SQL Assignment – Hospital Management System

Abstract:

The Hospital Management System database is designed to efficiently manage patient information, doctor schedules, appointments, medications, and medical records within a healthcare facility. This database incorporates various data types, including nominal, ordinal, interval, and ratio data, to represent different aspects of hospital operations. Through the use of multiple tables and foreign key relationships, the database ensures data integrity, reduces redundancy, and facilitates effective data retrieval and analysis.

Introduction:

In today's healthcare landscape, efficient management of patient information is essential for delivering quality care and optimising hospital operations. The Hospital Management System database serves as a centralised repository for storing and managing patient records, doctor information, appointments, medications, and medical records. By utilising various data types and establishing relationships between tables, the database enables healthcare providers to streamline administrative tasks, improve communication, and enhance patient care delivery. This report explores the design and implementation of the Hospital Management System database, highlighting its structure, data type representation, ethical considerations, and example queries.

Database Generation:

The database is about the hospital management system database. I ensured that all data types, including nominal, ordinal, interval, and ratio data, were represented in the database. The database is generated with 1000 rows of data with appropriate values assigned.

Patients table:

- This table stores information about patients who visit the hospital.
- This table allows hospital staff to maintain records of patients and contact them for appointments or follow-up visits.
- Columns include:
  - Patient_id: A unique identifier for each patient.
  - Patient_name: The name of the patient.
  - Age: The age of the patient.
  - Gender: The gender of the patient.
  - Address: The address of the patient.
  - Contact_number: The contact number of the patient.

patient_id	patient_name	age	gender	address	contact_number
Filter	Filter	Fil...	Filter	Filter	Filter
1	John Williams	77	Female	00780 Porter Turnpike...	453.359.9803
2	Crystal Stuart	62	Female	491 Chavez Rapids...	001-785-307-0447x1815
3	Jason Nelson	82	Female	67212 Mercedes Lakes Apt. 532...	990-422-2083
4	Timothy Jackson	57	Female	565 Thomas Branch...	001-425-649-9003x774
5	Sharon Anderson	19	Male	93866 Cordova Divide...	001-816-469-7001x2412
6	Ashley Krueger	58	Male	693 Ortiz Plaza Suite 091...	210-454-3572x8723
7	Jason Singh	79	Male	464 Reyes Passage Apt. 919...	+1-763-228-9781x293
8	Leslie Thomas	21	Female	72756 Norman Locks Suite 846...	875.431.7403
9	Alan Hernandez	29	Male	685 Daniel Mall...	+1-908-345-9864x211
10	Wendy Martinez	55	Male	136 Young Via Suite 980...	(268)860-2072x9782

Doctors table:

- This table stores information about doctors working at the hospital.
- This table helps in managing doctor schedules, assigning them to specific departments, and tracking their areas of expertise.
- Columns include:
  - Doctor_id: A unique identifier for each doctor.
  - Doctor_name: The name of the doctor.
  - Specialty: The specialty or field of expertise of the doctor.
  - Department_id: A foreign key referencing the department table, indicating the department to which the doctor belongs.

doctor_id	doctor_name	specialty	department_id
Filter	Filter	Filter	Filter
1	Kaitlin Hamilton	Pediatrics	4
2	Katherine Johnson	Orthopedics	5
3	Tammy Anderson	Oncology	3
4	Luis Robinson	Oncology	1
5	Heidi Lewis	Oncology	4
6	Karen Cantu	Cardiology	3
7	Tina Evans	Oncology	2
8	Catherine Joyce	Orthopedics	1
9	Juan Price	Pediatrics	4
10	Michael Jenkins	Pediatrics	2

Departments table:

- This table stores information about different departments within the hospital.
- This table helps in categorising hospital services into distinct departments, such as cardiology, orthopaedics, etc.
- Columns include:
  - Department_id: A unique identifier for each department.
  - Department_name: The name of the department.

department_id	department_name
Filter	Filter
1	Cardiology
2	Orthopedics
3	Pediatrics
4	Neurology
5	Oncology

Appointments Table:

- This table stores information about appointments scheduled between patients and doctors.
- This table facilitates the scheduling and tracking of appointments, allowing hospital staff to manage patient visits efficiently.
- Columns include:
  - Appointment_id: A unique identifier for each appointment.
  - Patient_id: A foreign key referencing the doctor's table, indicating the patient for the appointment.
  - Doctor_id: A foreign key referencing the doctor's table, indicating the doctor for the appointment.
  - Appointment_date: The date of the appointment.
  - Appointment_time: The time of the appointment.

	appointment_id	patient_id	doctor_id	appointment_date	appointment_time
	Filter	Filter	Filter	Filter	Filter
1	1	15	3	2024-03-06	21:54
2	2	79	7	2024-03-14	07:27
3	3	10	4	2024-03-10	10:42
4	4	67	2	2024-03-18	14:40
5	5	87	5	2024-03-12	22:46
6	6	35	8	2024-03-24	02:11
7	7	37	9	2024-03-30	11:05
8	8	60	3	2024-03-23	20:57
9	9	56	3	2024-03-16	02:55
10	10	87	3	2024-03-26	10:28

Medications Table:

- This table stores information about medications prescribed to patients.
- This table helps in maintaining records of prescribed medications and managing patient treatments.
- Columns include:
  - Medication_id: A unique identifier for each medication.
  - Medication_name: The name of the medication.
  - Dosage: The dosage of the medication.
  - Frequency: The frequency of the medication dosage.

	medication_id	medication_name	dosage	frequency
	Filter	Filter	Filter	Filter
1	1	Aspirin	100mg	Once daily
2	2	Ibuprofen	200mg	Twice daily
3	3	Acetaminophen	500mg	Three times daily
4	4	Amoxicillin	250mg	Four times daily

Medical Records Table:

- This table stores information about medical records, including diagnoses and prescribed medications.
- This table helps in maintaining a comprehensive history of patient diagnoses, treatments, and medications prescribed by doctors.
- Columns include:
  - Record_id: A unique identifier for each medical record.
  - patient_id: A foreign key referencing the patient's table, indicating the patient associated with the record.
  - doctor_id: A foreign key referencing the doctor's table, indicating the doctor who made the diagnosis.
  - diagnosis: The diagnosis made by the doctor.
  - prescription_id: A foreign key referencing the Medications table, indicating the prescribed medication.
  - date: The date of the medical record entry.

	record_id	patient_id	doctor_id	diagnosis	prescription_id	date
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	16	9	Record trial order lead employee program.	2	2023-08-04
2	2	37	6	Ok list understand leave song live.	1	2023-09-21
3	3	92	3	Class these thus third.	1	2023-09-13
4	4	94	2	Close lot up. Happen property turn benefit.	4	2023-04-26
5	5	80	9	Guess no conference amount city six property.	2	2023-12-10
6	6	99	4	Door public story act receive dinner follow.	2	2023-12-03
7	7	51	6	Quickly well protect already on several economic.	1	2023-10-05
8	8	78	10	Chance national central into leg production too.	4	2023-07-24
9	9	26	4	Because family court while fund.	1	2023-08-25
10	10	46	1	Late allow large success.	4	2023-07-21

Data Types Representation:

Nominal data:

- In the patients table, the gender column represents nominal data. The gender categories have no inherent order or ranking, and they simply represent distinct labels for different genders.

Ordinal data:

- In the doctors table, the specialty column represents ordinal data. Examples of ordinal values for specialty include cardiology, orthopaedics, and paediatrics. While these specialties have a specific order or ranking, the differences between them are not uniform or measurable.

Interval Data:

- In the appointments table, the appointment_date column represents the interval date. The intervals between appointment dates are consistent and measurable, but there is no true zero point.

Ratio Data:

- In the patients table, the age column represents' ratio data. Age represents numeric values with a true zero point, where zero indicates the absence of age. Arithmetic operations such as addition, subtraction, multiplication, and division are meaningful for age values.

Code:

```
import sqlite3
from faker import Faker
import random
import datetime

# Initialize Faker library
fake = Faker()

# Create a SQLite database
conn = sqlite3.connect('hospital.db')
c = conn.cursor()

# Create Patients table
c.execute('''CREATE TABLE IF NOT EXISTS Patients (
    patient_id INTEGER PRIMARY KEY,
    patient_name TEXT,
    age INTEGER,
    gender TEXT,
    address TEXT,
    contact_number TEXT
)''')

# Create Doctors table
c.execute('''CREATE TABLE IF NOT EXISTS Doctors (
    doctor_id INTEGER PRIMARY KEY,
    doctor_name TEXT,
    specialty TEXT,
    department_id INTEGER,
    FOREIGN KEY (department_id) REFERENCES Departments(department_id)
)''')

# Create Departments table
```

```
c.execute('''CREATE TABLE IF NOT EXISTS Departments (
    department_id INTEGER PRIMARY KEY,
    department_name TEXT
)''')

# Create Appointments table
c.execute('''CREATE TABLE IF NOT EXISTS Appointments (
    appointment_id INTEGER PRIMARY KEY,
    patient_id INTEGER,
    doctor_id INTEGER,
    appointment_date DATE,
    appointment_time TIME,
    FOREIGN KEY (patient_id) REFERENCES Patients(patient_id),
    FOREIGN KEY (doctor_id) REFERENCES Doctors(doctor_id)
)''')

# Create Medications table
c.execute('''CREATE TABLE IF NOT EXISTS Medications (
    medication_id INTEGER PRIMARY KEY,
    medication_name TEXT,
    dosage TEXT,
    frequency TEXT
)''')

# Create Medical Records table
c.execute('''CREATE TABLE IF NOT EXISTS MedicalRecords (
    record_id INTEGER PRIMARY KEY,
    patient_id INTEGER,
    doctor_id INTEGER,
    diagnosis TEXT,
    prescription_id INTEGER,
    date DATE,
    FOREIGN KEY (patient_id) REFERENCES Patients(patient_id),
    FOREIGN KEY (doctor_id) REFERENCES Doctors(doctor_id),
    FOREIGN KEY (prescription_id) REFERENCES Medications(medication_id)
)''')

# Populate Departments table
departments = ['Cardiology', 'Orthopedics', 'Pediatrics', 'Neurology', 'Oncology']
for dept_name in departments:
    c.execute("INSERT INTO Departments (department_name) VALUES (?)", (dept_name,))

# Populate Doctors table
for _ in range(10):
    doctor_name = fake.name()
    specialty = fake.random_element(elements=('Cardiology', 'Orthopedics', 'Pediatrics', 'Neurology', 'Oncology'))
    department_id = fake.random_int(min=1, max=len(departments))
    c.execute("INSERT INTO Doctors (doctor_name, specialty, department_id) VALUES (?, ?, ?)", (doctor_name, specialty,
                                                                                               department_id))

# Populate Patients table
for _ in range(100):
    patient_name = fake.name()
    age = fake.random_int(min=18, max=90)
    gender = fake.random_element(elements=('Male', 'Female'))
    address = fake.address()
    contact_number = fake.phone_number()
    c.execute("INSERT INTO Patients (patient_name, age, gender, address, contact_number) VALUES (?, ?, ?, ?, ?)",
              (patient_name, age, gender, address, contact_number))

# Populate Appointments table
for _ in range(500):
    patient_id = fake.random_int(min=1, max=100)
    doctor_id = fake.random_int(min=1, max=10)
```

```
    appointment_date = fake.date_between(start_date='today', end_date='+30d')
    appointment_time = fake.time(pattern='%H:%M')
    c.execute("INSERT INTO Appointments (patient_id, doctor_id, appointment_date, appointment_time) VALUES (?, ?, ?, ?)",
              (patient_id, doctor_id, appointment_date, appointment_time))

# Populate Medications table
medications = [('Aspirin', '100mg', 'Once daily'), ('Ibuprofen', '200mg', 'Twice daily'), ('Acetaminophen', '500mg',
                                                                                          'Three times daily'),
               ('Amoxicillin', '250mg', 'Four times daily')]
for med in medications:
    c.execute("INSERT INTO Medications (medication_name, dosage, frequency) VALUES (?, ?, ?)", med)

# Populate Medical Records table
for _ in range(200):
    patient_id = fake.random_int(min=1, max=100)
    doctor_id = fake.random_int(min=1, max=10)
    diagnosis = fake.text(max_nb_chars=50)
    prescription_id = fake.random_int(min=1, max=len(medications))
    date = fake.date_between(start_date='-1y', end_date='today')
    c.execute("INSERT INTO MedicalRecords (patient_id, doctor_id, diagnosis, prescription_id, date) VALUES (?, ?, ?, ?, ?)"
              , (patient_id, doctor_id, diagnosis, prescription_id, date))

# Commit changes and close connection
conn.commit()
conn.close()
```

Database Schema:

Name	Type	Schema
Appointments		CREATE TABLE Appointments (appointment_id INTEGER PRIMARY KEY, patient_id INTEGER, doctor_id INTEGER, appointment_date DATE, appointment_time TIME, FOREIGN KEY (patient_id) REFERENCES Patients(patient_id), FOREIGN KEY (doctor_id) REFERENCES Doctors(doctor_id))
appointment_id	INTEGER	"appointment_id" INTEGER
patient_id	INTEGER	"patient_id" INTEGER
doctor_id	INTEGER	"doctor_id" INTEGER
appointment_date	DATE	"appointment_date" DATE
appointment_time	TIME	"appointment_time" TIME
Departments		CREATE TABLE Departments (department_id INTEGER PRIMARY KEY, department_name TEXT)
department_id	INTEGER	"department_id" INTEGER
department_name	TEXT	"department_name" TEXT
Doctors		CREATE TABLE Doctors (doctor_id INTEGER PRIMARY KEY, doctor_name TEXT, specialty TEXT, department_id INTEGER, FOREIGN KEY (department_id) REFERENCES Departments(department_id))
doctor_id	INTEGER	"doctor_id" INTEGER
doctor_name	TEXT	"doctor_name" TEXT
specialty	TEXT	"specialty" TEXT
department_id	INTEGER	"department_id" INTEGER
MedicalRecords		CREATE TABLE MedicalRecords (record_id INTEGER PRIMARY KEY, patient_id INTEGER, doctor_id INTEGER, diagnosis TEXT,

Name	Type	Schema
		prescription_id INTEGER, date DATE, FOREIGN KEY (patient_id) REFERENCES Patients(patient_id), FOREIGN KEY (doctor_id) REFERENCES Doctors(doctor_id), FOREIGN KEY (prescription_id) REFERENCES Medications(medication_id))
record_id	INTEGER	"record_id" INTEGER
patient_id	INTEGER	"patient_id" INTEGER
doctor_id	INTEGER	"doctor_id" INTEGER
diagnosis	TEXT	"diagnosis" TEXT
prescription_id	INTEGER	"prescription_id" INTEGER
date	DATE	"date" DATE
Medications		CREATE TABLE Medications (medication_id INTEGER PRIMARY KEY, medication_name TEXT, dosage TEXT, frequency TEXT)
medication_id	INTEGER	"medication_id" INTEGER
medication_name	TEXT	"medication_name" TEXT
dosage	TEXT	"dosage" TEXT
frequency	TEXT	"frequency" TEXT
Patients		CREATE TABLE Patients (patient_id INTEGER PRIMARY KEY, patient_name TEXT, age INTEGER, gender TEXT, address TEXT, contact_number TEXT)
patient_id	INTEGER	"patient_id" INTEGER
patient_name	TEXT	"patient_name" TEXT
age	INTEGER	"age" INTEGER
gender	TEXT	"gender" TEXT
address	TEXT	"address" TEXT
contact_number	TEXT	"contact_number" TEXT

Justification of Separate Tables:

Separating the data into multiple tables allows for better organisation and reduces redundancy. For example, having a separate Patients table allows for storing patient information only once and referencing it in other tables like Appointments and Medical Records. Similarly, separating doctors and departments helps in managing doctor information and their respective specialties and departments efficiently.

Ethical Discussion:

- In the hospital database, patient information such as names, contact numbers, and medical records is sensitive and confidential. Unauthorised access to this data could lead to privacy breaches, identity theft, or discrimination. It is ethically imperative to implement robust security measures, access controls, and encryption techniques to safeguard patient data from unauthorised access or misuse.

Queries:

1. Retrieve all appointments scheduled for a specific date:

```
1  SELECT *
2  FROM Appointments
3  WHERE appointment_date = '2024-03-10';
4
```

	appointment_id	patient_id	doctor_id	appointment_date	appointment_time
1	3	10	4	2024-03-10	10:42
2	21	50	9	2024-03-10	13:14
3	36	54	8	2024-03-10	15:26
4	38	57	1	2024-03-10	18:05
5	72	11	3	2024-03-10	07:00
6	91	74	10	2024-03-10	13:48
7	128	4	2	2024-03-10	02:50
8	137	55	9	2024-03-10	07:23
9	231	53	6	2024-03-10	22:08
10	263	82	4	2024-03-10	20:48
11	277	24	3	2024-03-10	21:39
12	298	39	8	2024-03-10	06:42
13	370	89	8	2024-03-10	19:23
14	389	74	10	2024-03-10	17:42
15	394	75	9	2024-03-10	13:11
16	395	22	4	2024-03-10	02:36
17	400	48	3	2024-03-10	11:58
18	416	5	9	2024-03-10	05:34
19	459	40	10	2024-03-10	02:06
20	466	29	5	2024-03-10	16:53

2. Retrieve patient information along with their assigned doctor and department:

```
3   JOIN Appointments ON Patients.patient_id = Appointments.patient_id
4   JOIN Doctors ON Appointments.doctor_id = Doctors.doctor_id
5   JOIN Departments ON Doctors.department_id = Departments.department_id;
6
```

	patient_name	doctor_name	department_name
1	Benjamin Frazier	Tammy Anderson	Pediatrics
2	Wesley Chang	Tina Evans	Orthopedics
3	Wendy Martinez	Luis Robinson	Cardiology
4	Jessica Smith MD	Katherine Johnson	Oncology
5	Kristin Alexander	Heidi Lewis	Neurology
6	Vincent Wagner DVM	Catherine Joyce	Cardiology
7	Elizabeth Gibson	Juan Price	Neurology
8	Jason Lee	Tammy Anderson	Pediatrics
9	Brett Bell	Tammy Anderson	Pediatrics
10	Kristin Alexander	Tammy Anderson	Pediatrics
11	Michael Mcpherson	Catherine Joyce	Cardiology
12	Michael Nicholson	Kaitlin Hamilton	Neurology
13	Kristine Wagner	Luis Robinson	Cardiology
14	John Figueroa	Catherine Joyce	Cardiology
15	Joshua Roberts	Tammy Anderson	Pediatrics
16	Taylor Schneider	Catherine Joyce	Cardiology
17	Wendy Martinez	Tammy Anderson	Pediatrics
18	Alan Hernandez	Juan Price	Neurology
19	Timothy Mcgrath	Tammy Anderson	Pediatrics
20	Taylor Schneider	Catherine Joyce	Cardiology
21	Michael Steele	Juan Price	Neurology
22	Dr. Kelli Wolfe	Tina Evans	Orthopedics
23	Kenneth Mcknight	Tammy Anderson	Pediatrics
24	Shannon Knight	Juan Price	Neurology
25	Danielle Carter	Juan Price	Neurology

3. Retrieve the most prescribed medication and its dosage:

```
1  SELECT medication_name, dosage, COUNT(*) AS prescription_count
2  FROM MedicalRecords
3  JOIN Medications ON MedicalRecords.prescription_id = Medications.medication_id
4  GROUP BY medication_name
5  ORDER BY prescription_count DESC
6  LIMIT 1;
7
```

	medication_name	dosage	prescription_count
1	Ibuprofen	200mg	52

Conclusion:

The Hospital Management System database represents a comprehensive solution for managing hospital operations and patient care effectively. By incorporating nominal, ordinal, interval, and ratio data types, the database caters to diverse information needs within a healthcare facility. Ethical considerations, such as patient privacy and data security, are paramount in safeguarding sensitive information and maintaining trust between healthcare providers and patients.

Name: Ajay Prabhat Gorrumuchu
Student ID: 22069870