Name : Aditya Mahesh Pattar

Div: B

Roll no. 2401155

ADBMS SOLUTIONS SQL Operations

Scenario 1: Library

Schema:-

- BOOKS: (Book_ID, Title, Author, Genre, Price, Publication_Year, Copies)
- BORROWERS: (Borrower_ID, Name, Address, Phone, Membership_Type)
- ISSUES: (Issue_ID, Borrower_ID, Book_ID, Issue_Date, Return_Date)

Easy Questions :

1. Create a table BOOKS with the given schema

```
CREATE TABLE BOOKS (
Book_ID INT PRIMARY KEY,
Title VARCHAR(255),
Author VARCHAR(255),
Genre VARCHAR(100),
Price DECIMAL(10, 2),
Publication_Year INT,
Copies INT
);
```

2. Insert at least 5 rows into the BOOKS table.

```
INSERT INTO BOOKS (Book_ID, Title, Author, Genre, Price,
Publication_Year, Copies)
VALUES
(1, 'The Alchemist', 'Paulo Coelho', 'Fiction', 350, 1988, 10),
(2, 'Becoming', 'Michelle Obama', 'Biography', 500, 2018, 5),
(3, 'Sapiens', 'Yuval Noah Harari', 'History', 750, 2015, 8),
(4, 'Atomic Habits', 'James Clear', 'Self-help', 550, 2018, 2),
(5, '1984', 'George Orwell', 'Fiction', 300, 1949, 6);
```

3. Display all the details of books available in the library.

SELECT * FROM BOOKS;

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

Book_ID	Title	Author	Genre	Price	Publication_Year	Copies
1	The Alchemist	Paulo Coelho	Fiction	350.00	1988	10
2	Becoming	Michelle Obama	Biography	500.00	2018	5
3	Sapiens	Yuval Noah Harari	History	750.00	2015	8
4	Atomic Habits	James Clear	Self-help	550.00	2018	2
5	1984	George Orwell	Fiction	300.00	1949	6
NULL	NULL	NULL	NULL	NULL	NULL	NULL

BOOKS 4

4. Display the list of books published after 2015.

SELECT * FROM BOOKS WHERE Publication_Year > 2015;

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

Book_ID	Title	Author	Genre	Price	Publication_Year	Copies
2	Becoming	Michelle Obama	Biography	500.00	2018	5
4	Atomic Habits	James Clear	Self-help	550.00	2018	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL

5. Create a table BORROWERS with the given schema.

CREATE TABLE BORROWERS (

Borrower_ID INT PRIMARY KEY,

Name VARCHAR(255),

Address VARCHAR(255),

Phone VARCHAR(15),

Membership_Type VARCHAR(50)

);

6. Insert at least 5 rows into the BORROWERS table.

INSERT INTO BORROWERS (Borrower_ID, Name, Address, Phone,

Membership_Type)

VALUES

(1, 'Alice', '123 Elm Street', '1234567890', 'Gold'),

(2, 'Bob', '456 Oak Avenue', '9876543210', 'Silver'),

(3, 'Charlie', '789 Pine Lane', '5555555555', 'Gold'),

(4, 'Diana', '321 Maple Road', '4444444444', 'Bronze'),

(5, 'Edward', '654 Cedar Street', '3333333333', 'Gold');

7. Display the names and phone numbers of all borrowers.

SELECT Name, Phone FROM BORROWERS;

Result Grid | Filter Rows

Name	Phone
Alice	1234567890
Bob	9876543210
Charlie	5555555555
Diana	4444444444
Edward	3333333333

8. Display the list of borrowers who have a "Gold" membership type.

SELECT * FROM BORROWERS WHERE Membership_Type = 'Gold';

Result Grid | Filter Rows: | Edit: | Export/In

Borrower_ID	Name	Address	Phone	Membership_Type
1	Alice	123 Elm Street	1234567890	Gold
3	Charlie	789 Pine Lane	5555555555	Gold
5	Edward	654 Cedar Street	3333333333	Gold
NULL	NULL	NULL	NULL	NULL

9. Create a table ISSUES with the given schema.

CREATE TABLE ISSUES (

Issue_ID INT PRIMARY KEY,

Borrower_ID INT,

Book_ID INT,

Issue_Date DATE,

Return_Date DATE,

FOREIGN KEY (Borrower_ID) REFERENCES

BORROWERS(Borrower_ID),

FOREIGN KEY (Book_ID) REFERENCES BOOKS(Book_ID)

);

10. Insert 5 records into the ISSUES table.

INSERT INTO ISSUES (Issue_ID, Borrower_ID, Book_ID, Issue_Date,

Return_Date)

VALUES

(1, 1, 2, '2024-11-01', '2024-11-15'),

(2, 2, 3, '2024-11-02', '2024-11-12'),

(3, 3, 1, '2024-11-03', NULL),

(4, 4, 5, '2024-11-05', '2024-11-20'),

(5, 5, 4, '2024-11-07', NULL);

Result Grid | Filter Rows: | Edit:

Issue_ID	Borrower_ID	Book_ID	Issue_Date	Return_Date
1	1	2	2024-11-01	2024-11-15
2	2	3	2024-11-02	2024-11-12
3	3	1	2024-11-03	NULL
4	4	5	2024-11-05	2024-11-20
5	5	4	2024-11-07	NULL
NULL	NULL	NULL	NULL	NULL

Medium Questions :

1. Display the title and author of all books priced above 500.

SELECT Title, Author FROM BOOKS WHERE Price > 500;

Result Grid | Filter Rows:

Title	Author
Sapiens	Yuval Noah Harari
Atomic Habits	James Clear

2. Update the price of all books in the "Fiction" genre by increasing it by 10%.

UPDATE BOOKS SET Price = Price * 1.10 WHERE Genre = 'Fiction';

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

Book_ID	Title	Author	Genre	Price	Publication_Year	Copies
1	The Alchemist	Paulo Coelho	Fiction	385.00	1988	10
2	Becoming	Michelle Obama	Biography	500.00	2018	5
3	Sapiens	Yuval Noah Harari	History	750.00	2015	8
4	Atomic Habits	James Clear	Self-help	550.00	2018	2
5	1984	George Orwell	Fiction	330.00	1949	6
NULL	NULL	NULL	NULL	NULL	NULL	NULL

3. Delete the records of books that have no copies left.

DELETE FROM BOOKS WHERE Copies = 0;

4. Delete the records of books that have no copies left

DELETE FROM BOOKS WHERE Copies = 0;

5. Create a view AVAILABLE_BOOKS showing all books with more than 5 copies.

CREATE VIEW AVAILABLE_BOOKS AS

SELECT * FROM BOOKS WHERE Copies > 5;

6. Retrieve all the books sorted by Publication_Year in descending order.

SELECT * FROM BOOKS ORDER BY Publication_Year DESC;

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

Book_ID	Title	Author	Genre	Price	Publication_Year	Copies
2	Becoming	Michelle Obama	Biography	500.00	2018	5
4	Atomic Habits	James Clear	Self-help	550.00	2018	2
3	Sapiens	Yuval Noah Harari	History	750.00	2015	8
1	The Alchemist	Paulo Coelho	Fiction	385.00	1988	10
5	1984	George Orwell	Fiction	330.00	1949	6
NULL	NULL	NULL	NULL	NULL	NULL	NULL

7. Retrieve the details of borrowers who borrowed more than 2 books

SELECT b.Name, b.Phone, COUNT(i.Issue_ID) AS Books_Borrowed

FROM BORROWERS b

JOIN ISSUES i ON b.Borrower_ID = i.Borrower_ID

GROUP BY b.Borrower_ID, b.Name, b.Phone

HAVING COUNT(i.Issue_ID) > 2;

Result Grid | Filter Rows:

Name	Phone	Books_Borrowed

Hard Questions:

1. Find customers who rented a movie and never returned it (use LEFT JOIN and NULL check).

SELECT b.Name, b.Phone

FROM BORROWERS b

LEFT JOIN ISSUES i ON b.Borrower_ID = i.Borrower_ID

WHERE i.Return_Date IS NULL;

Result Grid | Filter Rov

Name	Phone
Charlie	5555555555
Edward	3333333333

2. Display the books rented the most times.

SELECT b.Title, COUNT(i.Book_ID) AS Times_Rented

FROM BOOKS b

JOIN ISSUES i ON b.Book_ID = i.Book_ID

GROUP BY b.Book_ID, b.Title

ORDER BY Times_Rented DESC

LIMIT 1;

Result Grid | Filter Rows:

Title	Times_Rented
The Alchemist	1

•

Scenario 2: Movie

Schema :

● MOVIES: (Movie_ID, Title, Genre, Release_Date, Rating, Director)

● CUSTOMERS: (Customer_ID, Name, Email, Phone, Membership_Type)

● RENTALS: (Rental_ID, Customer_ID, Movie_ID, Rental_Date, Return_Date)

Easy Questions :

1. Create a table MOVIES with the given schema.

```
CREATE TABLE MOVIES (
Movie_ID INT PRIMARY KEY,
Title VARCHAR(255),
Genre VARCHAR(100),
Release_Date DATE,
Rating DECIMAL(3, 2),
Director VARCHAR(255)
);
```

2. Insert at least 5 rows into the MOVIES table.

```
INSERT INTO MOVIES (Movie_ID, Title, Genre, Release_Date, Rating,
Director)
VALUES
(1, 'Inception', 'Sci-Fi', '2010-07-16', 8.8, 'Christopher Nolan'),
(2, 'The Avengers', 'Action', '2012-05-04', 8.0, 'Joss Whedon'),
(3, 'Titanic', 'Romance', '1997-12-19', 7.8, 'James Cameron'),
(4, 'The Godfather', 'Crime', '1972-03-24', 9.2, 'Francis Ford Coppola'),
(5, 'The Dark Knight', 'Action', '2008-07-18', 9.0, 'Christopher Nolan');
```

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

Movie_ID	Title	Genre	Release_Date	Rating	Director
1	Inception	Sci-Fi	2010-07-16	8.80	Christopher Nolan
2	The Avengers	Action	2012-05-04	8.00	Joss Whedon
3	Titanic	Romance	1997-12-19	7.80	James Cameron
4	The Godfather	Crime	1972-03-24	9.20	Francis Ford Coppola
5	The Dark Knight	Action	2008-07-18	9.00	Christopher Nolan
NULL	NULL	NULL	NULL	NULL	NULL

3. Display all the details of movies available.

SELECT * FROM MOVIES;

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

Movie_ID	Title	Genre	Release_Date	Rating	Director
1	Inception	Sci-Fi	2010-07-16	8.80	Christopher Nolan
2	The Avengers	Action	2012-05-04	8.00	Joss Whedon
3	Titanic	Romance	1997-12-19	7.80	James Cameron
4	The Godfather	Crime	1972-03-24	9.20	Francis Ford Coppola
5	The Dark Knight	Action	2008-07-18	9.00	Christopher Nolan
NULL	NULL	NULL	NULL	NULL	NULL

4. Display the list of movies in the "Action" genre.

SELECT * FROM MOVIES WHERE Genre = 'Action';

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

Movie_ID	Title	Genre	Release_Date	Rating	Director
2	The Avengers	Action	2012-05-04	8.00	Joss Whedon
5	The Dark Knight	Action	2008-07-18	9.00	Christopher Nolan
NULL	NULL	NULL	NULL	NULL	NULL

5. Create a table CUSTOMERS with the given schema.

CREATE TABLE CUSTOMERS (

Customer_ID INT PRIMARY KEY,

Name VARCHAR(255),

Email VARCHAR(255),

Phone VARCHAR(15),

Membership_Type VARCHAR(50)

);

6. Insert at least 5 rows into the CUSTOMERS table.

INSERT INTO CUSTOMERS (Customer_ID, Name, Email, Phone, Membership_Type)

VALUES

(1, 'Alice', 'alice@example.com', '1234567890', 'Premium'),

(2, 'Bob', 'bob@example.com', '9876543210', 'Standard'),

(3, 'Charlie', 'charlie@example.com', '5555555555', 'Premium'),

(4, 'Diana', 'diana@example.com', '4444444444', NULL),

(5, 'Edward', 'edward@example.com', '3333333333', 'Standard');

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

Customer_ID	Name	Email	Phone	Membership_Type
1	Alice	alice@example.com	1234567890	Premium
2	Bob	bob@example.com	9876543210	Standard
3	Charlie	charlie@example.com	5555555555	Premium
5	Edward	edward@example.com	3333333333	Standard
NULL	NULL	NULL	NULL	NULL

7. Display the names and emails of all customers.

SELECT Name, Email FROM CUSTOMERS;

Result Grid | Filter Rows:

Name	Email
Alice	alice@example.com
Bob	bob@example.com
Charlie	charlie@example.com
Diana	diana@example.com
Edward	edward@example.com

8. Display the list of customers with a "Premium" membership.

SELECT * FROM CUSTOMERS WHERE Membership_Type = 'Premium';

Result Grid | Filter Rows: | Edit: | Export/Impor

Customer_ID	Name	Email	Phone	Membership_Type
1	Alice	alice@example.com	1234567890	Premium
3	Charlie	charlie@example.com	5555555555	Premium
NULL	NULL	NULL	NULL	NULL

9. Create a table RENTALS with the given schema.

CREATE TABLE RENTALS (

Rental_ID INT PRIMARY KEY,

Customer_ID INT,

Movie_ID INT,

Rental_Date DATE,

Return_Date DATE

);

10. Insert 5 records into the RENTALS table.

INSERT INTO RENTALS (Rental_ID, Customer_ID, Movie_ID, Rental_Date, Return_Date)

VALUES

(1, 1, 2, '2024-12-01', '2024-12-05'),

(2, 2, 3, '2024-12-02', '2024-12-10'),

(3, 3, 1, '2024-12-03', NULL),

(4, 4, 5, '2024-12-04', '2024-12-15'),

(5, 5, 4, '2024-12-05', NULL);

Result Grid | Filter Rows: | Edit:

Rental_ID	Customer_ID	Movie_ID	Rental_Date	Return_Date
1	1	2	2024-12-01	2024-12-05
2	2	3	2024-12-02	2024-12-10
3	3	1	2024-12-03	NULL
4	4	5	2024-12-04	2024-12-15
5	5	4	2024-12-05	NULL
NULL	NULL	NULL	NULL	NULL

Medium Questions :

1. Add a NOT NULL constraint to the Genre column of the MOVIES table.

ALTER TABLE MOVIES MODIFY Genre VARCHAR(100) NOT NULL;

2. Add a UNIQUE constraint to the Email column in the CUSTOMERS table.

ALTER TABLE CUSTOMERS ADD CONSTRAINT UNIQUE (Email);

3. Add a foreign key constraint on Movie_ID in the RENTALS table referencing MOVIES (Movie_ID).

ALTER TABLE RENTALS

ADD CONSTRAINT fk_movie_id

FOREIGN KEY (Movie_ID) REFERENCES MOVIES(Movie_ID);

4. Create an index on the Rating column in the MOVIES table to optimize

queries

CREATE INDEX idx_rating ON MOVIES (Rating);

5. Find the average rating of all movies in the MOVIES table.

SELECT AVG(Rating) AS Average_Rating FROM MOVIES;

Result Grid

Average_Rating
8.560000

6. Find the total number of movies rented and the maximum fees paid in a single rental.

SELECT COUNT(*) AS Total_Rentals, MAX(Fees) AS Max_Fees FROM

RENTALS;

7. Update all movies with a rating below 5 to change their genre to 'Uncategorized'.

UPDATE MOVIES SET Genre = 'Uncategorized' WHERE Rating < 5;

Result Grid Filter Rows: Edit: Export/Import:

Movie_ID	Title	Genre	Release_Date	Rating	Director
1	Inception	Sci-Fi	2010-07-16	8.80	Christopher Nolan
2	The Avengers	Uncategorized	2012-05-04	4.00	Joss Whedon
3	Titanic	Romance	1997-12-19	7.80	James Cameron
4	The Godfather	Crime	1972-03-24	9.20	Francis Ford Coppola
5	The Dark Knight	Action	2008-07-18	9.00	Christopher Nolan
NULL	NULL	NULL	NULL	NULL	NULL

8. Delete all customer records where the Membership_Type is NULL.

DELETE FROM CUSTOMERS WHERE Membership_Type IS NULL;

Result Grid | Filter Rows: | Edit: | Export/Im

Customer_ID	Name	Email	Phone	Membership_Type
1	Alice	alice@example.com	1234567890	Premium
2	Bob	bob@example.com	9876543210	Standard
3	Charlie	charlie@example.com	5555555555	Premium
5	Edward	edward@example.com	3333333333	Standard
NULL	NULL	NULL	NULL	NULL

Hard Questions :

1. Retrieve customers who rented movies but have not returned them (use WHERE and NULL check).

SELECT c.Name, c.Email, r.Movie_ID

FROM CUSTOMERS c

JOIN RENTALS r ON c.Customer_ID = r.Customer_ID

WHERE r.Return_Date IS NULL;

Result Grid | Filter Rows:

Name	Email	Movie_ID
Charlie	charlie@example.com	1
Edward	edward@example.com	4

2. Find the most rented movie(s) and the number of times they were rented

SELECT m.Title, COUNT(r.Movie_ID) AS Times_Rented

FROM MOVIES m

JOIN RENTALS r ON m.Movie_ID = r.Movie_ID

GROUP BY m.Movie_ID, m.Title

HAVING COUNT(r.Movie_ID) = (

SELECT MAX(RentalCount) FROM (

SELECT COUNT(Movie_ID) AS RentalCount FROM RENTALS GROUP

BY Movie_ID

) AS SubQuery

);

Result Grid | Filter Rows:

Title	Times_Rented
Inception	1
The Avengers	1
Titanic	1
The Godfather	1
The Dark Knight	1

Scenario 3: Hospital

use hospital;

Schema :

1) DOCTORS :

```
create table Doctors(
D_ID INT primary key,
Name VARCHAR(50),
Specialty VARCHAR(50),
Phone VARCHAR(15),
Salary DECIMAL(10, 2)
);
```

Result Grid | Filter Rows: | Edit:

D_ID	Name	Specialty	Phone	Salary
1	Dr. Smith	Cardiology	1234567890	150000.00
2	Dr. Johnson	Pediatrics	0987654321	120000.00
3	Dr. Williams	Orthopedics	1122334455	130000.00
4	Dr. Brown	Neurology	2233445566	140000.00
5	Dr. Jones	Dermatology	3344556677	110000.00
6	Dr. Garcia	Cardiology	4455667788	160000.00
NULL	NULL	NULL	NULL	NULL

PATIENTS

```
create table Patient(
Patient_ID INT primary key,
Name VARCHAR(50),
Address VARCHAR(100),
Disease VARCHAR(50),
Phone VARCHAR(15)
);
```

Result Grid | Filter Rows: | Edit: | Expo

Patient_ID	Name	Address	Disease	Phone
1	John Doe	123 Main St	Flu	5551234567
2	Jane Smith	456 Elm St	Asthma	5552345678
3	Robert Brown	789 Oak St	Diabetes	5553456789
4	Emily Davis	101 Pine St	Hypertension	5554567890
5	Michael Wilson	202 Maple St	Allergy	5555678901
6	Sarah Johnson	303 Birch St	Migraine	5556789012
NULL	NULL	NULL	NULL	NULL

2) APPOINTMENTS :

```
create table Appointment(
Appointment_ID INT primary key,
Doctor_ID INT,
Patient_ID INT,
Appointment_Date DATE,
Fees DECIMAL(8, 2)
);
```

Result Grid | Filter Rows: | Edit:

Appointment_ID	Doctor_ID	Patient_ID	Appointment_Date	Fees
1	1	1	2023-12-01	500.00
2	2	2	2023-12-02	300.00
3	3	3	2023-12-03	400.00
4	4	4	2023-12-04	600.00
5	5	5	2023-12-05	350.00
6	6	6	2023-12-06	450.00
NULL	NULL	NULL	NULL	NULL

Appointment 3

Easy Level Questions :

1. Display all details of doctors specializing in "Cardiology".

SELECT * FROM DOCTORS WHERE Specialty = 'Cardiology';

Result Grid | Filter Rows: | Edit:

D_ID	Name	Specialty	Phone	Salary
1	Dr. Smith	Cardiology	1234567890	150000.00
6	Dr. Garcia	Cardiology	4455667788	160000.00
NULL	NULL	NULL	NULL	NULL

2. Retrieve the names and phone numbers of all patients.

SELECT Name, Phone FROM Patient;

Result Grid | Filter Rows:

Name	Phone
John Doe	5551234567
Jane Smith	5552345678
Robert Brown	5553456789
Emily Davis	5554567890
Michael Wilson	5555678901
Sarah Johnson	5556789012

3. Display the appointment details for a specific patient named "John Doe".

WHERE Patient_ID = (SELECT Patient_ID FROM Patient WHERE Name = 'John Doe');

Result Grid | Filter Rows: | Edit: | Exp

Appointment_ID	Doctor_ID	Patient_ID	Appointment_Date	Fees
1	1	1	2023-12-01	500.00
NULL	NULL	NULL	NULL	NULL

4. List all appointments scheduled on or after December 1, 2023.

SELECT * FROM Appointment WHERE Appointment_Date >= '2023-12-01';

Result Grid | Filter Rows: | Edit: | Export

Appointment_ID	Doctor_ID	Patient_ID	Appointment_Date	Fees
1	1	1	2023-12-01	500.00
2	2	2	2023-12-02	300.00
3	3	3	2023-12-03	400.00
4	4	4	2023-12-04	600.00
5	5	5	2023-12-05	350.00
6	6	6	2023-12-06	450.00
NULL	NULL	NULL	NULL	NULL

Appointment 10

5. Find all doctors with a salary greater than 1,00,000.

SELECT * FROM Doctors WHERE Salary > 100000;

Result Grid | Filter Rows: | Edit:

D_ID	Name	Specialty	Phone	Salary
1	Dr. Smith	Cardiology	1234567890	150000.00
2	Dr. Johnson	Pediatrics	0987654321	120000.00
3	Dr. Williams	Orthopedics	1122334455	130000.00
4	Dr. Brown	Neurology	2233445566	140000.00
5	Dr. Jones	Dermatology	3344556677	110000.00
6	Dr. Garcia	Cardiology	4455667788	160000.00
NULL	NULL	NULL	NULL	NULL

Doctors 11

Medium Questions :

1. Create an index on the Speciality column in the DOCTORS table.

CREATE INDEX idx_specialty ON Doctors(Specialty);

2. Update the salary of all doctors in the "Pediatrics" specialty by 10%.

UPDATE Doctors SET Salary = Salary * 1.10 WHERE Specialty = 'Pediatrics';

3. Delete the records of patients who have never been assigned an appointment.

DELETE FROM Patient WHERE Patient_ID NOT IN (SELECT DISTINCT Patient_ID FROM Appointment);

4. Find the total consultation fees collected by each doctor.

SELECT Doctor_ID, SUM(Fees) AS Total_Fees FROM Appointment GROUP BY Doctor_ID;

Result Grid | Filter Rov

Doctor_ID	Total_Fees
1	500.00
2	300.00
3	400.00
4	600.00
5	350.00
6	450.00

5. Display the first three characters of each doctor's name (use a single row function).

SELECT LEFT(Name, 3) AS Name_Prefix FROM Doctors;

Result Grid

Name_Prefix
Dr.
Dr.
Dr.
Dr.
Dr.
Dr.

6. Create a savepoint after updating the salary of a specific doctor.

BEGIN;

UPDATE Doctors SET Salary = Salary * 1.10 WHERE Doctor_ID = 1; -- Example Doctor_ID

SAVEPOINT salary_update;

7. Find the average consultation fees for appointments made in January 2024.

SELECT AVG(Fees) AS Average_Fees FROM Appointment WHERE Appointment_Date BETWEEN '2024-01-01' AND '2024-01-31';

8. Add a NOT NULL constraint to the Phone column in the DOCTORS table.

ALTER TABLE DOCTORS

MODIFY Phone VARCHAR(15) NOT NULL;

Hard Questions :

1. Display the details of patients treated by doctors specializing in "Cardiology" (use JOIN).

```
SELECT D.Name, COUNT(A.Patient_ID) AS Number_of_Patients
FROM Doctors D
JOIN Appointment A ON D.D_ID = A.Doctor_ID
GROUP BY D.Name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Co

Patient_ID	Name	Address	Disease	Phone
1	John Doe	123 Main St	Flu	5551234567
6	Sarah Johnson	303 Birch St	Migraine	5556789012

2. Find the doctor names along with the number of patients they treated.

```
SELECT D.Name, COUNT(A.Patient_ID) AS Number_of_Patients
FROM Doctors D
JOIN Appointment A ON D.Doctor_ID = A.Doctor_ID
GROUP BY D.Name;
```

Result Grid | Filter Rows:

Name	Number_of_Patients
Dr. Smith	1
Dr. Johnson	1
Dr. Williams	1
Dr. Brown	1
Dr. Jones	1
Dr. Garcia	1

3. Find doctors who treated patients with more than one disease (use

```
SELECT D.Name
FROM Doctors D
JOIN Appointment A ON D.D_ID = A.Doctor_ID
JOIN Patient P ON A.Patient_ID = P.Patient_ID
GROUP BY D.Name
HAVING COUNT(DISTINCT P.Disease) > 1;
```