')""")
```

```
cur.execute("""
INSERT INTO account
VALUES('admin', 'cusadmin', 'A', '00007')""")
```

```
cur.execute("""
INSERT INTO account
VALUES('admin2', 'cusadmin2', 'A', '00008')""")
```

```
cur.execute("""
INSERT INTO account
VALUES('akhan', 'admin', 'B', '00001')""")
```

```
cur.execute("""
INSERT INTO account
VALUES('jgammad', 'admin', 'B', '00002')""")
```

```
cur.execute("""
INSERT INTO doctors
VALUES('General Surgery', '00001')""")
```

```
cur.execute("""
INSERT INTO doctors
VALUES('Neurosurgery', '00002')""")
```

```
cur.execute("""
INSERT INTO nurses
VALUES('00003', 'Registered Nurse')""")
```

```
cur.execute("""
INSERT INTO nurses
VALUES('00004', 'Medical-surgical Nurse')""")
```

```
cur.execute("""
INSERT INTO nurses
VALUES('00005', 'Intensive Care Unit Registered Nurse')""")
```

```
cur.execute("""
INSERT INTO nurses
VALUES('00006', 'Nursing Assistant')""")
```

```
cur.execute("""
INSERT INTO Law_Enforcement
VALUES('Police', '00009')""")
```

```
cur.execute("""
INSERT INTO Law_Enforcement
VALUES('Fireman', '00010')""")
```

```
cur.execute("""
INSERT INTO Operations_Staff
VALUES('Front-desk Specialist', '00007')""")
```

```
cur.execute("""
INSERT INTO Operations_Staff
VALUES('Operations Manager', '00008')""")
```

```
cur.execute("""
INSERT INTO Patients
VALUES('1', 'Dominique Rodriguez', '20', 'M', '12312', '78012', 'Nausea')""")
```

```
cur.execute("""
INSERT INTO Patients
VALUES('2', 'Allyson Teofima', '32', 'F', '11312', '32012', 'Nausea')""")
```

```
cur.execute("""
INSERT INTO Patients
VALUES('3', 'Merci Tefioux', '26', 'M', '12332', '18542', 'Flu')""")
```

```
cur.execute("""
INSERT INTO Patients
VALUES('4', 'Anderson Silva', '29', 'M', '13222', '53902', 'Internal Bleeding')""")
```

```
cur.execute("""
INSERT INTO Patients
VALUES('5', 'Monica Mercedes', '16', 'F', '12312', '98012', 'Flu')""")
```

```
cur.execute("""
INSERT INTO Patients
VALUES('6', 'Brook Lopez', '20', 'M', '15512', '23012', 'Flu')""")
```

```
cur.execute("""
INSERT INTO Bed
VALUES('1', '1', 'DEFAULT')""")
```

```
cur.execute("""
```

```
INSERT INTO Bed
VALUES('2','2', DEFAULT)''''')
```

```
cur.execute('''''
INSERT INTO Bed
VALUES('3','3', 'Y')''''')
```

```
cur.execute('''''
INSERT INTO Bed
VALUES('4','4', 'Y')''''')
```

```
cur.execute('''''
INSERT INTO Bed
VALUES('5','5', DEFAULT)''''')
```

```
cur.execute('''''
INSERT INTO Bed
VALUES('6','6', 'Y')''''')
```

```
cur.execute('''''
INSERT INTO MEDICINE
VALUES('1','Ibuprofen','Painkiller','200')''''')
```

```
cur.execute('''''
INSERT INTO MEDICINE
VALUES('2','Zofran','Anti-nausea','50')''''')
```

```
cur.execute('''''
INSERT INTO MEDICINE
VALUES('3','Zanamivir','Flu medicine', '2000')''''')
```

```
cur.execute('''''
INSERT INTO Emergency_Contact
VALUES('Dayane Silva', 'Wife', '416322091','1532 John Street', '4')''''')
```

```
cur.execute('''''
INSERT INTO Emergency_Contact
VALUES ('Rafael Lopez', 'Brother', '3123244500','1420 Leli Drive', '6')''''')
```

```
cur.execute('''''
INSERT INTO Emergency_Contact
VALUES('Melina Mercedes', 'Mother', '415900213','501 Shelby Drive', '5')''''')
```

```
cur.execute('''''
INSERT INTO Emergency_Contact
VALUES('Dwayne Rodriguez', 'Father', '516321021','151 Touth Ave', '1')''''')
```

```
cur.execute('''''
INSERT INTO Medicine_Prescription
VALUES('1','4','3 times a day', '3 months')''''')
```

```
cur.execute('''''
INSERT INTO Medicine_Prescription
```



```

VALUES('2','1','1 times a day', '1 week')''''')

    cur.execute("""
INSERT INTO Medicine_Prescription
VALUES('3','3','2 times a day', '1 month')''''')

    cur.execute("""
INSERT INTO Date_Admitted
VALUES(TO_DATE('26/11/2020','DD/MM/YYYY'), TO_DATE('26/10/2020','DD/MM/YYYY'), '1')''''')

    cur.execute("""
INSERT INTO Date_Admitted
VALUES(TO_DATE('15/11/2020','DD/MM/YYYY'), TO_DATE('28/10/2020','DD/MM/YYYY'), '2')''''')

    cur.execute("""
INSERT INTO Date_Admitted
VALUES(DEFAULT, TO_DATE('28/10/2020','DD/MM/YYYY'), '3')''''')

    cur.execute("""
INSERT INTO Date_Admitted
VALUES(DEFAULT, TO_DATE('28/10/2020','DD/MM/YYYY'), '4')''''')

    cur.execute("""
INSERT INTO Date_Admitted
VALUES(TO_DATE('12/12/2020','DD/MM/YYYY'), TO_DATE('10/11/2020','DD/MM/YYYY'), '5')''''')

    cur.execute("""
INSERT INTO Date_Admitted
VALUES(DEFAULT, TO_DATE('10/11/2020','DD/MM/YYYY'), '6')''''')

if base == 'GUI DEMO':
    image = Image.open('hospital.jpg')

    st.image(image, use_column_width = True)
    st.title("Login")

    username = st.text_input('Username:', value = "")
    password = st.text_input('Password:', value = "", type = "password")
    stay = st.checkbox('Log In')

    _account = account()
    if username != "" and password != "" and stay:
        for users in range(len(_account)):
            if _account['USERNAME'][users] == username:
                print('here')
            if _account['PASSWORD'][users] == password:
                use = users
                print('here agin')
                loggedin = True

    if loggedin == True and stay and _account['EMP_PRIVILEGE'][use]=='A':

```

```

options = st.selectbox('Select Table to display',(
    'Employee', 'Logistics', 'Account', 'Bed Information',
    'Date of Admittance', 'List of Doctors','Emergency Contacts',
    'List of Law Enforcement','Medicine Inventory',
    'Medicine Prescriptions','List of Nurses','Operations Staff','Patients','View Doctors and Nurses','Occupied
Beds','Employee Accounts','View Employees'))

```

```

if options == 'Employee':
    source = employee
if options == 'Logistics':
    source = logic
if options == 'Account':
    source = account
if options == 'Bed Information':
    source = bed
if options == 'Date of Admittance':
    source = date_admit
if options == 'List of Doctors':
    source = doctor
if options == 'Emergency Contacts':
    source = emergency
if options == 'List of Law Enforcement':
    source = law
if options == 'Medicine Inventory':
    source = medicine
if options == 'Medicine Prescriptions':
    source = medicinepre
if options == 'List of Nurses':
    source = nurse
if options == 'Operations Staff':
    source = op_staff
if options == 'Patients':
    source = patients
if options == 'View Doctors and Nurses':
    source = Query1
if options == 'Occupied Beds':
    source = Query3
if options == 'Employee Accounts':
    source = Query4
if options == 'View Employees':
    source = Query8

```

```

edit = st.checkbox('Enable Editing')
if edit:
    add = st.radio('Select',('Add Information','Delete Information'))
    if add == 'Add Information':

        if options == 'Employee':
            Fname = st.text_input('Full Name')
            SIN = int(st.text_input('SIN',value = '1'))
            empid = int(st.text_input('Employee ID',value = '1'))

```

```

Emptype = st.text_input('Employee Type')

if options == 'Logistics':
    nump = int(st.text_input('Number of Patients',value = '1'))
    numd = int(st.text_input('Number of Doctors',value = '1'))
    numn = int(st.text_input('Number of Nurses',value = '1'))
    numb = int(st.text_input('Number of Beds',value = '1'))
    logdate = st.text_input('Logistic Date')

if options == 'Account':
    usern = st.text_input('USERNAME')
    passn = st.text_input('PASSWORD', type = 'password')
    empp = st.text_input('Employee Privilege')
    empid = int(st.text_input('Employee ID',value = '1'))

if options == 'List of Doctors':
    spec = st.text_input('Specialization')
    empid = int(st.text_input('Employee Id',value = '1'))

if options == 'List of Law Enforcement':
    proff = st.text_input('Profession')
    empid = int(st.text_input('Employee Id',value = '1'))

if options == 'List of Nurses':
    empid = int(st.text_input('Employee Id',value = '1'))
    spec = st.text_input('Specialization')

if options == 'Operations Staff':
    spec = st.text_input('Job Title')
    empid = int(st.text_input('Employee Id',value = '1'))

if options == 'Bed Information':
    Bed_id = st.text_input('Bed Id', value = '1')
    Bed_id = int(Bed_id)
    patid = st.text_input('Patient Id', value = '1')
    patid = int(patid)
    occupied = st.text_input('Occupied')

if options == 'Date of Admittance':
    datere = st.text_input('Date of Release')
    logdate = st.text_input('logistic Date')
    patid = st.text_input('Patient Id', value = '1')
    patid = int(patid)

if options == 'Emergency Contacts':
    Fname = st.text_input('Full name')
    Relation = st.text_input('Relation')
    Tnum = st.text_input('Telephone Number', value = '1')
    address = st.text_input('Address')
    patid = st.text_input('Patient Id', value = '1')
    Tnum = int(Tnum)
    patid = int(patid)

```

```

if options == 'Medicine Inventory':
    MedId = st.text_input('Medicine Id', value = '1')
    MedId = int(MedId)
    Name = st.text_input('Name')
    Desc = st.text_input('Description of Medication')
    inv = st.text_input('Inventory', value = '1')
    inv = int(inv)

if options == 'Medicine Prescriptions':
    MedId = st.text_input('Medicine Id', value = '1')
    MedId = int(MedId)
    patid = st.text_input('Patient Id', value = '1')
    patid = int(patid)
    dosage = st.text_input('Dosage Prescription')
    Consumpt = st.text_input('Consumption Cycle')

if options == 'Patients':
    patid = st.text_input('Patient Id', value = '1')
    patid = int(patid)
    Fname = st.text_input('FULL Name')
    Age = st.text_input('Age', value = '1')
    Age = int(Age)
    gender = st.text_input('Gender')
    sin = st.text_input('SIN', value = '1')
    sin = int(sin)
    Healthcn = st.text_input('Health Card Number', value = '1')
    Healthcn = int(Healthcn)
    symptoms = st.text_input('Symptoms')

if st.button('Submit'):
    if options == 'Employee':
        cur.execute("""INSERT INTO EMPLOYEE
VALUES('%s', '%d', '%d', 'nurses')"""%(Fname, SIN, empid, Emptye))

    if options == 'Logistics':
        cur.execute("""INSERT INTO LOGISTICS
VALUES('%d', '%d', '%d', '%d', TO_DATE('%s', 'DD/MM/YYYY'))"""%(nump, numd, numn, numb, logdate))

    if options == 'Account':
        cur.execute("""INSERT INTO account
VALUES('%s', '%s', '%s', '%d')"""%(usern, passn, empp, empid))

    if options == 'List of Doctors':
        cur.execute("""INSERT INTO doctors
VALUES('%s', '%d')"""%(spec, empid))

    if options == 'List of Law Enforcement':
        cur.execute("""INSERT INTO Law_Enforcement
VALUES('%s', '%d')"""%(proff, empid))

    if options == 'List of Nurses':
        cur.execute("""INSERT INTO nurses
VALUES('%d', '%s')"""%(empid, spec))

```

```

        if options == 'Operations Staff':
            cur.execute("""INSERT INTO Operations_Staff
VALUES('%s', '%d')"""%(spec,empid))

        if options == 'Bed Information':
            cur.execute("""INSERT INTO Bed
VALUES('%d','%d', '%s')"""%(Bed_id,patid,occupied))

        if options == 'Date of Admittance':
            cur.execute("""INSERT INTO Date_Admitted
VALUES(TO_DATE('%s','DD/MM/YYYY'), TO_DATE('%s','DD/MM/YYYY'), '%d')"""%(datere,logdate,patid))

        if options == 'Emergency Contacts':
            cur.execute("""INSERT INTO Emergency_Contact
VALUES('%s', '%s', '%d', '%s', '%d')"""%(Fname,Relation,Tnum,address,patid))

        if options == 'Medicine Inventory':
            cur.execute("""INSERT INTO MEDICINE
VALUES('%d', '%s', '%s', '%d')"""%(MedId,Name,Desc,inv))

        if options == 'Medicine Prescriptions':
            cur.execute("""INSERT INTO Medicine_Prescription
VALUES('%d', '%d', '%s', '%s')"""%(MedId,patid,dosage,Consumpt))

        if options == 'Patients':
            cur.execute("""INSERT INTO PATIENTS
VALUES('%d', '%s', '%d', '%s', '%d', '%d', '%s')"""%(patid,Fname,Age,gender,sin,Healthcn,symtoms))

    if add == 'Delete Information':
        if options == 'Employee':
            empid = int(st.text_input('Employee ID',value = '1'))

        if options == 'Logistics':
            logdate = st.text_input('Logistic Date')

        if options == 'Account':
            usern = st.text_input('USERNAME')

        if options == 'List of Doctors':
            empid = int(st.text_input('Employee Id',value = '1'))

        if options == 'List of Law Enforcement':
            empid = int(st.text_input('Employee Id',value = '1'))

        if options == 'List of Nurses':
            empid = int(st.text_input('Employee Id',value = '1'))

        if options == 'Operations Staff':
            empid = int(st.text_input('Employee Id',value = '1'))

        if options == 'Bed Information':
            Bed_id = st.text_input('Bed Id', value = '1')

```

```

    Bed_id = int(Bed_id)

    if options == 'Date of Admittance':
        patid = st.text_input('Patient Id', value = '1')
        patid = int(patid)

    if options == 'Emergency Contacts':
        patid = st.text_input('Patient Id', value = '1')
        patid = int(patid)

    if options == 'Medicine Inventory':
        MedId = st.text_input('Medicine Id', value = '1')
        MedId = int(MedId)

    if options == 'Medicine Prescriptions':
        MedId = st.text_input('Medicine Id', value = '1')
        MedId = int(MedId)

    if options == 'Patients':
        patid = st.text_input('Patient Id', value = '1')
        patid = int(patid)

    if st.button('Submit'):
        if options == 'Employee':
            cur.execute("""DELETE FROM EMPLOYEE
WHERE EmployeeID = %d
""%(empid))

        if options == 'Logistics':
            cur.execute("""DELETE FROM LOGISTICS
WHERE LogisticsDate = TO_DATE('%s','DD/MM/YYYY')""%(logdate))

        if options == 'Account':
            cur.execute("""DELETE FROM account
WHERE Username = %s""%(usern))

        if options == 'List of Doctors':
            cur.execute("""DELETE FROM doctors
WHERE EmployeeID = %d""%(empid))

        if options == 'List of Law Enforcement':
            cur.execute("""DELETE FROM Law_Enforcement
WHERE EmployeeID = %d""%(empid))

        if options == 'List of Nurses':
            cur.execute("""DELETE FROM nurses
WHERE EmployeeID = %d""%(empid))

        if options == 'Operations Staff':
            cur.execute("""DELETE FROM Operations_Staff
WHERE EmployeeID = %d""%(empid))

        if options == 'Bed Information':

```

```

        cur.execute("""DELETE FROM Bed
WHERE BED_id = %d"""%(Bed_id))

        if options == 'Date of Admittance':
            cur.execute("""DELETE FROM Date_Admitted
WHERE PatientID = %d"""%(patid))

        if options == 'Emergency Contacts':
            cur.execute("""DELETE FROM Emergency_Contact
WHERE PatientID = %d"""%(patid))

        if options == 'Medicine Inventory':
            cur.execute("""DELETE FROM MEDICINE
WHERE Medicine_ID = %d"""%(MedId))

        if options == 'Medicine Prescriptions':
            cur.execute("""DELETE FROM Medicine_Prescription
WHERE Medicine_ID = %d"""%(MedId))

        if options == 'Patients':
            cur.execute("""DELETE FROM Patients
WHERE PatientID = %d"""%(patid))

        conn.commit()

        st.table(source())

if loggedin == True and stay and _account['EMP_PRIVILEGE'][use]=='B':

    options = st.selectbox('Select Table to display',('Bed Information',
'Date of Admittance','Emergency Contacts','Medicine Inventory',
'Medicine Prescriptions','Patients','Patient Prescriptions','Patients Without Nausea','Medicine
Information','Young Patients'))

    if options == 'Bed Information':
        source = bed
    if options == 'Date of Admittance':
        source = date_admit
    if options == 'Emergency Contacts':
        source = emergency
    if options == 'Medicine Inventory':
        source = medicine
    if options == 'Medicine Prescriptions':
        source = medicinepre
    if options == 'Patients':
        source = patients
    if options == 'Patient Prescriptions':
        source = Query2
    if options == 'Patients Without Nausea':
        source = Query5
    if options == 'Medicine Information':

```

```

source = Query6
if options == 'Young Patients':
    source = Query7
edit = st.checkbox('Enable Editing')
if edit:
    add = st.radio('Select',('Add Information','Delete Information'))
    if add == 'Add Information':

        if options == 'Bed Information':
            Bed_id = st.text_input('Bed Id', value = '1')
            Bed_id = int(Bed_id)
            patid = st.text_input('Patient Id', value = '1')
            patid = int(patid)
            occupied = st.text_input('Occupied')

        if options == 'Date of Admittance':
            datere = st.text_input('Date of Release')
            logdate = st.text_input('logistic Date')
            patid = st.text_input('Patient Id', value = '1')
            patid = int(patid)

        if options == 'Emergency Contacts':
            Fname = st.text_input('Full name')
            Relation = st.text_input('Relation')
            Tnum = st.text_input('Telephone Number', value = '1')
            address = st.text_input('Address')
            patid = st.text_input('Patient Id', value = '1')
            Tnum = int(Tnum)
            patid = int(patid)

        if options == 'Medicine Inventory':
            MedId = st.text_input('Medicine Id', value = '1')
            MedId = int(MedId)
            Name = st.text_input('Name')
            Desc = st.text_input('Description of Medication')
            inv = st.text_input('Inventory', value = '1')
            inv = int(inv)

        if options == 'Medicine Prescriptions':
            MedId = st.text_input('Medicine Id', value = '1')
            MedId = int(MedId)
            patid = st.text_input('Patient Id', value = '1')
            patid = int(patid)
            dosage = st.text_input('Dosage Prescription')
            Consumpt = st.text_input('Consumption Cycle')

        if options == 'Patients':
            patid = st.text_input('Patient Id', value = '1')
            patid = int(patid)
            Fname = st.text_input('FULL Name')
            Age = st.text_input('Age', value = '1')
            Age = int(Age)
            gender = st.text_input('Gender')

```



```

sin = st.text_input('SIN', value = '1')
sin = int(sin)
Healthcn = st.text_input('Health Card Number', value = '1')
Healthcn = int(Healthcn)
symptoms = st.text_input('Symptoms')

if st.button('Submit'):
    if options == 'Bed Information':
        cur.execute("""INSERT INTO Bed
VALUES('%d','%d', '%s')"""%(Bed_id,patid,occupied))

    if options == 'Date of Admittance':
        cur.execute("""INSERT INTO Date_Admitted
VALUES( TO_DATE('%s','DD/MM/YYYY'), TO_DATE('%s','DD/MM/YYYY'), '%d')"""%(datere,logdate,patid))

    if options == 'Emergency Contacts':
        cur.execute("""INSERT INTO Emergency_Contact
VALUES('%s', '%s', '%d', '%s', '%d')"""%(Fname,Relation,Tnum,address,patid))

    if options == 'Medicine Inventory':
        cur.execute("""INSERT INTO MEDICINE
VALUES('%d','%s','%s','%d')"""%(MedId,Name,Desc,inv))

    if options == 'Medicine Prescriptions':
        cur.execute("""INSERT INTO Medicine_Prescription
VALUES('%d','%d','%s', '%s')"""%(MedId,patid,dosage,Consumpt))

    if options == 'Patients':
        cur.execute("""INSERT INTO PATIENTS
VALUES('%d','%s', '%d', '%s', '%d', '%d','%s')"""%(patid,Fname,Age,gender,sin,Healthcn,symptoms))

if add == 'Delete Information':
    if options == 'Bed Information':
        Bed_id = st.text_input('Bed Id', value = '1')
        Bed_id = int(Bed_id)

    if options == 'Date of Admittance':
        patid = st.text_input('Patient Id', value = '1')
        patid = int(patid)

    if options == 'Emergency Contacts':
        patid = st.text_input('Patient Id', value = '1')
        patid = int(patid)

    if options == 'Medicine Inventory':
        MedId = st.text_input('Medicine Id', value = '1')
        MedId = int(MedId)

    if options == 'Medicine Prescriptions':
        MedId = st.text_input('Medicine Id', value = '1')
        MedId = int(MedId)

    if options == 'Patients':

```

```
patid = st.text_input('Patient Id', value = '1')
patid = int(patid)

if st.button('Submit'):
    if options == 'Bed Information':
        cur.execute("""DELETE FROM Bed
WHERE BED_id = %d"""%(Bed_id))

    if options == 'Date of Admittance':
        cur.execute("""DELETE FROM Date_Admitted
WHERE PatientID = %d"""%(patid))

    if options == 'Emergency Contacts':
        cur.execute("""DELETE FROM Emergency_Contact
WHERE PatientID = %d"""%(patid))

    if options == 'Medicine Inventory':
        cur.execute("""DELETE FROM MEDICINE
WHERE Medicine_ID = %d"""%(MedId))

    if options == 'Medicine Prescriptions':
        cur.execute("""DELETE FROM Medicine_Prescription
WHERE Medicine_ID = %d"""%(MedId))

    if options == 'Patients':
        cur.execute("""DELETE FROM Patients
WHERE PatientID = %d"""%(patid))

    conn.commit()

st.table(source())

conn.commit()
```