USE CASE STUDY REPORT

Group 10
Manikandan Mohan
Kiran Tamilselvan

(857) 654-7863
(857) 654-7848

mohan.man@northeastern.edu
tamilselvan.k@northeastern.edu

Percentage of Effort Contributed by Student1:_50%___

Percentage of Effort Contributed by Student2:___50%_

Signature of Student 1: _Manikandan Mohan________

Signature of Student 2: _Kiran Tamilselvan__________

Submission Date:_12/08/2023________________________

USE CASE STUDY REPORT

Group No: Group 10

Student Names: Manikandan Mohan & Kiran Tamilselvan

Transforming Healthcare: Streamlined Patient Care Coordination Through Comprehensive Database Model

Executive Summary:

The healthcare database model represents a significant advancement in managing medical information. By centralizing crucial data such as patient records, healthcare provider details, appointment schedules, medical history, and medication information, it greatly simplifies information management. This system is adept at handling the complex relationships between patients, healthcare providers, and medical treatments. Its key features include assigning a unique identification to each patient for accurate record-keeping, enabling streamlined selection of healthcare providers based on their specialization, facilitating efficient scheduling of appointments, and maintaining detailed records of medications and prescriptions. The implementation of this model enhances the efficiency and precision of patient care coordination, leading to improved healthcare delivery. It significantly reduces administrative workloads, ensuring that patients receive quicker and more effective healthcare services.

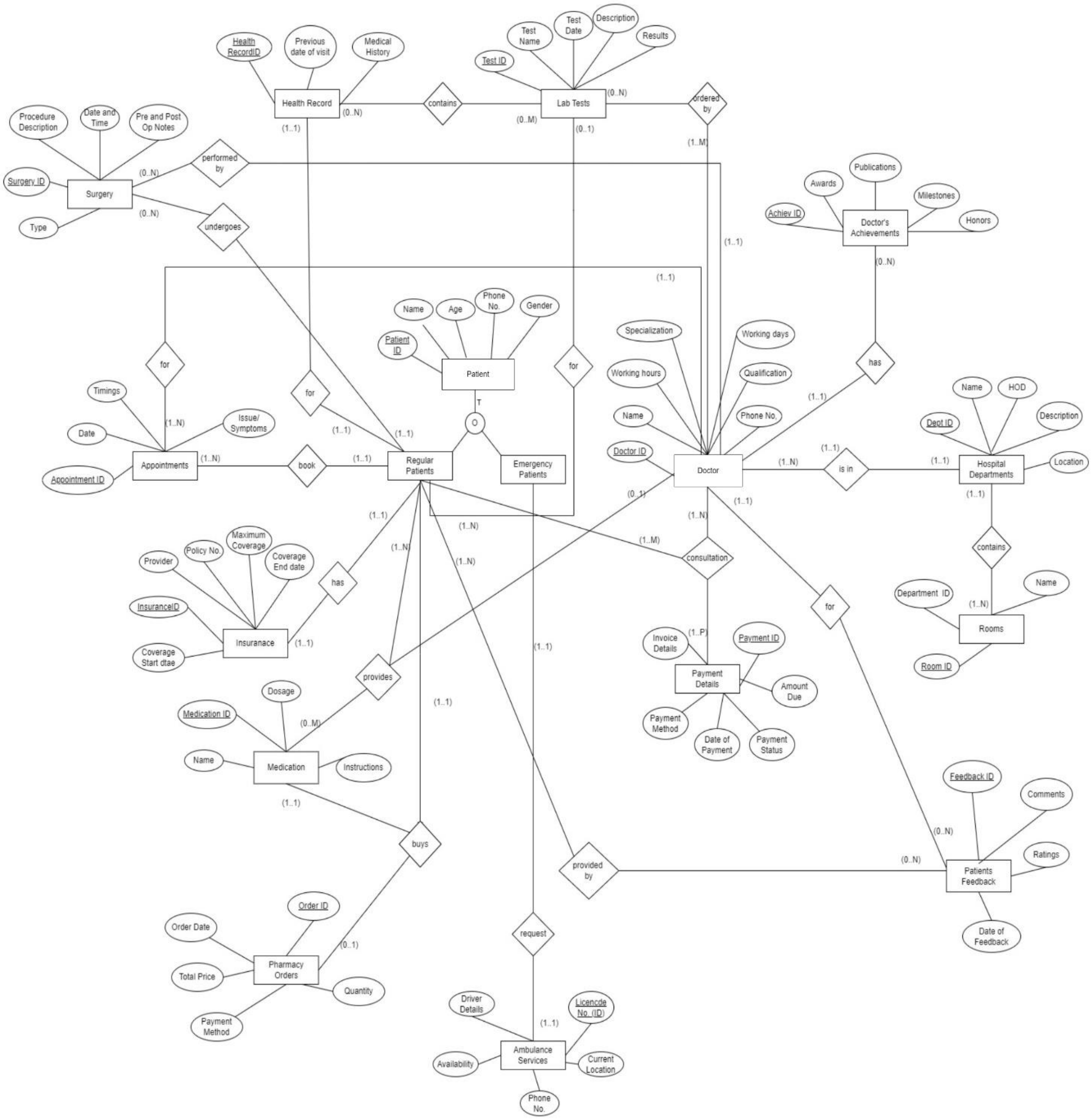
The EER and UML diagrams were modelled, followed by the mapping of the conceptual model to a relational model mapping with the required primary and foreign keys. This database was then implemented fully MySQL with twenty-three tables and was also implemented on MongoDB NoSQL database to study the feasibility of this database in a NoSQL environment. The created database is a great success, and by connecting it to Python in jupyter notebook, and some visualizations have been made by using scatter plot, bar plot and histogram. These queries can be very helpful tracking patient details, how payments have been made by the patients, how many appointments have been handled by the doctor for a particular period of the year and many more such queries.

Introduction:

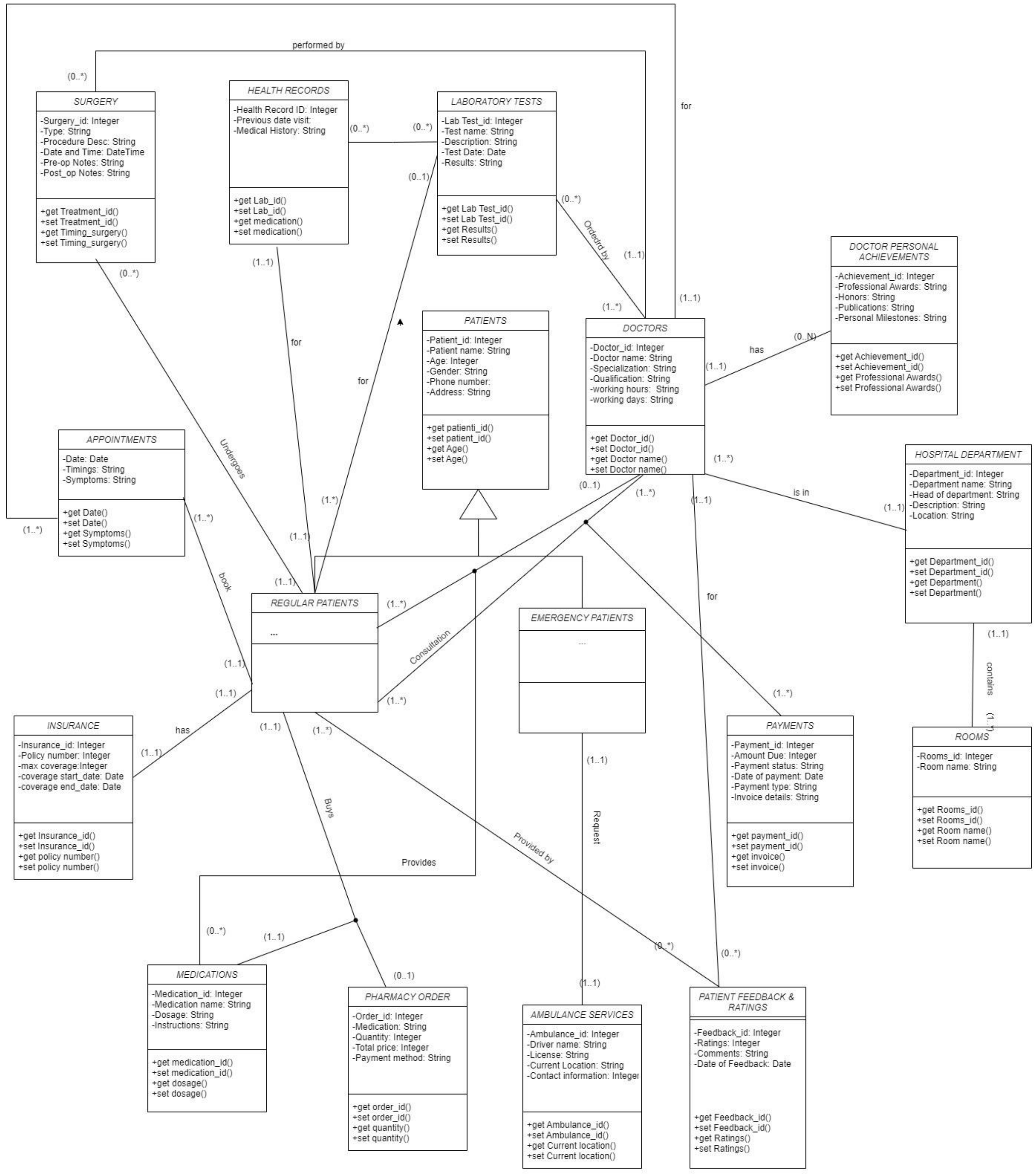
In the dynamic landscape of modern healthcare, efficient data management is critical to delivering high-quality patient care. The development of a comprehensive healthcare database model is a crucial step towards revolutionizing how medical information is stored, accessed, and utilized. This model is more than a mere repository of data; it is an integrated system that adeptly manages intricate relationships between various healthcare elements - from patient records and provider specialties to appointment schedules and medication prescriptions. Its implementation signifies a major leap forward in streamlining healthcare processes, enhancing decision-making, and ultimately improving service delivery. This introduction will explore the foundational aspects of the healthcare database model, delving into its functionality, benefits, and the profound impact it holds for the future of healthcare.

CONCEPTUAL MODELLING:

EER MODEL:



UML DIAGRAM:


performed by
for
(0..*)
SURGERY
-Surgery_id: Integer
-Type: String
-Procedure Desc: String
-Date and Time: DateTime
-Pre-op Notes: String
-Post_op Notes: String
+get Treatment_id()
+set Treatment_id()
+get Timing_surgery()
+set Timing_surgery()
HEALTH RECORDS
-Health Record ID: Integer
-Previous date visit:
-Medical History: String
+get Lab_id()
+set Lab_id()
+get medication()
+set medication()
(0..*)
(0..*)
(0..1)
LABORATORY TESTS
-Lab Test_id: Integer
-Test name: String
-Description: String
-Test Date: Date
-Results: String
+get Lab Test_id()
+set Lab Test_id()
+get Results()
+set Results()
(0..*)
Orderd by
(1..1)
(1..*)
(1..1)
DOCTOR PERSONAL ACHIEVEMENTS
-Achievement_id: Integer
-Professional Awards: String
-Honors: String
-Publications: String
-Personal Milestones: String
+get Achievement_id()
+set Achievement_id()
+get Professional Awards()
+set Professional Awards()
(0..N)
has
(1..1)
(0..*)
(1..1)
for
for
PATIENTS
-Patient_id: Integer
-Patient name: String
-Age: Integer
-Gender: String
-Phone number:
-Address: String
+get patienti_id()
+set patient_id()
+get Age()
+set Age()
DOCTORS
-Doctor_id: Integer
-Doctor name: String
-Specialization: String
-Qualification: String
-working hours: String
-working days: String
+get Doctor_id()
+set Doctor_id()
+get Doctor name()
+set Doctor name()
APPOINTMENTS
-Date: Date
-Timings: String
-Symptoms: String
+get Date()
+set Date()
+get Symptoms()
+set Symptoms()
Undergoes
(1..*)
(1..*)
(1..*)
(1..1)
(0..1)
(1..*)
(1..1)
(1..*)
is in
(1..1)
HOSPITAL DEPARTMENT
-Department_id: Integer
-Department name: String
-Head of department: String
-Description: String
-Location: String
+get Department_id()
+set Department_id()
+get Department()
+set Department()
book
(1..1)
REGULAR PATIENTS
...
(1..*)
EMERGENCY PATIENTS
...
Consultation
for
(1..1)
(1..1)
(1..*)
(1..1)
contains
(1..*)
(1..1)
INSURANCE
-Insurance_id: Integer
-Policy number: Integer
-max coverage:Integer
-coverage start_date: Date
-coverage end_date: Date
+get Insurance_id()
+set Insurance_id()
+get policy number()
+set policy number()
has
(1..1)
(1..1)
(1..*)
(1..1)
Buys
Request
PAYMENTS
-Payment_id: Integer
-Amount Due: Integer
-Payment status: String
-Date of payment: Date
-Payment type: String
-Invoice details: String
+get payment_id()
+set payment_id()
+get invoice()
+set invoice()
ROOMS
-Rooms_id: Integer
-Room name: String
+get Rooms_id()
+set Rooms_id()
+get Room name()
+set Room name()
Provided by
Provides
(0..*)
(1..1)
(0..1)
(0..*)
(0..*)
(1..1)
MEDICATIONS
-Medication_id: Integer
-Medication name: String
-Dosage: String
-Instructions: String
+get medication_id()
+set medication_id()
+get dosage()
+set dosage()
PHARMACY ORDER
-Order_id: Integer
-Medication: String
-Quantity: Integer
-Total price: Integer
-Payment method: String
+get order_id()
+set order_id()
+get quantity()
+set quantity()
AMBULANCE SERVICES
-Ambulance_id: Integer
-Driver name: String
-License: String
-Current Location: String
-Contact information: Integer
+get Ambulance_id()
+set Ambulance_id()
+get Current location()
+set Current location()
PATIENT FEEDBACK & RATINGS
-Feedback_id: Integer
-Ratings: Integer
-Comments: String
-Date of Feedback: Date
+get Feedback_id()
+set Feedback_id()
+get Ratings()
+set Ratings()

RELATIONAL MODEL:

Surgery (Procedure Description, surgeryID, Date and Time, Type, Pre n post Op notes, *doctorID*, *patientID*)

Here, doctorID is a foreign key and NULL is NOT allowed.

Health Record (Health RecordID, Previous date of visit, Medical-history)

Lab Tests (TestID, Test name, Test date, Description, Results)

Contains (*TestID*, *Health Record-id*)

Here, *TestID* is a foreign key and NULL is NOT allowed.

Here, *TestID* is a foreign key and NULL is NOT allowed.

Appointments (Appointment-id, date, timings, issue of symptoms, *doctorID, patientID*)

Here, *doctorID* is a foreign key and NULL is NOT allowed.

Here, *patientID* is a foreign key and NULL is NOT allowed.

Patient (Patient-id, name, age, phone number, gender)

Regular Patients (PatientID, *testID*,)

Here, *testID* is a foreign key and NULL is allowed.

Emergency Patients (PatiendID, *LicenseCode(ID)*)

Here, *LicenseCode(ID)* is a foreign key and NULL is NOT allowed.

Ambulance Services (Driver details, licenseCode(ID), current location, phone number, availability)

Insurance (InsuranceID, coverage-start-date, provider, policy no, max coverage, coverage-end-date, *patiendID*)

Here, *patiendID* is a foreign key and NULL is NOT allowed.

Medication (MedicationID, dosage, name, instructions)

Med Buys (*MedicationID*, *PatientID*, *OrderID*)

Here, *MedicationID* is a foreign key and NULL is NOT allowed.

Here, *PatientID* is a foreign key and NULL is NOT allowed.

Here, *OrderID* is a foreign key and NULL is allowed.

DocProvidesMed (*PatientID*, *MedicationID*, *DoctorID*) Here, *PatientID* is a foreign key and NULL is NOT allowed.
Here, *MedicationID* is a foreign key and NULL is NOT allowed.Here, *DoctorID* is a foreign key and NULL is allowed.

Pharmacy Orders (Order-id, quantity, order-date, total price, payment method)

Doctor (Doctor-id, name, working hours, specialization, working days, qualification, phonenumber, *deptID*)
Here, *deptID* is a foreign key and NULL is NOT allowed.

LabTestsOrderedBy (*TestID*, *DoctorID*,)
Here, *TestID* is a foreign key and NULL is NOT allowed. Here, *DoctorID* is a foreign key and NULL is NOT allowed.

Doctor Achievements (Achieve-id, awards, publications, milestones, hours, *doctrorID*)Here, *doctrorID* is a foreign key and NULL is NOT allowed.

Payment Details (PaymentID, Invoice details, amount due, payment method, date of payment,payment status)

Consultation (*PatientID*, *DoctorID*, *PaymentID*)
Here, *PatientID* is a foreign key and NULL is NOT allowed. Here, *DoctrorID* is a foreign key and NULL is NOT allowed. Here, *PaymentID* is a foreign key and NULL is NOT allowed.

Hospital Departments (DeptID, name, hod, description, location)

Rooms (RoomID, name, *deptID*)
Here, *deptID* is a foreign key and NULL is NOT allowed.

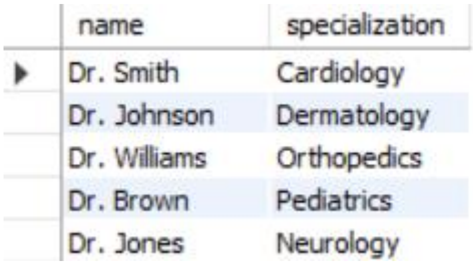
Patients Feedback (FeedbackID, comments, ratings, date of feedback, *doctorID*)Here, *doctorID* is a foreign key and NULL is NOT allowed.

Feedback Provided by (*FeedbackID*, *PatientID*)
Here, *FeedbackID* is a foreign key and NULL is NOT allowed.Here, *PatientID* is a foreign key and NULL is NOT allowed.

Implementation in MySQL:

1.) Simple query: Using select to display all the names and specialization of the doctor.

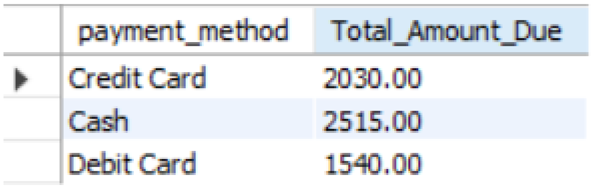
SELECT name, specialization FROM Doctors;



name	specialization
Dr. Smith	Cardiology
Dr. Johnson	Dermatology
Dr. Williams	Orthopedics
Dr. Brown	Pediatrics
Dr. Jones	Neurology

2.) Aggregate query: Total amount due for payment under each payment method.

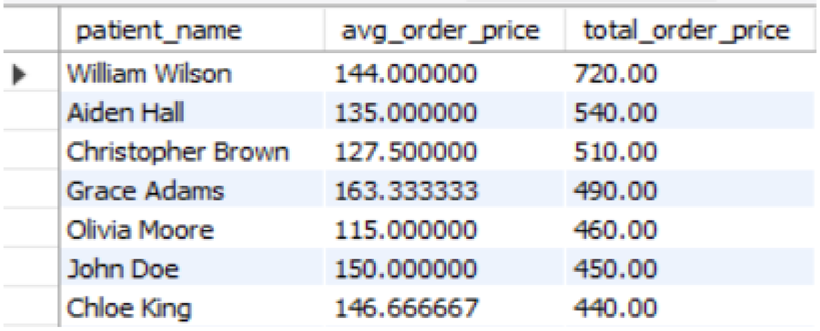
SELECT payment_method, SUM(amount_due) as Total_Amount_Due FROM payment_details
GROUP BY payment_method;



payment_method	Total_Amount_Due
Credit Card	2030.00
Cash	2515.00
Debit Card	1540.00

3.) Joins: Show the average and total price of pharmacy orders for each patient:

SELECT pd.name AS patient_name, AVG(po.total_price) AS avg_order_price,
SUM(po.total_price) AS total_order_price
FROM Patient_Details pd
JOIN Med_Buys mb ON pd.Patient_id = mb.PatientID
JOIN Pharmacy_Orders po ON mb.OrderID = po.Order_id
GROUP BY pd.name
ORDER BY total_order_price DESC;

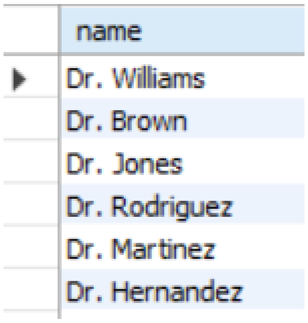


patient_name	avg_order_price	total_order_price
William Wilson	144.000000	720.00
Aiden Hall	135.000000	540.00
Christopher Brown	127.500000	510.00
Grace Adams	163.333333	490.00
Olivia Moore	115.000000	460.00
John Doe	150.000000	450.00
Chloe King	146.666667	440.00

4.) Nested query: Doctors that have lower average ratings than the overall average:

SELECT name
FROM Doctors
WHERE Doctor_id IN (

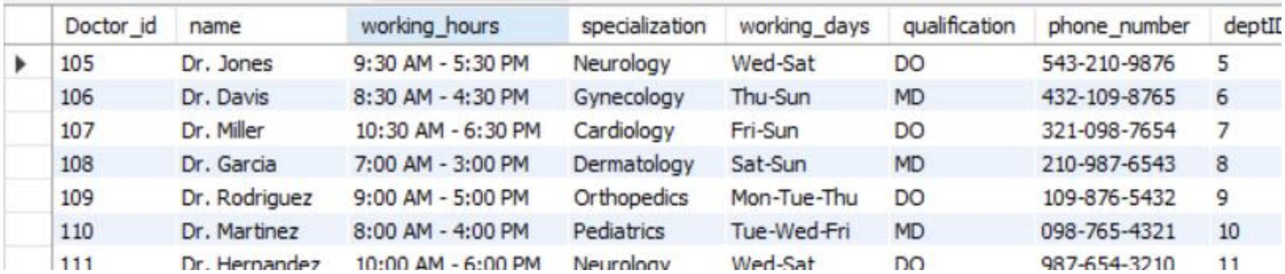
```
   SELECT doctorID  FROM Patients_Feedback GROUP BY doctorID
   HAVING AVG(ratings) < (SELECT AVG(ratings) FROM Patients_Feedback));
```



	name
▶	Dr. Williams
	Dr. Brown
	Dr. Jones
	Dr. Rodriguez
	Dr. Martinez
	Dr. Hernandez

5.) Correlated query: Find doctors with appointments between specific dates:

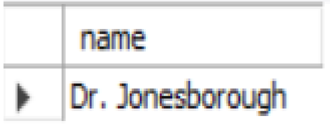
```
SELECT * FROM Doctors d
WHERE EXISTS (SELECT 1 FROM Appointments a
      WHERE a.doctorID = d.Doctor_id AND a.date >= '2023-03-20' AND a.date<= '2023-04-20');
```



	Doctor_id	name	working_hours	specialization	working_days	qualification	phone_number	deptID
▶	105	Dr. Jones	9:30 AM - 5:30 PM	Neurology	Wed-Sat	DO	543-210-9876	5
	106	Dr. Davis	8:30 AM - 4:30 PM	Gynecology	Thu-Sun	MD	432-109-8765	6
	107	Dr. Miller	10:30 AM - 6:30 PM	Cardiology	Fri-Sun	DO	321-098-7654	7
	108	Dr. Garcia	7:00 AM - 3:00 PM	Dermatology	Sat-Sun	MD	210-987-6543	8
	109	Dr. Rodriguez	9:00 AM - 5:00 PM	Orthopedics	Mon-Tue-Thu	DO	109-876-5432	9
	110	Dr. Martinez	8:00 AM - 4:00 PM	Pediatrics	Tue-Wed-Fri	MD	098-765-4321	10
	111	Dr. Hernandez	10:00 AM - 6:00 PM	Neurology	Wed-Sat	DO	987-654-3210	11

6.) Query using ALL/ANY: Find doctors who have appointments with patients that have the highest total number of appointment.

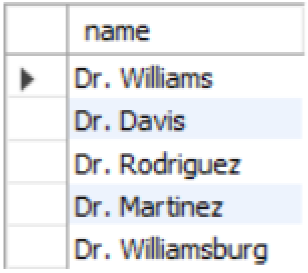
```
SELECT name
FROM Doctors d
WHERE (
   SELECT COUNT(Appointment_id) FROM Appointments a
   WHERE a.doctorID = d.Doctor_id
) >= ALL (
   SELECT COUNT(Appointment_id) FROM Appointments a2
   GROUP BY a2.doctorID
   HAVING COUNT(Appointment_id) IS NOT NULL AND a2.doctorID <> d.Doctor_id
);
```



	name
▶	Dr. Jonesborough

7.) Query using EXISTS: Query Doctors who have done more than 3 surgeries.

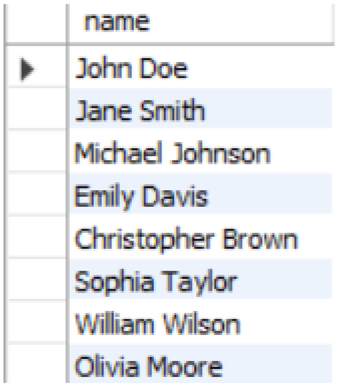
```
SELECT name FROM Doctors d
WHERE EXISTS (
   SELECT 1 FROM Surgery s
   WHERE s.doctorID = d.Doctor_id
   GROUP BY s.doctorID
   HAVING COUNT(s.surgeryID) > 3
);
```



name
Dr. Williams
Dr. Davis
Dr. Rodriguez
Dr. Martinez
Dr. Williamsburg

8.) Set operations: Retrieve the names of patients who have either undergone lab tests or surgeries:

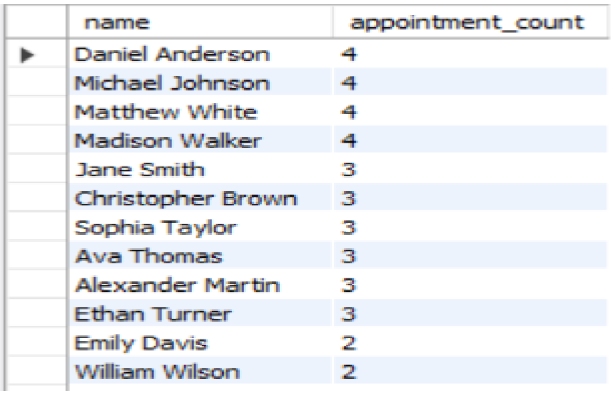
```
SELECT name FROM Patient_Details
WHERE EXISTS (
   SELECT 1 FROM Lab_Tests WHERE Lab_Tests.TestID = Patient_Details.Patient_id
)
UNION
SELECT name FROM Patient_Details
WHERE EXISTS (
   SELECT 1 FROM Surgery WHERE Surgery.patientID = Patient_Details.Patient_id
);
```



name
John Doe
Jane Smith
Michael Johnson
Emily Davis
Christopher Brown
Sophia Taylor
William Wilson
Olivia Moore

9.) Subqueries in select:

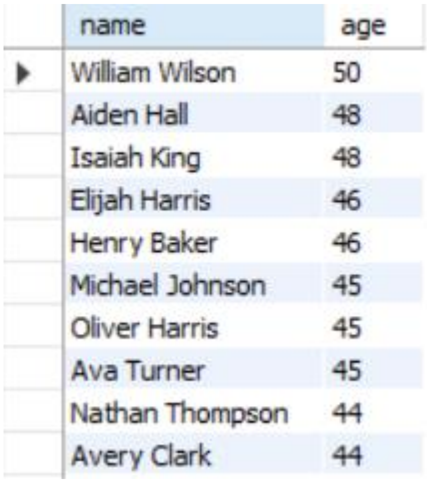
```
SELECT name, (SELECT COUNT(*) FROM Appointments a WHERE a.patientID = p.Patient_id)
AS appointment_count
FROM Patient_Details p ORDER BY appointment_count DESC ;
```



name	appointment_count
Daniel Anderson	4
Michael Johnson	4
Matthew White	4
Madison Walker	4
Jane Smith	3
Christopher Brown	3
Sophia Taylor	3
Ava Thomas	3
Alexander Martin	3
Ethan Turner	3
Emily Davis	2
William Wilson	2

#10 Subqueries: Retrieve patient name and age for those who have age greater than average age among all patients:

```
SELECT name, age
FROM patient_details
WHERE age > (SELECT AVG(age) FROM Patient_Details)
ORDER BY age DESC
```



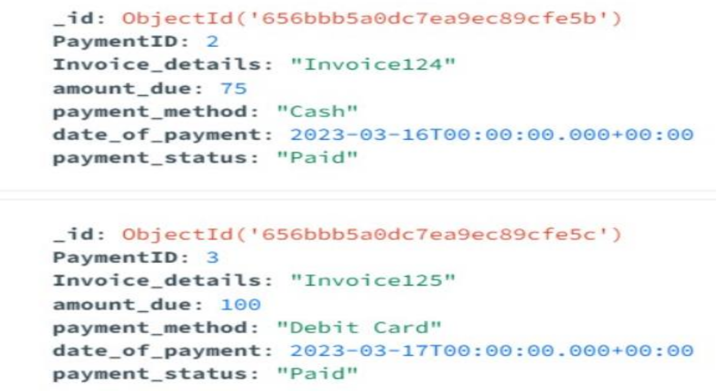
name	age
William Wilson	50
Aiden Hall	48
Isaiah King	48
Elijah Harris	46
Henry Baker	46
Michael Johnson	45
Oliver Harris	45
Ava Turner	45
Nathan Thompson	44
Avery Clark	44

Implementation in NoSQL:

SIMPLE QUERIES:

1.) Finding the amount due which is greater than 70 in the payment_details collection:

```
mydb.payment_details.find({ amount_due: { $gt: 70 } });
```



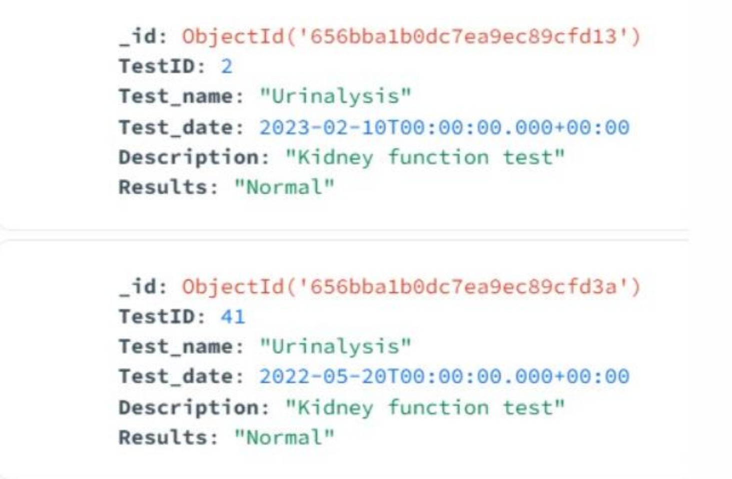
```
_id: ObjectId('656bbb5a0dc7ea9ec89cfe5b')
PaymentID: 2
Invoice_details: "Invoice124"
amount_due: 75
payment_method: "Cash"
date_of_payment: 2023-03-16T00:00:00.000+00:00
payment_status: "Paid"

_id: ObjectId('656bbb5a0dc7ea9ec89cfe5c')
PaymentID: 3
Invoice_details: "Invoice125"
amount_due: 100
payment_method: "Debit Card"
date_of_payment: 2023-03-17T00:00:00.000+00:00
payment_status: "Paid"
```

COMPLEX QUERIES:

2) Retrieving the lab tests collections where the description has the key work “kidney” in it.

mydb.lab_tests.find({ Description: { $regex: /kidney/i } });

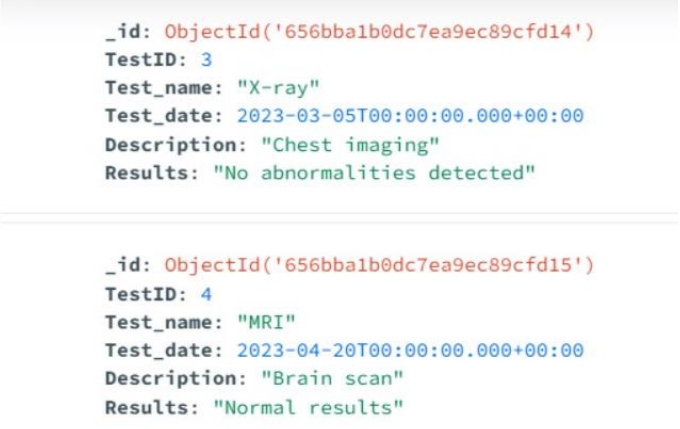


```
_id: ObjectId('656bba1b0dc7ea9ec89cfd13')
TestID: 2
Test_name: "Urinalysis"
Test_date: 2023-02-10T00:00:00.000+00:00
Description: "Kidney function test"
Results: "Normal"

_id: ObjectId('656bba1b0dc7ea9ec89cfd3a')
TestID: 41
Test_name: "Urinalysis"
Test_date: 2022-05-20T00:00:00.000+00:00
Description: "Kidney function test"
Results: "Normal"
```

3) Finding the lab test records which has the test date greater than a particular date.

mydb.lab_tests.find({ Test_date: { $gt: ISODate("2023-02-05T00:00:00.000Z") } })



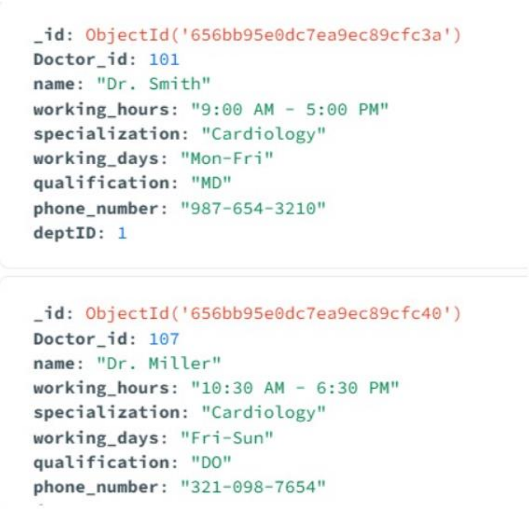
```
_id: ObjectId('656bba1b0dc7ea9ec89cfd14')
TestID: 3
Test_name: "X-ray"
Test_date: 2023-03-05T00:00:00.000+00:00
Description: "Chest imaging"
Results: "No abnormalities detected"

_id: ObjectId('656bba1b0dc7ea9ec89cfd15')
TestID: 4
Test_name: "MRI"
Test_date: 2023-04-20T00:00:00.000+00:00
Description: "Brain scan"
Results: "Normal results"
```

AGGREGATE QUERIES:

4) Using aggregate function match and finding the records that have ‘Cardiology’ as the specialization.

```
mydb.doctors.aggregate([
  {
    "$match": { "specialization": "Cardiology" }
  }
])
```

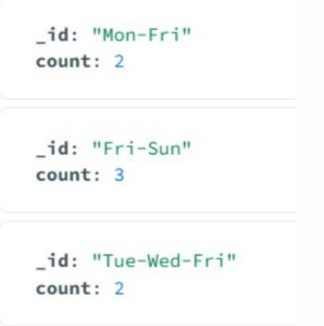


```
_id: ObjectId('656bb95e0dc7ea9ec89cfc3a')
Doctor_id: 101
name: "Dr. Smith"
working_hours: "9:00 AM - 5:00 PM"
specialization: "Cardiology"
working_days: "Mon-Fri"
qualification: "MD"
phone_number: "987-654-3210"
deptID: 1

_id: ObjectId('656bb95e0dc7ea9ec89cfc40')
Doctor_id: 107
name: "Dr. Miller"
working_hours: "10:30 AM - 6:30 PM"
specialization: "Cardiology"
working_days: "Fri-Sun"
qualification: "DO"
phone_number: "321-098-7654"
```

5) Using aggregate functions to group and find the count of the working days of the doctors.

```
mydb.doctors.aggregate([
  {
    "$group": {
      "_id": "$working_days",
      "count": { "$sum": 1 }
    }}
])
```



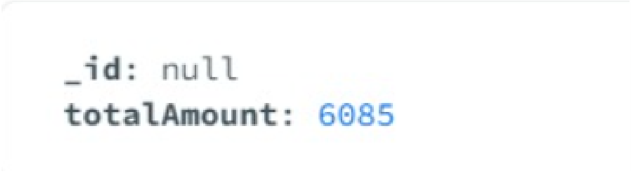
```
_id: "Mon-Fri"
count: 2

_id: "Fri-Sun"
count: 3

_id: "Tue-Wed-Fri"
count: 2
```

6) Finding the total amount due in the paymen_details collection by grouping aggregate function.

```
mydb.payment_details.aggregate([
  { $group: { _id: null, totalAmount: { $sum: "$amount_due" } } }
]);
```

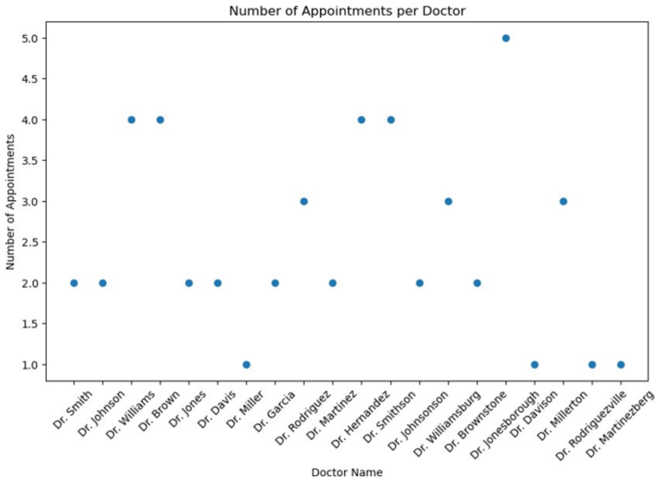


```
_id: null
totalAmount: 6085
```

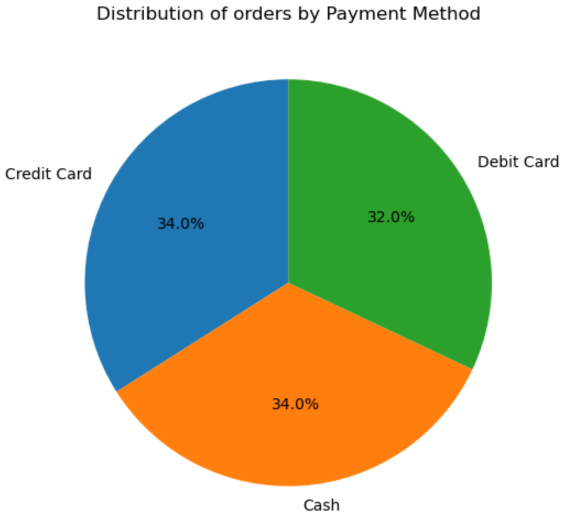
Implementation in Python via Database Access:

The database is accessed using Python and visualization of analyzed data is shown below. The connection of MySQL to Python is done using mysql alchemy, followed by pandas.read_sql fundtion to fetch the sql query and run it, storing it into a dataframe using the pandas library. Furthermore, used matplotlib to plot the graphs from the dataframes for analytics.

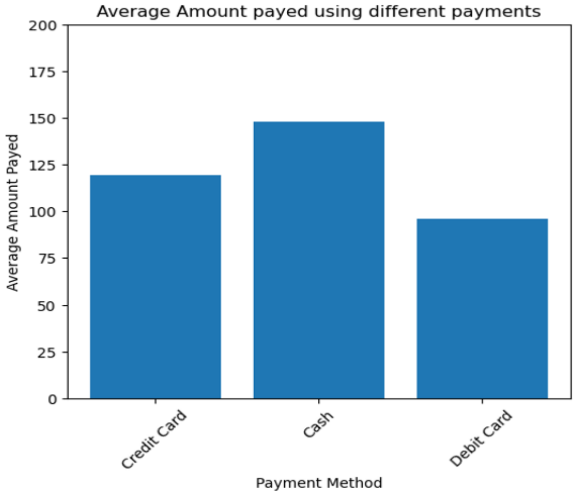
GRAPH 1:



GRAPH 2:



GRAPH 3:



Conclusion:

Our project is about making patient care better and simpler by using a new kind of database in healthcare. This database makes it easier to keep track of patient records, doctor information, appointments, and more. Our goal is to make healthcare faster and more organized, so doctors can focus more on taking care of patients. We believe this project will make a real difference in how hospitals work and help patients get better care. Together, we are shaping a healthier, more efficient future.