Database Design | CIS-552 | Final Project

Project Title	: Reduction of COVID-19 Exposure at Hospitals
Submission Date	: 12/15/2023
Submitted By	: Group 11
Group Members	: Balakrishna Vardhineni (02069565)
	Charitha Palli (02102438)
	Bala Rahul Kommareddy (02136603)
Advisor	: Yukui Luo, PhD

Database Design | CIS-552 | Final Project

Problem Statement:

The COVID-19 pandemic has posed significant challenges to healthcare systems worldwide, particularly in managing the risk of virus transmission in hospital settings. Hospitals face the critical issue of minimizing COVID-19 exposure among medical staff and patients. Efficient and accurate management of patient information is vital in this context. The lack of a centralized, up-to-date database for patient records, treatment details, and staff schedules exacerbates the risk of disease spread and complicates treatment processes.

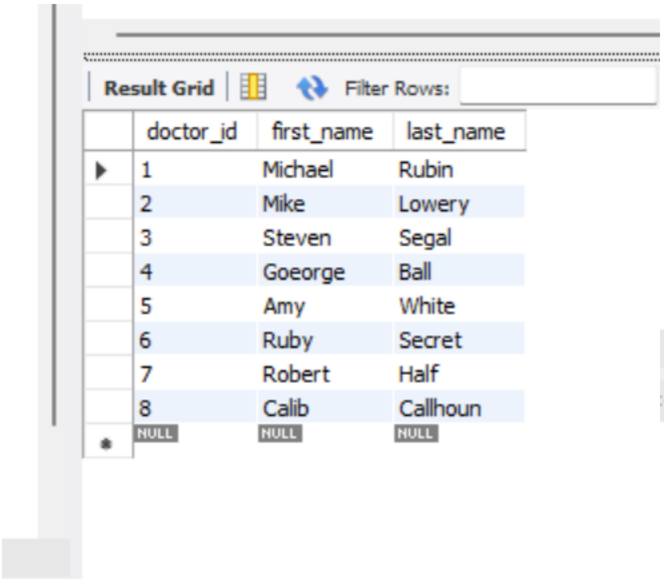
Introduction:

This project aims to develop a comprehensive database system, named "Hospital," designed to combat these challenges. The database will provide medical professionals with real-time access to crucial patient data, including admission dates, symptoms, treatment plans, and staff assignments. The database intends to enhance the hospital's operational efficiency, improve patient care, and reduce the risk of COVID-19 transmission among medical staff and patients by streamlining data access and management. The project encompasses the creation of a robust database structure, the implementation of stringent data integrity protocols, and the use of SQL for effective data handling and analysis.

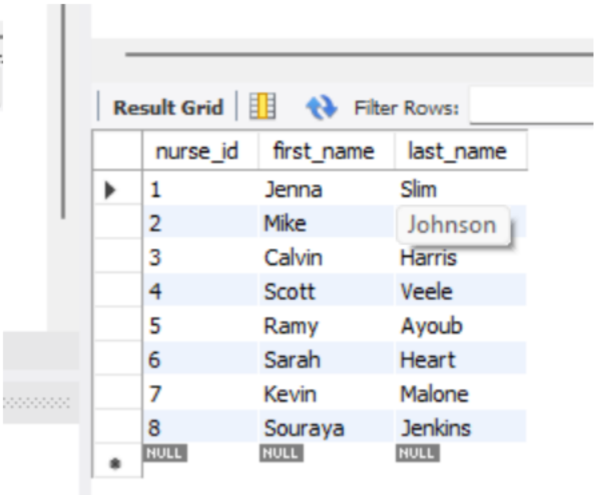
Database Design:

A completely new database “Hospital” is created which includes many tables. The tables that are included are contact info, contact relationship, COVID treatment, COVID wing, doctors, nurses, record admissions, record contacts, doctor shifts, nurses’ shifts, record of patients, symptoms, testing, treatments, and testing methods. Every table has its primary key, foreign key, and attributes. The tables are related using a primary key (uniquely identifies each record in a table), and a foreign key (a field in one table that refers to the primary key in another table. It establishes a link between the tables, ensuring referential integrity). SQL uses various commands and constraints to establish and manage relationships between tables:

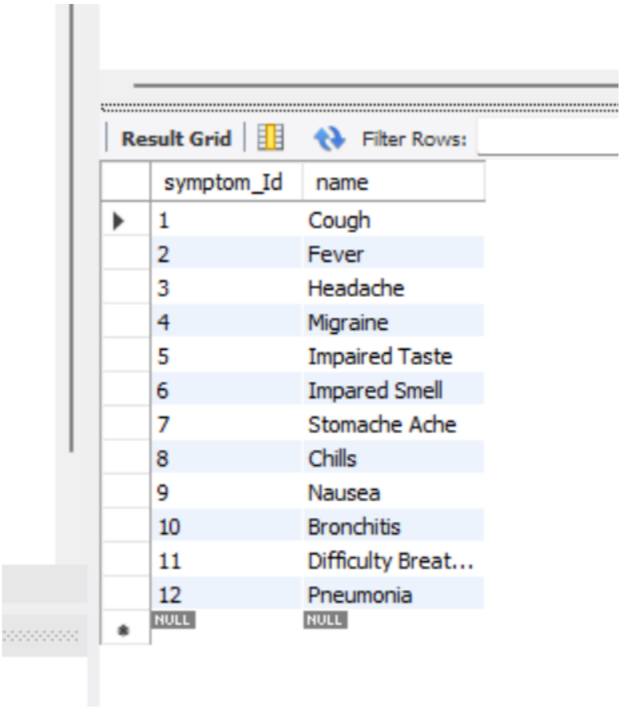
- CREATE TABLE: This command is used to create tables and define columns, primary keys, foreign keys, and constraints.



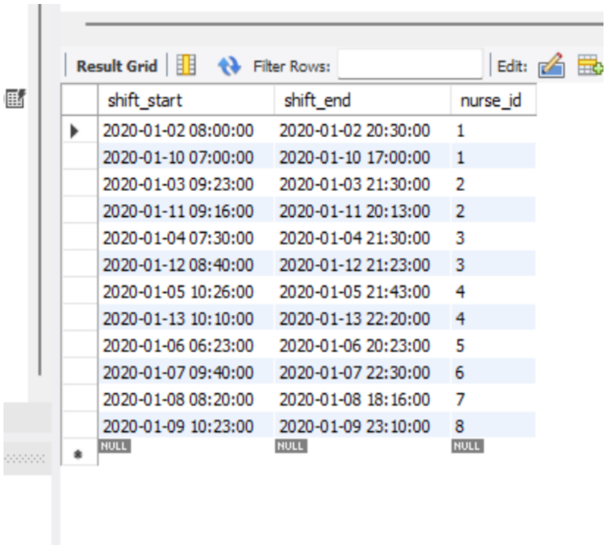
doctor_id	first_name	last_name
1	Michael	Rubin
2	Mike	Lowery
3	Steven	Segal
4	Goeorge	Ball
5	Amy	White
6	Ruby	Secret
7	Robert	Half
8	Calib	Callhoun
NULL	NULL	NULL



nurse_id	first_name	last_name
1	Jenna	Slim
2	Mike	Johnson
3	Calvin	Harris
4	Scott	Veele
5	Ramy	Ayoub
6	Sarah	Heart
7	Kevin	Malone
8	Souraya	Jenkins
NULL	NULL	NULL

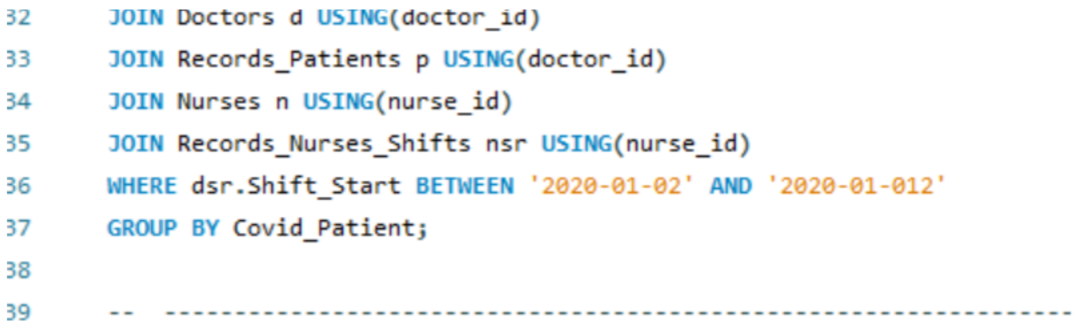


symptom_Id	name
1	Cough
2	Fever
3	Headache
4	Migraine
5	Impaired Taste
6	Impared Smell
7	Stomache Ache
8	Chills
9	Nausea
10	Bronchitis
11	Difficulty Breat...
12	Pneumonia
NULL	NULL



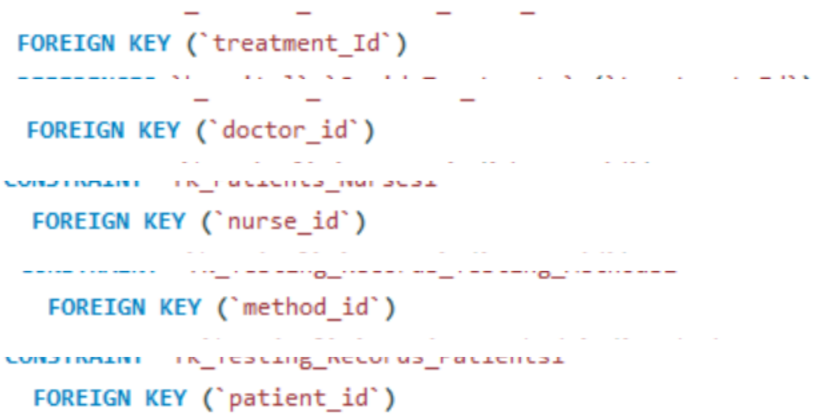
shift_start	shift_end	nurse_id
2020-01-02 08:00:00	2020-01-02 20:30:00	1
2020-01-10 07:00:00	2020-01-10 17:00:00	1
2020-01-03 09:23:00	2020-01-03 21:30:00	2
2020-01-11 09:16:00	2020-01-11 20:13:00	2
2020-01-04 07:30:00	2020-01-04 21:30:00	3
2020-01-12 08:40:00	2020-01-12 21:23:00	3
2020-01-05 10:26:00	2020-01-05 21:43:00	4
2020-01-13 10:10:00	2020-01-13 22:20:00	4
2020-01-06 06:23:00	2020-01-06 20:23:00	5
2020-01-07 09:40:00	2020-01-07 22:30:00	6
2020-01-08 08:20:00	2020-01-08 18:16:00	7
2020-01-09 10:23:00	2020-01-09 23:10:00	8
NULL	NULL	NULL

- ALTER TABLE: It allows modification of an existing table, including adding or dropping columns, defining primary and foreign keys, etc.
- JOIN: SQL uses JOIN operations (e.g., INNER JOIN, LEFT JOIN, RIGHT JOIN) to combine rows from multiple tables based on related columns between them.



```sql
JOIN Doctors d USING(doctor_id)
JOIN Records_Patients p USING(doctor_id)
JOIN Nurses n USING(nurse_id)
JOIN Records_Nurses_Shifts nsr USING(nurse_id)
WHERE dsr.Shift_Start BETWEEN '2020-01-02' AND '2020-01-012'
GROUP BY Covid_Patient;

--  ----------------------------------------------------------------
```

- FOREIGN KEY Constraint: This constraint ensures the referential integrity of the data in a table by specifying that the values in a column must match values in a related table's primary key.



```sql
FOREIGN KEY (`treatment_Id`)
FOREIGN KEY (`doctor_id`)
FOREIGN KEY (`nurse_id`)
FOREIGN KEY (`method_id`)
FOREIGN KEY (`patient_id`)
```

There are mainly three types of relationships used:

- one-to-one relationships
- one-to-many relationships
- many-to-many relationships

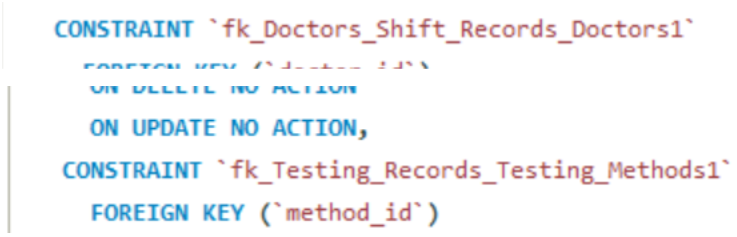
By establishing and maintaining these relationships using keys and SQL commands, databases organize data efficiently and ensure data consistency across multiple tables.

Data integrity:

Data integrity in the database is maintained using:

- Referential Integrity: This is maintained through foreign keys. It ensures the relationships between tables are preserved. A foreign key in one table points to the primary key in another table.

- Constraints: SQL provides various constraints like NOT NULL, UNIQUE, CHECK, and DEFAULT constraints. These enforce rules at the column level.



```
CONSTRAINT `fk_Doctors_Shift_Records_Doctors1`
  FOREIGN KEY (`doctor_id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
CONSTRAINT `fk_Testing_Records_Testing_Methods1`
  FOREIGN KEY (`method_id`)
```

- Indexes: While not directly related to integrity, indexes improve data retrieval performance. They ensure that data is stored and retrieved efficiently while maintaining the integrity of relationships.
- Triggers: Triggers are special stored procedures that are automatically executed or fired when certain events occur (like an INSERT, UPDATE, or DELETE operation). They help maintain data integrity by allowing custom checks, modifications, or actions to be performed when specific conditions are met.

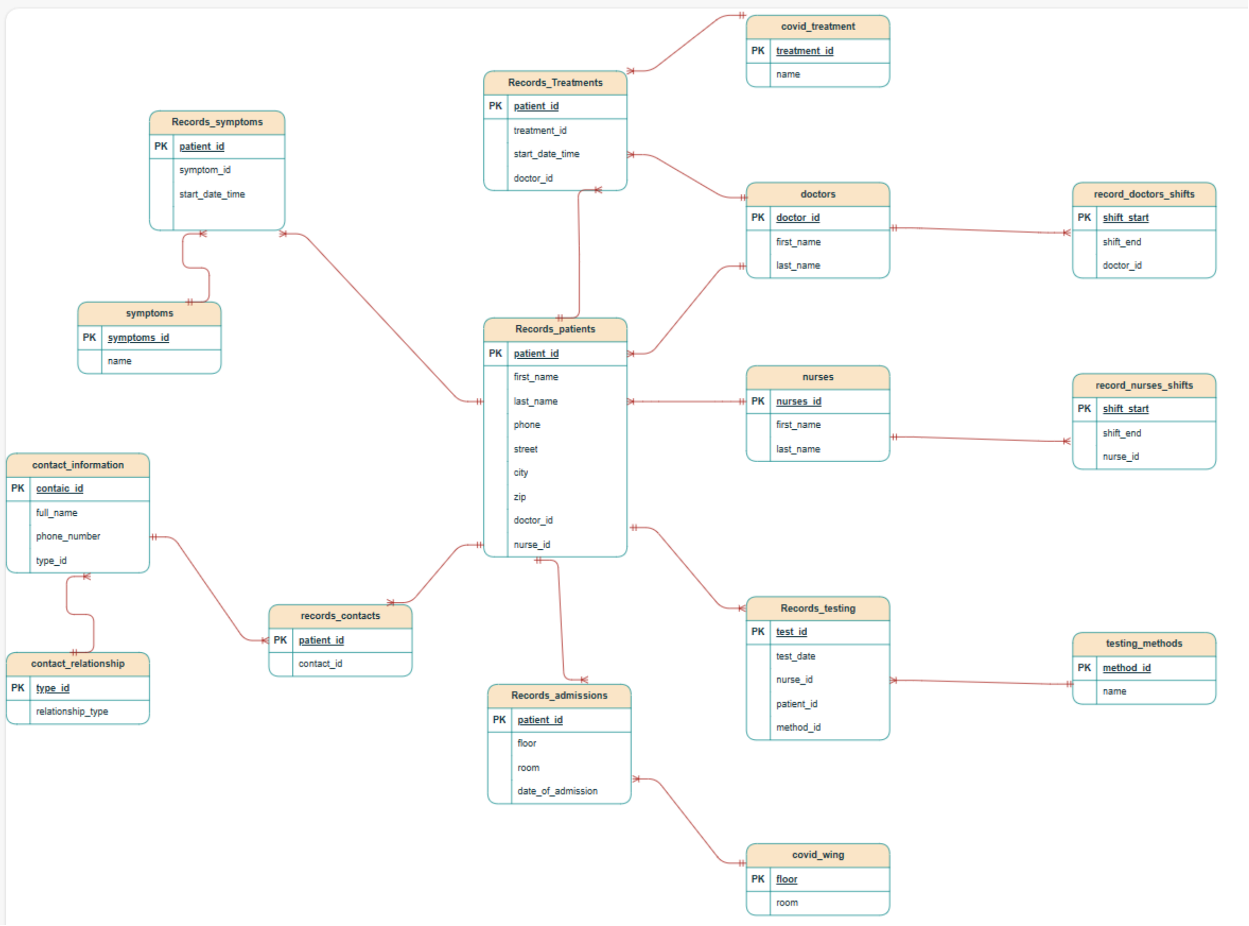
Databases maintain data integrity, ensuring the accuracy, consistency, and reliability of the stored data.

UML Diagrams:

Unified Modeling Language (UML) diagrams are graphical representations for graphical representations of the database where all the schemas are related. These representations include how the schemas are related to each other in the database.

The database has many schemas which are represented below and are made using draw.io.

Database Design | CIS-552 | Final Project



The UML diagram describes the logical model of the database. Which gives an idea of how the tables are connected in the database. Every table contains primary keys such as patient_id, doctor_id, nurse_id, etc… The tables are connected using a foreign key which is the primary key in some other tables.

Data Collection & SQL Development:

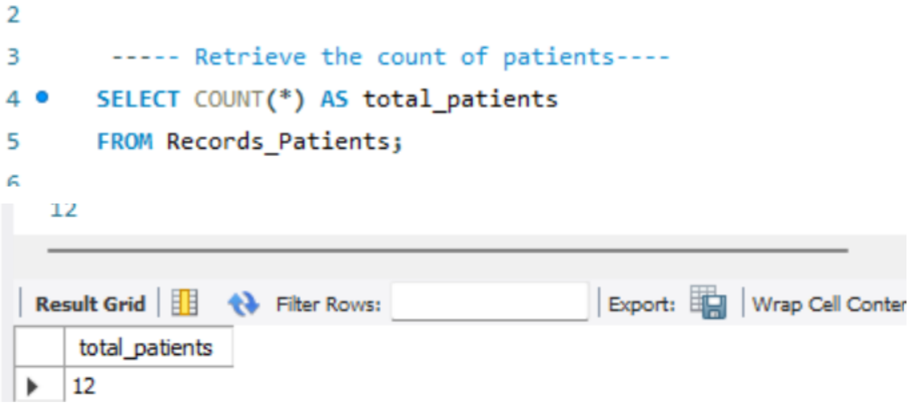
Collecting relevant data is a time-consuming process because it must maintain integrity in the database and the structural schema. The database describes the completely new healthcare environment. The data in the database is not very high because it must be computed and have a clear understanding of it. All the data that is present in the database is manual imputation and is not hospital data nor some online data source.

Once the database is created and the tables are inserted into it then the next phase of running the queries for information retrieval is ready. The main aim is to facilitate seamless data sharing, thereby reducing potential exposure to infections for both patients and healthcare staff. The overarching goal is to simplify data processes, mitigate human errors, and ultimately enhance the precision and efficiency of patient care.

Analyzing database:

Analyzing the database will give more insights into the data. As shown in the given figures:

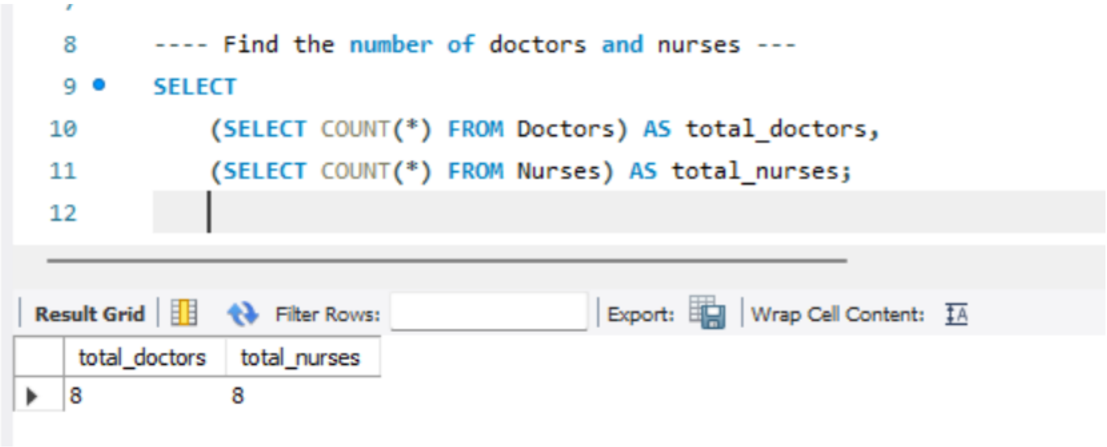
- This query will retrieve the count of patients. This will give a clear idea of how many patients are affected, administered, and treated in the hospital. This is how the information about patients is accessible and gives a clear idea about how the management is handling viruses in the hospital. This will also help in analyzing and finding the patient-to-doctor count in the hospital.



```
 ----- Retrieve the count of patients----
SELECT COUNT(*) AS total_patients
FROM Records_Patients;
```

```
  total_patients
▶ 12
```

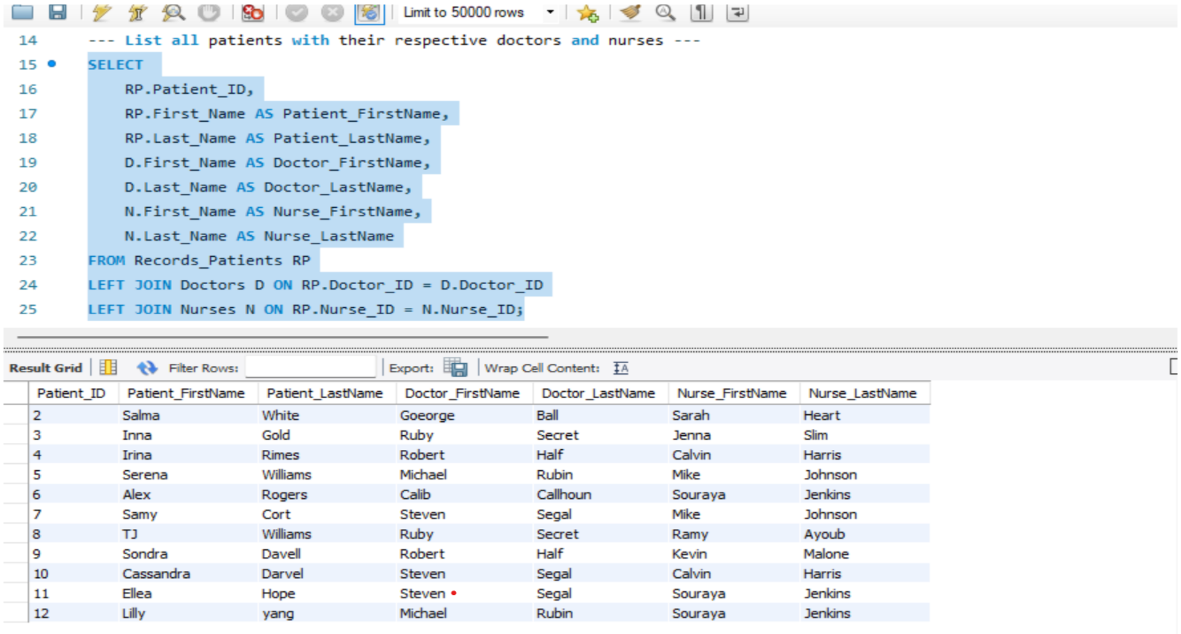
- To know the number of doctors and nurses in the hospital. This is very important as the patients are to be treated based on how many doctors are present in the hospital and can be managed for the post-administration process.



```
---- Find the number of doctors and nurses ---
SELECT
    (SELECT COUNT(*) FROM Doctors) AS total_doctors,
    (SELECT COUNT(*) FROM Nurses) AS total_nurses;
```

```
  total_doctors  total_nurses
▶ 8              8
```

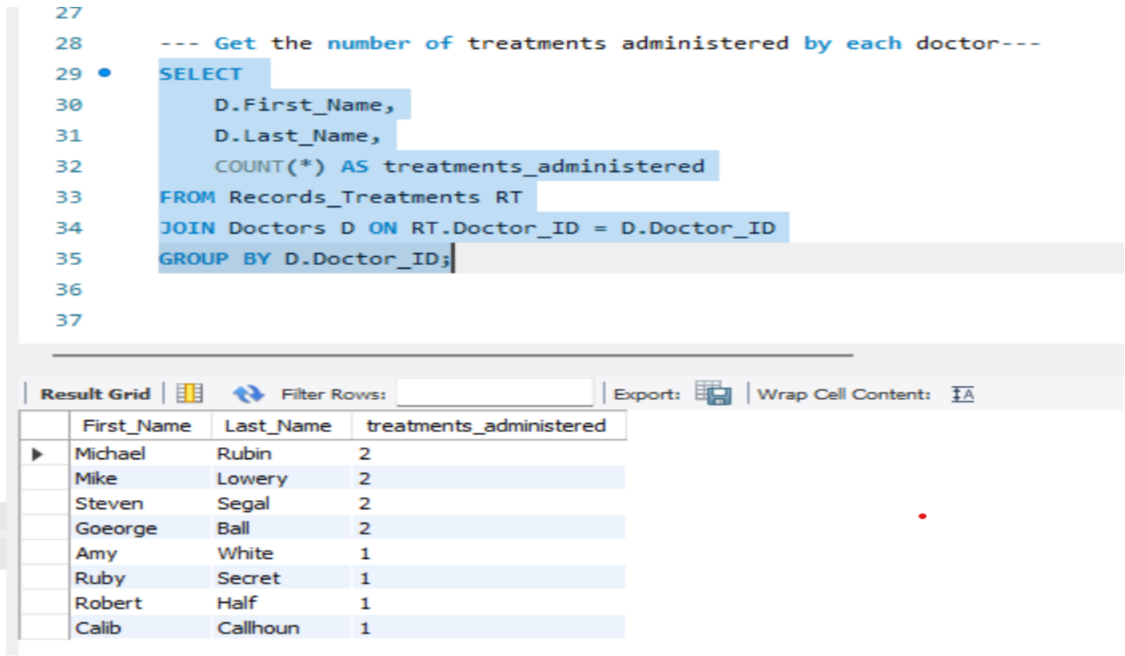
- To know which patient belongs to the respective doctor the following query will help to provide the information. This can also be tracked with doctor records with which patient the doctor has interacted with and the time of exposure to the virus.
- This is also done with the nurses as they are also in contact with the patients.



```sql
--- List all patients with their respective doctors and nurses ---
SELECT
    RP.Patient_ID,
    RP.First_Name AS Patient_FirstName,
    RP.Last_Name AS Patient_LastName,
    D.First_Name AS Doctor_FirstName,
    D.Last_Name AS Doctor_LastName,
    N.First_Name AS Nurse_FirstName,
    N.Last_Name AS Nurse_LastName
FROM Records_Patients RP
LEFT JOIN Doctors D ON RP.Doctor_ID = D.Doctor_ID
LEFT JOIN Nurses N ON RP.Nurse_ID = N.Nurse_ID;
```

Patient_ID	Patient_FirstName	Patient_LastName	Doctor_FirstName	Doctor_LastName	Nurse_FirstName	Nurse_LastName
2	Salma	White	Goeorge	Ball	Sarah	Heart
3	Inna	Gold	Ruby	Secret	Jenna	Slim
4	Irina	Rimes	Robert	Half	Calvin	Harris
5	Serena	Williams	Michael	Rubin	Mike	Johnson
6	Alex	Rogers	Calib	Callhoun	Souraya	Jenkins
7	Samy	Cort	Steven	Segal	Mike	Johnson
8	TJ	Williams	Ruby	Secret	Ramy	Ayoub
9	Sondra	Davell	Robert	Half	Kevin	Malone
10	Cassandra	Darvel	Steven	Segal	Calvin	Harris
11	Ellea	Hope	Steven	Segal	Souraya	Jenkins
12	Lilly	yang	Michael	Rubin	Souraya	Jenkins

- The below query will give how many treatments the doctor has done and will help analyze how long the doctor has been exposed to the virus and the risk factors that need to be considered.



```sql
--- Get the number of treatments administered by each doctor---
SELECT
    D.First_Name,
    D.Last_Name,
    COUNT(*) AS treatments_administered
FROM Records_Treatments RT
JOIN Doctors D ON RT.Doctor_ID = D.Doctor_ID
GROUP BY D.Doctor_ID;
```

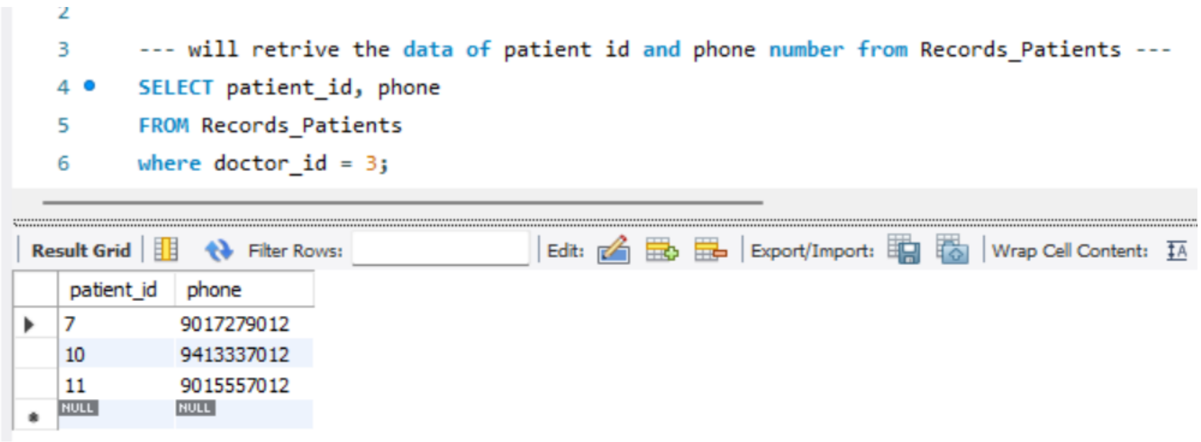
First_Name	Last_Name	treatments_administered
Michael	Rubin	2
Mike	Lowery	2
Steven	Segal	2
Goeorge	Ball	2
Amy	White	1
Ruby	Secret	1
Robert	Half	1
Calib	Callhoun	1

In this way, the hospital management will have a record of all the data of patients, doctors, and nurses. And is easy to access the required data at any time. This data can retrieve all the shift timings of the staff in this way it is easy to know the time the doctor or nurse is treating the patient and analyze whether the doctor or nurse is affected by the virus.

All this data together will give the staff exposure to the disease and enhance staff efficiency and productivity. This allows medical professionals to allocate more time and focus on direct patient care.

Performance Tuning:

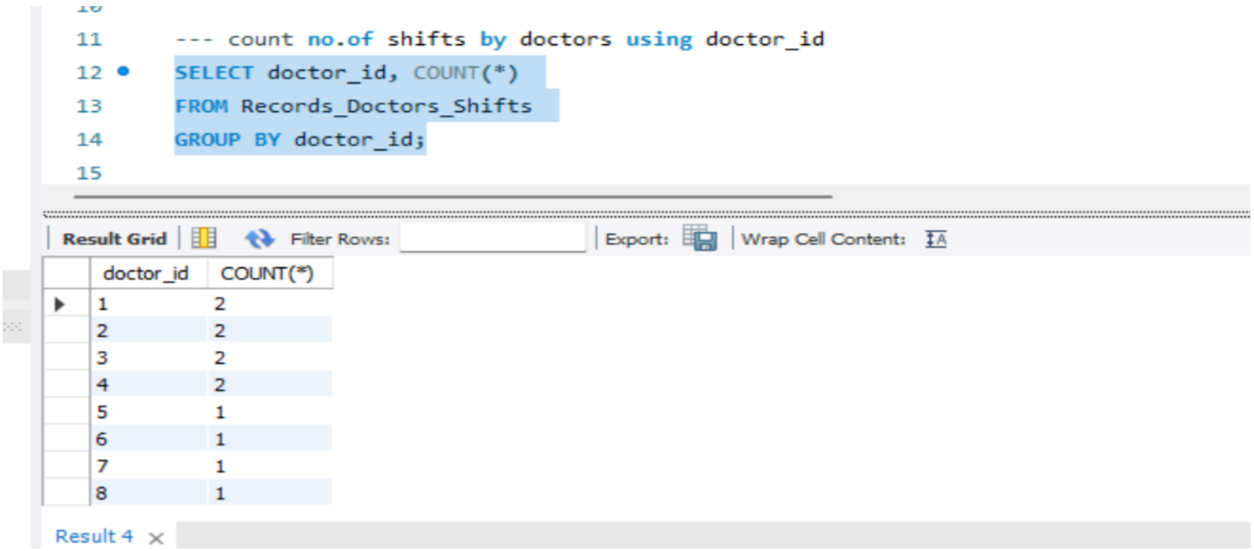
Performance tuning in a database involves optimizing queries, improving indexes, managing database configurations, and enhancing overall database performance.



```
--- will retrive the data of patient id and phone number from Records_Patients ---
SELECT patient_id, phone
FROM Records_Patients
where doctor_id = 3;
```

patient_id	phone
7	9017279012
10	9413337012
11	9015557012
NULL	NULL

The above code will retrieve the patient_id and phone number from the records_patients table and the doctor_id = 3. This is how the queries can be performed so that data of patients can be retrieved based on the doctor_id.



```
--- count no.of shifts by doctors using doctor_id
SELECT doctor_id, COUNT(*)
FROM Records_Doctors_Shifts
GROUP BY doctor_id;
```

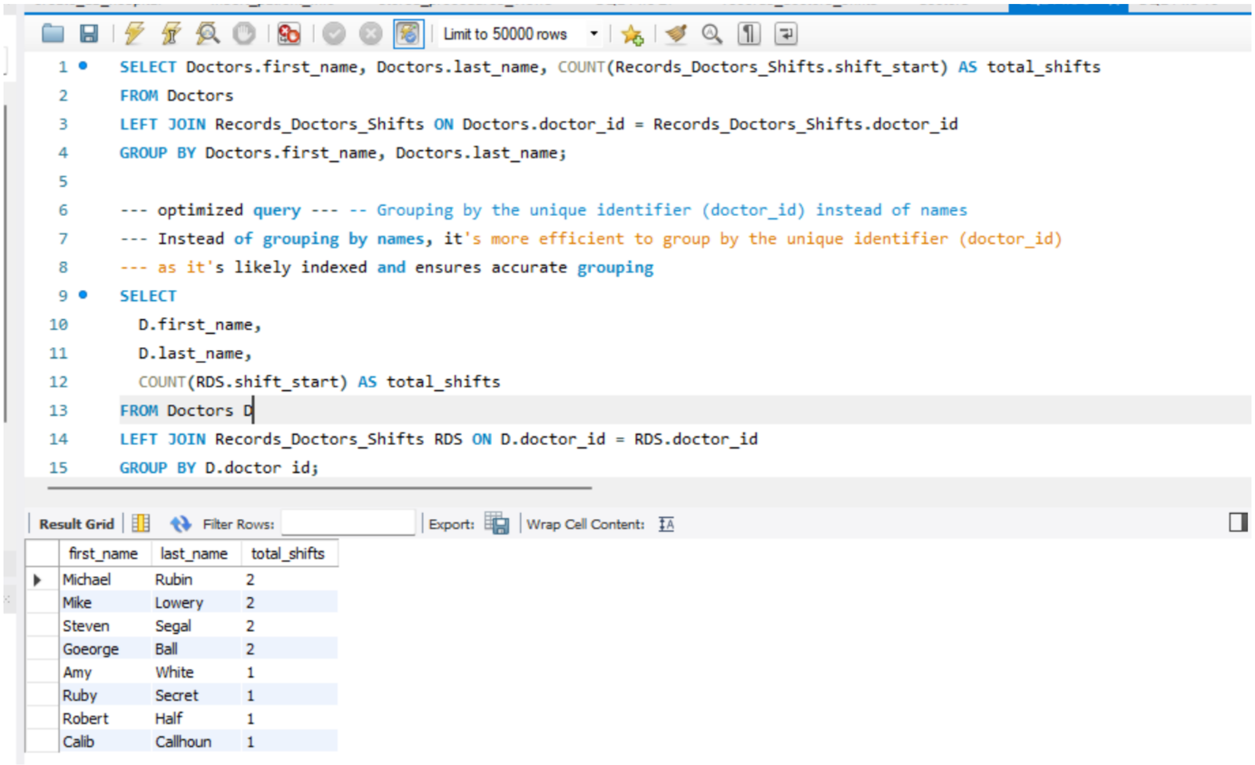
doctor_id	COUNT(*)
1	2
2	2
3	2
4	2
5	1
6	1
7	1
8	1

This optimized query will count the number of shifts the doctor has done and will display the total shifts the doctor has done.

The first query joins these tables based on the patient_id column and filters the result set to include only records where the admitted_date falls within the date range '2022-01-01' to '2022-12-31'.

Database Design | CIS-552 | Final Project

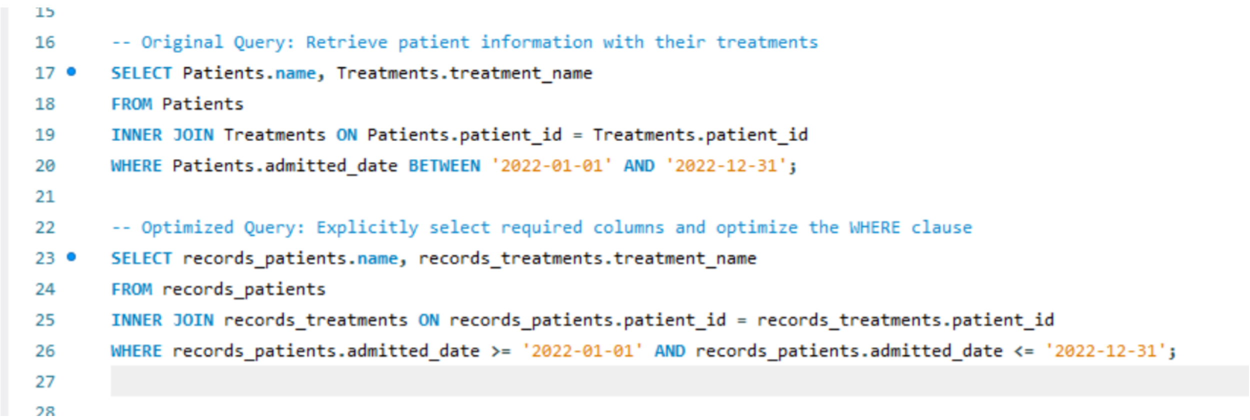
The second query filters records where the admitted_date in the records_patients table falls within the specified date range.



```
SELECT Doctors.first_name, Doctors.last_name, COUNT(Records_Doctors_Shifts.shift_start) AS total_shifts
FROM Doctors
LEFT JOIN Records_Doctors_Shifts ON Doctors.doctor_id = Records_Doctors_Shifts.doctor_id
GROUP BY Doctors.first_name, Doctors.last_name;

--- optimized query --- -- Grouping by the unique identifier (doctor_id) instead of names
--- Instead of grouping by names, it's more efficient to group by the unique identifier (doctor_id)
--- as it's likely indexed and ensures accurate grouping
SELECT
  D.first_name,
  D.last_name,
  COUNT(RDS.shift_start) AS total_shifts
FROM Doctors D
LEFT JOIN Records_Doctors_Shifts RDS ON D.doctor_id = RDS.doctor_id
GROUP BY D.doctor_id;
```

first_name	last_name	total_shifts
Michael	Rubin	2
Mike	Lowery	2
Steven	Segal	2
Goeorge	Ball	2
Amy	White	1
Ruby	Secret	1
Robert	Half	1
Calib	Callhoun	1

The EXPLAIN command is used to analyze the execution plan for the queries to understand how the database executes them and accesses the tables and indexes.



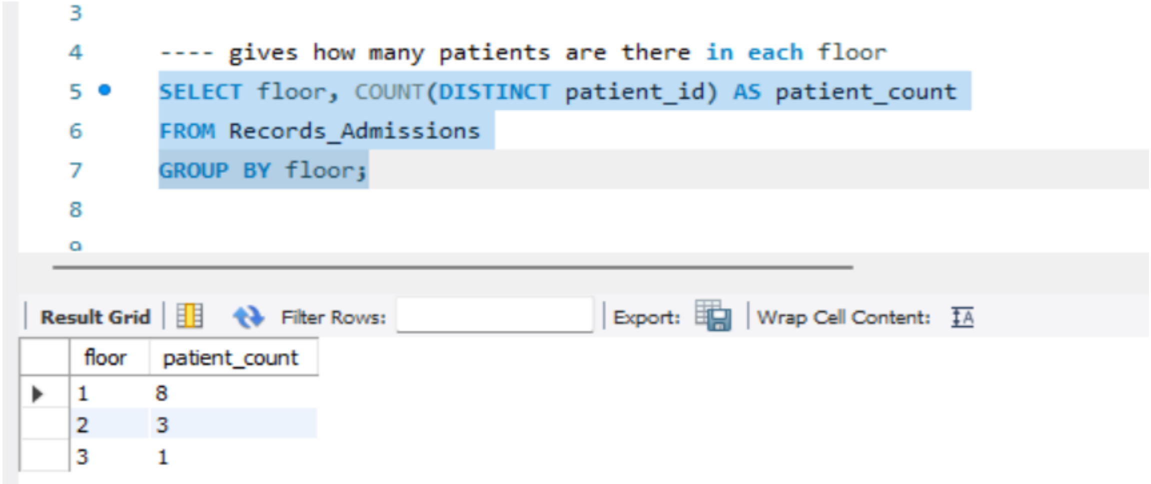
```
-- Original Query: Retrieve patient information with their treatments
SELECT Patients.name, Treatments.treatment_name
FROM Patients
INNER JOIN Treatments ON Patients.patient_id = Treatments.patient_id
WHERE Patients.admitted_date BETWEEN '2022-01-01' AND '2022-12-31';

-- Optimized Query: Explicitly select required columns and optimize the WHERE clause
SELECT records_patients.name, records_treatments.treatment_name
FROM records_patients
INNER JOIN records_treatments ON records_patients.patient_id = records_treatments.patient_id
WHERE records_patients.admitted_date >= '2022-01-01' AND records_patients.admitted_date <= '2022-12-31';
```

The first query retrieves the first_name and last_name of doctors from the Doctors table and counts the number of shifts each doctor has in the Records_Doctors_Shifts table by performing a left join between the Doctors and Records_Doctors_Shifts tables based on the doctor_id. The GROUP BY clause groups the results by the doctors' names to provide the total shifts for each doctor.

The second query, Similar to the first query, retrieves the first_name and last_name of doctors from the Doctors table and counts the number of shifts each doctor has in the Records_Doctors_Shifts table by performing a left join between the Doctors and Records_Doctors_Shifts tables based on the doctor_id. However, in this query, the GROUP BY clause is based on D.doctor_id.

Both queries essentially aim to retrieve a count of shifts for each doctor, but they differ in how they group the results. The first query groups by doctor names (first_name and last_name), while the second query groups by the doctor_id.

The following query will retrieve the information on how many patients are there on each floor. This helps in better room allocation and the working staff will know how many rooms are filled on each floor.



```
---- gives how many patients are there in each floor
SELECT floor, COUNT(DISTINCT patient_id) AS patient_count
FROM Records_Admissions
GROUP BY floor;
```

floor	patient_count
1	8
2	3
3	1

Questions and Solutions:

The below Tuple Relational Calculus (TRC) is a declarative language used to formulate queries in relational databases without giving a specific method of how to retrieve the data.

1. The below query solves the question of accessing patient information and will retrieve the information of patients by their treatment. In this way, the management will better enhance the hospital's patient information.

{ <p.name, t.treatment_name> | ∃p ∈ Patients, ∃t ∈ Treatments, p.patient_id = t.patient_id ∧ p.admitted_date >= '2022-01-01' ∧ p.admitted_date <= '2022-12-31' }

2. This is an optimized query for the above given TRC.
   ∃p ∈ records_patients and ∃t ∈ records_treatments represent "there exist" quantifiers over the records_patients and records_treatments relations, respectively.
   { <p.name, t.treatment_name> | ∃p ∈ records_patients, ∃t ∈ records_treatments, p.patient_id = t.patient_id ∧ p.admitted_date >= '2022-01-01' ∧ p.admitted_date <= '2022-12-31' }

3. This solves our first question of disease exposure reduction. The following query will give how many shifts did the doctor do which gives information on how long the doctor is exposed to the disease.
   Instead of grouping by names, it's more efficient to group by the unique identifier (doctor_id) as it's likely indexed and ensures accurate grouping.
   { <d.first_name, d.last_name, c> | ∃d ∈ Doctors, c = COUNT({ rds.shift_start | rds ∈ Records_Doctors_Shifts ∧ d.doctor_id = rds.doctor_id }) }

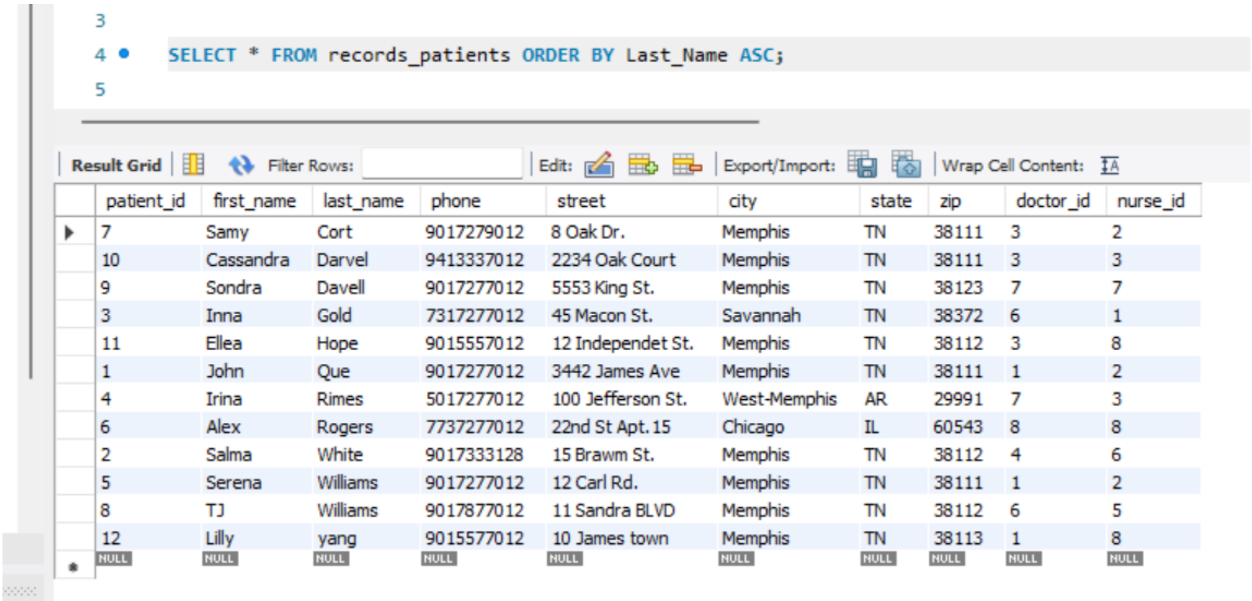
4. This solves our final question of enhanced staff efficiency. The following query will retrieve how many treatments the doctor administrates which helps in a way to reduce the exposure and maintain the count for better staff enhancement.
   {<d.First_Name, d.Last_Name, c> | ∃d ∈ Doctors, c = COUNT({ rt | rt ∈ Records_Treatments ∧ d.Doctor_ID = rt.Doctor_ID }) }

Limitations:

There are a few limitations to the database “hospital” and the query optimization. Below are the limitations mentioned.

1. Data:
   - The data that is available is less and is not true. This data serves only as a reference to build a database and manage access to the hospital management system for a more efficient and effective way of handling patients.
   - The database has records of patients, doctors, and nurses but it does not contain information on other working staff i.e., cleaning workers, drivers to run logistics, and other staff. They are indirectly in contact with the disease. Thus, the database lacks relevant information that is needed to maintain efficiency in the hospital.

2. Scalability:
   - This design might need to be improved in handling a large volume of data efficiently. As the database grows, performance issues might arise, affecting query speed and system responsiveness.
3. Performance Optimization:
   - Optimizing queries, indexing strategies, and database configuration is crucial for maintaining good performance.
4. Data Integrity:
   - Although foreign key constraints are in place, there might be scenarios where data integrity can be compromised, especially during complex transactions or when multiple users are accessing and modifying the database concurrently.
5. ORDER BY clause for sorting rows:
   - The limitation below is that it imposes an ordering on the result set.



```
SELECT * FROM records_patients ORDER BY Last_Name ASC;
```

patient_id	first_name	last_name	phone	street	city	state	zip	doctor_id	nurse_id
7	Samy	Cort	9017279012	8 Oak Dr.	Memphis	TN	38111	3	2
10	Cassandra	Darvel	9413337012	2234 Oak Court	Memphis	TN	38111	3	3
9	Sondra	Davell	9017277012	5553 King St.	Memphis	TN	38123	7	7
3	Inna	Gold	7317277012	45 Macon St.	Savannah	TN	38372	6	1
11	Ellea	Hope	9015557012	12 Independet St.	Memphis	TN	38112	3	8
1	John	Que	9017277012	3442 James Ave	Memphis	TN	38111	1	2
4	Irina	Rimes	5017277012	100 Jefferson St.	West-Memphis	AR	29991	7	3
6	Alex	Rogers	7737277012	22nd St Apt. 15	Chicago	IL	60543	8	8
2	Salma	White	9017333128	15 Brawm St.	Memphis	TN	38112	4	6
5	Serena	Williams	9017277012	12 Carl Rd.	Memphis	TN	38111	1	2
8	TJ	Williams	9017877012	11 Sandra BLVD	Memphis	TN	38112	6	5
12	Lilly	yang	9015577012	10 James town	Memphis	TN	38113	1	8
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Conclusion:

The conclusion of the document for the "Hospital" database design project would encapsulate the achievements and implications of the project. It would emphasize how the database effectively addresses the challenge of minimizing COVID-19 exposure in hospital settings by providing real-time, accurate patient data to medical staff. The conclusion would highlight the successful integration of various database elements, such as patient records, staff schedules, and treatment details, ensuring efficient hospital operations and improved patient care. It would also reflect on the project's contributions to data management in healthcare, noting any limitations encountered and suggesting future enhancements or areas for further research. The conclusion serves as a testament to the project's role in advancing healthcare data management in response to the pandemic.

Github:

The complete project details can be found in my github account by following this link,
[Covid19-Exposure-Database-Design]