IE6700 Data Management for Analytics

VITAL SIGN HEALTH MANAGER

Project Report

Group 21

Student 1: Simran Sinha

Student 2: Sinduja Vengadesan

857-381-5028 (Tel of Simran)

617-943-3248 (Tel of Sinduja)

sinha.sim@northeastern.edu

vengadesan.s@northeastern.edu

Percentage of Effort Contributed by Student 1: 50%

Percentage of Effort Contributed by Student 2: 50%

Signature of Student 1: Simran Sinha

Signature of Student 2: Sinduja Vengadesan

Submission Date: 20-April-2024

VITAL SIGN HEALTH MANAGER

(Hospital Management System)

Simran Sinha & Sinduja Vengadesan

Executive Summary

The suggested Hospital Management System combines relational and NoSQL databases, providing a proactive response to the changing demands of contemporary healthcare operations. Hospitals need to adjust to cutting edge technologies that improve operational efficiencies and raise the standard of patient care in the quickly changing healthcare environment of today. The proactive approach of the system links databases to manage a variety of medical data, including organized data like electronic health records and unstructured and semi-structured data like patient-generated data, medical pictures, and clinical notes. NoSQL databases provide the flexibility required to easily handle this data fluctuation. This all-inclusive system manages and creates links between wards, physicians, personnel, pharmacies, patients, treatment plans, drugs, and appointments, among other hospital components.

I. Introduction

As hospitals develop, the need for a dynamic and flexible data management system becomes increasingly obvious. The recommended Hospital Management System, which employs both NoSQL and Relational databases, is designed to meet the unique requirements of modern healthcare operations. Considering the rapidly evolving healthcare landscape, the hospital must remain at the forefront of deploying innovative solutions to enhance operational efficiencies and elevate the quality of patient care. The proposed idea connects two databases in a proactive manner to meet the evolving needs of our healthcare environment. The range of data utilized in healthcare is growing, including unstructured and semi-structured data such as patient-generated data, medical photos, and clinical notes in addition to standard organized data such as electronic health records. NoSQL databases ensure that our hospital is ready to handle the full spectrum of medical data by offering the flexibility required to manage this variability with simplicity.

Theory:

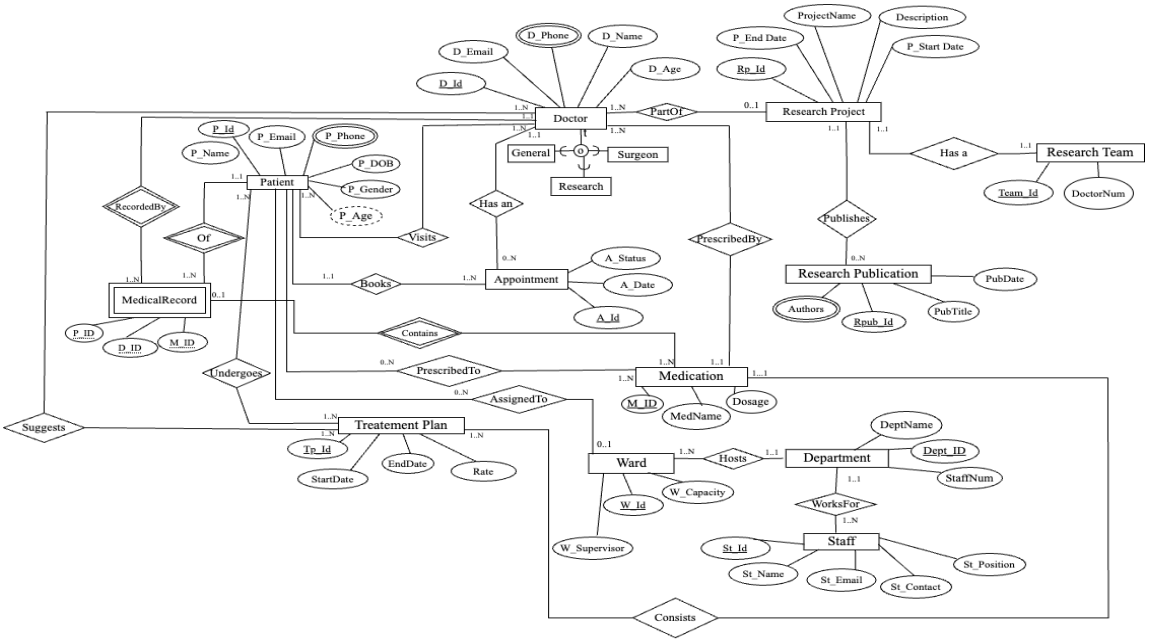
It is essential to study the ideas behind this tactical technological approach to comprehend the projected VitalSign Manager that integrates relational and NoSQL databases. This section aims to provide a theoretical framework that makes sense of the rationale behind merging these two database designs. This system oversees all aspects of a certain hospital, including its wards, doctors, staff, pharmacy, and patients. The system also includes treatment plans, medications, and appointments. Doctors, prescription drugs, and treatment plans are all connected to patients. A hospital would have multiple departments under it, as well as multiple staff members. Plus, it would house at least one Ward. The doctors at the hospital who perform research and development are subject to oversight. Every research project has a research team and, if needed, a research publication attached to it.

Requirements:

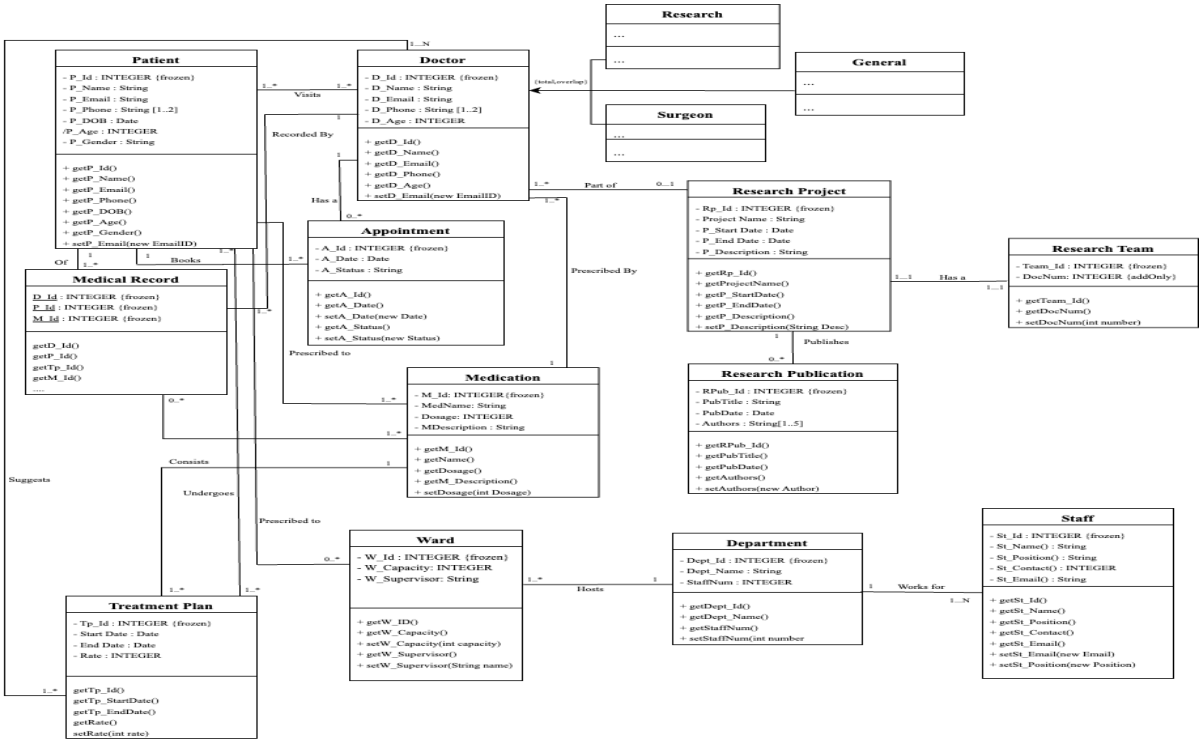
1. A Patient can have multiple Appointments, Treatment and Medications; an appointment, treatment and medication must be associated with one patient only.
2. A Doctor can have multiple Appointments, Treatment and Medications; an appointment must be associated with one doctor only.
3. A Staff member is associated with only one Department; a department can have multiple staff members.
4. A Ward belongs to one Department, a department can be associated with multiple wards.
5. A Treatment can have multiple Medications.
6. A Pharmacy can be associated with multiple Treatment and Medication, a Treatment and Medication must be from one Pharmacy.
7. A Pharmacy relates to many departments, each department is associated with only one pharmacy.

II. Conceptual Data Modelling

Extended Entity Relation Model



Unified Modelling Language



III. Mapping Conceptual Model to Relational Model

Relations and Attributes:

1. **Patient:** Gives the details of the Patients being treated at the hospital
    - (P_Id, P_Name, P_Email, P_DOB, P_Gender, P_Age, *W_Id*)
        - P_Id: Primary key
        - W_Id: Foreign key refers to Ward Relation, NULL ALLOWED
            - If any patient is residing in a Ward, that particular W_Id is stored

2. **Patient_Phone:** Gives the various phone numbers a patient might have
    - (P_Id, P_Phone)
        - Multivalued Attribute Type relation
        - P_Id and P_Phone together make up the primary key

3. **Patient_Visits:** Gives details of which patient visited which doctor corresponding to an appointment id.
    - (*P_Id*, *D_Id,* AppointmentID)
        - P_Id: Refers to P_Id in Patient Relation, NOT NULL
        - D_Id: Refers to D_Id in Doctor Relation, NOT NULL
        - AppointmentID: Refers to A_Id from Appointment Table, NOT NULL
            - Indicates under which appointment the patient is visiting the doctor

4. **Doctor:** Gives the details of Doctors that are working at the hospital
    - (D_Id, D_Name, D_Email, D_Age)
        - D_Id: Primary key

5. **Doctor_Phone:** Gives the various phone numbers a doctor might have
    - (D_Id, D_Phone)
        - Multivalued Attribute Type relation
        - D_Id and D_Phone together make up the primary key

6. **General:** Gives the details of doctors who are general physicians
    - (*D_Id)*
        - Subclass of Doctor with D_Id as Foreign key

7. **Researcher:** Has the details of Doctors who do research
    - (*D_Id, Rp_Id)*
        - Subclass of Doctor with D_Id as Foreign key
        - Rp_Id: Refers to Rp_Id in ResearchProject Relation, NOT NULL
            - Indicates which Research Project this subclass of Doctor is associated with

8. **Surgeon:** Gives the details of doctors who are surgeons
    - (*D_Id*)
        - Subclass of Doctor with D_Id as Foreign key

9. **ResearchProject:** Gives the details of the Research Project being conducted at the hospital
    - (Rp_Id, ProjectName, Description, P_Startdate, P_Enddate,Status)
        - Rp_Id: Primary key

10. **ResearchTeam:** Gives the details of the members in a research team
    - (Team_Id, DoctorNum, *Rp_Id*)
        - Team_Id: Primary key
        - Rp_Id: Refers to Rp_Id in ResearchProject Relation, NOT NULL
            - Indicates what Research Project a particular Team is currently working on

11. **ResearchPublication :** Gives the details of the research publications that is associated with a particular research project
    - (Rpub_Id, PubTitle1, PubDate, *Rp_Id*)
        - Rpub_Id: Primary key
        - Rp_Id: Refers to Rp_Id in ResearchProject Relation, NOT NULL
            - Tells which ResearchPublication is published on which ResearchProject

12. **ResearchPublication_Authors :** Has the details of various authors of a particular research publication

- (Rpub_Id, Authors)
    - Multivalued Attribute Type relation
    - Rpub_Id and Authors together make up the primary key

13. **Department:** Details of the various department housed by the hospital
    - (Dept_ID, DeptName, StaffNum)
        - Dept_ID: Primary key

14. **Staff**: Contains the details of the various staffs (other than doctors) that work at the hospital
    - (St_Id, St_Name, St_Email, St_Contact ,St_Position, *Dept_ID*)
        - St_Id: Primary key
        - Dept_ID: Refers to Dept_ID in Department Relation, NOT NULL
            - Tells which Department a staff is working under

15. **Ward**: Has basic details of the various wards the hospital hosts
    - (W_Id, W_Supervisor, W_Capacity, *Dept_ID*)
        - W_Id: Primary key
        - Dept_ID: Refers to Dept_ID in Department Relation, NOT NULL
            - Tells which Department a ward is associated with

16. **Medication**: Contains the details of the basic medications that is available in the hospital prescribed to the Patient by the Doctor
    - (M_ID, MedName, Dosage)
        - M_ID: Primary key

17. **Appointment**: Has the details of the various appointments made by the Patient to visit a Doctor
    - (A_Id, A_Date, A_Status)
        - A_Id: Primary key

18. **TreatmentPlan**: Gives details of various Treatment plans a patient undergoes that is suggested by a doctor
    - (Tp_Id, StartDate, EndDate, Rate, *M_ID*)

- Tp_Id: Primary key
- M_ID: Foreign key refers to Medication Table; NOT NULL
    - Shows what is the associated Medication that is given to a Treatment Plan

19. **MedicalRecord**: Weak Entity type, Gives the medical record for a patient along with Doctor and Medication
    - (RecordID, P_Id, D_Id, M_ID)
        - RecordID: partial key
        - P_Id, D_Id, M_ID: Combination of primary key

20. **Doctor_Medication**: Contains details of medication prescribed by a Doctor
    - (D_Id, M_ID)
        - D_Id, M_ID: Together makes the primary key
        - D_Id: Foreign key refers to Doctor relation, NOT NULL
        - M_ID: Foreign key refers to Medication relation, NOT NULL

21. **Patient_Medication**: Contains details of medication prescribed to a Patient
    - (P_Id, M_ID)
        - P_Id, M_ID: Together makes the primary key
        - P_Id: Foreign key refers to Patient relation, NOT NULL
        - M_ID: Foreign key refers to Medication relation, NOT NULL

22. **TreatmentPlan_Patient**: Contains details of associated Treatment Plans and Patients
    - (P_Id, Tp_Id)
        - P_Id, Tp_ID: Together makes the primary key
        - P_Id: Foreign key refers to Patient relation, NOT NULL
        - Tp_ID: Foreign key refers to Treatment Plan relation, NOT NULL

23. **TreatmentPlan_Doctor**: Contains details of associated Treatment Plans and Medications
    - (D_Id, Tp_Id)
        - D_Id, M_ID: Together makes the primary key
        - D_Id: Foreign key refers to Doctor relation, NOT NULL
        - Tp_ID: Foreign key refers to Treatment Plan relation, NOT NUL

IV. Implementation of Relational Model: MySQL & NoSQL

MySQL

Schema Name used: VitalSign_HealthManager

Total No. Of tables after Normalization: 23

Database Schema Creation:

```
CREATE DATABASE VitalSign_HealthManager;
USE VitalSign_HealthManager;
```

Query 1: Give the Patient Details who are adults

```
SELECT P_id, P_Name, P_Email, P_Gender
FROM Patients
WHERE P_Age >=18;
```

Output:

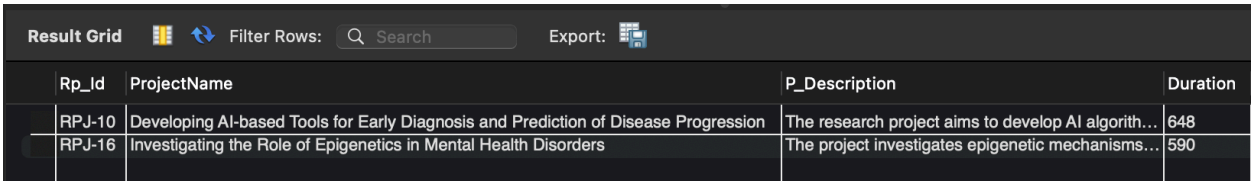
Result Grid Filter Rows: Search Edit: Export/Import:

P_id	P_Name	P_Email	P_Gen...
PT-10	Lara T. Graves	laratgraves@aol.org	Male
PT-100	Maile G. Page	mailegpage@hotmail.couk	Male
PT-110	Nita D. Christian	nitadchristian@aol.net	Female
PT-120	Joel V. Shields	joelvshields3561@protonmail.com	Male
PT-125	Lane F. Rush	lanefrush456@icloud.edu	Female
PT-130	Quemby H. Harmon	quembyhharmon@protonmail.couk	Female
PT-140	Ryan X. Hawkins	ryanxhawkins@outlook.org	Male
PT-145	Catherine W. Osborn	catherinewosborn1711@icloud.net	Male
PT-15	Dominic E. Finley	dominicefinley6017@google.couk	Male
PT-150	Quemby G. Young	quembygyoung985@protonmail.org	Female
PT-155	Dustin C. Swanson	dustincswanson9437@hotmail.ca	Male
PT-160	Christopher R. Mac...	christopherrmacias@aol.ca	Male
PT-170	Amir I. Suarez	amirisuarez@yahoo.couk	Male
PT-185	Cruz C. Lowery	cruzclowery@outlook.org	Female
PT-20	Theodore I. Hogan	theodoreihogan4942@yahoo.com	Male
PT-200	Colby F. Gutierrez	colbyfgutierrez@protonmail.edu	Male
PT-205	Raja H. Snider	rajahsnider6638@yahoo.edu	Male
PT-230	Hedwig M. Heath	hedwigmheath@hotmail.edu	Male
PT-235	Carolyn T. Hurley	carolynthurley@google.com	Male
PT-240	Maris Y. Solomon	marisysolomon3742@aol.com	Female
PT-245	Edan M. Wiley	edanmwiley955@outlook.ca	Male
PT-25	Robert P. Hickman	robertphickman@google.ca	Male
PT-250	Aaron X. Bradford	aaronxbradford2029@hotmail.net	Male
PT-255	Vera T. Chang	veratchang1539@hotmail.org	Male
PT-35	Ronan T. Mcmillan	ronantmcmillan@hotmail.edu	Female
PT-40	Forrest O. Mcintosh	forrestomcintosh9170@hotmail.com	Male
PT-45	Thaddeus O. Garcia	thaddeusogarcia@yahoo.edu	Female
PT-55	Timothy V. Becker	timothyvbecker@outlook.edu	Female
PT-60	Adena B. Velasquez	adenabvelasquez@protonmail.net	Male
PT-70	Duncan S. Crane	duncanscrane@hotmail.edu	Male
PT-75	Ferdinand H. Spencer	ferdinandhspencer9453@hotmail....	Male
PT-80	Sybill F. Slater	sybillfslater9704@outlook.org	Female
PT-85	Quentin S. Emerson	quentinsemerson6299@google.org	Male
PT-90	Lamar S. Huber	lamarshuber7787@google.org	Male

Query 2: Return the details of the Completed Research Projects

```
SELECT Rp_Id, ProjectName,
P_Description,
(DATEDIFF(P_EndDate, P_StartDate)) AS Duration
FROM ResearchProject
WHERE Project_Status = 'Completed';
```

Output:



Result Grid | Filter Rows: Search | Export:

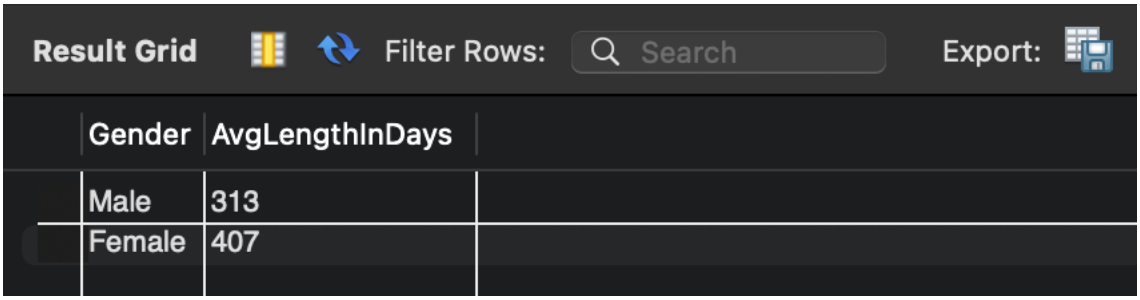
Rp_Id	ProjectName	P_Description	Duration
RPJ-10	Developing AI-based Tools for Early Diagnosis and Prediction of Disease Progression	The research project aims to develop AI algorith...	648
RPJ-16	Investigating the Role of Epigenetics in Mental Health Disorders	The project investigates epigenetic mechanisms...	590

Query 3: Find the average length of treatment plans for patients for each gender

By calculating the average length of treatment plans separately for each gender, hospital administrators can compare the duration of treatment received by male and female patients. Understanding any differences in treatment duration can provide insights into potential gender-related healthcare disparities or variations in healthcare needs.

```
SELECT p.P_Gender as Gender, ROUND(avg(DATEDIFF(t.End_Date, t.Start_Date))) AS AvgLengthInDays
FROM TreatmentPlan t
INNER JOIN TreatmentPlan_Patient tp ON t.Tp_Id = tp.Tp_Id
INNER JOIN Patients p ON tp.P_Id = p.P_Id
GROUP BY p.P_Gender;
```

Output:



Result Grid | Filter Rows: Search | Export:

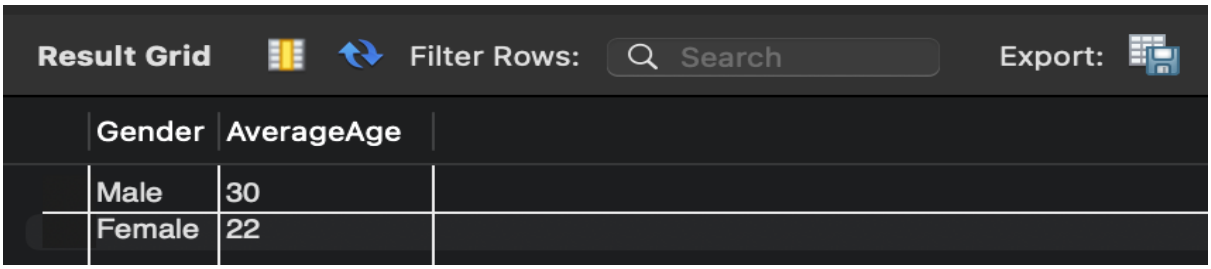
Gender	AvgLengthInDays
Male	313
Female	407

Query 4: Retrieve the average age of male and female patients who visited the hospital more than twice

This is useful in understanding patient demographics, healthcare providers gain insights into the age distribution of frequent visitors.

```
SELECT p.P_Gender as Gender, round(avg(p.P_Age)) as AverageAge
FROM Patients p
INNER JOIN Patient_Visits pv ON p.P_Id = pv.P_Id
GROUP BY p.P_Gender
HAVING count(pv.AppointmentID) >= 2;
```

Output:



Query 5: Revenue generated by the Hospital from Each Treatment plan and Medication

By calculating the revenue generated from each treatment plan and medication, hospital administrators gain insights into the financial performance of different healthcare services and pharmaceutical products. This information helps in assessing revenue streams

```
SELECT TreatmentPlanID, MedicationName, Revenue
FROM (
    SELECT tp.Tp_Id AS TreatmentPlanID,
           m.MedName AS MedicationName,
           SUM(CAST(REPLACE(REPLACE(tp.Rate,"$",""),",","") AS UNSIGNED INTEGER)) as Revenue
    FROM TreatmentPlan tp
    INNER JOIN Medication m ON tp.M_ID = m.M_ID
    GROUP BY tp.Tp_Id, m.MedName
) AS Subquery
ORDER BY Revenue DESC;
```

Output:

Result Grid | Filter Rows: Search | Export:

TreatmentPlanID	MedicationName	Revenue
TP-22-VSHM	Clonazepam	9922
TP-8-VSHM	Simvastatin	9851
TP-18-VSHM	Allopurinol	8115
TP-17-VSHM	Albuterol	8032
TP-12-VSHM	Lipitor	7842
TP-25-VSHM	Methylprednisolone	7413
TP-14-VSHM	Oxycontin	7255
TP-10-VSHM	Trazodone HCl	6699
TP-5-VSHM	Lovastatin	6446
TP-7-VSHM	Fluticasone Propionate	6196
TP-2-VSHM	Lovastatin	5938
TP-20-VSHM	Albuterol	5387
TP-1-VSHM	Gianvi	4947
TP-6-VSHM	APAP/Codeine	3788
TP-9-VSHM	Amlodipine Besylate	3624
TP-21-VSHM	Methylprednisolone	3301
TP-19-VSHM	Levothyroxine Sodium	3154
TP-23-VSHM	Nasonex	3092
TP-15-VSHM	Niaspan	2920
TP-4-VSHM	Naproxen	2824
TP-24-VSHM	Oxycontin	2180
TP-13-VSHM	Nasonex	2171
TP-3-VSHM	Lantus	2123
TP-16-VSHM	Ciprofloxacin HCl	1782
TP-11-VSHM	Clindamycin HCl	1621

Query 6: Find the Patient and assigned Doctor details where the appointment is “Cancelled”.

By identifying appointments that are canceled, hospital administrators can monitor appointment scheduling and cancellation trends.

```
SELECT DISTINCT p.P_Id, p.P_Name, p.P_Email, p.P_Age, p.P_Gender,
                d.D_Id, d.D_Name, d.D_Age, d.D_Email
FROM Patients p
INNER JOIN Patient_Visits pv on p.P_Id = pv.P_Id
INNER JOIN Doctor d on d.D_Id = pv.D_Id
WHERE EXISTS (
    SELECT *
    FROM Appointment a
    WHERE a.A_Id = pv.AppointmentID
    AND a.A_Status = 'Cancelled'
);
```

Output:

Result Grid | Filter Rows: Search | Export:

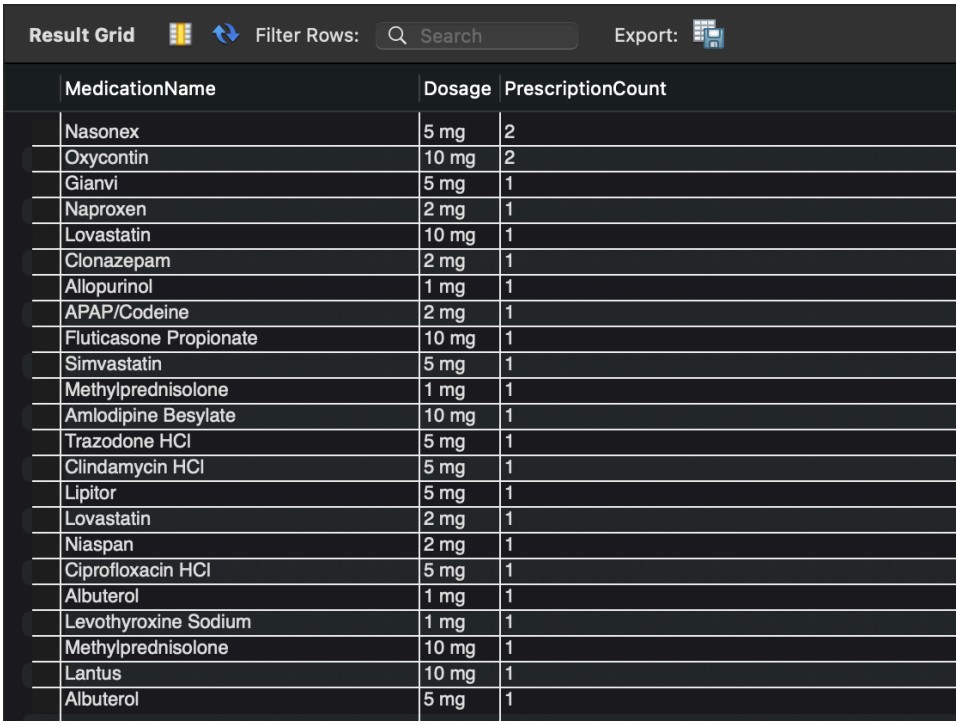
P_Id	P_Name	P_Email	P_Age	P_Gen...	D_Id	D_Name	D_Age	D_Email
PT-10	Lara T. Graves	laratgraves@aol.org	29	Male	DR-1	Shannon P. Levine	57	shannonplevine@yahoo.couk
PT-15	Dominic E. Finley	dominicefinley6017@google.couk	26	Male	DR-2	Kieran Y. Floyd	38	kieranyfloyd6146@outlook.edu
PT-20	Theodore I. Hogan	theodoreihogan4942@yahoo.com	26	Male	DR-3	Leigh J. Weeks	63	leighjweeks3@yahoo.couk
PT-30	Quamar B. Miles	quamarbmiles1933@google.net	15	Female	DR-5	Blake Y. Leach	61	blakeyleach5968@hotmail.edu
PT-40	Forrest O. Mcintosh	forrestomcintosh9170@hotmail.com	47	Male	DR-7	Aaron N. Moody	35	aaronnmoody@protonmail.couk
PT-55	Timothy V. Becker	timothyvbecker@outlook.edu	33	Female	DR-10	Colt J. Barry	54	coltjbarry4357@protonmail.org
PT-60	Adena B. Velasquez	adenabvelasquez@protonmail.net	41	Male	DR-11	Harrison M. Carey	33	harrisonmcarey2569@aol.org
PT-65	Veronica Q. Clarke	veronicaqclarke@protonmail.org	12	Female	DR-12	Shelby V. Alexander	32	shelbyvalexander@hotmail.ca
PT-70	Duncan S. Crane	duncanscrane@hotmail.edu	42	Male	DR-13	Valentine O. Guerra	60	valentineoguerra8176@outlook.ca
PT-75	Ferdinand H. Spencer	ferdinandhspencer9453@hotmail.edu	35	Male	DR-14	Belle I. Zamora	60	belleizamora@protonmail.edu
PT-140	Ryan X. Hawkins	ryanxhawkins@outlook.org	24	Male	DR-2	Kieran Y. Floyd	38	kieranyfloyd6146@outlook.edu
PT-185	Cruz C. Lowery	cruzclowery@outlook.org	46	Female	DR-11	Harrison M. Carey	33	harrisonmcarey2569@aol.org
PT-190	Regina O. Oneil	reginaooneil5545@google.ca	9	Female	DR-12	Shelby V. Alexander	32	shelbyvalexander@hotmail.ca
PT-215	Clark X. Vinson	clarkxvinson@outlook.com	14	Female	DR-13	Valentine O. Guerra	60	valentineoguerra8176@outlook.ca
PT-220	Jescie S. Wolf	jescieswolf2034@yahoo.com	12	Female	DR-15	Yoko E. Gilmore	34	yokoegilmore6020@hotmail.edu
PT-225	Roanna P. Roach	roannaproach8508@hotmail.net	15	Female	DR-6	Ronan G. Vincent	58	ronangvincent@hotmail.couk
PT-230	Hedwig M. Heath	hedwigmheath@hotmail.edu	29	Male	DR-7	Aaron N. Moody	35	aaronnmoody@protonmail.couk
PT-235	Carolyn T. Hurley	carolynthurley@google.com	45	Male	DR-5	Blake Y. Leach	61	blakeyleach5968@hotmail.edu
PT-240	Maris Y. Solomon	marisysolomon3742@aol.com	22	Female	DR-12	Shelby V. Alexander	32	shelbyvalexander@hotmail.ca
PT-250	Aaron X. Bradford	aaronxbradford2029@hotmail.net	25	Male	DR-4	Patricia B. Wolfe	64	patriciabwolfe@google.couk
PT-255	Vera T. Chang	veratchang1539@hotmail.org	18	Male	DR-5	Blake Y. Leach	61	blakeyleach5968@hotmail.edu
PT-90	Lamar S. Huber	lamarshuber7787@google.org	26	Male	DR-2	Kieran Y. Floyd	38	kieranyfloyd6146@outlook.edu
PT-95	Chastity E. Puckett	chastityepuckett@google.com	13	Male	DR-3	Leigh J. Weeks	63	leighjweeks3@yahoo.couk
PT-100	Maile G. Page	mailegpage@hotmail.couk	46	Male	DR-5	Blake Y. Leach	61	blakeyleach5968@hotmail.edu
PT-110	Nita D. Christian	nitadchristian@aol.net	38	Female	DR-11	Harrison M. Carey	33	harrisonmcarey2569@aol.org
PT-125	Lane F. Rush	lanefrush456@icloud.edu	36	Female	DR-4	Patricia B. Wolfe	64	patriciabwolfe@google.couk
PT-135	Sara M. Espinoza	saramespinoza@google.edu	6	Male	DR-1	Shannon P. Levine	57	shannonplevine@yahoo.couk

Query 7: Top 5 Medications prescribed by doctors, grouped by their dosages

This information helps in understanding which medications are most prescribed and in what dosages, facilitating inventory management, procurement planning, and resource allocation.

```
SELECT
m.MedName AS MedicationName,
CONCAT(m.Dosage, " mg") AS Dosage,
COUNT(tp.Tp_Id) AS PrescriptionCount
FROM Treatmentplan tp
INNER JOIN Medication m ON tp.M_ID = m.M_ID
GROUP BY m.MedName, m.Dosage
HAVING COUNT(tp.Tp_Id) >= ALL (SELECT COUNT(tp2.Tp_Id)
                               FROM Treatmentplan tp2
                               GROUP BY tp2.M_ID)
ORDER BY
    COUNT(tp.Tp_Id) DESC;
```

Output:



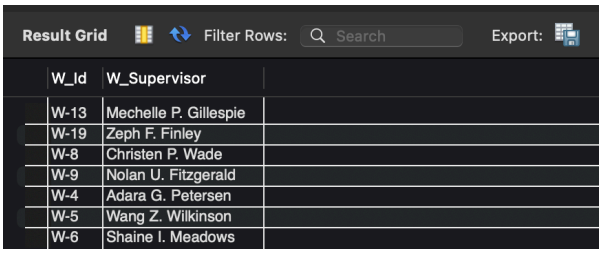
Result Grid | Filter Rows: Search | Export:

MedicationName	Dosage	PrescriptionCount
Nasonex	5 mg	2
Oxycontin	10 mg	2
Gianvi	5 mg	1
Naproxen	2 mg	1
Lovastatin	10 mg	1
Clonazepam	2 mg	1
Allopurinol	1 mg	1
APAP/Codeine	2 mg	1
Fluticasone Propionate	10 mg	1
Simvastatin	5 mg	1
Methylprednisolone	1 mg	1
Amlodipine Besylate	10 mg	1
Trazodone HCl	5 mg	1
Clindamycin HCl	5 mg	1
Lipitor	5 mg	1
Lovastatin	2 mg	1
Niaspan	2 mg	1
Ciprofloxacin HCl	5 mg	1
Albuterol	1 mg	1
Levothyroxine Sodium	1 mg	1
Methylprednisolone	10 mg	1
Lantus	10 mg	1
Albuterol	5 mg	1

Query 8: Select the Ward Details where the ward has a capacity greater than 40 or the ward is associated with the Emergency Department

```
SELECT W_Id, W_Supervisor
FROM Ward
WHERE W_Capacity > 40
UNION
SELECT W_Id, W_Supervisor
FROM Ward
WHERE Dept_Id = 'VSHM-14';
```

Output:



Result Grid | Filter Rows: Search | Export:

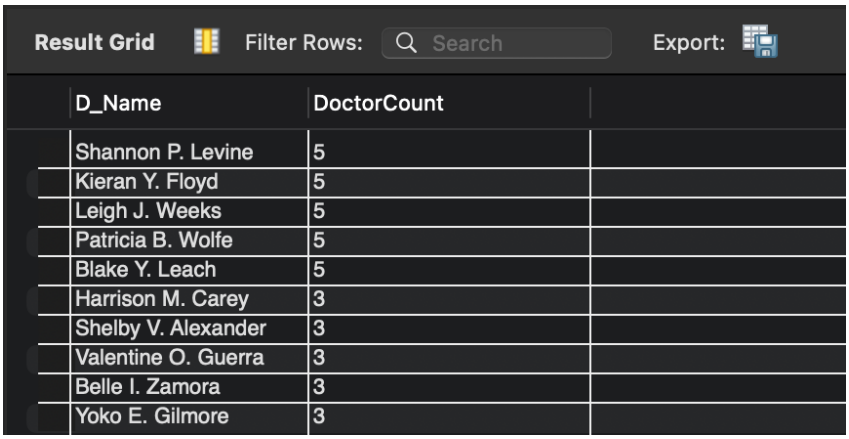
W_Id	W_Supervisor
W-13	Mechelle P. Gillespie
W-19	Zeph F. Finley
W-8	Christen P. Wade
W-9	Nolan U. Fitzgerald
W-4	Adara G. Petersen
W-5	Wang Z. Wilkinson
W-6	Shaine I. Meadows

Query 9: To get the Doctor names who has the maximum and minimum number of medical records in the hospital

```
WITH Doctor_Count AS (
    SELECT COUNT(*) as DoctorCount, mr.D_Id, d.D_Name
    FROM MedicalRecord mr
    INNER JOIN Doctor d on mr.D_Id = d.D_Id
    GROUP BY mr.D_Id, d.D_Name
)

SELECT
D_Name,
DoctorCount
FROM
Doctor_Count
HAVING DoctorCount IN
    (SELECT Max(DoctorCount)
     FROM Doctor_Count
     UNION
     SELECT Min(DoctorCount)
     FROM Doctor_Count)
ORDER BY DoctorCount DESC;
```

Output:



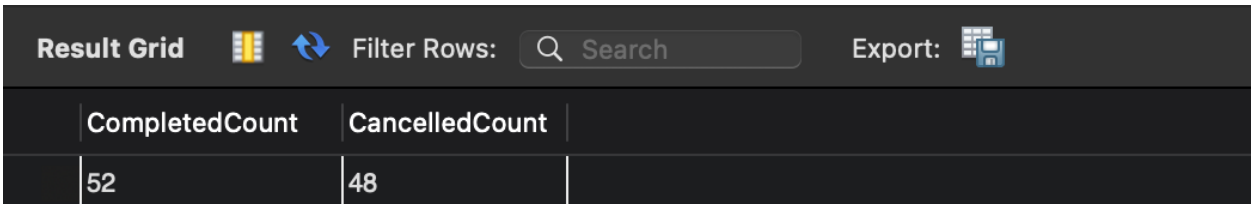
Result Grid | Filter Rows: Search | Export:

D_Name	DoctorCount
Shannon P. Levine	5
Kieran Y. Floyd	5
Leigh J. Weeks	5
Patricia B. Wolfe	5
Blake Y. Leach	5
Harrison M. Carey	3
Shelby V. Alexander	3
Valentine O. Guerra	3
Belle I. Zamora	3
Yoko E. Gilmore	3

Query 10: Get the count of Cancelled and Completed Appointments, each in a seperate column

```
SELECT DISTINCT
(SELECT count(*) FROM Appointment WHERE A_Status = 'Completed') AS CompletedCount,
(SELECT count(*) FROM Appointment WHERE A_Status = 'Cancelled') AS CancelledCount
from Appointment;
```

Output:



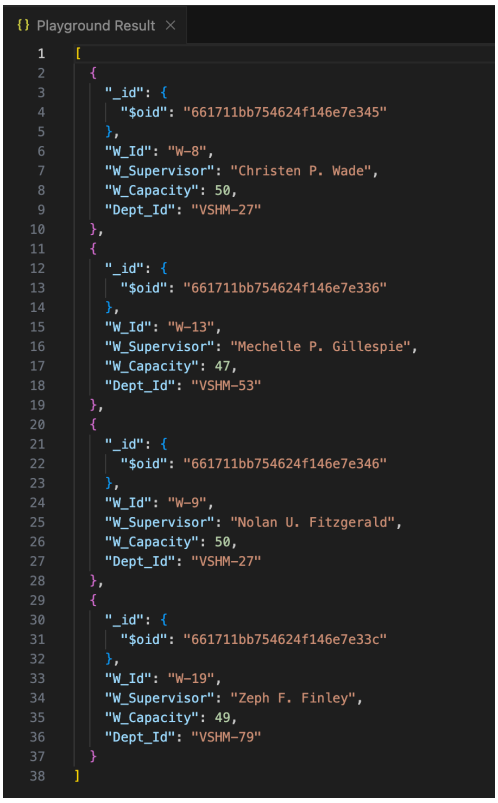
NoSQL using Mongo DB

Query 1: Find the Ward and return the details if the capacity is greater than 40.

```
// Select the database to use.
use('VitalSignHealthManager');

db.Ward.find({ W_Capacity: { $gt: 40 } });
```

Output:

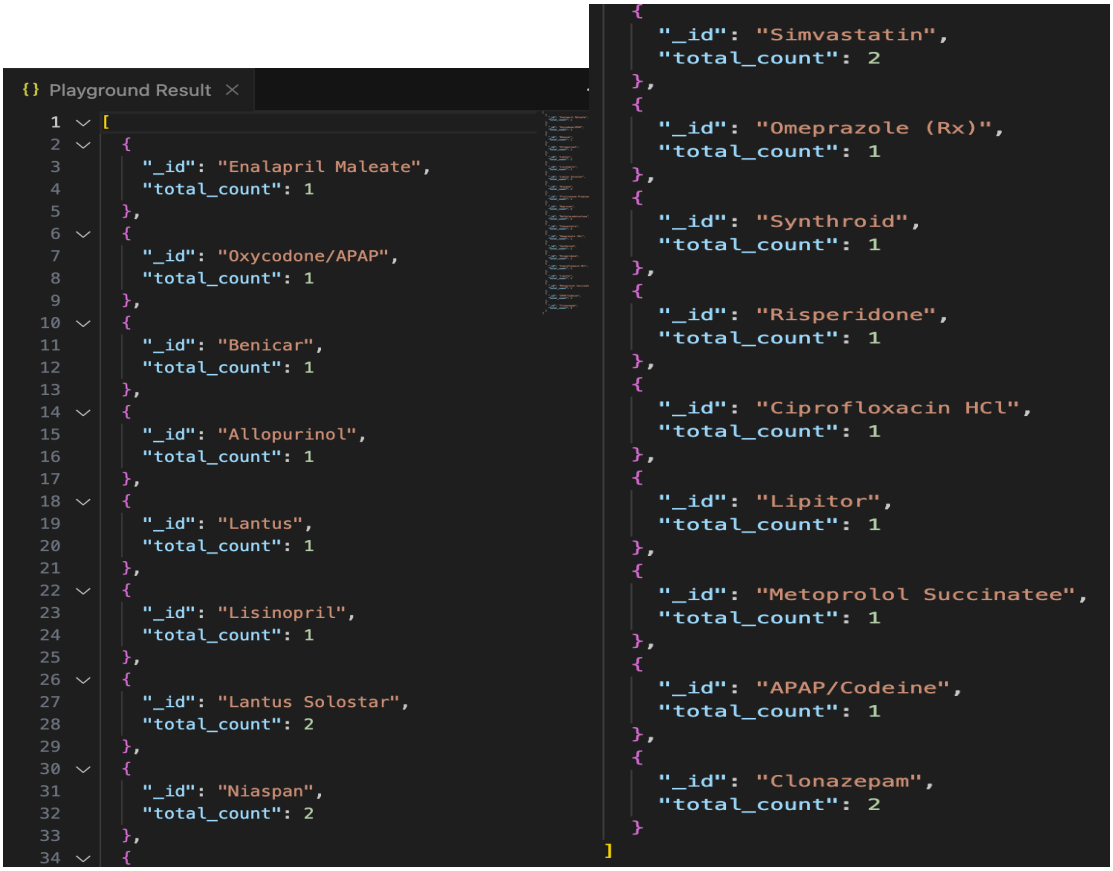


Query 2: Find the count of Each Medication

```
use('VitalSignHealthManager');

db.Medication.aggregate([
    {
        $group: {
            _id: "$MedName",
            total_count: { $sum: 1 }
        }
    }
]);
```

Output:



Query 3: Give the average no. Of Staffs in each Position across the hospital

```
use('VitalSignHealthManager');

db.Staff.aggregate([
    {
        $group: {
            _id: "$St_Position",
            positionCount: { $sum: 1 }
        }
    },
    {
        $group: {
            _id: null,
            totalPositions: { $sum: "$positionCount" },
            positions: {
                $push: {
                    position: "$_id",
                    count: "$positionCount"
                }
            }
        }
    },
    {
        $unwind: "$positions"
    },
    {
        $project: {
            _id: 0,
            position: "$positions.position",
            averageCount: { $divide: ["$positions.count", "$totalPositions"] }
        }
    }
]);
```

Output:

Playground Result


```
[
  {
    "position": "Admin Staff",
    "averageCount": 0.0967741935483871
  },
  {
    "position": "Medical Assistant",
    "averageCount": 0.07526881720430108
  },
  {
    "position": "Nurse",
    "averageCount": 0.1935483870967742
  },
  {
    "position": "Technician",
    "averageCount": 0.10752688172043011
  },
  {
    "position": "Maintanance Staff",
    "averageCount": 0.07526881720430108
  },
  {
    "position": "IT Support Staff",
    "averageCount": 0.043010752688172046
  },
  {
    "position": "Midwife",
    "averageCount": 0.08602150537634409
  },
  {
    "position": "Billing and Coding Specialist",
    "averageCount": 0.06451612903225806
  },
  {
    "position": "Security Personnel",
    "averageCount": 0.03225806451612903
  },
```

```
  {
    "position": "Social Worker",
    "averageCount": 0.03225806451612903
  },
  {
    "position": "Receptionist",
    "averageCount": 0.1935483870967742
  }
]
```


Query 4: Return the count of Adult and Non-Adult Patients, Grouped by Gender

```
db.Patients.aggregate([
    {
        $group: {
            _id: {
                ageCategory: {
                    $cond: { if: { $gte: ["$P_Age", 18] }, then: "Greater than 18", else: "Less than 18" }
                },
                gender: "$P_Gender"
            },
            count: { $sum: 1 }
        }
    }
]);
```

Output:

Playground Result


```
[
  {
    "_id": {
      "ageCategory": "Less than 18",
      "gender": "Female"
    },
    "count": 12
  },
  {
    "_id": {
      "ageCategory": "Greater than 18",
      "gender": "Female"
    },
    "count": 10
  },
  {
    "_id": {
      "ageCategory": "Greater than 18",
      "gender": "Male"
    },
    "count": 24
  },
  {
    "_id": {
      "ageCategory": "Less than 18",
      "gender": "Male"
    },
    "count": 4
  }
]
```

IV. Database access via Python

Connecting to DB:

```
import mysql.connector
import pandas as pd
import matplotlib.pyplot as plt
import os
import pymysql
import numpy as np
import seaborn as sns
```

[1] ✓ 3.8s Python

```
host = os.getenv('Local')
```

[2] ✓ 0.0s Python

```
conn= pymysql.connect(
    host='localhost',
    user='root',
    passwd='sinduja123', # Put your password in this place
    db='VitalSign_HealthManager', charset='utf8mb4')
```

[57] ✓ 0.0s Python

Collecting Data from DB:

```
patients = pd.read_sql('select * from Patients', conn)
doctor = pd.read_sql('select * from Doctor', conn)
staff = pd.read_sql('select * from Staff', conn)
treatmentplan = pd.read_sql('select * from TreatmentPlan', conn)
medication = pd.read_sql('select * from Medication', conn)
department = pd.read_sql('select * from Department', conn)
doc_med = pd.read_sql('select * from Doctor_Medication', conn)
tp_patient = pd.read_sql('select * from TreatmentPlan_Patient', conn)
appointment = pd.read_sql('select * from Appointment', conn)
ap_visits = pd.read_sql('select * from Patient_Visits', conn)
doc_general = pd.read_sql('select * from General', conn)
medicalrecord = pd.read_sql('select * from MedicalRecord', conn)
pat_med = pd.read_sql('select * from Patient_Medication', conn)
doc_researcher = pd.read_sql('select * from Researcher', conn)
researchproject = pd.read_sql('select * from ResearchProject', conn)
researchpublication = pd.read_sql('select * from ResearchPublication', conn)
rp_author = pd.read_sql('select * from ResearchPublication_Authors', conn)
researchteam = pd.read_sql('select * from ResearchTeam', conn)
doc_surgeon = pd.read_sql('select * from Surgeon', conn)
tp_doc = pd.read_sql('select * from TreatmentPlan_Doctor', conn)
ward = pd.read_sql('select * from Ward', conn)
```

[5] ✓ 0.0s Python

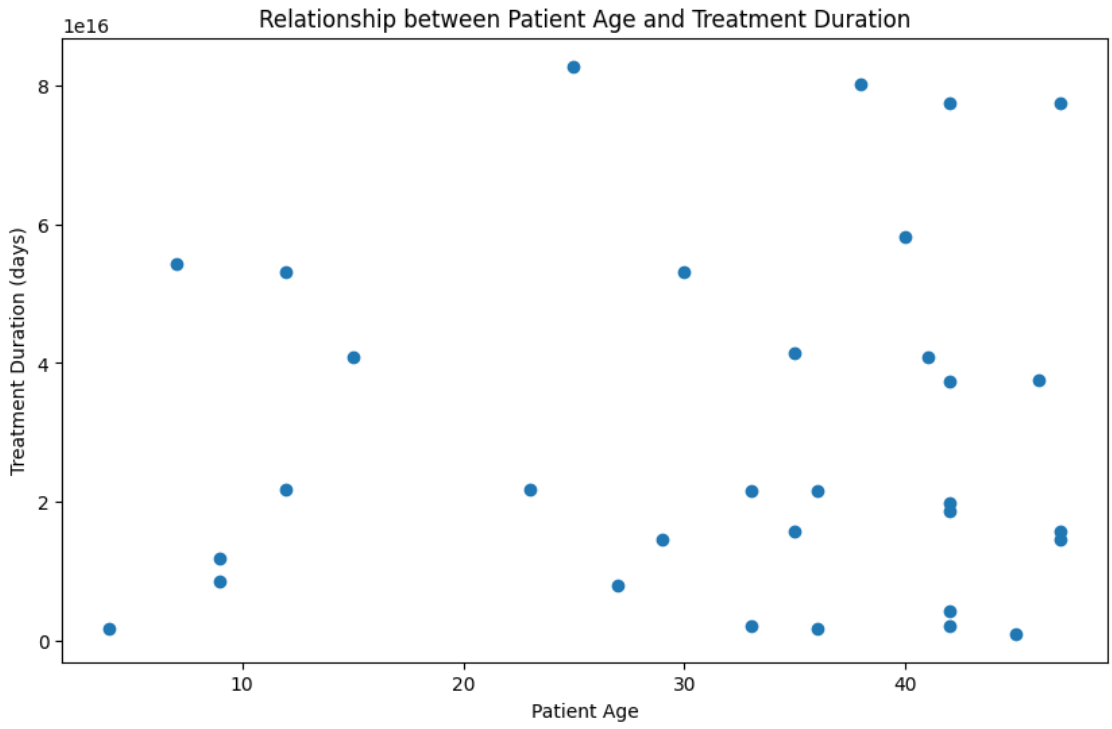
Query 1: Is there a correlation between patient age and treatment duration?

```
[34]  treatmentplan['duration'] = treatmentplan['End_Date'] - treatmentplan['Start_Date']
      tp_data = pd.merge(treatmentplan, tp_patient, on = 'Tp_Id', how = 'inner')
      ✓ 0.0s                                                                    Python

[42]  patients = patients.rename(columns={'P_id': 'P_Id'})
      merged_data = pd.merge(patients, tp_data, on='P_Id', how='inner')
      ✓ 0.0s                                                                    Python

[46]  # Create a scatter plot to visualize the relationship between patient age and treatment duration
      plt.figure(figsize=(10, 6))
      plt.scatter(merged_data['P_Age'], merged_data['duration'])
      plt.title('Relationship between Patient Age and Treatment Duration')
      plt.xlabel('Patient Age')
      plt.ylabel('Treatment Duration (days)')
      plt.show()
      ✓ 0.1s                                                                    Python
```

Output:



Insights based on the graph:

From the scatterplot, we can see that there is no correlation between the Patient Age and the duration of the Treatment. It further confirms that the duration of the Treatment solely depends on the severity of the sickness, rather than the age.

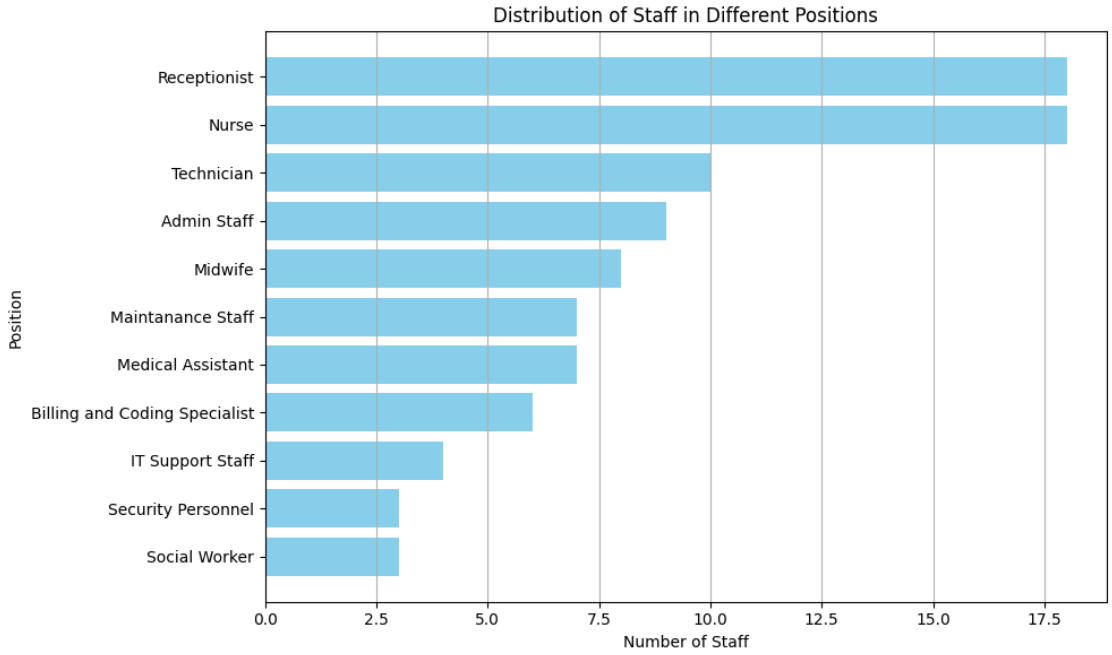
Query 2: What can we say about the distribution of staff in different positions?

```
# Count the number of staff in each position
staff_position_count = staff['St_Position'].value_counts().reset_index()
staff_position_count.columns = ['Position', 'Count']

# Sort the data by count in descending order
staff_position_count = staff_position_count.sort_values(by='Count', ascending=True)

# Create a horizontal bar plot to visualize the distribution of staff positions
plt.figure(figsize=(10, 6))
plt.barh(staff_position_count['Position'], staff_position_count['Count'], color='skyblue')
plt.title('Distribution of Staff in Different Positions')
plt.xlabel('Number of Staff')
plt.ylabel('Position')
plt.grid(axis='x')
plt.tight_layout()
plt.show()
```

[60] ✓ 0.1s Python



Insights based on the graph:

The Number of Receptionists and Nurse is the highest, followed by Technicians, Admin Staff, Midwife and so on.

Query 3: Which doctor has the highest number of appointments, and how does their appointment load compare to other doctors?

Query 3

Which doctor has the highest number of appointments, and how does their appointment load compare to other doctors?

```
doctor_appointment_count = ap_visits['D_Id'].value_counts().reset_index()
doctor_appointment_count.columns = ['D_Id', 'AppointmentCount']
```

[71] ✓ 0.0s Python

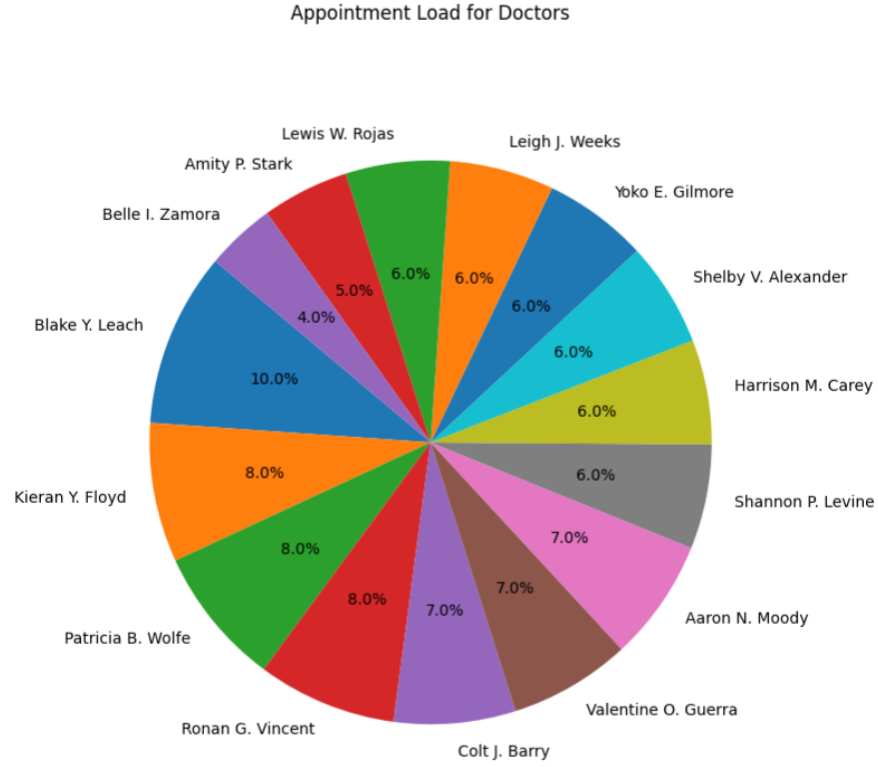
```
# Merge with the Doctor table to get doctor names
doctor_appointment_count = pd.merge(doctor_appointment_count, doctor, on='D_Id', how='inner', suffixes=('_left', '_right'))

# Sort the data by appointment count in descending order
doctor_appointment_count = doctor_appointment_count.sort_values(by='AppointmentCount', ascending=False)
```

[73] ✓ 0.0s Python

```
plt.figure(figsize=(8, 8))
plt.pie(doctor_appointment_count['AppointmentCount'], labels=doctor_appointment_count['D_Name'], autopct='%1.1f%%', startangle=140)
plt.title('Appointment Load for Doctors')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.tight_layout()
plt.show()
```

[76] ✓ 0.1s Python



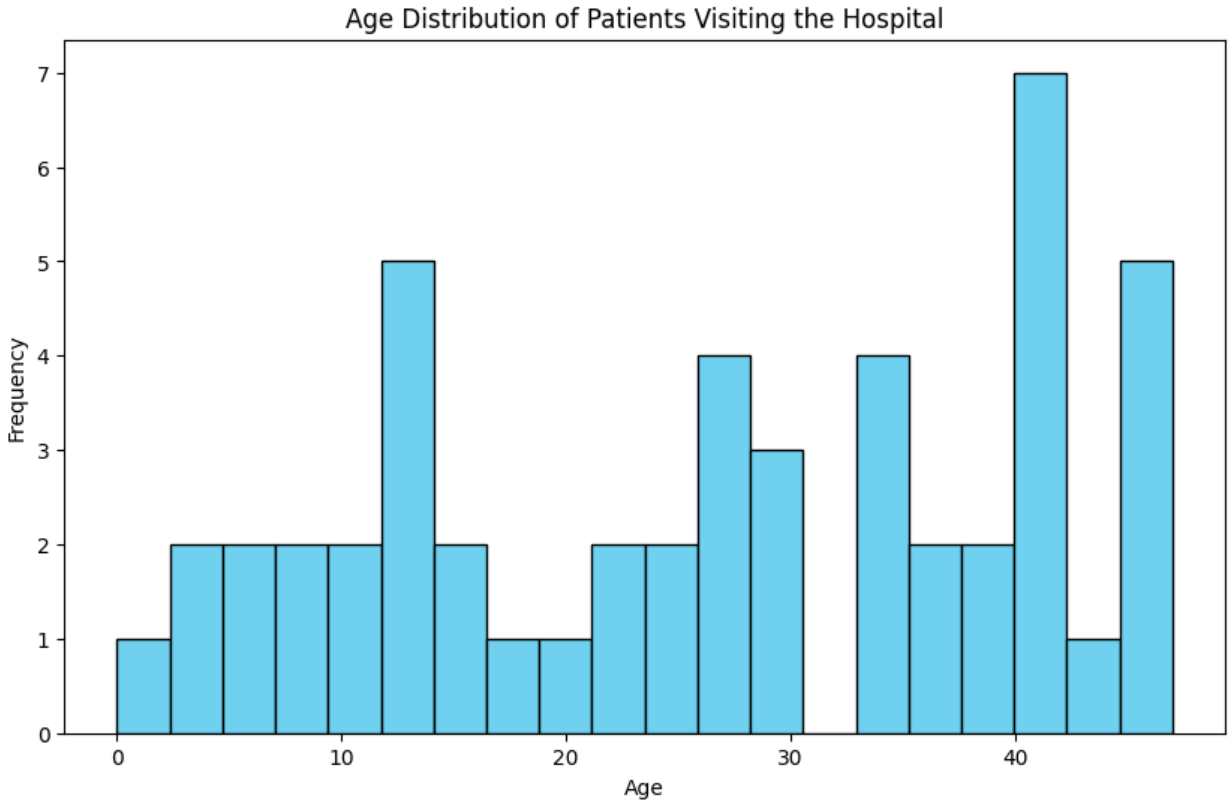
Insights based on the graph:

From the pie chart, almost all the doctors have the same appointment load, with just a difference of + or – 2%.

Query – 4: What is the age distribution of patients visiting the hospital?

```
plt.figure(figsize=(10, 6))
plt.hist(patients['P_Age'], bins=20, color='skyblue', edgecolor='black')
plt.title('Age Distribution of Patients Visiting the Hospital')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```

Output:



Insights from the Graph:

The age of most people visiting the hospital is between 10 – 40, with the highest being in the bucket 30-40

V. Summary and Recommendations

The hospital management database model offers a comprehensive structure for allocating and overseeing important hospital operations components. It effectively gathers vital data, enabling faster procedures and high-quality medical care, including personnel specifics, patient demographics, treatment plans, appointments, and prescriptions.

The model facilitates informed decision-making and effective resource allocation by providing healthcare professionals with timely access to pertinent information and ensuring data integrity through well-defined entities and relationships.

However, to further enhance the system, it is recommended to focus on improving scalability, optimizing performance, and integrating advanced analytics capabilities.

Furthermore, the integration of resilient security protocols, intuitive user interfaces, and feedback mechanisms will bolster the system's dependability, usefulness, and ongoing enhancement, ultimately augmenting the caliber of medical services rendered by the hospital.

In conclusion, although the hospital management database model in use today provides a strong framework for managing hospital operations, continuous improvements and modifications are necessary to successfully address changing patient needs. The hospital can guarantee system dependability, efficiency, and the capacity to employ data-driven insights for improved patient outcomes and operational excellence by placing a high priority on scalability, performance optimization, security, and user experience.