USE CASE STUDY REPORT

Hospital Management System

Executive Summary:

The primary objective of this project was to design and create a relational database that can manage various activities of a hospital. This system will enable hospital managers to understand the current patterns of faculty, keep track of all patients, medical records, treatment provided, prescription, and the use of medicine and other medical supplies. By reducing the time required to manage medical records, maintaining faculty and patient details, and improving the efficiency and effectiveness of health and welfare services, this system will help to enhance the quality of care provided by the hospital.

We populated our dataset with the information of faculty (doctors and nurses), patients, medical records, and medicine to analyze and generate the insights about health records, staff, medicine management and other activities of the hospital. To create our sample database, we generated 14 different datasets, and analyzed them to ensure compatibility with the proposed model.

We created EER and UML diagrams to model the conceptual structure of the database and mapped it to a relational model using the required primary and foreign keys. This database was implemented in MySQL and MongoDB to study its feasibility in a NoSQL environment. The database can then be accessed and manipulated by connecting it to Python, allowing further and extensive analysis, as demonstrated in the report.

I. Introduction

The healthcare industry is the lifeblood for society, while hospitals are vital in saving lives. Unfortunately, many health care providers face challenges to offer active services to patients, or fail to provide health-related critical information in the crucial time and urgencies when needed most. In this project, we are going to build a Hospital Management System, which is an organized computerized system designed to deal with daily operations and management of hospital activities, to reduce that type of burden and to provide both information and management capabilities to a large variety of users.

In this system, we define the following entity types, their attributes, and their relationships as follow:

- Faculty member: The hospital contains information on faculty members, including doctors and nurses with specialized roles. Each faculty member's record contains personal information (ID, name, gender, date of birth, phone), specialty, working schedule.
- Patient: The system also includes registration of patients, including patient ID, name, gender, address. It also records some more detailed information, e.g. height, weight, blood type. A patient may be treated by multiple but at least one doctor; a doctor can treat zero to many patients.
- Room: A hospital has many rooms. Room ID, room type (standard, VIP) and capacity (single, 2-bed, 3-bed, 4-bed) will be recorded. A patient is assigned to 0 or exactly 1 room; a nurse is assigned to 0 to many rooms for monitoring duty.
- Medical record: Doctors will fill in medical records. For each medical record, the system includes record ID, date admitted, date discharged, diagnosis and treatment. Doctors and patients are

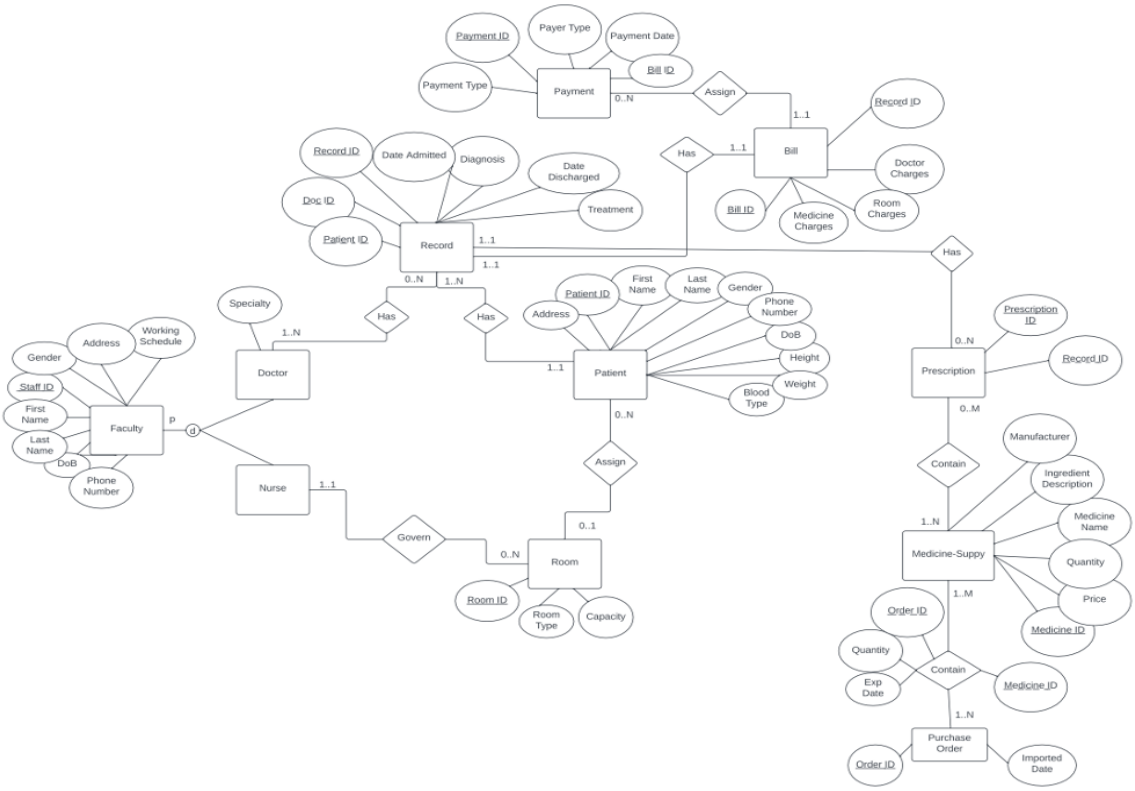
connected indirectly through medical records. A patient may have multiple but at least 1 record; a record contains exactly 1 patient, and 1 to many doctors.

- Medicine: The hospital needs to record data regarding drugs and supplies currently in stock and allows ordering them in advance. Each item contains ID, name, quantity, price, ingredient description and manufacturer.
- Prescription: Doctors may give patients prescriptions, which contain a list of prescribed medicine, quantity, which should be associated with the treatment process. A prescription is linked to exactly 1 medical record. 1 medical record may have one or many prescriptions.
- Bill: Bills are issued when the treatment process is completed. It contains medicine charges and other bill-able services (room charges, diagnosis fee). Bills are linked to a medical record.
- Payment: For each payment, the system includes payment ID, payment type (cash, debit card, credit card, PayPal). One payment can be assigned by exactly one bill, and vice versa.
- Order: Medicine orders will be recorded when the hospital purchases medicine. Order records contain medicine ID, quantity, and expiration date. One order can have 1 to many medicines, and 1 medicine can be in 1 to many orders.

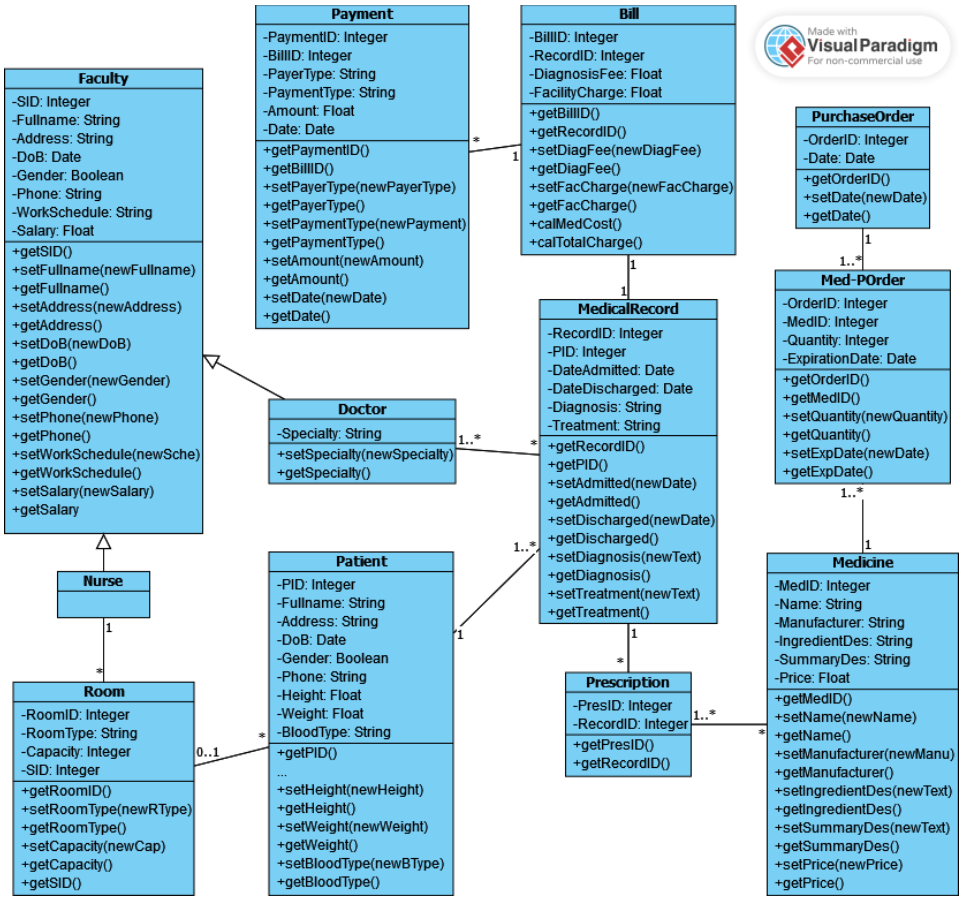
The system is designed for multispecialty hospitals and aims to aid in managing inpatient and outpatient care, health records, database treatments, medicine and supplies, billings in the pharmacy. It also maintains hospital information such as doctors in charge and rooms in the hospital. The users of this hospital management system may be hospital administration; doctors and other authorized employees.

II. Conceptual Data Modelling

1. EER Diagram



2. UML Diagram



III. Mapping Conceptual Model to Relational Model

FACULTY (staff_id, first_name, last_name, DoB, phone_number, gender, address, working_schedule)

- staff_id: primary key

DOCTOR (*doctor_id*, specialty)

- doctor_id: primary and foreign key refers to staff_id in FACULTY, NOT NULL

NURSE (*nurse_id*)

- nurse_id: primary and foreign key refers to staff_id in FACULTY, NOT NULL

RECORD (record_id, date_admitted, date_discharged, diagnosis, treatment, *patient_id*)

- record_id: primary key
- patient_id foreign key refers to nurse_id in PATIENT, NOT NULL

DOCTOR_CASES (*record_id*, *doctor_id*)

- The combination of record_id and doctor_id is primary key of DOCTOR-RECORD, so they are NOT NULL
- record_id: foreign key refers to record_id in RECORD
- doctor_id: foreign key refers to doctor_id in DOCTOR

PATIENT (patient_id, first_name, last_name, DoB, phone_number, gender, address, height, weight, blood_type, *room_id*)

- patient_id: primary key
- room_id: foreign key refers to room_id in ROOM, NULL ALLOWED

ROOM (room_id, room_type, capacity, *nurse_id*)

- room_id: primary key
- nurse_id: foreign key refers to nurse_id in NURSE, NULL ALLOWED

BILL (bill_id, medicine_charges, room_charges, doctor_charges, *record_id*)

- bill_id: primary key
- record_id: foreign key refers to record_id in RECORD, NOT NULL

PAYMENT (payment_id, payment_type, payer_type, payment_date, *bill_id*)

- payment_id: primary key
- bill_id: foreign key refers to bill_id in BILL, NOT NULL

PRESCRIPTION (prescription_id, *record_id*)

- prescription_id: primary key
- record_id: foreign key refers to record_id in RECORD, NOT NULL

MEDICINE (medicine_id, name, price, quantity, ingredient_description, manufacturer)

- medicine_id: primary key

PRESCRIPTION_CONTENT (*medicine_id*, *prescription_id*, quantity, note)

- The combination of medicine_id and prescription_id is primary key of MEDICINE-PRESCRIPTION, so they are NOT NULL
- medicine_id foreign key refers to medicine_id in MEDICINE
- prescription_id foreign key refers to prescription_id in PRESCRIPTION

PURCHASE_ORDER (order_id, import_date)

- order_id: primary key

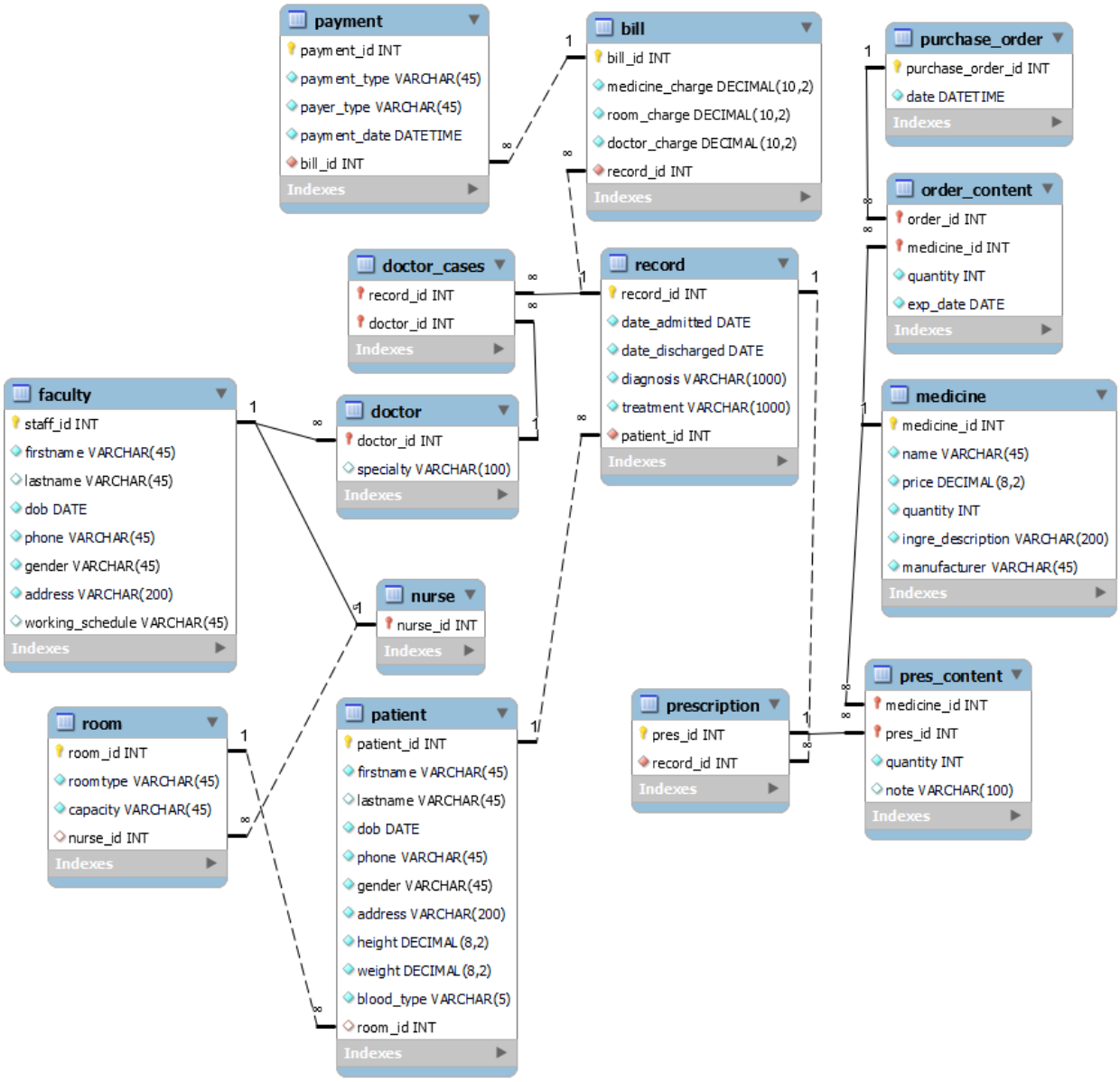
ORDER_CONTENT (*order_id*, *medicine_id*, quantity, exp_date)

- The combination of medicine_id and order_id is primary key of MEDICINE-ORDER, so they are NOT NULL
- order_id foreign key refers to order_id in PURCHASE_ORDER
- medicine_id foreign key refers to medicine_id in MEDICINE

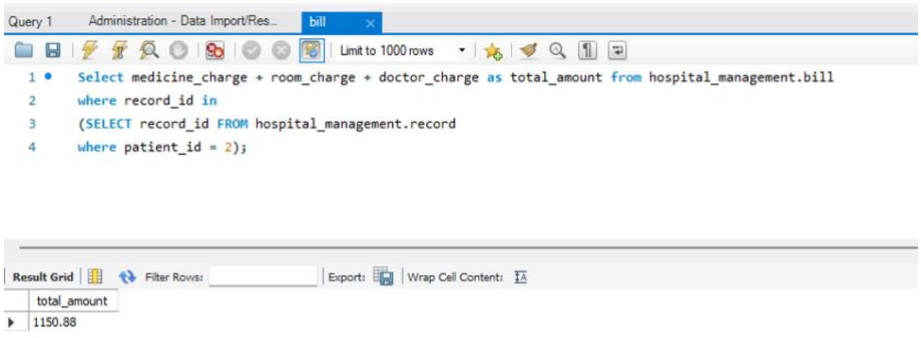
IV. Implementation of Relation Model via MySQL and NoSQL

1. MySQL Implementation

Our implementation consists of 14 tables using the 14 relations from our relational model.

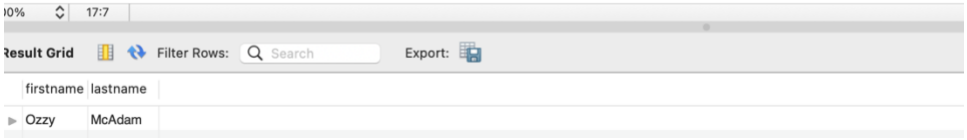


Query 1: Show billing amount of all bills of patient with ID 2 (NESTED QUERY)



Query 2: Show the name of all patients at the rooms monitored by the nurse with ID 396 (NESTED QUERY)

```sql
-- Show the name of all patients at the rooms monitored by the nurse with ID 396 (Nested query)
SELECT firstname, lastname
FROM patient
WHERE patient_id IN
(SELECT patient_id
FROM patient
WHERE room_id IN
(SELECT room_id
FROM room
WHERE nurse_id = 396))
```

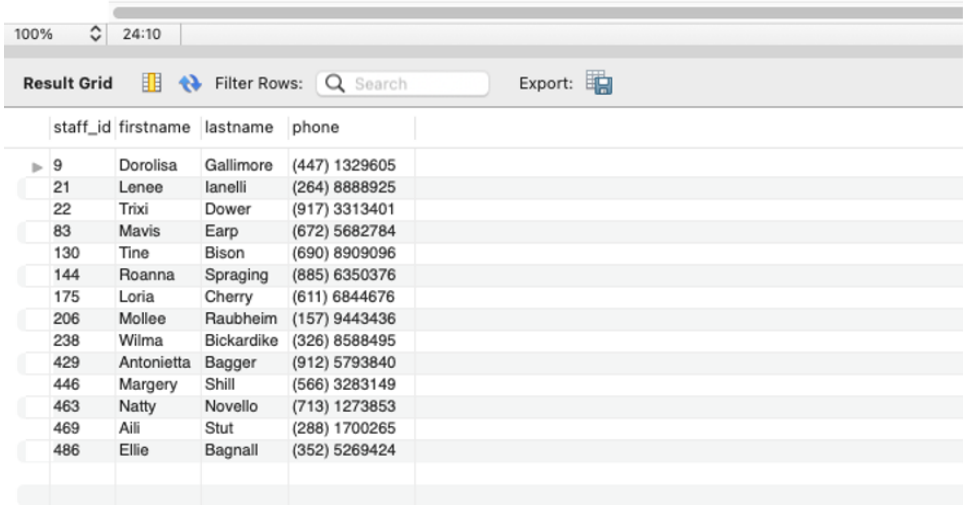


Result Grid | Filter Rows: Search | Export:

firstname	lastname
Ozzy	McAdam

Query 3: Display female nurses working from Wednesday to Saturday (JOIN)

```sql
-- Display female nurses working from Wednesday to Saturday
SELECT staff_id, firstname, lastname, phone
FROM faculty f
JOIN nurse n
ON f.staff_id = n.nurse_id
WHERE gender = "Female" AND working_schedule LIKE "Wednesday - Saturday%";
```



Result Grid | Filter Rows: Search | Export:

staff_id	firstname	lastname	phone
9	Dorolisa	Gallimore	(447) 1329605
21	Lenee	Ianelli	(264) 8888925
22	Trixi	Dower	(917) 3313401
83	Mavis	Earp	(672) 5682784
130	Tine	Bison	(690) 8909096
144	Roanna	Spraging	(885) 6350376
175	Loria	Cherry	(611) 6844676
206	Mollee	Raubheim	(157) 9443436
238	Wilma	Bickardike	(326) 8588495
429	Antonietta	Bagger	(912) 5793840
446	Margery	Shill	(566) 3283149
463	Natty	Novello	(713) 1273853
469	Aili	Stut	(288) 1700265
486	Ellie	Bagnall	(352) 5269424

Query 4: Calculate total medicine_charges of patients according to blood_type (AGGREGATE)

```sql
-- Calculate total medicine_charges of patients according to blood_type
SELECT  blood_type, SUM(medicine_charge) AS total_medicine_charge
FROM patient p
JOIN record r
    ON p.patient_id = r.patient_id
JOIN bill b
    ON r.record_id = b.record_id
GROUP BY blood_type;
```

100% 12:21

Result Grid | Filter Rows: Search | Export:

blood_type	total_medicine_charge
A	87657.82
AB	169052.48
B	81953.32
O	331655.28

Query 5: Display the medicine that has price of more than 40, quantity >80 and has "hydrochloride" in description

```sql
-- Display the medicine that has price of more than 40, quantity >80 and has "hydrochloride" in description
SELECT *
FROM medicine
WHERE price > 40 AND quantity >80 AND ingre_description LIKE "%hydrochloride";
```

100% 79:4

Result Grid | Filter Rows: Search | Edit: | Export/Import:

medicine_id	name	price	quantity	ingre_description	manufacturer
7	equaline nasal	45.35	82	oxymetazoline hydrochloride	Supervalu Inc
38	Lidocaine Hydrochloride	40.41	99	Lidocaine Hydrochloride	Cardinal Health
116	Mucinex	45.89	101	acetaminophen, dextromethorphan hydrobromid...	Reckitt Benckiser LLC
137	TAMSULOSIN HYDROCHLORIDE	42.38	85	TAMSULOSIN HYDROCHLORIDE	WOCKHARDT LIMITED
236	NUCYNTA	45.98	123	tapentadol hydrochloride	Stat Rx USA
410	Amiodarone Hydrochloride	42.06	101	Amiodarone Hydrochloride	General Injectables & Vaccines, Inc.
418	Trihexyphenidyl Hydrochloride	47.76	89	trihexyphenidyl hydrochloride	Qualitest Pharmaceuticals
423	Allergy Relief	41.02	106	cetirizine Hydrochloride	CVS Pharmacy
434	ropinirole hydrochloride	41.63	125	ropinirole hydrochloride	Zydus Pharmaceuticals (USA) Inc.
486	Dicyclomine Hydrochloride	46.44	105	Dicyclomine Hydrochloride	A-S Medication Solutions LLC
515	Procainamide Hydrochloride	47.36	112	PROCAINAMIDE HYDROCHLORIDE	Cardinal Health
567	FLUOXETINE	41.85	123	fluoxetine hydrochloride	Bryant Ranch Prepack
577	Terazosin Hydrochloride	40.97	117	Terazosin Hydrochloride	State of Florida DOH Central Phar...
588	Metformin Hydrochloride	47.26	103	Metformin Hydrochloride	NCS HealthCare of KY, Inc dba Va...
618	Cyclopentolate Hydrochloride	49.53	130	Cyclopentolate Hydrochloride	Akorn, Inc.
640	Extra Strength Acetaminophen PM	43.18	113	ACETAMINOPHEN, DIPHENHYDRAMINE HY...	McKesson
655	eLenza Patch	41.45	119	Lidocaine Hydrochloride	Pharmaceutics Corporation
721	Clindamycin hydrochloride	43.25	103	Clindamycin hydrochloride	Preferred Pharmaceuticals, Inc
778	Burn	46.54	132	Lidocaine Hydrochloride	First Aid Only, Inc
NULL	NULL	NULL	NULL	NULL	NULL

Query 6: Show any room that is not housing any patient (NOT EXISTS)

```sql
SELECT * FROM hospital_management.room r WHERE NOT EXISTS
    (SELECT * FROM hospital_management.patient p WHERE p.room_id = r.room_id)
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

room_id	roomtype	capacity	nurse_id
510	VIP	single	306
NULL	NULL	NULL	NULL

Query 7: Retrieve all patients who have at least 2 medical records (CORRELATED QUERY)

```sql
-- Retrieve all patients who have at least 2 medical records
SELECT firstname, lastname
FROM patient p
WHERE
    (SELECT COUNT(*)
     FROM record r
     WHERE p.patient_id = r.patient_id) >= 2
```

100% 45:59

Result Grid Filter Rows: Search Export:

firstname	lastname
Peyter	Stollberger
Wiatt	Ludlom
Bearnard	Grayer
Kingsley	Seefus
Danya	Dey
Kippy	Aikenhead
Filberto	Lundbech
Tatiana	Paynton
Horacio	Milan
Jerrie	Birchenhead
Georgena	O' Kelleher
Tandy	Altree
Borg	Berriman
Alonzo	Selwyn
Tuckie	MacPaik
Vick	Vallis
Kort	Castles
Kyle	Burdass
Daffy	Brito
Wait	Hinze
Trumaine	Gwinnel
Billy	Shier

Query 8: Show all patients with record of being treated by "pediatrics" doctors or have "cold" in their diagnosis (UNION)

```sql
-- Show all patient with record of being treated by "Pediatrics" doctors or have "cold" in their diagnosis
SELECT * FROM hospital_management.patient WHERE patient_id IN
    ((SELECT patient_id FROM hospital_management.record WHERE record_id IN
        (SELECT record_id FROM hospital_management.doctor_cases WHERE doctor_id IN
            (SELECT doctor_id FROM hospital_management.doctor WHERE specialty = "Pediatrics")))
    UNION
    (SELECT patient_id FROM hospital_management.record WHERE diagnosis LIKE "%cold%"))
```

Result Grid Filter Rows: Export: Wrap Cell Content:

patient_id	firstname	lastname	dob	phone	gender	address	height	weight	blood_type	room_id
6	Peyter	Stollberger	1992-02-06	(702) 7189711	Male	391 Quincy Street,North Las Vegas,NV	168.00	145.91	B	602
16	Rockwell	Linge	1957-08-11	(727) 6387087	Male	5417 Laurel Way,Saint Petersburg,FL	161.75	144.08	O	NULL
24	Woodrow	Boor	1974-05-08	(303) 5619415	Male	22073 Porter Terrace,Denver,CO	160.25	161.72	O	115
30	Berrie	Wellfare	1953-02-27	(412) 7596150	Female	4701 Gerald Road,Pittsburgh,PA	181.25	147.09	O	615
44	Fawne	Penberthy	1970-01-23	(763) 1628792	Female	65556 Warrior Center,Loretto,MN	163.75	111.99	A	NULL
47	Violet	Beames	1971-11-06	(619) 8061081	Female	741 Mayer Hill,San Diego,CA	186.25	167.50	AB	NULL
49	Leicester	Eastop	1993-11-28	(704) 8519844	Male	08080 Eagle Crest Place,Charlotte,NC	154.00	138.13	O	NULL
53	Justen	Paik	1990-06-10	(214) 9054756	Male	15671 Killdeer Plaza,Garland,TX	163.25	169.60	A	NULL
59	Kingsley	Seefus	1950-08-01	(951) 3451480	Male	68557 3rd Crossing,Riverside,CA	165.00	156.52	AB	NULL
76	Thea	Cawthorn	1983-08-27	(318) 2633348	Female	0 Donald Place,Monroe,LA	172.50	137.25	O	NULL
81	Tatiana	Paynton	1965-10-30	(772) 9045192	Female	2 Charing Cross Avenue,Port Saint Lu...	178.50	196.98	O	310
97	Ardeen	Raftery	2011-03-26	(704) 2371170	Female	28981 Duke Point,Charlotte,NC	168.00	120.53	O	NULL
126	Vick	Vallis	1993-12-19	(724) 2619532	Male	438 Welch Street,Pittsburgh,PA	159.50	125.91	O	NULL
169	Hazel	Goodread	1947-08-14	(702) 2577589	Male	1580 Thompson Avenue,Las Vegas,NV	171.25	182.77	A	NULL
174	Axe	Juliano	1954-11-21	(972) 6409999	Male	8 Morrow Avenue,Dallas,TX	156.50	159.44	O	NULL
177	Lorilee	Chiplen	2021-06-02	(646) 5445314	Female	45 Donald Pass,New York City,NY	110.75	42.44	O	NULL
178	Stephine	Bunkle	2003-01-27	(859) 5321905	Female	488 Lakewood Gardens Terrace,Lexin...	172.50	160.32	O	119
186	Leena	Curm	1986-08-26	(781) 4055690	Female	557 Ryan Plaza,Boston,MA	160.00	156.42	O	NULL
210	Rancell	Calan	1964-05-05	(615) 4887536	Male	340 Mallard Plaza,Nashville,TN	161.25	153.74	O	NULL
218	Alicia	Antonelli	2003-09-06	(601) 1049574	Female	2790 Barby Drive,Jackson,MS	174.50	185.09	AB	508
222	Matilda	Biggin	2000-12-12	(858) 7670396	Female	9 Summer Ridge Pass,San Diego,CA	173.00	168.86	O	NULL
235	Davidson	Cosson	1994-12-31	(909) 4234694	Male	59854 Fuller Lane,San Bernardino,CA	170.00	183.94	O	NULL
236	Keary	Macci	1954-10-24	(415) 7320080	Male	2 Maple Road,San Francisco,CA	161.25	175.30	B	NULL

Query 9: Retrieve the manufacturer that charge the highest price for the medicine "Nicotine" (ALL)

```
62    -- Retrieve the manufacturer that charge the highest price for the medicine "Nicotine"
63    SELECT manufacturer
64    FROM medicine
65    WHERE name = "Nicotine"
66    AND price >= ALL
67        (SELECT price
68         FROM medicine
69         WHERE name = "Nicotine")
70
71
72
73
```

100% 1:71

Result Grid | Filter Rows: Search | Export:

manufacturer
Walgreen Company

Query 10: Retrieve patient_id, patient name, total number of medical records and total amount of medical charges for each patient (Subquery in Select clause)

```
1  select patient_id, firstname, lastname,
2      (select count(*)
3       from record r1
4       where r1.patient_id = p.patient_id) AS number_of_records,
5       (select sum(medicine_charge + room_charge + doctor_charge)
6        from bill b, record r2
7        where b.record_id = r2.record_id and r2.patient_id = p.patient_id) as total_medicine_charge
8  from patient p
```

100% 50:5

Result Grid | Filter Rows: Search | Export:

patient_id	firstname	lastname	number_of_recor...	total_medicine_charge
1	Erv	Wyd	1	1328.34
2	Renault	Scrimshaw	1	1150.88
3	Marc	Zielinski	1	727.96
4	Ozzy	McAdam	1	1476.21
5	Bernadina	Lelievre	1	1557.38
6	Peyter	Stollberger	2	2234.31
7	Elias	Ecclestone	1	1641.25
8	Emory	Sprouls	1	1320.95
9	Ervin	Kembery	1	1638.67
10	Garvy	Yerson	1	1701.71
11	Sarena	Collerd	1	435.80
12	Broddy	Brushfield	1	1430.35
13	Joyce	Bleddon	1	656.29
14	Nona	Fish	1	519.65
15	Moritz	Le Noir	1	962.78

2. NoSQL Implementation

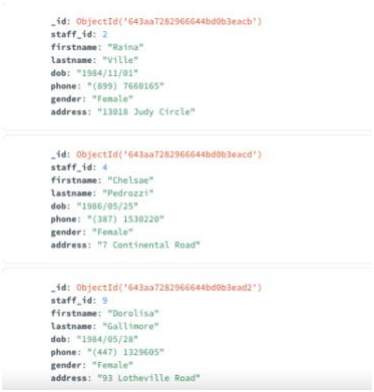
Our choice of NoSQL database management system is MongoDB. We implemented a simplified version of the system database in MySQL. This database consists of 7 collections equivalent to 7 tables/relations from the MySQL database: patient, doctor, record, doctor_cases, prescription, medicine, pres_content

We use the latest version of MongoDB and its GUI MongoDB Compass for our implementation.

The following queries were performed:

Query 1: Show all female faculty having year of birth greater than or equal 1980 and less than or equal 1990

```
db.faculty.find({
        $and: [
          {gender:"Female"},
          {dob:{$gte:"1980-01-01", $lte:"1990-01-01"}}
        ]
})
```

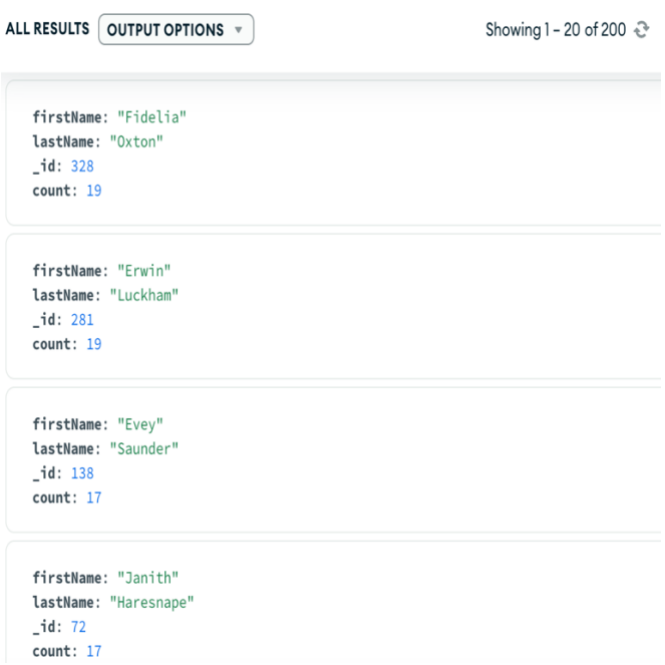


Query 2: Show all doctor's specialties and count how many doctors with that specialty, in descending order.

```
db.doctor.aggregate([
  {
    $group:
      {
        _id: "$specialty",
        count: { $sum: 1, },
      },
  },
  { $sort: { count: -1, }, },
])
```

Query 3: Show all doctors' names, and the number of cases/records they are responsible for in descending order.

```
db.doctor_cases.aggregate([
  {$group: {_id: "$doctor_id",count: { $sum: 1, },},},
  { $sort: { count: -1, }, },
  {$lookup:{from: "faculty",
        localField: "_id",
        foreignField: "staff_id",
        pipeline: [{$project: {
              _id: 0,
              firstName: "$firstname",
              lastName: "$lastname",
              specialty: "$specialty",},},],
        as: "result",},},{
    $replaceRoot: {newRoot: { $mergeObjects: [ { $arrayElemAt: ["$result", 0], }, "$$ROOT", ], },},},{ $project: { result: 0, },},])
```



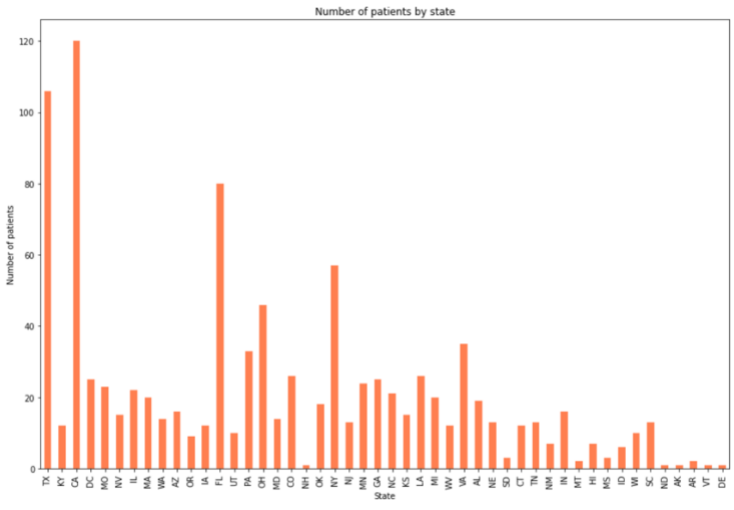
Query 4: Show all medicines and the total amount as TotalQuantity (in descending order) from their appearances in prescriptions.

```
db.pres_content.aggregate([{
    $group:{_id: "$medicine_id",
    TotalQuantity: { $sum: "$quantity", },},},
  { $sort: { TotalQuantity: -1, }, },{
    $lookup:{
        from: "medicine",
        localField: "_id",
        foreignField: "medicine_id",
        as: "result",
      },},{
    $replaceRoot: {newRoot: { $mergeObjects: [ { $arrayElemAt: ["$result", 0], }, "$$ROOT", ], },},},{ $project: { result: 0, }, },])
```

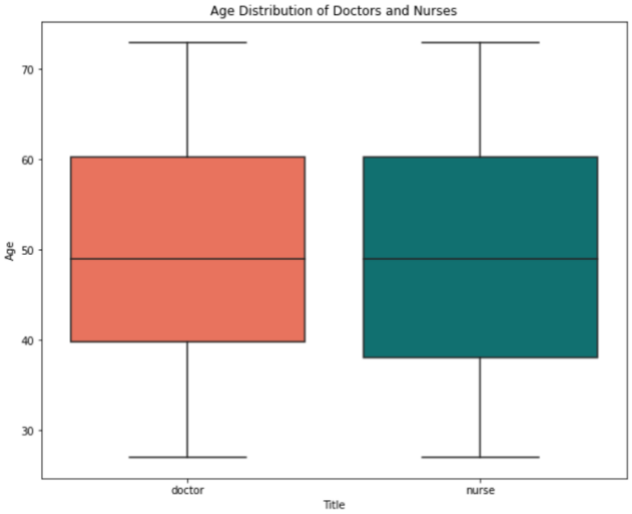
V. Database Access via Python

The connection to MySQL database server is established using mysql.connector library in Python. The library also provides methods for retrieving the data by sending literal SQL querying in string. The queried data is then imported into a DataFrame object, allowing further analysis in the Python environment. The following visualizations were performed.

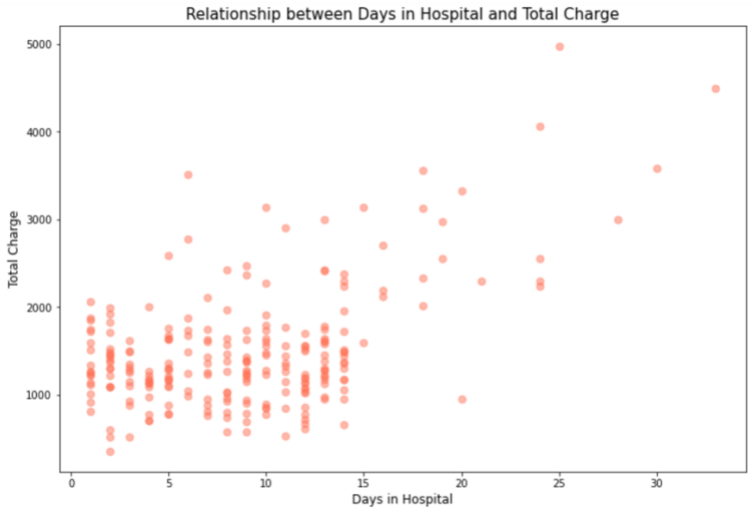
Graph 1: Number of patients by state



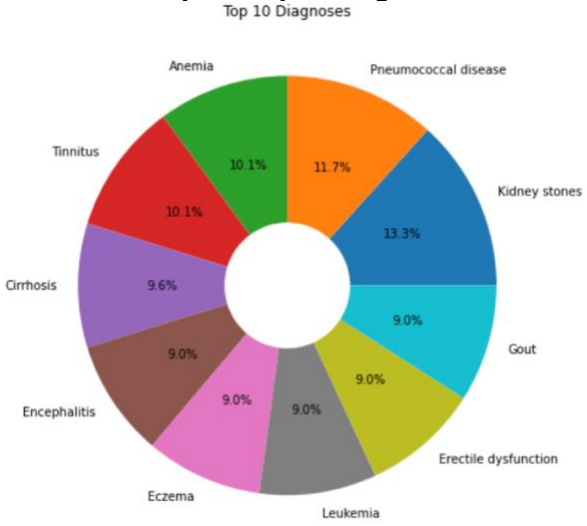
Graph 2: Age distribution of doctors and nurses



Graph 3: The correlation between the number of days patient staying in the hospital and total charge



Graph 4: Top 10 diagnosis



Summary and Recommendation

The Hospital Management System is a comprehensive relational database that can be implemented in any hospital. It aims to enhance the effectiveness and efficiency of major hospital activities, resulting in better healthcare services. In addition, by utilizing Python, useful insights can be analyzed and visualized, helping hospital managers gain a better understanding of various aspects of hospital operations, such as patient and staff information. For instance, through visualization, hospital managers can identify the top 10 diagnoses in the hospital, enabling them to prepare sufficient medicine or hire more doctors with specialties in those areas. Such insights can lead to better decision-making and more informed resource allocation. Overall, the Hospital Management System can help hospitals provide high-quality healthcare services while optimizing their operations.

Additional features can be implemented to extend the hospital's operations based on specific need. For example, diagnosis and treatment can be expanded to include more details, like image, machinery, elaborated procedure.

The current users of this system mostly are hospital administration, doctors, and other authorized employees. Therefore, improvement can be made to accommodate patient users, by creating views that allow patients to view their medical records or schedule appointments, helping them to have a better understanding of their health and treatment options. This can increase transparency in the patient-provider relationship, leading to better trust and patient satisfaction. Also, hospital staff can spend less time on administrative tasks and more time providing patient care. This can increase efficiency and reduce costs for the hospital.