

DELHI TECHNOLOGICAL UNIVERSITY



NAME:	<u>Ishaan Gupta</u>
ROLL NO:	<u>23/MC/68</u>
BRANCH:	<u>MCE</u>
SUBJECT:	<u>DATABASE MANAGEMENT SYSTEM</u>
SUBJECT CODE: <u>MC209</u>	
SUBMITTED TO: <u>Prof. Brij Kishore Tyagi & Miss Shweta Kalson</u>	

No. Of Experiments	Name of Experiment	Date	Signature
1	Synopsis of the Project, along with its E-R Diagram	12/08/2024	
2	Implement the following DDL statements: a. Create b. Create table with constraints (NOT NULL, UNIQUE, DEFAULT, CHECK, PRIMARY KEY, FOREIGN KEY) c. Alter Table: i. Add column ii. Drop column iii. Add/drop constraint iv. Rename column d. Drop Table	2/09/24	
3	Implement the following DML statements: a. Insert b. Update c. Delete d. Truncate	9/09/24	
4	Implement the following SELECT statements: a. Simple SELECT statement b. Where clause+ IN/NOT IN c. Aggregate functions d. Group by + Having e. Order by f. Views g. Inbuilt Functions (e.g., Date)	16/09/24	
5	Implement and perform Nested Queries along with Joins (Inner join, Outer join, Left join, Right join)	30/09/24	
6	Introduction to PL/SQL. Create a PL/SQL block and implement the following: a. Variables	7/10/24	

	b. Packages c. Procedures d. Functions		
7	Perform Exception handling in a PL/SQL block.	7/10/24	
8	Project Report with Code and Screenshots	14/10/24	
9	Implement Triggers in SQL.	14/10/24	
10	Implement the following transaction statements: a. Commit b. Rollback c. Savepoint	21/10/24	

Practical -1

Synopsis of the Project, along with its E-R Diagram

Synopsis of Hospital Management System

Project Title: Hospital Management System (HMS)

Objective: The primary objective of this Hospital Management System (HMS) is to create a comprehensive database solution that manages the daily operations and administrative tasks of a hospital. This system aims to improve the efficiency and accuracy of various processes, including patient registration, appointment scheduling, medical records management, billing, and staff management.

Introduction: In the modern healthcare environment, hospitals face challenges such as managing patient data, scheduling appointments, and ensuring seamless communication among healthcare providers. The Hospital Management System seeks to address these challenges by providing a centralized platform that streamlines operations, enhances patient care, and optimizes resource management.

Key Features:

1. Patient Registration:

- Collect and store patient information, including personal details, medical history, and contact information.
- Assign a unique patient ID for easy identification.

2. Appointment Scheduling:

- Allow patients to book appointments with doctors online.
- Manage appointment availability and send reminders to patients.

3. Medical Records Management:

- Maintain electronic medical records (EMRs) for each patient.
- Facilitate easy access to patient history, diagnoses, treatments, and prescriptions.

4. Billing and Invoicing:

- Automate the billing process for medical services rendered.
- Generate invoices and manage payment records.

5. Staff Management:

- Manage doctor, nurse, and administrative staff information.

- Track staff schedules, availability, and performance.

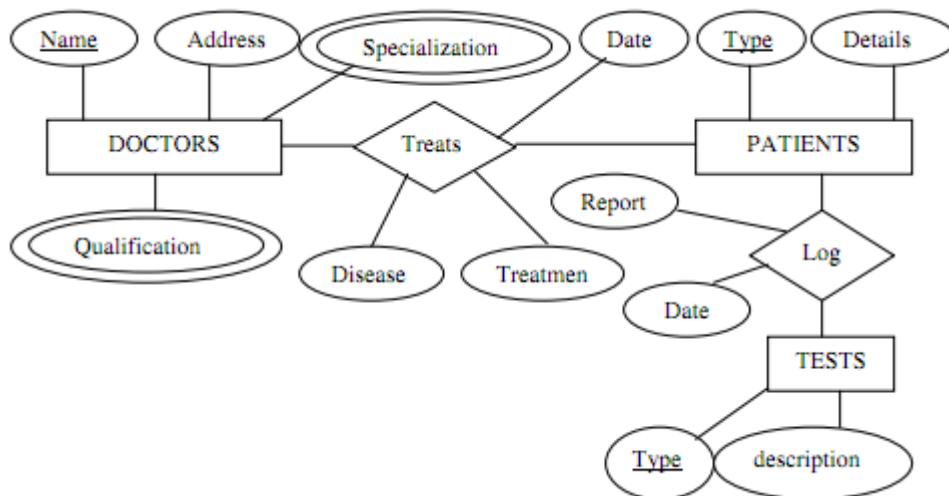
6. Reporting:

- Generate reports on patient statistics, financials, and operational efficiency.
- Provide insights to aid decision-making and improve services.

Technologies Used:

- **Database Management System:** MySQL

Conclusion: The Hospital Management System will significantly enhance the efficiency of hospital operations, improve patient experience, and ensure accurate management of medical records. By implementing a robust database solution, the system will facilitate better healthcare delivery, ultimately leading to improved patient outcomes.



Practical -2

Implement the following DDL statements:

a. Create

b. Create table with constraints (NOT NULL, UNIQUE, DEFAULT, CHECK, PRIMARY KEY, FOREIGN KEY)

c. Alter Table:

i. Add column

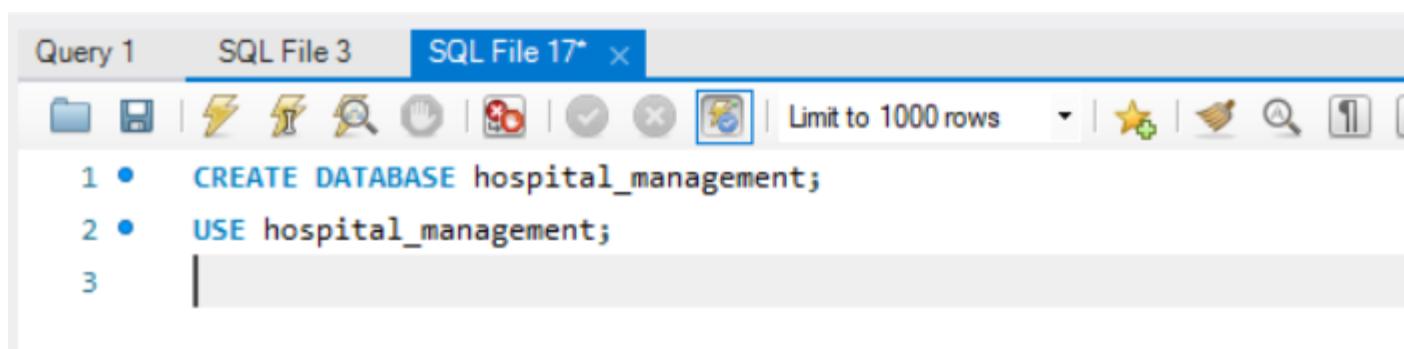
ii. Drop column

iii. Add/drop constraint

iv. Rename column

d. Drop Table

a. Create



The screenshot shows the MySQL Workbench interface with three tabs at the top: 'Query 1', 'SQL File 3', and 'SQL File 17*'. The 'SQL File 17*' tab is active. Below the tabs is a toolbar with various icons. The main area contains the following SQL code:

```
1 • CREATE DATABASE hospital_management;
2 • USE hospital_management;
```

b. Create table with constraints (NOT NULL, UNIQUE, DEFAULT, CHECK, PRIMARY KEY, FOREIGN KEY)

Query 1 SQL File 3 SQL File 17* x

The screenshot shows the MySQL Workbench interface with three tabs: Query 1, SQL File 3, and SQL File 17*. The SQL File 17* tab is active, displaying the following SQL code:

```
1 • CREATE TABLE Patients (
2     patient_id INT PRIMARY KEY,
3     patient_name VARCHAR(100) NOT NULL,
4     age INT CHECK (age > 0),
5     gender VARCHAR(10),
6     phone VARCHAR(15) UNIQUE,
7     disease VARCHAR(100),
8     date_of_admission DATE DEFAULT (CURRENT_DATE())
9 );
10 |
```

c. Alter Table:

i. Add column

Query 1 SQL File 3 SQL File 17* x

The screenshot shows the MySQL Workbench interface with three tabs: Query 1, SQL File 3, and SQL File 17*. The SQL File 17* tab is active, displaying the following SQL code:

```
1 • ALTER TABLE Patients ADD doctor_assigned VARCHAR(100);
2 |
```

ii. Drop column

patients patients patients patients patients doctors doctors doctors x

The screenshot shows the MySQL Workbench interface with three tabs: Query 1, SQL File 3, and SQL File 17*. The SQL File 17* tab is active, displaying the following SQL code:

```
1 • ALTER TABLE Patients DROP COLUMN disease;
2 |
```

iii. Add/drop constraint

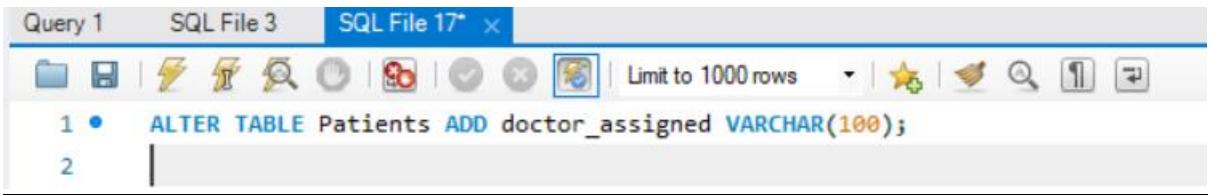
Add a NOT NULL constraint on doctor_assigned

Query 1 SQL File 3 SQL File 17* x

The screenshot shows the MySQL Workbench interface with three tabs: Query 1, SQL File 3, and SQL File 17*. The SQL File 17* tab is active, displaying the following SQL code:

```
1 • ALTER TABLE Patients MODIFY doctor_assigned VARCHAR(100) NOT NULL;
2 |
```

Drop the NOT NULL constraint

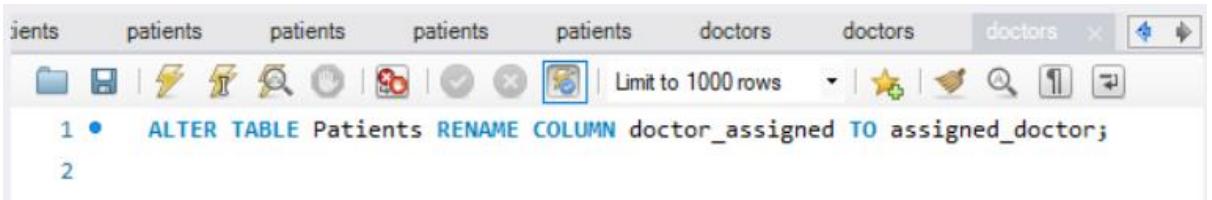


A screenshot of the MySQL Workbench interface. The top menu bar shows "Query 1", "SQL File 3", and "SQL File 17*". The main query editor window contains the following SQL code:

```
1 • ALTER TABLE Patients ADD doctor_assigned VARCHAR(100);
2
```

The interface includes standard database navigation buttons (back, forward, search) and a toolbar with icons for file operations and database management.

iv. Rename column

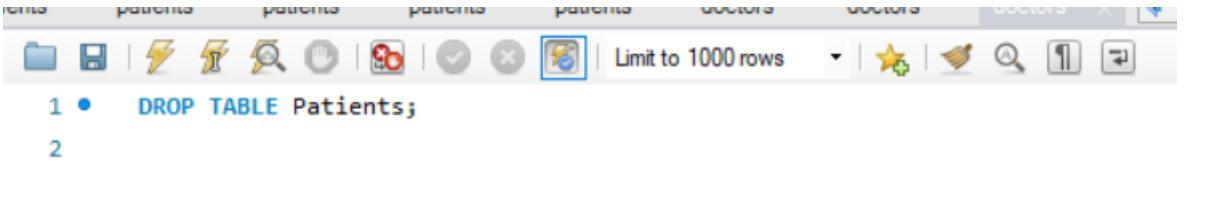


A screenshot of the MySQL Workbench interface. The top menu bar shows "patients", "patients", "patients", "patients", "patients", "doctors", "doctors", "doctors". The main query editor window contains the following SQL code:

```
1 • ALTER TABLE Patients RENAME COLUMN doctor_assigned TO assigned_doctor;
2
```

The interface includes standard database navigation buttons (back, forward, search) and a toolbar with icons for file operations and database management.

d. Drop Table



A screenshot of the MySQL Workbench interface. The top menu bar shows "patients", "patients", "patients", "patients", "patients", "doctors", "doctors", "doctors". The main query editor window contains the following SQL code:

```
1 • DROP TABLE Patients;
2
```

The interface includes standard database navigation buttons (back, forward, search) and a toolbar with icons for file operations and database management.

Practical -3

Implement the following DML statements:

- a. Insert
- b. Update
- c. Delete
- d. Truncate

a. Insert

The screenshot shows the MySQL Workbench interface with two queries in the query editor.

Query 1 (Top):

```
1 • INSERT INTO Patients (patient_id, patient_name, age, gender, phone, disease,
2   VALUES (1, 'John Doe', 30, 'Male', '1234567890', 'Fever', '2024-10-05');
```

Query 2 (Bottom):

```
1 • SELECT * FROM patients;
```

Result Grid (Bottom):

	patient_id	patient_name	age	gender	phone	disease	date_of_admission	doctor_a
▶	1	John Doe	30	Male	1234567890	Fever	2024-10-05	HULL
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

b. Update

The screenshot shows the MySQL Workbench interface with a single update query in the query editor.

Query 1 (Top):

```
1 • UPDATE Patients
2   SET disease = 'COVID-19'
3   WHERE patient_id = 1;
```

4
5

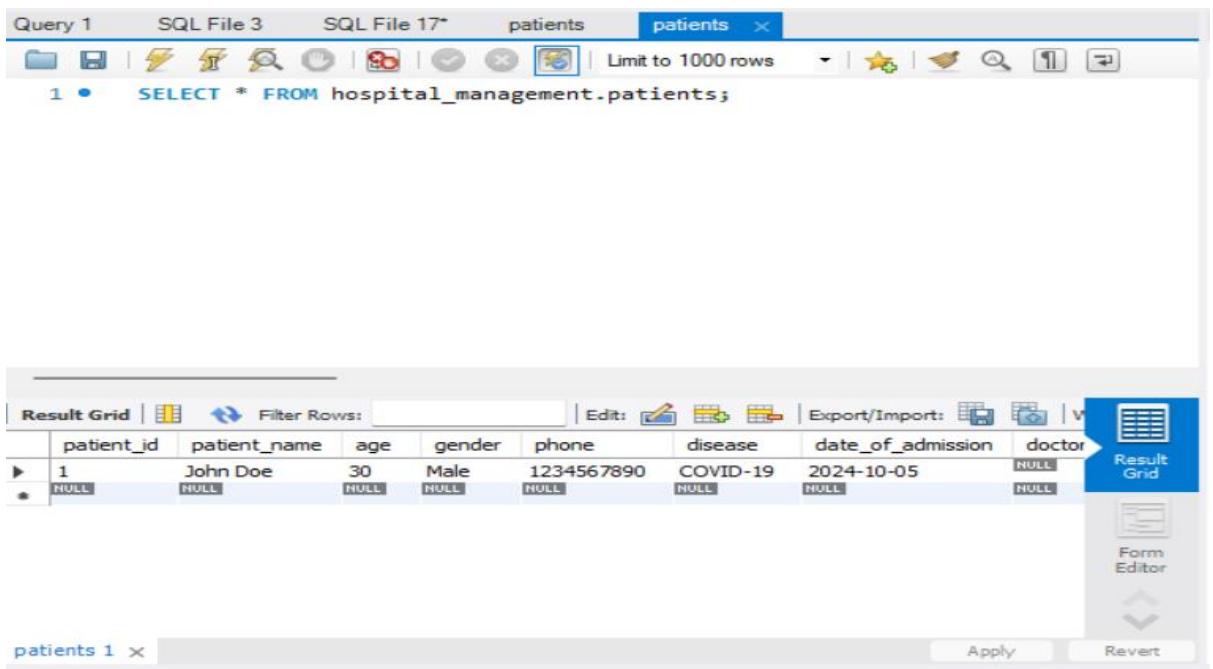
Query 1 SQL File 3 SQL File 17* patients patients ×

1 • `SELECT * FROM hospital_management.patients;`

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

patient_id	patient_name	age	gender	phone	disease	date_of_admission	doctor_assigned
1	John Doe	30	Male	1234567890	COVID-19	2024-10-05	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

patients 1 × Apply Revert

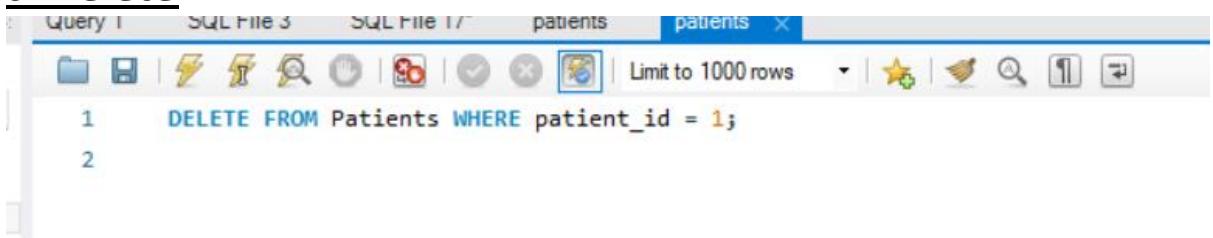


c. Delete

Query 1 SQL File 3 SQL File 17* patients patients ×

1 `DELETE FROM Patients WHERE patient_id = 1;`

2

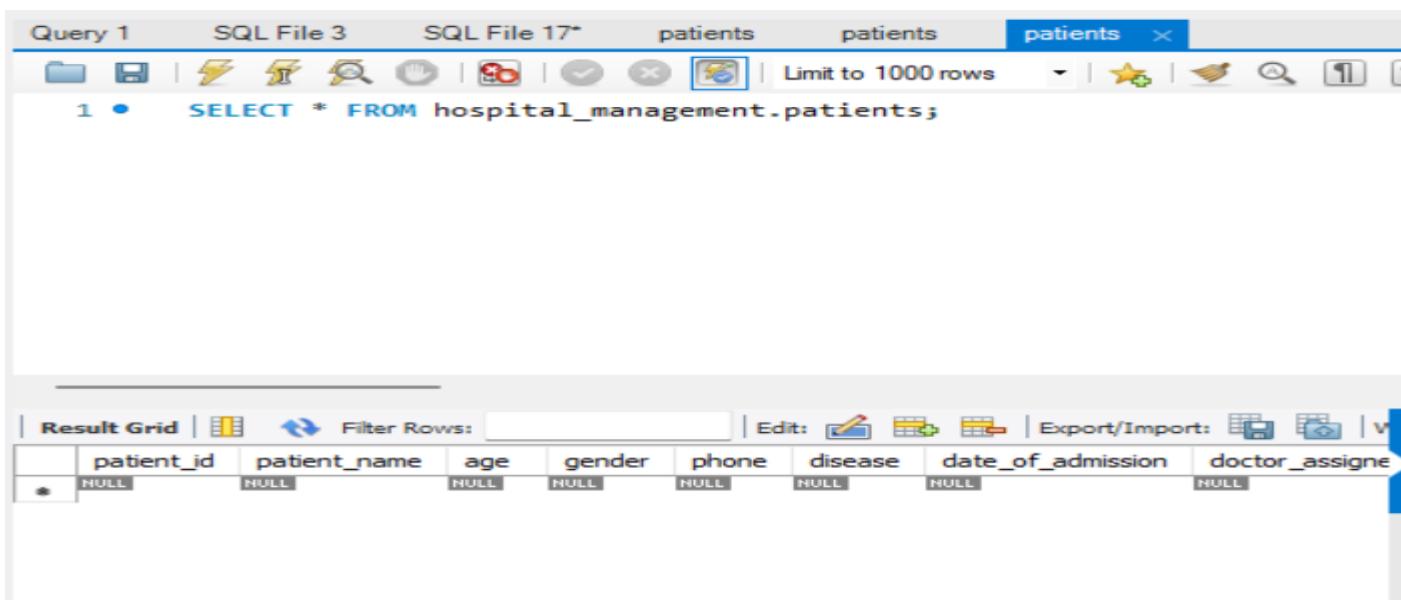


Query 1 SQL File 3 SQL File 17* patients patients patients ×

1 • `SELECT * FROM hospital_management.patients;`

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

patient_id	patient_name	age	gender	phone	disease	date_of_admission	doctor_assigned
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



d. Truncate

Query 1 SQL File 3 SQL File 17* patients patients patients x

Limit to 1000 rows

1 TRUNCATE TABLE Patients;

2

Query 1 SQL File 3 SQL File 17* patients patients patients x

Limit to 1000 rows

1 • SELECT * FROM hospital_management.patients;

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

patient_id	patient_name	age	gender	phone	disease	date_of_admission	doctor_assigned
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

patients 1 x

Apply Revert

Action Output

#	Time	Action	Message	Duration / Fetch
1	13:10:40	SELECT * FROM patients LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
2	13:11:47	INSERT INTO Patients (patient_id, patient_name, age, gender, phone, disease, date_of_admission, doctor_assigned)	1 row(s) affected	0.000 sec
3	13:11:58	INSERT INTO Patients (patient_id, patient_name, age, gender, phone, disease, date_of_admission, doctor_assigned)	Error Code: 1062. Duplicate entry '1' for key 'patients.PRIMARY'	0.000 sec
4	13:12:52	SELECT * FROM table_name LIMIT 0, 1000	Error Code: 1146. Table 'hospital_management.table_name' doesn't exist	0.000 sec
5	13:13:02	SELECT * FROM patients LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
6	13:13:54	UPDATE Patients SET disease = 'COVID-19' WHERE patient_id = 1	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
7	13:14:30	SELECT * FROM hospital_management.patients LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
8	13:15:04	DELETE FROM Patients WHERE patient_id = 1	1 row(s) affected	0.000 sec
9	13:15:45	SELECT * FROM hospital_management.patients LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
10	13:16:14	TRUNCATE TABLE Patients	0 row(s) affected	0.047 sec

Practical -4

Implement the following SELECT statements:

- a. Simple SELECT statement
- b. Where clause + IN/NOT IN
- c. Aggregate functions
- d. Group by + Having
- e. Order by
- f. Views
- g. Inbuilt Functions (e.g., Date)

a. Simple SELECT statement

The screenshot shows the MySQL Workbench interface. The query editor window contains the following SQL code:

```
1 •  SELECT * FROM hospital_management.patients;
```

The results grid displays the following data:

patient_id	patient_name	age	gender	phone	disease	date_of_admission	do
1	John Doe	45	Male	1234567890	Flu	2024-10-01	HULL
2	Jane Smith	32	Female	0987654321	Diabetes	2024-10-02	HULL
3	Mike Johnson	28	Male	1122334455	Asthma	2024-10-03	HULL
4	Emma Davis	54	Female	2233445566	Hypertension	2024-10-04	HULL
5	Chris Lee	38	Male	3344556677	COVID-19	2024-10-05	HULL
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL

b. Where clause + IN/NOT IN:

The screenshot shows the MySQL Workbench interface. The top navigation bar includes tabs for 'Query 1', 'SQL File 3', 'SQL File 17...', 'patients', 'patients', 'patients', and 'patients'. The main area displays a SQL query:

```
1 • SELECT patient_name, disease
2 FROM Patients
3 WHERE disease IN ('Fever', 'COVID-19');
4
5
```

The results grid below shows one row of data:

patient_name	disease
Chris Lee	COVID-19

A context help message is displayed on the right: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

c. Aggregate Functions (e.g., COUNT, AVG):

sql

The screenshot shows the SQL Server Management Studio interface. The top pane displays a T-SQL query:

```
1 • SELECT COUNT(patient_id) AS total_patients, AVG(age) AS average_age
2   FROM Patients;
3
```

The results grid shows the output of the query:

	total_patients	average_age
▶	5	39.4000

The bottom pane shows a history of recent actions:

#	Time	Action	Message	Duration / Fetch
11	13:16:38	SELECT * FROM hospital_management.patients LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
12	13:19:07	INSERT INTO Patients (patient_id, patient_name, age, gender, phone, disease, date...)	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.015 sec
13	13:19:11	SELECT * FROM hospital_management.patients LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
14	13:20:12	SELECT * FROM Patients LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
15	13:20:45	SELECT patient_name, disease FROM Patients WHERE disease IN ('Fever', 'COVID...')	1 row(s) returned	0.000 sec / 0.000 sec
16	13:21:30	SELECT COUNT(patient_id) AS total_patients, AVG(age) AS average_age FROM Pa...	1 row(s) returned	0.000 sec / 0.000 sec

d. Group By + Having

The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query is:

```
1 •  SELECT disease, COUNT(patient_id) AS total_cases
2   FROM Patients
3   GROUP BY disease
4
5
```

The results grid displays the following data:

disease	total_cases
Flu	1
Diabetes	1
Asthma	1
Hypertension	1
COVID-19	1

e. Order By

The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query is:

```
1 •  SELECT patient_name, age FROM Patients ORDER BY age DESC;
2
```

The results grid displays the following data:

patient_name	age
Emma Davis	54
John Doe	45
Chris Lee	38
Jane Smith	32
Mike Johnson	28

f. Views

The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query is:

```
1 •  CREATE VIEW PatientInfo AS
2   SELECT patient_name, age, disease FROM Patients;
3
```

The screenshot shows the MySQL Workbench interface. The top menu bar includes tabs for 'Query 1', 'SQL File 3', 'SQL File 17*', 'patients' (repeated three times), and 'patients'. Below the menu is a toolbar with various icons for database management. The main area contains a query editor window with the following content:

```
1 •  SELECT * FROM PatientInfo;
2
```

Below the query editor is a result grid titled 'Result Grid' with columns for 'patient_name', 'age', and 'disease'. The results are as follows:

	patient_name	age	disease
▶	John Doe	45	Flu
	Jane Smith	32	Diabetes
	Mike Johnson	28	Asthma
	Emma Davis	54	Hypertension
	Chris Lee	38	COVID-19

The bottom status bar indicates the current connection is 'PatientInfo 8' and shows a 'Read Only' status.

g. Inbuilt Functions (e.g., Date)

```
Query 1 SQL File 3 SQL File 17* patients patients patients patients
  • SELECT patient_name, date_of_admission
  FROM Patients
 WHERE date_of_admission = CURRENT_DATE;
  4

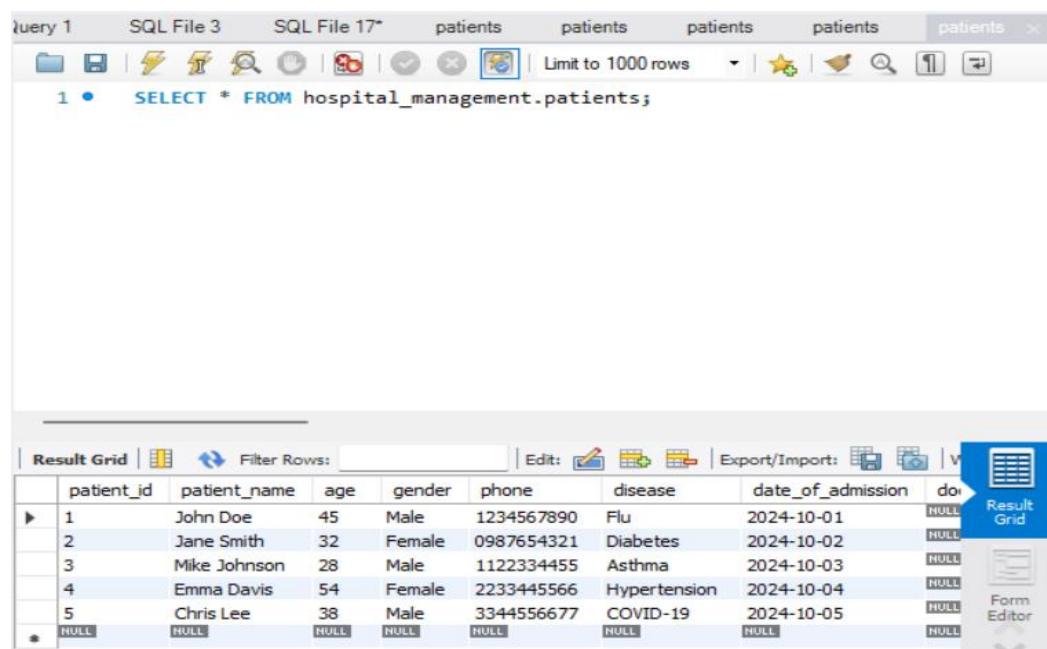
Result Grid | Filter Rows: _____ | Export: | Wrap Cell Content: | □ | Result Grid | Form Editor
```

	patient_name	date_of_admission
▶	Chris Lee	2024-10-05

Practical -5

**Implement and perform Nested Queries along with Joins
(Inner join, Outer join, Left join, Right join)**

We have 2 tables named Patients and doctors



The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor: Query 1, SQL File 3, SQL File 17*, patients, patients, patients, patients, patients.
- Query Text: 1 • `SELECT * FROM hospital_management.patients;`
- Result Grid: Shows a table with 6 columns: patient_id, patient_name, age, gender, phone, disease, date_of_admission, doctor_id. The data includes 5 rows of patient information.

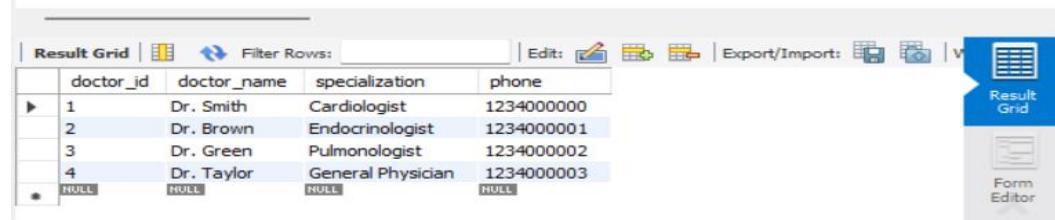
	patient_id	patient_name	age	gender	phone	disease	date_of_admission	doctor_id
▶	1	John Doe	45	Male	1234567890	Flu	2024-10-01	NULL
2	Jane Smith	32	Female	0987654321	Diabetes	2024-10-02	NULL	
3	Mike Johnson	28	Male	1122334455	Asthma	2024-10-03	NULL	
4	Emma Davis	54	Female	2233445566	Hypertension	2024-10-04	NULL	
5	Chris Lee	38	Male	3344556677	COVID-19	2024-10-05	NULL	
*	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL



The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor: patients, patients, patients, patients, patients, doctors, doctors, doctors, doctors.
- Query Text: 1 • `SELECT * FROM hospital_management.doctors;`
- Result Grid: Shows a table with 4 columns: doctor_id, doctor_name, specialization, phone. The data includes 4 rows of doctor information.

	doctor_id	doctor_name	specialization	phone
▶	1	Dr. Smith	Cardiologist	1234000000
2	Dr. Brown	Endocrinologist	1234000001	
3	Dr. Green	Pulmonologist	1234000002	
4	Dr. Taylor	General Physician	1234000003	
*	NUL	NUL	NUL	NUL



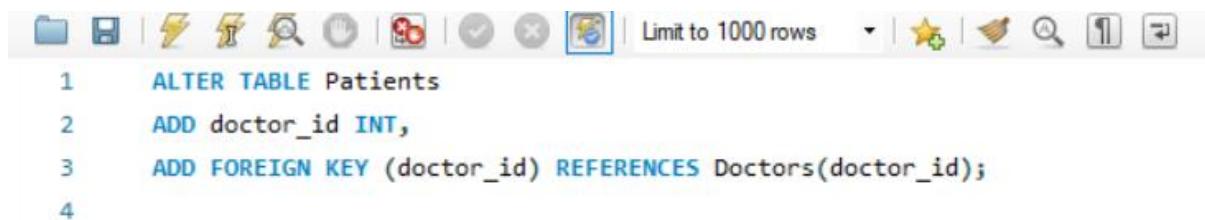
The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor: patients, patients, patients, patients, patients, doctors, doctors, doctors, doctors.
- Query Text: 1 • `SELECT * FROM hospital_management.doctors;`
- Result Grid: Shows a table with 4 columns: doctor_id, doctor_name, specialization, phone. The data includes 4 rows of doctor information.

	doctor_id	doctor_name	specialization	phone
▶	1	Dr. Smith	Cardiologist	1234000000
2	Dr. Brown	Endocrinologist	1234000001	
3	Dr. Green	Pulmonologist	1234000002	
4	Dr. Taylor	General Physician	1234000003	
*	NUL	NUL	NUL	NUL

Add a doctor_id to the Patients Table (for relational queries)

We need to update the Patients table to include a reference to the Doctors table, linking patients to doctors using a foreign key.



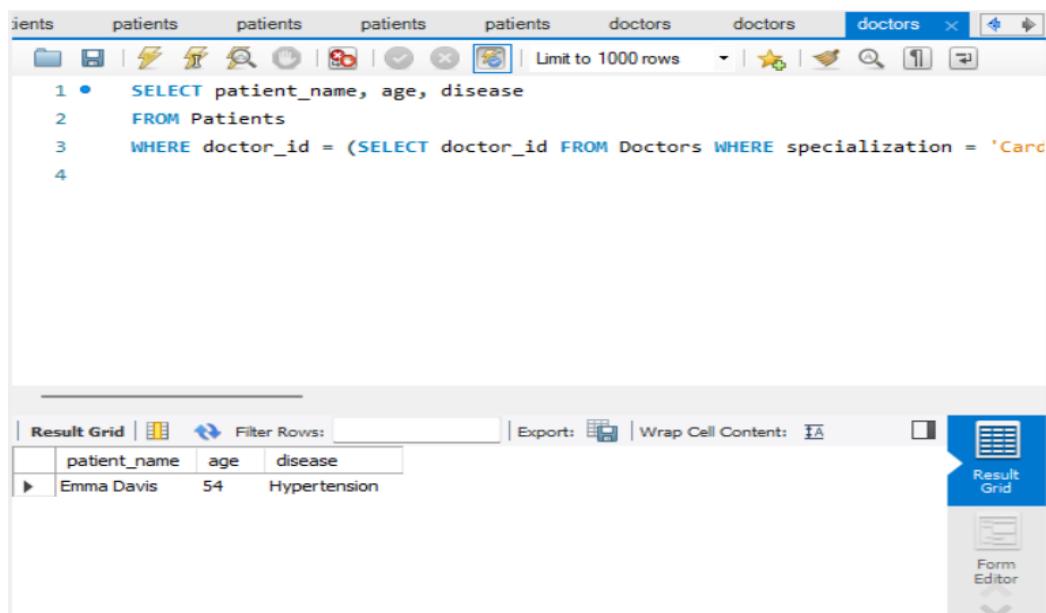
```
1 ALTER TABLE Patients
2 ADD doctor_id INT,
3 ADD FOREIGN KEY (doctor_id) REFERENCES Doctors(doctor_id);
4
```

Update Patient Records to Assign Doctors



```
1 • UPDATE Patients
2 SET doctor_id = 1 WHERE disease = 'Hypertension';
3 • UPDATE Patients
4 SET doctor_id = 2 WHERE disease = 'Diabetes';
5 • UPDATE Patients
6 SET doctor_id = 3 WHERE disease = 'Asthma';
7 • UPDATE Patients
8 SET doctor_id = 4 WHERE disease = 'Flu';
9 • UPDATE Patients
10 SET doctor_id = 4 WHERE disease = 'COVID-19';
11
12
```

Nested Queries



```
1 • SELECT patient_name, age, disease
2 FROM Patients
3 WHERE doctor_id = (SELECT doctor_id FROM Doctors WHERE specialization = 'Cardiology')
4
```

The screenshot shows the results of the query in a Result Grid:

patient_name	age	disease
Emma Davis	54	Hypertension

On the right side of the interface, there are icons for "Result Grid" (selected), "Form Editor", and "Table Editor".

Inner join

The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it is a SQL editor window containing the following code:

```
1 •   SELECT P.patient_name, P.age, P.disease, D.doctor_name, D.specialization
2     FROM Patients P
3   INNER JOIN Doctors D ON P.doctor_id = D.doctor_id;
4
```

Below the SQL editor is a results grid titled "Result Grid". It displays the following data:

	patient_name	age	disease	doctor_name	specialization
▶	John Doe	45	Flu	Dr. Taylor	General Physician
	Jane Smith	32	Diabetes	Dr. Brown	Endocrinologist
	Mike Johnson	28	Asthma	Dr. Green	Pulmonologist
	Emma Davis	54	Hypertension	Dr. Smith	Cardiologist
	Chris Lee	38	COVID-19	Dr. Taylor	General Physician

On the right side of the results grid, there's a vertical toolbar with two buttons: "Result Grid" (highlighted in blue) and "Form Editor".

Left join

The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it is a SQL editor window containing the following code:

```
1 •   SELECT P.patient_name, P.age, P.disease, D.doctor_name, D.specialization
2     FROM Patients P
3   LEFT JOIN Doctors D ON P.doctor_id = D.doctor_id;
4
```

Below the SQL editor is a results grid titled "Result Grid". It displays the same data as the previous screenshot:

	patient_name	age	disease	doctor_name	specialization
▶	John Doe	45	Flu	Dr. Taylor	General Physician
	Jane Smith	32	Diabetes	Dr. Brown	Endocrinologist
	Mike Johnson	28	Asthma	Dr. Green	Pulmonologist
	Emma Davis	54	Hypertension	Dr. Smith	Cardiologist
	Chris Lee	38	COVID-19	Dr. Taylor	General Physician

On the right side of the results grid, there's a vertical toolbar with two buttons: "Result Grid" (highlighted in blue) and "Form Editor".

Right join

The screenshot shows the MySQL Workbench interface with a query editor and a result grid.

Query Editor:

```
1 •   SELECT P.patient_name, P.age, P.disease, D.doctor_name, D.specialization
2     FROM Patients P
3   RIGHT JOIN Doctors D ON P.doctor_id = D.doctor_id;
4
```

Result Grid:

patient_name	age	disease	doctor_name	specialization
Emma Davis	54	Hypertension	Dr. Smith	Cardiologist
Jane Smith	32	Diabetes	Dr. Brown	Endocrinologist
Mike Johnson	28	Asthma	Dr. Green	Pulmonologist
Chris Lee	38	COVID-19	Dr. Taylor	General Physician
John Doe	45	Flu	Dr. Taylor	General Physician

Right Grid Panel:

- Result Grid
- Form Editor

Outer join

The screenshot shows the MySQL Workbench interface with a query editor and a result grid.

Query Editor:

```
1 •   SELECT P.patient_name, P.age, P.disease, D.doctor_name, D.specialization
2     FROM Patients P
3   LEFT JOIN Doctors D ON P.doctor_id = D.doctor_id
4   UNION
5   SELECT P.patient_name, P.age, P.disease, D.doctor_name, D.specialization
6     FROM Patients P
7   RIGHT JOIN Doctors D ON P.doctor_id = D.doctor_id;
8
```

Result Grid:

patient_name	age	disease	doctor_name	specialization
John Doe	45	Flu	Dr. Taylor	General Physician
Jane Smith	32	Diabetes	Dr. Brown	Endocrinologist
Mike Johnson	28	Asthma	Dr. Green	Pulmonologist
Emma Davis	54	Hypertension	Dr. Smith	Cardiologist
Chris Lee	38	COVID-19	Dr. Taylor	General Physician

Right Grid Panel:

- Result Grid
- Form Editor

Practical -6

Introduction to PL/SQL. Create a PL/SQL block and implement the following:

- a. Variables**
- b. Packages**
- c. Procedures**
- d. Functions**

a. Variables

The screenshot shows the OneCompiler IDE interface. The code editor window contains a file named 'queries.sql' with the following content:

```
1 DECLARE
2     patient_id NUMBER := 1001;
3     patient_name VARCHAR2(50) := 'John Doe';
4 BEGIN
5     DBMS_OUTPUT.PUT_LINE('Patient ID: ' || patient_id);
6     DBMS_OUTPUT.PUT_LINE('Patient Name: ' || patient_name);
7 END;
8 /
9 |
```

The code is executed, and the output pane shows the results:

STDIN
Input for the program (O|
Output:
Patient ID: 1001
Patient Name: John Doe

b. Packages

The screenshot shows the OneCompiler IDE interface. The code editor window contains a file with the following content:

```
1 CREATE OR REPLACE PACKAGE hospital_pkg IS
2     PROCEDURE add_patient(patient_name VARCHAR2, patient_age NUMBER);
3     FUNCTION get_patient_count RETURN NUMBER;
4 END hospital_pkg;
5 /
6 CREATE OR REPLACE PACKAGE BODY hospital_pkg IS
7     v_patient_count NUMBER := 0;
8
9     PROCEDURE add_patient(patient_name VARCHAR2, patient_age NUMBER) IS
10    BEGIN
11        v_patient_count := v_patient_count + 1;
12        DBMS_OUTPUT.PUT_LINE('Added patient: ' || patient_name);
13    END add_patient;
14
15    FUNCTION get_patient_count RETURN NUMBER IS
16    BEGIN
17        RETURN v_patient_count;
18    END get_patient_count;
19
20 END hospital_pkg;
21 /
22 DECLARE
23     patient_count NUMBER;
24 BEGIN
25     hospital_pkg.add_patient('Alice Green', 29);
26     hospital_pkg.add_patient('Bob Brown', 45);
27     patient_count := hospital_pkg.get_patient_count;
28     DBMS_OUTPUT.PUT_LINE('Total Patients: ' || patient_count);
29 END;
30 /
```

The code is executed, and the output pane shows the results:

STDIN
Input for the program (Opti
Output:
Added patient: Alice Green
Added patient: Bob Brown
Total Patients: 2

c. Procedures

```
1 v DECLARE
2   patient_count NUMBER;
3 v BEGIN
4   patient_count := hospital_pkg.get_patient_count;
5   DBMS_OUTPUT.PUT_LINE('Total Patients: ' || patient_count);
6 END;
7 /
8
```

Statement processed.
Total Patients: 0

d. Functions

```
1 v DECLARE
2   patient_count NUMBER;
3 v BEGIN
4   patient_count := hospital_pkg.get_patient_count;
5   DBMS_OUTPUT.PUT_LINE('Total Patients: ' || patient_count);
6 END;
7 /
8
```

Statement processed.
Total Patients: 0

Practical -7

Perform Exception handling in a PL/SQL block.

```
1 v DECLARE
2   patient_id NUMBER := 1;           -- Example patient ID
3   patient_name VARCHAR2(50);        -- Variable to hold patient name
4   age NUMBER := 0;                 -- Example variable for division by zero
5   result NUMBER;
6 v BEGIN
7   -- Attempt to retrieve patient name
8   SELECT name INTO patient_name
9   FROM patients
10  WHERE id = patient_id;
11
12  -- Display patient name
13  DBMS_OUTPUT.PUT_LINE('Patient Name: ' || patient_name);
14
15  -- Example of division by zero to trigger an exception
16  result := 100 / age;
17  DBMS_OUTPUT.PUT_LINE('Result: ' || result);
18
19 v EXCEPTION
20   -- Handle the NO_DATA_FOUND exception
21   WHEN NO_DATA_FOUND THEN
22     DBMS_OUTPUT.PUT_LINE('Error: Patient with ID ' || patient_id || ' was not found.');
23
--
```

```
14
15  -- Example of division by zero to trigger an exception
16  result := 100 / age;
17  DBMS_OUTPUT.PUT_LINE('Result: ' || result);
18
19 v EXCEPTION
20   -- Handle the NO_DATA_FOUND exception
21   WHEN NO_DATA_FOUND THEN
22     DBMS_OUTPUT.PUT_LINE('Error: Patient with ID ' || patient_id || ' was not found.');
23
24   -- Handle division by zero
25   WHEN ZERO_DIVIDE THEN
26     DBMS_OUTPUT.PUT_LINE('Error: Division by zero occurred.');
27
28   -- Handle any other exception
29   WHEN OTHERS THEN
30     DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
31 END;
32 /
33
```

Statement processed.
Patient Name: Alice Green
Error: Division by zero occurred.

Practical -9

Implement Triggers in SQL.

Step 1: Create the Trigger

```
1  DELIMITER $$  
2  
3  •  CREATE TRIGGER set_default_age  
4      BEFORE INSERT ON Patients  
5      FOR EACH ROW  
6      BEGIN  
7          IF NEW.age IS NULL THEN  
8              SET NEW.age = 30;  
9          END IF;  
10     END$$  
11  
12    DELIMITER ;  
13
```

Step 2: Insert Data

```
Query 1 × SQL File 3* SQL File 17* patients patients patients patients patients doctors doctors  
1 •  INSERT INTO Patients (patient_id, patient_name, gender, phone, disease, date_of_admission)  
2   VALUES (6, 'John Doe', 'Male', '1234567891', 'Flu', CURRENT_DATE());  
3
```



```
Query 1 × SQL File 3* SQL File 17* patients patients patients patients patients doctors doctors  
1 •  INSERT INTO Patients (patient_id, patient_name, age, gender, phone, disease, date_of_admission)  
2   VALUES (7, 'Jane Doe', NULL, 'Female', '0987654321', 'Cold', CURRENT_DATE());  
3
```

Step 3: Verify the Output

The screenshot shows the MySQL Workbench interface with a query editor window titled "Query 1". The query is:

```

1 •  SELECT * FROM Patients WHERE patient_id IN (6, 7);
2

```

The result grid displays two rows of patient data:

	patient_id	patient_name	age	gender	phone	disease	date_of_admission	doctor_assigned	doctor_id
▶	6	John Doe	30	Male	1234567891	Flu	2024-10-20	NULL	NULL
*	7	Jane Doe	30	Female	0987654371	Cold	2024-10-20	NULL	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

On the right side of the result grid, there are icons for "Result Grid" and "Form Editor".

Practical -10

Implement the following transaction statements:

- a. Commit**
- b. Rollback**
- c. Savepoint**

a. Commit

c. Savepoint

The screenshot shows the MySQL Workbench interface with a query editor window titled "Query 1". The script contains numbered comments explaining a transaction process:

```

1 -- Step 1: Start the transaction
2 • START TRANSACTION;
3
4 -- Step 2: Update patient to assign a doctor
5 • UPDATE Patients SET doctor_id = 2 WHERE patient_id = 1;
6
7 -- Step 3: Create a savepoint after the first update
8 • SAVEPOINT Update1;
9
10 -- Step 4: Update another patient
11 • UPDATE Patients SET doctor_id = 1 WHERE patient_id = 2;
12
13 -- Step 5: Check whether to commit or rollback
14 -- If something goes wrong, rollback to savepoint
15 -- ROLLBACK TO Update1;
16
17 -- Step 6: Commit the transaction if everything is fine
18 • COMMIT;
19

```

The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query editor at the top contains the following SQL code:

```
1 • SELECT * FROM Patients;
2
```

The results grid below displays the data from the Patients table:

patient_id	patient_name	age	gender	phone	disease	date_of_admission	doctor_assigned	doctor_id
1	John Doe	45	Male	1234567890	Flu	2024-10-01	NULL	2
2	Jane Smith	32	Female	0987654321	Diabetes	2024-10-02	NULL	1
3	Mike Johnson	28	Male	1122334455	Asthma	2024-10-03	NULL	3
4	Emma Davis	54	Female	2233445566	Hypertension	2024-10-04	NULL	1
5	Chris Lee	38	Male	3344556677	COVID-19	2024-10-05	NULL	4
6	John Doe	30	Male	1234567891	Flu	2024-10-20	NULL	NULL
7	Jane Doe	30	Female	0987654371	Cold	2024-10-20	NULL	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

On the right side of the interface, there are three floating panes: 'Result Grid', 'Form Editor', and 'Field Types'.

b. Rollback