



PROJECT REPORT

A report submitted to the

Department of Electrical and Information Engineering
Faculty of Engineering
University of Ruhuna

Sri Lanka

On 11th of September 2023

In completing a Report for the module
EE4350 Database Systems

By:

Gunathilake A. P. B. - EG/2020/3947

Hanan Ahamed M. R. - EG/2020/3949

Hariharasakthy N. - EG/2020/3956

Table of Contents

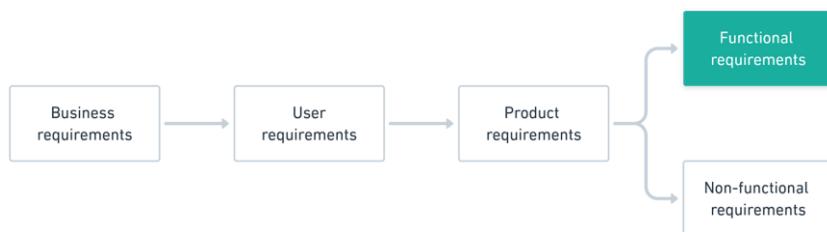
Chapter 1 : Requirement Analysis	1
1.1 Functional Requirements	1
1.2 Data Requirements	1
Chapter 2 : Conceptual Design	4
2.1 ER Diagram.....	4
Chapter 3 : Implementation.....	5
3.1 Schema Creation	5
3.2 Creation of Tables & Insertion of data.....	5
3.3 Alteration & deriving age from date of birth	21
3.4 Table Definitions.....	22
3.5 Updating & Deleting data of tables.....	38
Chapter 4 : Transactions.....	54
4.1 Simple Queries	54
4.2 Complex Queries.....	57
Chapter 5 : Tuning	64

Chapter 1 : Requirement Analysis

Here are the Functional Requirements and the Data Requirements of the Hospital DBMS.

1.1 Functional Requirements

The primary purpose of a Database Management System (DBMS) is to securely store data. For effective health management, a DBMS is very useful in current times. To keep track of the historical records of a hospital, for effective and efficient data handling, a DBMS is a very helpful tool for the hospital staff. In that case, this system will help in improving the quality of the service provided. Functional requirements are product features or the actions that developers must implement to be capable for the users to achieve their goals. These are the ‘must do requirements’ of a system.



Source: <https://www.nuclino.com/img/articles/functional-requirements.png>

In this project, we have identified 12 entities. They are “PATIENT”, “DOCTOR”, “NURSE”, “STAFF”, “DEPARTMENT”, “WARD”, “APPOINTMENT”, “TEST_REPORT”, “MEDICINE”, “PRESCRIPTION”, “BILL” “RECORD”. All the entities are interconnected from many relations. The Entity-Relationship Diagram will be illustrated in the project report.

- 01) Registration Process: Add new patients, doctors, nurses, ... records.
- 02) Modifying Entries: Modify patient, doctor, nurses, Records.
- 03) Deleting Entries: Removing records.
- 04) Modify data content.
- 05) Data Retrieval.

1.2 Data Requirements

Attributes of Entity “PATIENT”:

- Patient_id (Should not be NULL)
- Gender
- Age
- Patient_name (Should not be NULL)
- Address
- Doctor_id (Should not be NULL)
- Test_id (Should not be NULL)
- Phone_number (Should not be NULL)
- Medicine_id (Should not be NULL)
- Prescription_id (Should not be NULL)

Attributes of Entity “DOCTOR”:

- Doctor_id (Should not be NULL)
 - Specialization
 - Gender
 - Name_Birth (Should not be NULL)
 - Address
 - Phone_Number (Should not be NULL)

Attributes of Entity “NURSE”:

- Nurse_id (Should not be NULL)
 - Gender
 - Age
 - Date_of_birth (Should not be NULL)
 - Name (Should not be NULL)
 - Patient_id (Should not be NULL)
 - Phone_number (Should not be NULL)

Attributes of Entity “NON MEDICAL STAFF”:

- Staff_id (Should not be NULL)
 - Gender
 - Job_title (Should not be NULL)
 - Name_birth (Should not be NULL)
 - Address
 - Phone-number

Attributes of Entity “DEPARTMENT”:

- Department_id (Should not be NULL)
 - Head_id (Should not be NULL)
 - Department_name
 - Doctor_id (Should not be NULL)
 - Staff_id (Should not be NULL)

Attributes of Entity “WARD”:

- Ward_no (Should not be NULL)
 - Capacity
 - Availability
 - Nurse_id (Should not be NULL)
 - Staff_id (Should not be NULL)
 - Patient_id (Should not be NULL)
 - Department_id (Should not be NULL)

Attributes of Entity “APPOINTMENT”:

- Reason (Should not be NULL)
- Current Status
- Appointment_date_time (Should not be NULL)

Attributes of Entity “TEST_REPORT”:

- Test_id (Should not be NULL)
- Test_name
- Result
- Tested_date_time

Attributes of Entity “MEDICINE”:

- Quantity (Should not be NULL)
- Description
- Medicine_id (Should not be NULL)
- Medicine_name
- Price (Should not be NULL)

Attributes of Entity “PRESCRIPTION”:

- Prescribed_date
- No_of_days
- Dosage
- Medicine_id (Should not be NULL)
- Doctor_id (Should not be NULL)
- Prescription_id (Should not be NULL)

Attributes of Entity “BILL”:

- Bill_number (Should not be NULL)
- Bill_date
- Medicine_charge (Should not be NULL)
- Ward_charge
- Test_charge
- Patient_id (Should not be NULL)

Attributes of Entity “RECORD”:

- Record_id (Should not be NULL)
- Date_admitted
- Date_discharged
- Doctor_id (Should not be NULL)
- Medicine_id (Should not be NULL)
- Test_id
- Patient_id (Should not be NULL)

Chapter 2 : Conceptual Design

2.1 ER Diagram

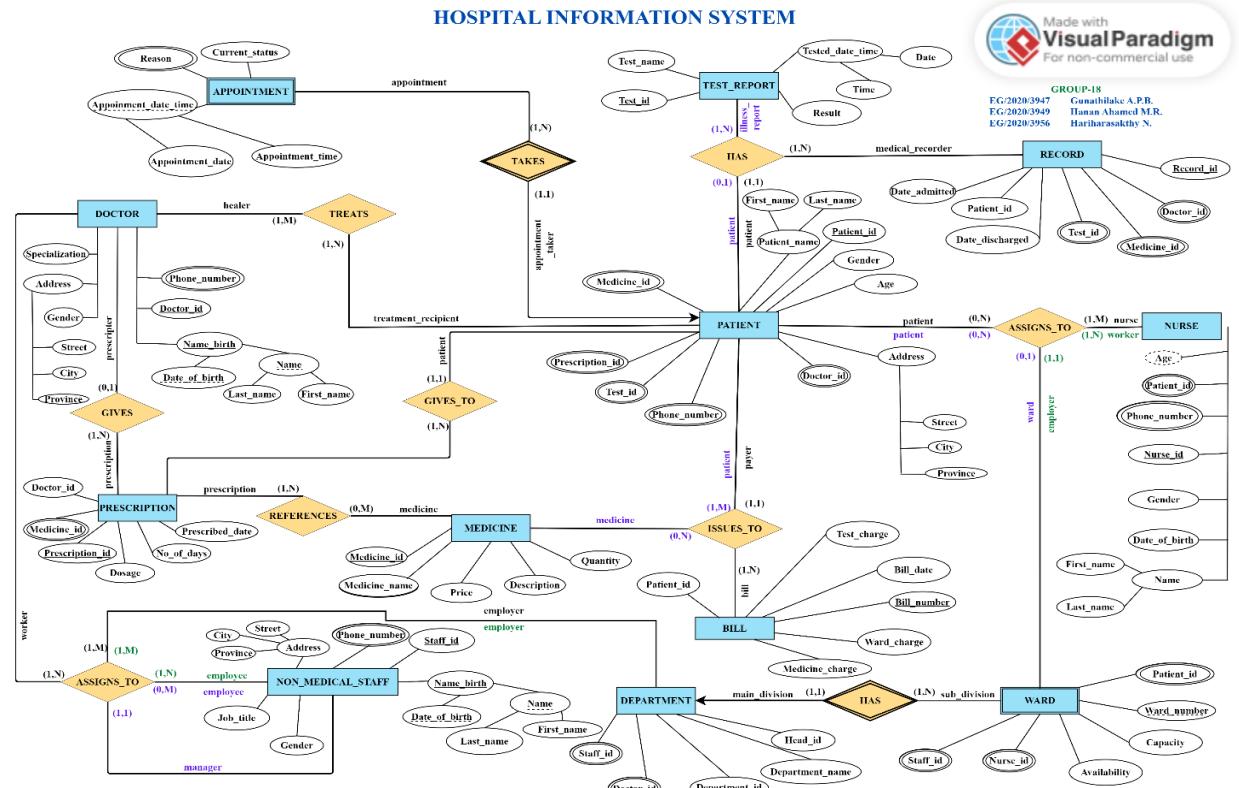


Figure 1:ER Diagram Design

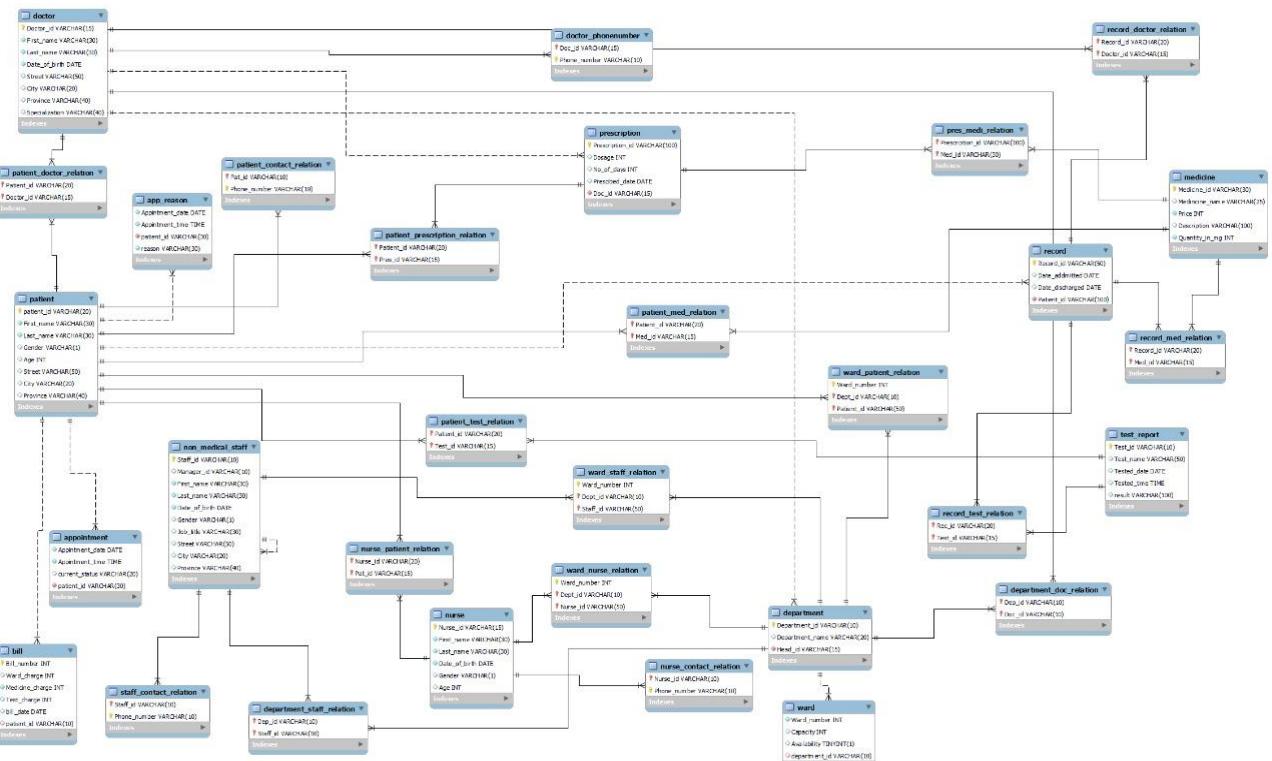
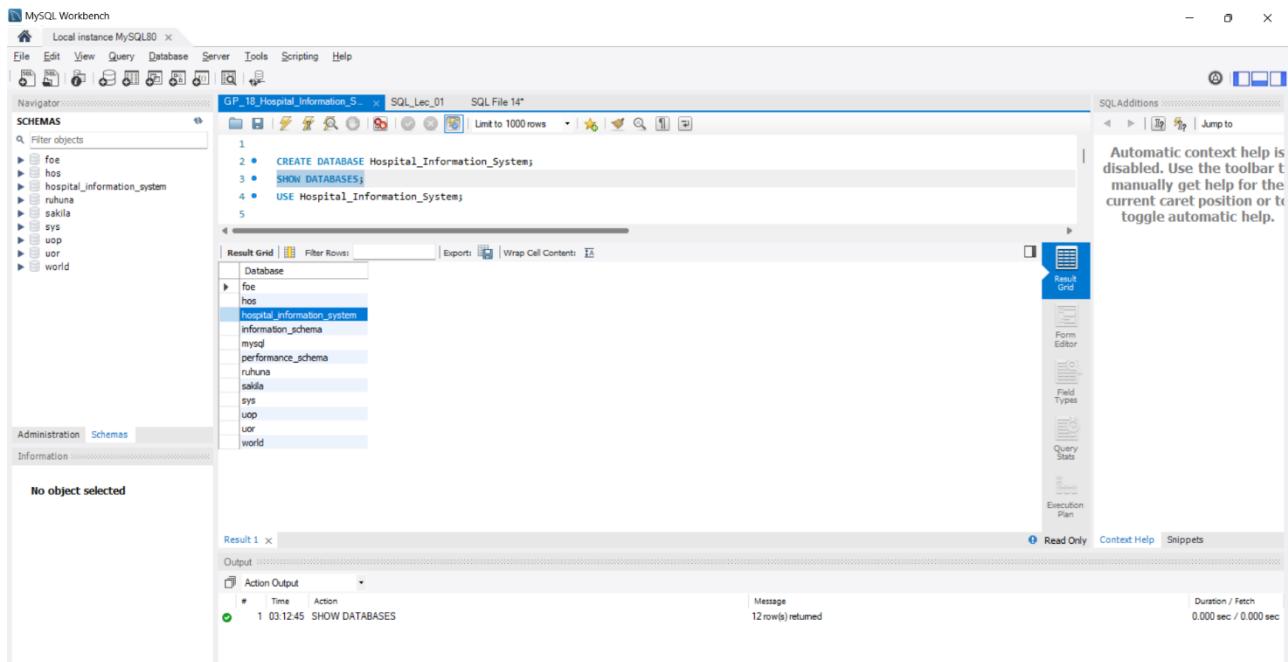


Figure 2:ER Diagram design from MySQL Workbench

Chapter 3 : Implementation

3.1 Schema Creation



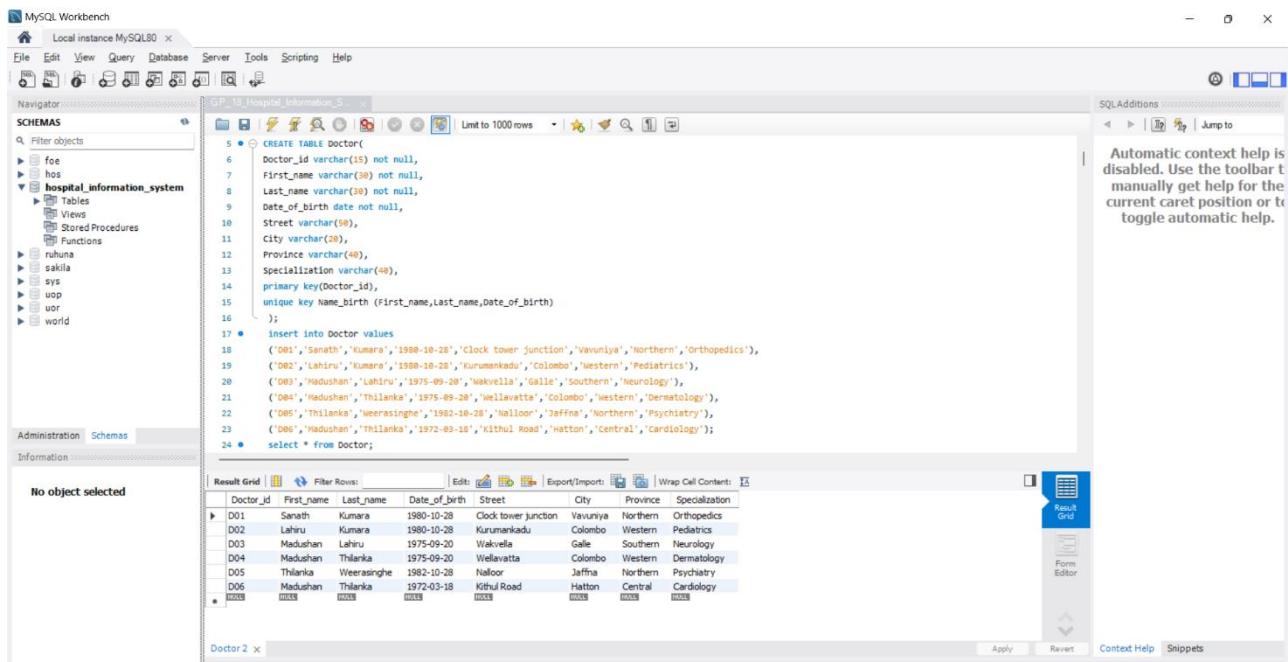
The screenshot shows the MySQL Workbench interface with the following details:

- Title Bar:** Local instance MySQL80, SQL_Lec_01, SQL File 14*
- Navigator:** Shows the schema tree with the newly created `Hospital_Information_System` database.
- SQL Editor:** Contains the following SQL code:

```
1 CREATE DATABASE Hospital_Information_System;
2 SHOW DATABASES;
3 USE Hospital_Information_System;
```
- Result Grid:** Displays the output of the `SHOW DATABASES` command, listing the databases: `foe`, `hos`, `information_schema`, `mysql`, `performance_schema`, `ruhuna`, `sakila`, `sys`, `uop`, `uor`, and `world`.
- Message Area:** Shows "12 rows(s) returned".
- Toolbar:** Includes icons for Save, Undo, Redo, Copy, Paste, and Execute.
- Help:** A note says "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

3.2 Creation of Tables & Insertion of data

3.2.1 Doctor Table



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

- Title Bar:** Local instance MySQL80, SQL_Lec_01, SQL File 14*
- Navigator:** Shows the schema tree with the `hospital_information_system` database selected, displaying Tables, Views, Stored Procedures, and Functions.
- SQL Editor:** Contains the following SQL code for creating the `Doctor` table and inserting data:

```
5 CREATE TABLE Doctor(
6     Doctor_id varchar(15) not null,
7     First_name varchar(30) not null,
8     Last_name varchar(30) not null,
9     Date_of_birth date not null,
10    Street varchar(50),
11    City varchar(30),
12    Province varchar(40),
13    Specialization varchar(40),
14    primary key(Doctor_id),
15    unique key_name_birth (First_name,Last_name,Date_of_birth)
16 );
17 insert into Doctor values
18 ('D01','Sanath','Kumara','1980-10-28','Clock tower junction','Vavuniya','Northern','Orthopedics'),
19 ('D02','Lahiru','Kumara','1980-10-28','Kurumankadu','Colombo','Western','Pediatrics'),
20 ('D03','Madushan','Lahiru','1975-09-20','Wakellawa','Galle','Southern','Neurology'),
21 ('D04','Madushan','Thilanka','1975-09-20','Wellawatta','Colombo','Western','Dermatology'),
22 ('D05','Thilanka','Weerasinghe','1982-10-28','Maloon','Jaffna','Northern','Psychiatry'),
23 ('D06','Madushan','Thilanka','1972-03-18','Kithul Road','Matton','Central','Cardiology');
```
- Result Grid:** Displays the inserted data in the `Doctor` table:

Doctor_id	First_name	Last_name	Date_of_birth	Street	City	Province	Specialization
D01	Sanath	Kumara	1980-10-28	Clock tower junction	Vavuniya	Northern	Orthopedics
D02	Lahiru	Kumara	1980-10-28	Kurumankadu	Colombo	Western	Pediatrics
D03	Madushan	Lahiru	1975-09-20	Wakellawa	Galle	Southern	Neurology
D04	Madushan	Thilanka	1975-09-20	Wellawatta	Colombo	Western	Dermatology
D05	Thilanka	Weerasinghe	1982-10-28	Maloon	Jaffna	Northern	Psychiatry
D06	Madushan	Thilanka	1972-03-18	Kithul Road	Matton	Central	Cardiology
- Message Area:** Shows "24 rows(s) inserted".
- Toolbar:** Includes icons for Save, Undo, Redo, Copy, Paste, and Execute.
- Help:** A note says "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

3.2.2 Doctor_PhoneNumber Table

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

- Schemas:** hospital_information_system
- Table:** Doctor_PhoneNumber
- SQL Editor:** Contains the SQL code for creating the table and inserting 10 rows of data.
- Result Grid:** Displays the inserted data.
- Information:** Shows "No object selected".
- Toolbar:** Includes icons for file operations, search, and help.
- Right Panel:** Shows context help for automatic context help being disabled.

```
CREATE TABLE Doctor_PhoneNumber
(
    Doc_Id varchar(15) not null,
    Phone_number varchar(10) not null,
    primary key(Doc_Id,Phone_number),
    constraint fkDoc foreign key(Doc_Id) references Doctor(Doc_Id) on delete cascade on update cascade);

insert into Doctor_PhoneNumber values
('D01','0768993412'),
('D01','0728939412'),
('D02','077693412'),
('D03','0769341412'),
('D04','0728972812'),
('D04','0768283934'),
('D05','0740993412'),
('D06','0728123412');

select * from Doctor_PhoneNumber;
```

Doc_Id	Phone_number
D01	0728939412
D01	0768993412
D02	077693412
D03	0769341412
D04	0728972812
D04	0768283934
D05	0740993412
D06	0728123412

3.2.3 Medicine Table

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

- Schemas:** hospital_information_system
- Table:** medicine
- SQL Editor:** Contains the SQL code for creating the table and inserting 8 rows of data.
- Result Grid:** Displays the inserted data.
- Information:** Shows "No object selected".
- Toolbar:** Includes icons for file operations, search, and help.
- Right Panel:** Shows context help for automatic context help being disabled.

```
CREATE TABLE medicine(
    Medicine_id varchar(30) not null,
    Medicine_name varchar(25),
    Price Integer not null,
    Description varchar(100),
    Quantity_in_mg int not null,
    primary key(Medicine_id));

insert into medicine values
('M0001','Carisoprodol ','485.00','Pain reliever',50),
('M0002','Cetirizine ','600.00','Allergy medication',50),
('M0003','amoxicillin ',385 ,null,50),
('M0004','diclofenac gel ',100.00,'Pain reliever',50),
('M0005','Delsym (syrup) ',250.00,'Cough suppressant',50),
('M0006','Chlorzoxazone ','980.00,'Muscle relaxant',50);

select * from medicine;
```

Medicine_id	Medicine_name	Price	Description	Quantity_in_mg
M0001	Carisoprodol	485	Pain reliever	50
M0002	Cetirizine	600	Allergy medication	50
M0003	amoxicillin	385		50
M0004	diclofenac gel	100	Pain reliever	50
M0005	Delsym (syrup)	250	Cough suppressant	50
M0006	Chlorzoxazone	980	Muscle relaxant	50

3.2.4 Prescription Table

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

- Schemas:** The current schema is "GT_10_Hospital_Information_S".
- SQL Editor:** The code pane contains the SQL script for creating the Prescription table and inserting data. The table structure includes columns: Prescription_id (primary key), Dosage, No_of_days, Prescribed_date, and Doc_id. A foreign key constraint "fk_PresDoc" links Doc_id to the Doctor table.
- Result Grid:** The data pane displays the inserted data into the Prescription table. The columns are Prescription_id, Dosage, No_of_days, Prescribed_date, and Doc_id. The data rows are:

Prescription_id	Dosage	No_of_days	Prescribed_date	Doc_id
PR001	2	7	2023-08-15	D01
PR002	1	14	2023-08-20	D01
PR003	3	5	2023-08-25	D01
PR004	2	10	2023-08-28	D01
PR005	1	7	2023-08-30	D01
PR006	2	3	2023-09-02	D01

- Information:** The "Administration" tab is selected.
- Toolbar:** Standard MySQL Workbench toolbar icons are visible.
- Help:** A note in the top right says: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

3.2.5 Pres_Medi_Relation Table

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

- Schemas:** The current schema is "GT_10_Hospital_Information_S".
- SQL Editor:** The code pane contains the SQL script for creating the Pres_Medi_Relation table and inserting data. The table structure includes columns: Prescription_id (primary key) and Med_id. Foreign keys "fk_pres" and "fk_med" link to the Prescription and Medicine tables respectively.
- Result Grid:** The data pane displays the inserted data into the Pres_Medi_Relation table. The columns are Prescription_id and Med_id. The data rows are:

Prescription_id	Med_id
PR001	M0001
PR002	M0001
PR003	M0001
PR004	M0001
PR005	M0001
PR001	M0002
PR002	M0003
PR002	M0004

- Information:** The "Administration" tab is selected.
- Toolbar:** Standard MySQL Workbench toolbar icons are visible.
- Help:** A note in the top right says: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

3.2.6 Non_medical_staff Table

The screenshot shows the MySQL Workbench interface with the database 'GP_10_Hospital_Information_S' selected. In the Navigator pane, the 'Tables' section under the 'hospital_information' schema is expanded, showing the 'Non_medical_staff' table. The SQL Editor pane contains the DDL and DML code for creating the table and inserting data. The Result Grid pane displays the inserted data.

```

create table Non_medical_staff(
    Staff_id varchar(10) not null,
    Manager_id varchar(10) ,
    First_name varchar(20) not null,
    Last_name varchar(30) not null,
    Date_of_birth date not null,
    Gender varchar(1),
    Job_title varchar(30),
    Street varchar(50),
    City varchar(20),
    Province varchar(40),
    unique key Name_birth (First_name,Last_name,Date_of_birth),
    primary key(Staff_id),
    constraint fk_staff foreign key(Manager_id) references Non_medical_staff(Staff_id));
    INSERT INTO Non_medical_staff VALUES
    ('S001','null','Christi','Smith','1985-05-10','M','Manager','123 Main St','Cityville','CA'),
    ('S002','S001','Emily','Williams','1978-01-10','M','Admin Asst','456 Elm St','Townsville','NY'),
    ('S003','S001','Jane','Williams','1970-01-10','F','IT Support','123 second St','Southwest','Central'),
    ('S004','S002','Emily','Doe','1970-01-10','F','HR Asst','null','Vavuniya','Northern'),
    ('S005','S002','Michael','Doe','1975-05-10','M','Ambulance driver','101 Pine Ave','Galle','Southern'),
    ('S006','S002','Christi','Smith','1995-05-10','F','Ward clerk','13 Main St','Hatton','Central');
    select * from Non_medical_staff;
  
```

Staff_id	Manager_id	First_name	Last_name	Date_of_birth	Gender	Job_title	Street	City	Province
S001		Christi	Smith	1985-05-10	M	Manager	123 Main St	Cityville	CA
S002	S001	Emily	Williams	1970-01-10	M	Admin Asst	456 Elm St	Townsville	NY
S003	S001	Jane	Williams	1970-01-10	F	IT Support	123 second St	Southwest	Central
S004	S002	Emily	Doe	1970-01-10	F	HR Asst	null	Vavuniya	Northern
S005	S002	Michael	Doe	1975-05-10	M	Ambulance driver	101 Pine Ave	Galle	Southern
S006	S002	Christi	Smith	1995-05-10	F	Ward clerk	13 Main St	Hatton	Central

3.2.7 Staff_contact_relation Table

The screenshot shows the MySQL Workbench interface with the database 'GP_10_Hospital_Information_S' selected. In the Navigator pane, the 'Tables' section under the 'hospital_information' schema is expanded, showing the 'Staff_contact_relation' table. The SQL Editor pane contains the DDL and DML code for creating the table and inserting data. The Result Grid pane displays the inserted data.

```

create table Staff_contact_relation(
    Staff_id varchar(10) not null,
    Phone_number varchar(10) not null,
    primary key(Staff_id,Phone_number),
    constraint fkStaff foreign key(Staff_id) references Non_medical_staff(Staff_id) on delete cascade on update cascade);
    insert into Staff_contact_relation values
    ('S001','0768993412'),
    ('S001','0728393412'),
    ('S002','0776493412'),
    ('S003','0769341412'),
    ('S004','0728972812'),
    ('S005','0768283934'),
    ('S006','0740993412'),
    ('S006','0728123412');
    select * from Staff_contact_relation;
  
```

Staff_id	Phone_number
S001	0728393412
S001	0768993412
S002	0776493412
S003	0769341412
S004	0728972812
S005	0768283934
S006	0728123412
S006	0740993412

3.2.8 Department Table

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

- Schemas:** Local instance MySQL80, hospital_information_S.
- Query Editor:** Contains the SQL code for creating the Department table and inserting data into it.
- Result Grid:** Displays the data inserted into the Department table, showing 6 rows with columns: Department_id, Department_name, and Head_id.
- Information:** Shows "No object selected".
- Toolbar:** Includes icons for File, Edit, View, Query, Database, Server, Tools, Scripting, Help, and various navigation and search tools.
- Right Panel:** Shows a message about automatic context help being disabled and provides a toolbar for navigating help.

```
149
150 • create table Department(
151     Department_id varchar(10) not null,
152     Department_name varchar(20),
153     Head_id varchar(15) not null,
154     primary key(Department_id),
155     constraint fk_dep foreign key(Head_id) references doctor(Doctor_id) on delete cascade on update cascade);
156
157 • insert into department values
158     ('DE01','Pediatrics ','D02'),
159     ('DE02','Orthopedic ','D01'),
160     ('DE03','Psychiatry ','D05'),
161     ('DE04','Neurology ','D03'),
162     ('DE05','Dermatology ','D04'),
163     ('DE06','Inpatient ','D06');
164
165 • select * from department;
```

Department_id	Department_name	Head_id
DE01	Pediatrics	D02
DE02	Orthopedic	D01
DE03	Psychiatry	D05
DE04	Neurology	D03
DE05	Dermatology	D04
DE06	Inpatient	D06
DE01	Hospital	H001

3.2.9 Department_Staff_Relation Table

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

- Schemas:** Local instance MySQL80, hospital_information_S.
- Query Editor:** Contains the SQL code for creating the Department_Staff_Relation table and inserting data into it.
- Result Grid:** Displays the data inserted into the Department_Staff_Relation table, showing 10 rows with columns: Dep_id and Staff_id.
- Information:** Shows "No object selected".
- Toolbar:** Includes icons for File, Edit, View, Query, Database, Server, Tools, Scripting, Help, and various navigation and search tools.
- Right Panel:** Shows a message about automatic context help being disabled and provides a toolbar for navigating help.

```
166 • create table department_staff_relation(
167     Dep_id varchar(10) not null,
168     Staff_id varchar(10) not null,
169     primary key(Dep_id,Staff_id),
170     constraint fkDepstaff foreign key(Staff_id) references Non_Medical_staff(Staff_id) on delete cascade on update cascade,
171     constraint fkDep foreign key(Dep_id) references Department(Department_id) on delete cascade on update cascade);
172 • insert into Department_Staff_Relation values
173     ('DE01','S001'),
174     ('DE01','S002'),
175     ('DE02','S002'),
176     ('DE02','S003'),
177     ('DE02','S004'),
178     ('DE03','S001'),
179     ('DE04','S001'),
180     ('DE05','S005'),
181     ('DE06','S006');
182
183 • select * from Department_Staff_Relation;
```

Dep_id	Staff_id
DE01	S001
DE03	S001
DE04	S001
DE01	S002
DE02	S002
DE02	S003
DE02	S004
DE05	S005
DE06	S006
DE01	S003

3.2.10 Department_Doc_Relation Table

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

- Schemas:** hospital_information_5
- SQL Editor:** Contains the SQL code for creating the table and inserting data.
- Result Grid:** Displays the inserted data.
- Information:** Shows the table structure.

```
create table Department_Doc_Relation
(
    Dep_Id varchar(10) not null,
    Doc_Id varchar(10) not null,
    primary key(Dep_Id,Doc_Id),
    constraint fkDepDoc foreign key(Dep_Id) references Doctor(Doctor_Id) on delete cascade on update cascade,
    constraint fkDocDoc foreign key(Doc_Id) references Department(Department_Id) on delete cascade on update cascade
);

insert into Department_Doc_Relation values
('DE01','D01'),
('DE01','D01'),
('DE01','D05'),
('DE02','D01'),
('DE03','D02'),
('DE03','D05'),
('DE04','D03'),
('DE05','D04'),
('DE05','D05'),
('DE06','D06'),
('DE06','D06');

select * from Department_Doc_Relation;
```

Dep_Id	Doc_Id
DE01	D01
DE02	D01
DE01	D02
DE03	D02
DE04	D03
DE05	D04
DE01	D05
DE03	D05
DE05	D06
DE06	D06
DE06	D06

3.2.11 Test_report Table

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

- Schemas:** hospital_information_5
- SQL Editor:** Contains the SQL code for creating the table and inserting data.
- Result Grid:** Displays the inserted data.
- Information:** Shows the table structure.

```
create table Test_report
(
    Test_Id varchar(10) not null,
    Test_name varchar(50),
    Tested_date date,
    Tested_time time,
    result varchar(100),
    primary key(Test_Id)
);

insert into Test_report values
('TR0001','Blood test','2023-07-13','15:00','All are normal'),
('TR0002','Hematocrit','2023-07-14','11:00','All are normal'),
('TR0003','Liver function test','2023-07-14','09:00','All are normal'),
('TR0004','Pregnancy test','2023-07-16','15:00','positive'),
('TR0005','Kidney function test','2023-07-18','14:00','In the middle level of kidney failure'),
('TR0006','Syphilis test','2023-07-19','15:30','All are normal');

select * from Test_report;
```

Test_Id	Test_name	Tested_date	Tested_time	result
TR0001	Blood test	2023-07-13	15:00:00	All are normal
TR0002	Hematocrit	2023-07-14	11:00:00	All are normal
TR0003	Liver function test	2023-07-14	09:00:00	All are normal
TR0004	Pregnancy test	2023-07-16	15:00:00	positive
TR0005	Kidney function test	2023-07-18	14:00:00	In the middle level of kidney failure
TR0006	Syphilis test	2023-07-19	15:30:00	All are normal

3.2.12 Patient Table

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

- Schemas:** Local instance MySQL80, hospital_information_5
- SQL Editor:** Contains the SQL code for creating the Patient table and inserting data. The table structure includes columns: Patient_Id (primary key), First_name, Last_name, Gender, Age, Street, City, and Province. The inserted data consists of 10 rows of patient information.
- Result Grid:** Displays the inserted patient data in a tabular format.
- Information:** Shows the table structure with columns: Patient_Id, First_name, Last_name, Gender, Age, Street, City, and Province.

```

221 • create table Patient(
222     Patient_Id varchar(20)not null,
223     First_name varchar(30) not null,
224     Last_name varchar(30) not null,
225     Gender varchar(1),
226     Age int,
227     Street varchar(50),
228     City varchar(20),
229     Province varchar(40),
230     primary key(Patient_Id)
231 );
232 • INSERT INTO Patient VALUES
233 ('P001', 'John', 'Smith', 'M', 45, '123 Main St', 'Anytown', 'California'),
234 ('P002', 'Jane', 'Doe', 'F', 32, '456 Elm Ave', 'Springfield', 'Illinois'),
235 ('P003', 'Michael', 'Johnson', 'M', 28, '789 Oak Rd', 'Oakville', 'New York'),
236 ('P004', 'Emily', 'Williams', 'F', 60, '101 Pine St', 'Pineville', 'Louisiana'),
237 ('P005', 'David', 'Brown', 'M', 50, '222 Maple Lane', 'Maple City', 'Michigan'),
238 ('P006', 'Sarah', 'Anderson', 'F', 22, '555 Cedar Ave', 'Cederville', 'Texas');
239 • select * from patient;
...

```

3.2.13 Patient_Doctor_relation Table

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

- Schemas:** Local instance MySQL80, hospital_information_5
- SQL Editor:** Contains the SQL code for creating the Patient_Doctor_relation table and inserting data. The table structure includes columns: Patient_Id (foreign key) and Doctor_Id (foreign key). It features constraints for cascading deletes.
- Result Grid:** Displays the inserted data in a tabular format.
- Information:** Shows the table structure with columns: Patient_Id and Doctor_Id.

```

242 • create table Patient_Doctor_relation(
243     Patient_Id varchar(20)not null,
244     Doctor_Id varchar(15) not null,
245     primary key(Patient_Id,Doctor_Id),
246     constraint fkPat foreign key(Patient_Id) references patient(Patient_Id) on delete cascade on update cascade,
247     constraint fkDoc foreign key(Doctor_Id) references Doctor(Doctor_Id) on delete cascade on update cascade
248 );
249 • INSERT INTO Patient_Doctor_relation VALUES
250 ('P001','D01'),
251 ('P001','D02'),
252 ('P001','D03'),
253 ('P002','D01'),
254 ('P002','D03'),
255 ('P003','D01'),
256 ('P003','D05'),
257 ('P005','D02'),
258 ('P005','D03'),
259 ('P006','D06');
260 • select * from Patient_Doctor_relation;
...

```

3.2.14 Patient_Med_relation Table

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

- Schemas:** Local instance MySQL80, hospital_information_5
- SQL Editor:** Contains the SQL code for creating the table and inserting data:

```
261 •  create table Patient_Med_relation
262   (
263     Patient_Id varchar(10) not null,
264     Med_Id varchar(10) not null,
265     primary key(Patient_Id,med_id),
266     constraint fk_Pat foreign key(Patient_Id) references patient(Patient_Id) on delete cascade on update cascade,
267     constraint fk_PMed foreign key(Med_Id) references Medicine(Medicine_id) on delete cascade on update cascade
268   );
269 •  insert into Patient_Med_relation values
270   ('P001','M0001'),
271   ('P002','M0001'),
272   ('P003','M0002'),
273   ('P002','M0004'),
274   ('P003','M0005'),
275   ('P004','M0006'),
276   ('P005','M0006'),
277   ('P006','M0004'),
278   ('P007','M0002');
279 •  select * from Patient_Med_relation;
```
- Result Grid:** Shows the data inserted into the table:

Patient_Id	Med_Id
P001	M0001
P002	M0001
P003	M0001
P002	M0002
P006	M0002
P002	M0004
P005	M0004
P003	M0005
P004	M0006
P005	M0006
P006	M0004
P007	M0002
- Information:** No object selected.

3.2.15 Patient_contact_relation Table

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

- Schemas:** Local instance MySQL80, hospital_information_5
- SQL Editor:** Contains the SQL code for creating the table and inserting data:

```
282 •  create table Patient_contact_relation
283   (
284     Pat_Id varchar(10) not null,
285     Phone_number varchar(10) not null,
286     primary key(Pat_Id,Phone_number),
287     constraint fk_PatC foreign key(Pat_Id) references Patient(Patient_Id) on delete cascade on update cascade;
288
289 •  insert into Patient_contact_relation values
290   ('P001','0768993412'),
291   ('P001','0728393412'),
292   ('P002','0724787912'),
293   ('P003','0769001412'),
294   ('P004','0728977864'),
295   ('P005','0765776684'),
296   ('P006','07409993412'),
297
298 •  select * from Patient_contact_relation;
```
- Result Grid:** Shows the data inserted into the table:

Pat_Id	Phone_number
P001	0728393412
P001	0768993412
P002	0724787912
P003	0769001412
P004	0728977864
P005	0765776684
P006	0728123412
P006	07409993412
- Information:** No object selected.

3.2.16 Patient_Test_relation Table

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

- Schemas:** hospital_information
- Table:** Patient_Test_relation
- SQL Editor:** Contains the CREATE TABLE statement and an INSERT INTO statement with 10 rows of data.
- Result Grid:** Displays the 10 rows of data inserted into the table.
- Action Output:** Shows two log entries for the SELECT statements run on the table.

```

CREATE TABLE Patient_Test_relation (
    Patient_id varchar(20) NOT NULL,
    Test_id varchar(15) NOT NULL,
    PRIMARY KEY(Patient_id,Test_id),
    constraint fk_TPat foreign key(Patient_id) references patient(Patient_id) on delete cascade on update cascade,
    constraint fk_PTest foreign key(Test_id) references Test_report(Test_id) on delete cascade on update cascade
);

INSERT INTO Patient_Test_relation values
('P001','TR0001'),
('P001','TR0003'),
('P002','TR0002'),
('P002','TR0004'),
('P003','TR0005'),
('P004','TR0006'),
('P004','TR0008'),
('P005','TR0007');

SELECT * FROM Patient_Test_relation;
  
```

3.2.17 Patient_Prescription_relation Table

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

- Schemas:** hospital_information
- Table:** Patient_Prescription_relation
- SQL Editor:** Contains the CREATE TABLE statement and an INSERT INTO statement with 10 rows of data.
- Result Grid:** Displays the 10 rows of data inserted into the table.
- Action Output:** Shows two log entries for the SELECT statements run on the table.

```

CREATE TABLE Patient_Prescription_relation (
    Patient_id varchar(20) NOT NULL,
    Pres_id varchar(15) NOT NULL,
    PRIMARY KEY(Patient_id,Pres_id),
    constraint fk_PPat foreign key(Patient_id) references patient(Patient_id) on delete cascade on update cascade,
    constraint fk_PPres foreign key(Pres_id) references Prescription(Prescription_id) on delete cascade on update cascade
);

INSERT INTO Patient_Prescription_relation values
('P001','PR001'),
('P001','PR002'),
('P001','PR003'),
('P002','PR004'),
('P002','PR005'),
('P003','PR006'),
('P004','PR007'),
('P004','PR008'),
('P005','PR009');

SELECT * FROM Patient_Prescription_relation;
  
```

3.2.18 Appointment Table

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

- Schemas:** hospital_info
- SQL Editor:** Contains the SQL code for creating the appointment table and inserting sample data.
- Result Grid:** Displays the data inserted into the appointment table.

```

329 • create table appointment
330   Appointment_date date not null,
331   Appointment_time time not null,
332   current_status varchar(20),
333   patient_id varchar(30) not null,
334   constraint fk_PatApp foreign key(Patient_id) references patient(Patient_id) on delete cascade on update cascade);
335
336 • insert into appointment values
337   ('2023-08-20','17:00','pending','P001'),
338   ('2023-08-21','15:00','confirmed','P001'),
339   ('2023-08-23','15:00','completed','P001'),
340   ('2023-08-20','15:00','canceled','P001'),
341   ('2023-08-20','15:00','confirmed','P005'),
342   ('2023-08-21','17:00','pending','P004');
343
344 • select * from appointment;
345

```

Appointment_date	Appointment_time	current_status	patient_id
2023-08-20	17:00:00	pending	P001
2023-08-21	15:00:00	confirmed	P001
2023-08-23	15:00:00	completed	P001
2023-08-20	15:00:00	canceled	P001
2023-08-20	15:00:00	confirmed	P005
2023-08-21	17:00:00	pending	P004

3.2.19 App_reason Table

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

- Schemas:** hospital_info
- SQL Editor:** Contains the SQL code for creating the app_reason table and inserting sample data.
- Result Grid:** Displays the data inserted into the app_reason table.

```

347 • create table app_reason(
348   Appointment_date date not null,
349   Appointment_time time not null,
350   patient_id varchar(30) not null,
351   reason varchar(30) not null,
352   constraint fk_AppReason foreign key(Patient_id) references patient(Patient_id) on delete cascade on update cascade);
353
354 • insert into app_reason values
355   ('2023-08-20','17:00','P001','ill'),
356   ('2023-08-20','17:00','P001','Diabetics'),
357   ('2023-08-21','15:00','P001','Full checkup'),
358   ('2023-08-23','15:00','P001','Knee pain'),
359   ('2023-08-22','15:00','P001','ill'),
360   ('2023-08-22','15:00','P005','Tooth pain'),
361   ('2023-08-21','17:00','P004','ill');
362
363 • select * from app_reason;
364

```

Appointment_date	Appointment_time	patient_id	reason
2023-08-20	17:00:00	P001	ill
2023-08-20	17:00:00	P001	Diabetics
2023-08-21	15:00:00	P001	Full checkup
2023-08-23	15:00:00	P001	Knee pain
2023-08-20	15:00:00	P001	ill
2023-08-22	15:00:00	P005	Tooth pain
2023-08-21	17:00:00	P004	ill

3.2.20 Record Table

The screenshot shows the MySQL Workbench interface with the 'GP_18_Hospital_Information_S' database selected. In the Navigator pane, the 'hospital_information' schema is expanded, showing its tables, views, stored procedures, and functions. The 'Tables' section contains a single table named 'record'. The SQL Editor pane displays the creation script for the 'record' table:

```
372
373 • create table record(
374     Record_id varchar(50) not null,
375     Date_admitted date ,
376     Date_discharged date,
377     primary key(Record_id),
378     Patient_id varchar(100) not null default 'P004',
379     constraint fk_r_Patient foreign key(Patient_id) references patient(Patient_id) on delete cascade on update cascade);
380
381 • insert into record(Record_id,Date_admitted,Date_discharged) values
382     ('R0001','2023-08-23','2023-08-26'),
383     ('R0002',null,null),
384     ('R0003','2023-07-23','2023-07-26'),
385     ('R0004','2023-05-13','2023-05-16'),
386     ('R0005','2023-04-27','2023-04-29'),
387     ('R0006','2023-02-23','2023-02-26');
388
389 • select * from record;
390
```

The Result Grid pane shows the data inserted into the 'record' table:

Record_id	Date_admitted	Date_discharged	Patient_id
R0001	2023-08-23	2023-08-26	P004
R0002	NULL	NULL	P004
R0003	2023-07-23	2023-07-26	P004
R0004	2023-05-13	2023-05-16	P004
R0005	2023-04-27	2023-04-29	P004
R0006	2023-02-23	2023-02-26	P004

3.2.21 Record_Doctor_relation Table

The screenshot shows the MySQL Workbench interface with the 'GP_18_Hospital_Information_S' database selected. In the Navigator pane, the 'hospital_information' schema is expanded, showing its tables, views, stored procedures, and functions. The 'Tables' section contains a table named 'Record_Doctor_relation'. The SQL Editor pane displays the creation script for the 'Record_Doctor_relation' table:

```
393 • create table Record_Doctor_relation
394     (Record_id varchar(20) not null,
395      Doctor_id varchar(15) not null,
396      primary key(Record_id,Doctor_id),
397      constraint fkrec foreign key(Record_id) references record(Record_id) on delete cascade on update cascade,
398      constraint fk_reDoc foreign key(Doctor_id) references Doctor(Doctor_id) on delete cascade on update cascade);
399
400 • INSERT INTO Record_Doctor_relation VALUES
401     ('R0001','D01'),
402     ('R0001','D02'),
403     ('R0002','D03'),
404     ('R0002','D04'),
405     ('R0002','D05'),
406     ('R0003','D05'),
407     ('R0004','D06');
408
409 • select * from Record_Doctor_relation;
410
```

The Result Grid pane shows the data inserted into the 'Record_Doctor_relation' table:

Record_id	Doctor_id
R0001	D01
R0001	D02
R0002	D03
R0002	D04
R0003	D05
R0004	D06

3.2.22 Record_Test_relation Table

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

- Schemas:** hospital_information
- Table:** Record_Test_relation
- SQL Editor:** Contains the following code:

```
402 • create table Record_Test_relation
403   (Rec_id varchar(20)not null,
404    Test_Id varchar(15) not null,
405    primary key(Rec_id,Test_id),
406    constraint fk_TRec foreign key(Rec_id) references record(record_id) on delete cascade on update cascade,
407    constraint fk_RecTest foreign key(Test_id) references Test_report(Test_id) on delete cascade on update cascade);
408
409 • insert into Record_Test_relation values
410   ('R0001','TR0001'),
411   ('R0001','TR0003'),
412   ('R0002','TR0002'),
413   ('R0003','TR0004'),
414   ('R0003','TR0005'),
415   ('R0003','TR0006')
416 • select * from Record_Test_relation;
```
- Result Grid:** Shows the inserted data:

Rec_id	Test_Id
R0001	TR0001
R0002	TR0002
R0001	TR0003
R0003	TR0004
R0003	TR0005
R0003	TR0006
- Output:** Shows the execution log:

Action	Time	Message	Duration / Fetch
select * from Patient_Test_relation LIMIT 0, 1000	5 10:45:56	6 rows(s) returned	0.000 sec / 0.000 sec
select * from Record_Test_relation LIMIT 0, 1000	6 10:48:02	6 row(s) returned	0.000 sec / 0.000 sec

3.2.23 Record_Med_relation Table

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

- Schemas:** hospital_information
- Table:** Record_Med_relation
- SQL Editor:** Contains the following code:

```
429 • create table Record_Med_relation
430   (Record_id varchar(20)not null,
431    Med_Id varchar(15) not null,
432    primary key(Record_id,Med_Id),
433    constraint fk_Record foreign key(Record_id) references record(record_id) on delete cascade on update cascade,
434    constraint fk_RecordMed foreign key(Med_Id) references Medicine(Medicine_id) on delete cascade on update cascade);
435
436 • insert into Record_Med_relation values
437   ('R0001','M0001'),
438   ('R0002','M0002'),
439   ('R0002','M0003'),
440   ('R0002','M0004'),
441   ('R0003','M0005'),
442   ('R0004','M0006'),
443   ('R0005','M0003');
444
445 • select * from Record_Med_relation;
```
- Result Grid:** Shows the inserted data:

Record_id	Med_Id
R0001	M0001
R0002	M0002
R0002	M0003
R0005	M0003
R0002	M0004
R0003	M0005
R0004	M0006
NULL	NULL

3.2.24 Nurse Table

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Schemas:** Navigator pane shows the 'GP_10_Hospital_Information_S' database containing schemas like 'foe', 'hos', 'ruhuna', 'sakila', 'sys', 'uop', 'uor', and 'world'.
- SQL Editor:** Displays the SQL code for creating the 'nurse' table and inserting data. The table has columns: Nurse_id (varchar(15) not null), First_name (varchar(30) not null), Last_name (varchar(30) not null), Date_of_birth (date not null default '2000-04-12'), Gender (varchar(1) default 'F'), and primary key(Nurse_id). Data is inserted for 8 rows.
- Result Grid:** Shows the resulting data from the 'select * from nurse' query, displaying columns: Nurse_id, First_name, Last_name, Date_of_birth, Gender, and Age.
- Output:** Shows the output of the 'nurse 198' command.

3.2.25 Nurse_contact_relation Table

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Schemas:** Navigator pane shows the 'GP_10_Hospital_Information_S' database containing schemas like 'foe', 'hos', 'ruhuna', 'sakila', 'sys', 'uop', 'uor', and 'world'. It also lists Tables, Views, Stored Procedures, and Functions.
- SQL Editor:** Displays the SQL code for creating the 'Nurse_contact_relation' table. The table has columns: Nurse_id (varchar(15) not null), Phone_number (varchar(15) not null), and primary key(Nurse_id,Phone_number). A constraint fkNc foreign key(Nurse_id) references nurse(Nurse_id) on delete cascade is defined. Data is inserted for 8 rows.
- Result Grid:** Shows the resulting data from the 'select * from Nurse_contact_relation' query, displaying columns: Nurse_id and Phone_number.
- Output:** Shows the output of the 'Nurse_contact_relation 29' command.

3.2.26 NursePat_relation Table

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

- Navigator:** Shows the schema `GP_10_Hospital_Information_S` with objects `foe`, `hos`, and `hospital_information`.
- SQL Editor:** Displays the SQL code for creating the `NursePat_relation` table and inserting data. The table has columns `Nurse_id` and `Pat_id`. The inserted data is as follows:

Nurse_id	Pat_id
N001	P001
N005	P001
N002	P002
N003	P003
N003	P004
N006	P005
N003	P006
N004	P006
N005	P007
N008	P001

- Result Grid:** Shows the result of the `select * from NursePat_relation;` query.
- SQL Additions:** A note states: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

3.2.27 Bill Table

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

- Navigator:** Shows the schema `GP_10_Hospital_Information_S` with objects `foe`, `hos`, `ruhuna`, `sakila`, `sys`, `uop`, `uor`, and `world`.
- SQL Editor:** Displays the SQL code for creating the `bill` table and inserting data. The table has columns `Bill_number`, `Ward_charge`, `Medicine_charge`, `Test_charge`, `bill_date`, and `patient_id`. The inserted data is as follows:

Bill_number	Ward_charge	Medicine_charge	Test_charge	bill_date	patient_id
1	1000	1500	2000	2023-04-23	P001
2	500	1200	2500	2023-02-13	P002
3	1000	1500	1000	2023-04-03	P001
4	1000	2500	1500	2023-03-15	P003
5	900	1500	2000	2023-01-30	P003
6	1000	1500	3000	2023-12-16	P004

- Result Grid:** Shows the result of the `select * from bill;` query.
- SQL Additions:** A note states: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

3.2.28 Ward Table

The screenshot shows the MySQL Workbench interface with the database 'GP_10_Hospital_Information_S' selected. In the Navigator pane, under the 'hospital_information' schema, the 'Tables' node is expanded, showing the 'ward' table. The SQL editor pane contains the following code:

```
522 • create table ward
523   (Ward_number int not null,
524    Capacity int ,
525    Availability bool ,
526    department_id varchar(10) ,
527    foreign key(department_id) references department(department_id) on delete cascade on update cascade);
528
529 • INSERT INTO ward
530   VALUES
531   ('1,10', TRUE, 'DE01'),
532   ('1,20', FALSE, 'DE02'),
533   ('2,15', TRUE, 'DE03'),
534   ('2,12', TRUE, 'DE01'),
535   ('3,25', TRUE, 'DE02'),
536   ('3,18', TRUE, 'DE03');
537
538 • select * from ward;
539
```

The Result Grid pane displays the data inserted into the 'ward' table:

Ward_number	Capacity	Availability	department_id
1	10	1	DE01
1	20	0	DE02
2	15	1	DE03
2	12	1	DE01
3	25	1	DE02
3	18	1	DE03

3.2.29 Ward_staff_relation Table

The screenshot shows the MySQL Workbench interface with the database 'GP_10_Hospital_Information_S' selected. In the Navigator pane, under the 'hospital_information' schema, the 'Tables' node is expanded, showing the 'ward_staff_relation' table. The SQL editor pane contains the following code:

```
540 • create table ward_staff_relation
541   (Ward_number int not null,
542    Dept_id varchar(10) not null,
543    Staff_id varchar(50) not null,
544    primary key(Ward_number,Dept_id,Staff_id),
545    foreign key(Dept_id) references department(department_id) on delete cascade on update cascade,
546    foreign key(Staff_id) references Non_medical_staff(Staff_id) on delete cascade on update cascade);
547
548 • insert into ward_staff_relation values
549   ('1','DE02','S001'),
550   ('1','DE02','S002'),
551   ('2','DE01','S002'),
552   ('2','DE01','S003'),
553   ('3','DE02','S004'),
554   ('3','DE03','S005');
555
556 • select * from ward_staff_relation;
557
```

The Result Grid pane displays the data inserted into the 'ward_staff_relation' table:

Ward_number	Dept_id	Staff_id
2	DE01	S002
2	DE01	S003
1	DE02	S001
1	DE02	S002
3	DE02	S004
3	DE03	S005

3.2.30 Ward_nurse_relation Table

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Schemas Navigator:** Shows available schemas: foe, hos, ruhuna, sakila, sys, uop, uor, world, and GP_10_Hospital_Information_S.
- SQL Editor:** Displays the SQL code for creating the table and inserting data. The table structure is defined with columns: Ward_number (int), Dept_id (varchar(10)), and Nurse_id (varchar(50)). Primary key and foreign key constraints are also defined.
- Result Grid:** Shows the inserted data into the Ward_nurse_relation table. The data consists of 10 rows with the following values:

Ward_number	Dept_id	Nurse_id
2	DE01	N004
1	DE02	N001
1	DE02	N002
1	DE02	N003
3	DE02	N006
3	DE03	N005
1	DE02	N007
2	DE01	N008
1	DE02	N009

3.2.31 Ward_patient_relation Table

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Schemas Navigator:** Shows available schemas: foe, hos, ruhuna, sakila, sys, uop, uor, world, and GP_10_Hospital_Information_S.
- SQL Editor:** Displays the SQL code for creating the table and inserting data. The table structure is defined with columns: Ward_number (int), Dept_id (varchar(10)), and Patient_id (varchar(50)). Primary key and foreign key constraints are also defined.
- Result Grid:** Shows the inserted data into the Ward_patient_relation table. The data consists of 10 rows with the following values:

Ward_number	Dept_id	Patient_id
2	DE01	P004
2	DE01	P005
1	DE02	P001
1	DE02	P002
1	DE02	P003
3	DE03	P005
1	DE02	P006
2	DE01	P007
2	DE01	P008
3	DE03	P009

3.3 Alteration & deriving age from date of birth

3.3.1 Nurse Table

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Schemas:** Schemas pane showing databases: foe, hos, ruhuna, sakila, sys, uop, uor, world.
- Query Editor:** SQL Additions pane containing the following SQL code:

```
615  
616  
617  
618  
619 • Alter table Nurse  
620 add Age int;  
621  
622 • update Nurse  
623 set Age =Date_format(from_days(datediff(now(),Date_of_birth)), '%Y')+0;  
624 • select * FROM Nurse;
```
- Result Grid:** Result Grid pane showing the data in the Nurse table:

Nurse_id	First_name	Last_name	Date_of_birth	Gender	Age
N001	Mary	Johnson	1985-07-15	F	38
N002	Robert	Anderson	1990-02-20	M	33
N003	Jennifer	Smith	1988-11-05	F	34
N004	William	Brown	1982-09-10	M	40
N005	Jessica	Williams	1995-06-25	F	28
N006	Daniel	Davis	1998-03-18	F	25
- Toolbar:** Includes icons for Home, New Connection, Local instance MySQL80, Undo, Redo, Save, Print, Find, Replace, Copy, Paste, Cut, Delete, Refresh, Filter, Sort, Limit, Jump to, and Context Help.
- Right Panel:** Shows context help for the current caret position and a sidebar with tabs: Result Grid, Form Editor, Field Types, Query Stats, Execution Plan, Context Help, and Snippets.

3.4 Table Definitions

3.4.1 Doctor Table

The screenshot shows the MySQL Workbench interface with the 'Local instance MySQL80' connection selected. In the Navigator pane, under the 'hospital_information_system' schema, the 'Tables' node is expanded, showing various tables like app_reason, appointment, bill, department, etc. A context menu is open over the 'doctor' table, and the 'Edit Data for Create Table (VARCHAR)' option is selected. This opens a new window titled 'Edit Data for Create Table (VARCHAR)' with the SQL code for creating the 'doctor' table:

```
CREATE TABLE `doctor` (
  `Doctor_id` varchar(15) NOT NULL,
  `First_name` varchar(30) NOT NULL,
  `Last_name` varchar(30) NOT NULL,
  `Date_of_birth` date NOT NULL,
  `Street` varchar(50) DEFAULT NULL,
  `City` varchar(20) DEFAULT NULL,
  `Province` varchar(40) DEFAULT NULL,
  `Specialization` varchar(40) DEFAULT NULL,
  PRIMARY KEY (`Doctor_id`),
  UNIQUE KEY `Name_birth` (`First_name`, `Last_name`, `Date_of_birth`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

The 'Result Grid' tab shows the output of the SHOW CREATE TABLE Doctor command, which includes the table definition and its constraints.

3.4.2 Doctor_PhoneNumber Table

The screenshot shows the MySQL Workbench interface with the 'Local_Instance(1)' connection selected. In the Navigator pane, under the 'hospital_information_system' schema, the 'Tables' node is expanded, showing various tables like app_reason, appointment, bill, department, etc. A context menu is open over the 'doctor' table, and the 'Edit Data for Create Table (VARCHAR)' option is selected. This opens a new window titled 'Edit Data for Create Table (VARCHAR)' with the SQL code for creating the 'doctor_phonenumber' table:

```
CREATE TABLE `doctor_phonenumber` (
  `Doc_id` varchar(15) NOT NULL,
  `Phone_number` varchar(10) NOT NULL,
  PRIMARY KEY (`Doc_id`, `Phone_number`),
  CONSTRAINT `fkDoc` FOREIGN KEY (`Doc_id`) REFERENCES `doctor` (`Doctor_id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

The 'Result Grid' tab shows the output of the SHOW CREATE TABLE Doctor and SHOW CREATE TABLE Doctor_PhoneNumber commands.

3.4.3 Medicine Table

The screenshot shows the MySQL Workbench interface with the database 'GP_18_Hospital_Information_S...' selected. In the Navigator pane, under the 'hospital_information_system' schema, the 'Tables' section lists various tables including 'app_reason', 'appointment', 'bill', 'department', 'department_doc_relation', 'department_staff_relation', 'doctor', 'doctor_phonenumber', 'medicine', 'non_medical_staff', 'nurse', 'nurse_contact_relation', 'nursepat_relation', 'patient', 'patient_contact_relation', 'patient_doctor_relation', and 'patient_med_relation'. The central pane displays the SQL code for creating the 'medicine' table:

```
CREATE TABLE `medicine` (
  `Medicine_id` varchar(30) NOT NULL,
  `Medicine_name` varchar(25) DEFAULT NULL,
  `Price` int NOT NULL,
  `Description` varchar(100) DEFAULT NULL,
  `Quantity_in_mg` int NOT NULL,
  PRIMARY KEY (`Medicine_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

The 'Result Grid' shows the execution of several 'SHOW CREATE TABLE' statements, and the 'Output' pane shows the results of these statements along with the creation of the 'medicine' table.

3.4.4 Prescription Table

The screenshot shows the MySQL Workbench interface with the database 'GP_18_Hospital_Information_S...' selected. In the Navigator pane, under the 'hospital_information_system' schema, the 'Tables' section lists various tables including 'app_reason', 'appointment', 'bill', 'department', 'department_doc_relation', 'department_staff_relation', 'doctor', 'doctor_phonenumber', 'medicine', 'non_medical_staff', 'nurse', 'nurse_contact_relation', 'nursepat_relation', 'patient', 'patient_contact_relation', 'patient_doctor_relation', and 'patient_med_relation'. The central pane displays the SQL code for creating the 'prescription' table:

```
CREATE TABLE `prescription` (
  `Prescription_id` varchar(100) NOT NULL,
  `Dosage` int DEFAULT NULL,
  `No_of_days` int DEFAULT NULL,
  `Prescribed_date` date DEFAULT NULL,
  `Doc_id` varchar(15) NOT NULL DEFAULT 'D01',
  PRIMARY KEY (`Prescription_id`),
  KEY `fk_PresDoc` (`Doc_id`),
  CONSTRAINT `fk_PresDoc` FOREIGN KEY (`Doc_id`) REFERENCES `doctor` (`Doctor_id`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

The 'Result Grid' shows the execution of several 'SHOW CREATE TABLE' statements, and the 'Output' pane shows the results of these statements along with the creation of the 'prescription' table.

3.4.5 Pres_Medi_Relation Table

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

- Schemas:** Local_Instance(1) is selected.
- Tables:** The `hospital_information_system` schema is expanded, showing tables like `app_reason`, `appointment`, `bill`, `department`, `department_doc_relation`, `department_staff_relation`, `doctor`, `doctor_phonenumber`, `medicine`, `non_medical_staff`, `nurse`, `nurse_contact_relation`, `nursepat_relation`, `patient`, `patient_contact_relation`, `patient_doctor_relation`, and `patient_med_relation`.
- Query Editor:** A query titled "GP_18_Hospital_Information_S..." is running, showing the creation of the `Pres_Medi_Relation` table. The code is as follows:

```

1 CREATE TABLE `pres_medi_relation` (
2   `Prescription_id` varchar(100) NOT NULL,
3   `Med_id` varchar(30) NOT NULL,
4   PRIMARY KEY (`Prescription_id`, `Med_id`),
5   KEY `fk_medic` (`Med_id`),
6   CONSTRAINT `fk_pres` FOREIGN KEY (`Med_id`) REFERENCES `medicine` (`Medicine_id`) ON DELETE CASCADE ON UPDATE CASCADE,
7   CONSTRAINT `fk_pres` FOREIGN KEY (`Prescription_id`) REFERENCES `prescription` (`Prescription_id`) ON DELETE CASCADE ON UPDATE CASCADE
8 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

```

- Result Grid:** Shows the execution of the query, returning 43 rows.
- Action Output:** Shows the execution history with the following details:

#	Time	Action	Data Length
109	11:43:37	SHOW CREATE TABLE Doctor	506 bytes
110	11:44:50	SHOW CREATE TABLE Doctor_Phonenumber	
111	11:45:56	SHOW CREATE TABLE Medicine	
112	11:46:41	SHOW CREATE TABLE Prescription	
113	11:47:27	SHOW CREATE TABLE Pres_Medi_Relation	

3.4.6 Non_medical_staff Table

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

- Schemas:** Local instance MySQL80 is selected.
- Tables:** The `hospital_information_system` schema is expanded, showing tables like `app_reason`, `appointment`, `bill`, `department`, `department_doc_relation`, `department_staff_relation`, `doctor`, `doctor_phonenumber`, `medicine`, `non_medical_staff`, `nurse`, `nurse_contact_relation`, `nursepat_relation`, and `patient`.
- Query Editor:** A query titled "recorrection_3" is running, showing the creation of the `Non_medical_staff` table. The code is as follows:

```

1 CREATE TABLE `non_medical_staff` (
2   `Staff_id` varchar(10) NOT NULL,
3   `Manager_id` varchar(10) DEFAULT NULL,
4   `First_name` varchar(30) NOT NULL,
5   `Last_name` varchar(30) NOT NULL,
6   `Date_of_birth` date NOT NULL,
7   `Gender` varchar(1) DEFAULT NULL,
8   `Job_title` varchar(30) DEFAULT NULL,
9   `Street` varchar(50) DEFAULT NULL,
10  `City` varchar(20) DEFAULT NULL,
11  `Province` varchar(40) DEFAULT NULL,
12  PRIMARY KEY (`Staff_id`),
13  UNIQUE KEY `Name_birth` (`First_name`, `Last_name`, `Date_of_birth`),
14  KEY `fk_staff` (`Manager_id`),
15  CONSTRAINT `fk_staff` FOREIGN KEY (`Manager_id`) REFERENCES `non_medical_staff` (`Staff_id`)
16 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

```

- Result Grid:** Shows the execution of the query, returning 194 rows.
- Action Output:** Shows the execution history with the following details:

#	Time	Action	Data Length
463	15:41:18	SHOW CREATE TABLE Medicine	696 bytes
464	15:41:53	SHOW CREATE TABLE Prescription	
465	15:42:34	SHOW CREATE TABLE Pres_Medi_Relation	
466	15:43:38	SHOW CREATE TABLE Non_medical_staff	

3.4.7 Staff_contact_relation Table

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

- Schemas:** Local_Instance(1) is selected.
- Tables:** The `hospital_information_system` schema is expanded, showing tables like `app_reason`, `appointment`, `bill`, `department`, `department_doc_relation`, `department_staff_relation`, `doctor`, `doctor_phonenumber`, `medicine`, `non_medical_staff`, `nurse`, `nurse_contact_relation`, `nursepat_relation`, `patient`, `patient_contact_relation`, `patient_doctor_relation`, and `patient_med_relation`.
- Query Editor:** The query `CREATE TABLE `staff_contact_relation` (`Staff_id` varchar(10) NOT NULL, `Phone_number` varchar(10) NOT NULL, PRIMARY KEY (`Staff_id`), `Phone_number`), CONSTRAINT `fkStaff` FOREIGN KEY (`Staff_id`) REFERENCES `non_medical_staff` (`Staff_id`) ON DELETE CASCADE ON UPDATE CASCADE) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci` is being edited in the Text tab.
- Result Grid:** Shows the execution of the query, returning 1 row(s) returned.
- Output:** Shows the execution history with entries like "SHOW CREATE TABLE Medicine", "SHOW CREATE TABLE Prescription", etc.

3.4.8 Department Table

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

- Schemas:** Local_Instance(1) is selected.
- Tables:** The `hospital_information_system` schema is expanded, showing tables like `app_reason`, `appointment`, `bill`, `department`, `department_doc_relation`, `department_staff_relation`, `doctor`, `doctor_phonenumber`, `medicine`, `non_medical_staff`, `nurse`, `nurse_contact_relation`, `nursepat_relation`, `patient`, `patient_contact_relation`, `patient_doctor_relation`, and `patient_med_relation`.
- Query Editor:** The query `CREATE TABLE `department` (`Department_id` varchar(10) NOT NULL, `Department_name` varchar(20) DEFAULT NULL, `Head_id` varchar(15) NOT NULL, PRIMARY KEY (`Department_id`), KEY `fk_dep` (`Head_id`), CONSTRAINT `fk_dep` FOREIGN KEY (`Head_id`) REFERENCES `doctor` (`Doctor_id`) ON DELETE CASCADE ON UPDATE CASCADE) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci` is being edited in the Text tab.
- Result Grid:** Shows the execution of the query, returning 1 row(s) returned.
- Output:** Shows the execution history with entries like "SHOW CREATE TABLE Prescription", "SHOW CREATE TABLE Pres_Med_Relation", etc.

3.4.9 Department_Staff_Relation Table

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

- Schemas:** Local_Instance(1) is selected.
- Tables:** The `hospital_information_system` schema is expanded, showing tables like `app_reason`, `appointment`, `bill`, `department`, `department_doc_relation`, `department_staff_relation`, `doctor`, `doctor_phonenumber`, `medicine`, `non_medical_staff`, `nurse`, `nurse_contact_relation`, `nursepat_relation`, `patient`, `patient_contact_relation`, `patient_doctor_relation`, and `patient_med_relation`.
- Query Editor:** The query `SHOW CREATE TABLE Department_Staff_Relation ;` is run, and the results are displayed in the "Edit Data for Create Table (VARCHAR)" pane.
- Output Window:** The "Result 47" window shows the execution history of the command, with a data length of 501 bytes.
- System Bar:** The taskbar at the bottom shows various application icons and the system clock at 11:50 AM.

3.4.10 Department_Doc_Relation Table

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

- Schemas:** Local_Instance(1) is selected.
- Tables:** The `hospital_information_system` schema is expanded, showing tables like `app_reason`, `appointment`, `bill`, `department`, `department_doc_relation`, `department_staff_relation`, `doctor`, `doctor_phonenumber`, `medicine`, `non_medical_staff`, `nurse`, `nurse_contact_relation`, `nursepat_relation`, `patient`, `patient_contact_relation`, `patient_doctor_relation`, and `patient_med_relation`.
- Query Editor:** The query `SHOW CREATE TABLE Department_Doc_Relation ;` is run, and the results are displayed in the "Edit Data for Create Table (VARCHAR)" pane.
- Output Window:** The "Result 48" window shows the execution history of the command, with a data length of 479 bytes.
- System Bar:** The taskbar at the bottom shows various application icons and the system clock at 11:51 AM.

3.4.11 Test_report Table

The screenshot shows the MySQL Workbench interface with the database 'GP_18_Hospital_Information_S' selected. In the 'Edit Data for Create Table (VARCHAR)' window, the SQL code for creating the 'test_report' table is displayed:

```
CREATE TABLE `test_report` (
  `Test_id` varchar(10) NOT NULL,
  `Test_name` varchar(50) DEFAULT NULL,
  `Tested_date` date DEFAULT NULL,
  `Tested_time` time DEFAULT NULL,
  `result` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`Test_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

The 'Result Grid' shows the execution of the CREATE TABLE command, returning 1 row(s). The 'Output' pane displays the execution log with the following entries:

#	Time	Action	Data Length
115	11:49:18	SHOW CREATE TABLE Staff_contact_Relation	0.000 sec / 0.000 sec
116	11:50:02	SHOW CREATE TABLE Department	1row(s) returned 0.000 sec / 0.000 sec
117	11:50:34	SHOW CREATE TABLE Department_Staff_Relation	1row(s) returned 0.000 sec / 0.000 sec
118	11:51:07	SHOW CREATE TABLE Department_Doc_Relation	1row(s) returned 0.000 sec / 0.000 sec
119	11:51:42	SHOW CREATE TABLE Test_report	1row(s) returned 0.000 sec / 0.000 sec

3.4.12 Patient Table

The screenshot shows the MySQL Workbench interface with the database 'GP_18_Hospital_Information_S' selected. In the 'Edit Data for Create Table (VARCHAR)' window, the SQL code for creating the 'patient' table is displayed:

```
CREATE TABLE `patient` (
  `patient_id` varchar(20) NOT NULL,
  `First_name` varchar(30) NOT NULL,
  `Last_name` varchar(30) NOT NULL,
  `Gender` varchar(1) DEFAULT NULL,
  `Age` int DEFAULT NULL,
  `Street` varchar(50) DEFAULT NULL,
  `City` varchar(20) DEFAULT NULL,
  `Province` varchar(40) DEFAULT NULL,
  PRIMARY KEY (`patient_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

The 'Result Grid' shows the execution of the CREATE TABLE command, returning 1 row(s). The 'Output' pane displays the execution log with the following entries:

#	Time	Action	Data Length
116	11:50:02	SHOW CREATE TABLE Department	0.000 sec / 0.000 sec
117	11:50:34	SHOW CREATE TABLE Department_Staff_Relation	1row(s) returned 0.000 sec / 0.000 sec
118	11:51:07	SHOW CREATE TABLE Department_Doc_Relation	1row(s) returned 0.000 sec / 0.000 sec
119	11:51:42	SHOW CREATE TABLE Test_report	1row(s) returned 0.000 sec / 0.000 sec
120	11:52:23	SHOW CREATE TABLE Patient	1row(s) returned 0.000 sec / 0.000 sec

3.4.13 Patient_Doctor_relation Table

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

- Schemas:** Local_Instance(1) is selected.
- Tables:** The `hospital_information_system` schema is expanded, showing tables like `app_reason`, `appointment`, `bill`, `department`, `department_doc_relation`, `department_staff_relation`, `doctor`, `doctor_phonenumber`, `medicine`, `non_medical_staff`, `nurse`, `nurse_contact_relation`, `nursespat_relation`, `patient`, `patient_contact_relation`, `patient_doctor_relation`, and `patient_med_relation`.
- Query Editor:** The query `SHOW CREATE TABLE Patient ;` is run, followed by `SHOW CREATE TABLE Patient_Doctor_relation ;`. The results show the table structure and constraints.
- Result Grid:** Shows the creation of the `Patient_Doctor_relation` table.
- Output Window:** Displays the execution history with actions like `SHOW CREATE TABLE` and `CREATE TABLE`.
- System Bar:** Shows the taskbar with various icons and the system clock at 11:53 AM.

3.4.14 Patient_Med_relation Table

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

- Schemas:** Local_Instance(1) is selected.
- Tables:** The `hospital_information_system` schema is expanded, showing tables like `app_reason`, `appointment`, `bill`, `department`, `department_doc_relation`, `department_staff_relation`, `doctor`, `doctor_phonenumber`, `medicine`, `non_medical_staff`, `nurse`, `nurse_contact_relation`, `nursespat_relation`, `patient`, `patient_contact_relation`, `patient_doctor_relation`, and `patient_med_relation`.
- Query Editor:** The query `SHOW CREATE TABLE Patient_Doctor_relation ;` is run, followed by `SHOW CREATE TABLE Patient_Med_relation ;`. The results show the table structure and constraints.
- Result Grid:** Shows the creation of the `Patient_Med_relation` table.
- Output Window:** Displays the execution history with actions like `SHOW CREATE TABLE` and `CREATE TABLE`.
- System Bar:** Shows the taskbar with various icons and the system clock at 11:54 AM.

3.4.15 Patient_contact_relation Table

The screenshot shows the MySQL Workbench interface with the 'GP_18_Hospital_Information_S...' database selected. In the 'Tables' section of the Navigator, the 'patient_contact_relation' table is highlighted. A context menu is open over the table, with the 'Edit Data for Create Table (VARCHAR)' option selected. This opens a new window titled 'Edit Data for Create Table (VARCHAR)' containing the SQL code for creating the table:

```
CREATE TABLE `patient_contact_relation` (
  `Pat_id` varchar(10) NOT NULL,
  `Phone_number` varchar(10) NOT NULL,
  PRIMARY KEY (`Pat_id`, `Phone_number`),
  CONSTRAINT `fkPC` FOREIGN KEY (`Pat_id`) REFERENCES `patient` (`patient_id`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Below this window, the 'Result Grid' shows the execution history of the commands:

#	Time	Action	Rows(s) returned	Time
119	11:51:42	SHOW CREATE TABLE Test_report	1	0.000 sec / 0.000 sec
120	11:52:23	SHOW CREATE TABLE Patient	1	0.000 sec / 0.000 sec
121	11:53:10	SHOW CREATE TABLE Patient_Doctor	1	0.000 sec / 0.000 sec
122	11:53:45	SHOW CREATE TABLE Patient_Med_relation	1	0.000 sec / 0.000 sec
123	11:55:34	SHOW CREATE TABLE Patient_contact_relation	1	0.000 sec / 0.000 sec

3.4.16 Patient_Test_relation Table

The screenshot shows the MySQL Workbench interface with the 'GP_18_Hospital_Information_S...' database selected. In the 'Tables' section of the Navigator, the 'Patient_Test_relation' table is highlighted. A context menu is open over the table, with the 'Edit Data for Create Table (VARCHAR)' option selected. This opens a new window titled 'Edit Data for Create Table (VARCHAR)' containing the SQL code for creating the table:

```
CREATE TABLE `patient_test_relation` (
  `Patient_id` varchar(20) NOT NULL,
  `Test_id` varchar(15) NOT NULL,
  PRIMARY KEY (`Patient_id`, `Test_id`),
  KEY `fk_PTest` (`Test_id`),
  CONSTRAINT `fk_PTest` FOREIGN KEY (`Test_id`) REFERENCES `test_report` (`Test_id`) ON DELETE CASCADE
  ON UPDATE CASCADE,
  CONSTRAINT `fk_TPat` FOREIGN KEY (`Patient_id`) REFERENCES `patient` (`patient_id`) ON DELETE CASCADE
  ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Below this window, the 'Result Grid' shows the execution history of the commands:

#	Time	Action	Rows(s) returned	Time
120	11:52:23	SHOW CREATE TABLE Patient	1	0.000 sec / 0.000 sec
121	11:53:10	SHOW CREATE TABLE Patient_Doctor	1	0.000 sec / 0.000 sec
122	11:53:45	SHOW CREATE TABLE Patient_Med_relation	1	0.000 sec / 0.000 sec
123	11:55:34	SHOW CREATE TABLE Patient_contact_relation	1	0.000 sec / 0.000 sec
124	11:56:10	SHOW CREATE TABLE Patient_Test_relation	1	0.000 sec / 0.000 sec

3.4.17 Patient_Prescription_relation Table

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

- Schemas:** Local_Instance(1) is selected.
- Tables:** The `hospital_information_system` schema is expanded, showing tables like `app_reason`, `appointment`, etc.
- Query Editor:** The query `SHOW CREATE TABLE Patient_Prescription_relation ;` is run, displaying the table's definition.
- Result Grid:** The result shows the table structure and constraints.
- Output Window:** Shows the execution history of the commands.
- System Bar:** Displays the date and time as 11:57 AM.

```
CREATE TABLE `patient_prescription_relation` (
  `Patient_id` varchar(20) NOT NULL,
  `Pres_id` varchar(15) NOT NULL,
  PRIMARY KEY (`Patient_id`, `Pres_id`),
  KEY `fk_PPres` (`Pres_id`),
  CONSTRAINT `fk_PPat` FOREIGN KEY (`Patient_id`) REFERENCES `patient` (`patient_id`) ON DELETE CASCADE,
  CONSTRAINT `fk_PPres` FOREIGN KEY (`Pres_id`) REFERENCES `prescription` (`Prescription_id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

3.4.18 Appointment Table

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

- Schemas:** Local_Instance(1) is selected.
- Tables:** The `hospital_information_system` schema is expanded, showing tables like `app_reason`, `appointment`, etc.
- Query Editor:** The query `SHOW CREATE TABLE Appointment ;` is run, displaying the table's definition.
- Result Grid:** The result shows the table structure and constraints.
- Output Window:** Shows the execution history of the commands.
- System Bar:** Displays the date and time as 11:57 AM.

```
CREATE TABLE `appointment` (
  `Appointment_date` date NOT NULL,
  `Appointment_time` time NOT NULL,
  `current_status` varchar(20) DEFAULT NULL,
  `patient_id` varchar(30) NOT NULL,
  KEY `fk_PatApp` (`patient_id`),
  CONSTRAINT `fk_PatApp` FOREIGN KEY (`patient_id`) REFERENCES `patient` (`patient_id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

3.4.19 App_reason Table

The screenshot shows the MySQL Workbench interface with the 'GP_18_Hospital_Information_S...' database selected. In the 'Edit Data for Create Table (VARCHAR)' window, the SQL code for creating the 'app_reason' table is displayed:

```
CREATE TABLE `app_reason` (
  `Appointment_date` date NOT NULL,
  `Appointment_time` time NOT NULL,
  `patient_id` varchar(30) NOT NULL,
  `reason` varchar(30) NOT NULL,
  KEY `fk_AppReason` (`patient_id`),
  CONSTRAINT `fk_AppReason` FOREIGN KEY (`patient_id`) REFERENCES `patient` (`patient_id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

The 'Result Grid' shows the execution of the CREATE TABLE command, returning 1 row(s) returned. The 'Object Info' tab shows the table structure with columns: Appointment_date, Appointment_time, patient_id, and reason.

3.4.20 Record Table

The screenshot shows the MySQL Workbench interface with the 'GP_18_Hospital_Information_S...' database selected. In the 'Edit Data for Create Table (VARCHAR)' window, the SQL code for creating the 'record' table is displayed:

```
CREATE TABLE `record` (
  `Record_id` varchar(50) NOT NULL,
  `Date_admitted` date DEFAULT NULL,
  `Date_discharged` date DEFAULT NULL,
  `Patient_id` varchar(100) NOT NULL DEFAULT 'P004',
  PRIMARY KEY (`Record_id`),
  KEY `fkP_rec` (`Patient_id`),
  CONSTRAINT `fkP_rec` FOREIGN KEY (`Patient_id`) REFERENCES `patient` (`patient_id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

The 'Result Grid' shows the execution of the CREATE TABLE command, returning 1 row(s) returned. The 'Object Info' tab shows the table structure with columns: Record_id, Date_admitted, Date_discharged, and Patient_id.

3.4.21 Record_Doctor_relation Table

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

- Schemas:** Local_Instance(1) is selected.
- Tables:** The `hospital_information_system` schema is expanded, showing tables like `app_reason`, `appointment`, `bill`, etc.
- Query Editor:** The query `SHOW CREATE TABLE Record_Doctor_relation ;` is run, displaying the table's definition.
- Result Grid:** The result shows the table creation command.
- Output Window:** Shows the execution log with entries for creating the table and other related tables.
- System Bar:** Shows the date and time as 11:59 AM.

```
CREATE TABLE `record_doctor_relation` (
  `Record_id` varchar(20) NOT NULL,
  `Doctor_id` varchar(15) NOT NULL,
  PRIMARY KEY (`Record_id`, `Doctor_id`),
  KEY `fk_ReDoc` (`Doctor_id`),
  CONSTRAINT `fk_ReDoc` FOREIGN KEY (`Doctor_id`) REFERENCES `doctor` (`Doctor_id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fkRec` FOREIGN KEY (`Record_id`) REFERENCES `record` (`Record_id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

3.4.22 Record_Test_relation Table

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

- Schemas:** Local_Instance(1) is selected.
- Tables:** The `hospital_information_system` schema is expanded, showing tables like `app_reason`, `appointment`, `bill`, etc.
- Query Editor:** The query `SHOW CREATE TABLE Record_Test_relation ;` is run, displaying the table's definition.
- Result Grid:** The result shows the table creation command.
- Output Window:** Shows the execution log with entries for creating the table and other related tables.
- System Bar:** Shows the date and time as 12:00 PM.

```
CREATE TABLE `record_test_relation` (
  `Rec_id` varchar(20) NOT NULL,
  `Test_id` varchar(15) NOT NULL,
  PRIMARY KEY (`Rec_id`, `Test_id`),
  KEY `fk_ReTest` (`Test_id`),
  CONSTRAINT `fk_ReTest` FOREIGN KEY (`Test_id`) REFERENCES `test_report` (`Test_id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_TRec` FOREIGN KEY (`Rec_id`) REFERENCES `record` (`Record_id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

3.4.23 Record_Med_relation Table

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

- Schemas:** Local_Instance(1) is selected.
- Tables:** The `hospital_information_system` schema is expanded, showing tables like `app_reason`, `appointment`, etc.
- Query Editor:** A script window titled "GP_18_Hospital_Information_S..." contains the following SQL code:

```
621 • SHOW CREATE TABLE Record_Test_relation ;
622 • SHOW CREATE TABLE Record_Med_relation ;
623
624
625
626
627
628
629
630
631
```
- Create Table Window:** An "Edit Data for Create Table (VARCHAR)" window is open, showing the definition for the `Record_Med_relation` table:

```
1 CREATE TABLE `record_med_relation` (
2   `Record_id` varchar(20) NOT NULL,
3   `Med_id` varchar(15) NOT NULL,
4   PRIMARY KEY (`Record_id`,`Med_id`),
5   KEY `fk_RecMed` (`Med_id`),
6   CONSTRAINT `fk_RecMed` FOREIGN KEY (`Record_id`) REFERENCES `record` (`Record_id`) ON DELETE CASCADE,
7   CONSTRAINT `fk_RecMed` FOREIGN KEY (`Med_id`) REFERENCES `medicine` (`Medicine_id`) ON DELETE CASCADE ON UPDATE CASCADE
8 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```
- Result Grid:** A "Result 61" grid shows the execution of the create statements, with a total data length of 482 bytes.
- Session Bar:** Shows the session information and a system tray at the bottom.

3.4.24 Nurse Table

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

- Schemas:** Local_Instance(1) is selected.
- Tables:** The `hospital_information_system` schema is expanded, showing tables like `app_reason`, `appointment`, etc.
- Query Editor:** A script window titled "GP_18_Hospital_Information_S..." contains the following SQL code:

```
622 • SHOW CREATE TABLE Record_Med_relation ;
623 • SHOW CREATE TABLE Nurse ;
624
625
626
627
628
629
630
631
```
- Create Table Window:** An "Edit Data for Create Table (VARCHAR)" window is open, showing the definition for the `Nurse` table:

```
1 CREATE TABLE `nurse` (
2   `Nurse_id` varchar(15) NOT NULL,
3   `First_name` varchar(30) NOT NULL,
4   `Last_name` varchar(30) NOT NULL,
5   `Date_of_birth` date NOT NULL DEFAULT '2000-04-12',
6   `Gender` varchar(1) DEFAULT 'F',
7   `Age` int DEFAULT NULL,
8   PRIMARY KEY (`Nurse_id`),
9 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```
- Result Grid:** A "Result 62" grid shows the execution of the create statements, with a total data length of 339 bytes.
- Session Bar:** Shows the session information and a system tray at the bottom.

3.4.25 Nurse_contact_relation Table

The screenshot shows the MySQL Workbench interface with the database 'GP_18_Hospital_Information_S...' selected. In the 'Edit Data for Create Table (VARCHAR)' dialog, the SQL code for creating the 'nurse_contact_relation' table is displayed:

```
CREATE TABLE `nurse_contact_relation` (
  `Nurse_id` varchar(10) NOT NULL,
  `Phone_number` varchar(10) NOT NULL,
  PRIMARY KEY (`Nurse_id`, `Phone_number`),
  CONSTRAINT `fkNC` FOREIGN KEY (`Nurse_id`) REFERENCES `nurse` (`Nurse_id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

The 'Result 63' tab shows the execution history of the commands:

#	Time	Action
129	11:59:40	SHOW CREATE TABLE Record_Doc...
130	12:00:26	SHOW CREATE TABLE Record_Test...
131	12:01:07	SHOW CREATE TABLE Record_Med...
132	12:01:46	SHOW CREATE TABLE Nurse
133	12:02:20	SHOW CREATE TABLE Nurse_contact_relation

3.4.26 NursePat_relation Table

The screenshot shows the MySQL Workbench interface with the database 'GP_18_Hospital_Information_S...' selected. In the 'Edit Data for Create Table (VARCHAR)' dialog, the SQL code for creating the 'nursepat_relation' table is displayed:

```
CREATE TABLE `nursepat_relation` (
  `Nurse_id` varchar(20) NOT NULL,
  `Pat_id` varchar(15) NOT NULL,
  PRIMARY KEY (`Nurse_id`, `Pat_id`),
  KEY `fk_NuPat` (`Pat_id`),
  CONSTRAINT `fk_NuPat` FOREIGN KEY (`Pat_id`) REFERENCES `patient` (`patient_id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_PNu` FOREIGN KEY (`Nurse_id`) REFERENCES `nurse` (`Nurse_id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

The 'Result 64' tab shows the execution history of the commands:

#	Time	Action
130	12:00:26	SHOW CREATE TABLE Record_Test...
131	12:01:07	SHOW CREATE TABLE Record_Med...
132	12:01:46	SHOW CREATE TABLE Nurse
133	12:02:20	SHOW CREATE TABLE Nurse_contact_relation
134	12:02:55	SHOW CREATE TABLE NursePat_relation

3.4.27 Bill Table

The screenshot shows the MySQL Workbench interface with the database 'GP_18_Hospital_Information_S...' selected. In the 'Edit Data for Create Table (VARCHAR)' window, the 'Text' tab displays the SQL code for creating the 'bill' table:

```
CREATE TABLE `bill` (
  `Bill_number` int NOT NULL AUTO_INCREMENT,
  `Ward_charge` int DEFAULT NULL,
  `Medicine_charge` int NOT NULL,
  `Test_charge` int DEFAULT NULL,
  `bill_date` date DEFAULT NULL,
  `patient_id` varchar(10) DEFAULT NULL,
  PRIMARY KEY (`Bill_number`),
  KEY `patient_id` (`patient_id`),
  CONSTRAINT `bill_ibfk_1` FOREIGN KEY (`patient_id`) REFERENCES `patient` (`patient_id`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

The 'Result 65' window shows the execution history with five entries related to table creation:

#	Time	Action	Data Length
131	12:01:07	SHOW CREATE TABLE Record_Med...	518 bytes
132	12:01:46	SHOW CREATE TABLE Nurse	1 row(s) returned
133	12:02:20	SHOW CREATE TABLE Nurse_contact_relation	1 row(s) returned
134	12:02:55	SHOW CREATE TABLE NursePat_relation	1 row(s) returned
135	12:04:35	SHOW CREATE TABLE Bill	1 row(s) returned

3.4.28 Ward Table

The screenshot shows the MySQL Workbench interface with the database 'GP_18_Hospital_Information_S...' selected. In the 'Edit Data for Create Table (VARCHAR)' window, the 'Text' tab displays the SQL code for creating the 'ward' table:

```
CREATE TABLE `ward` (
  `Ward_number` int NOT NULL,
  `Capacity` int DEFAULT NULL,
  `Availability` tinyint(1) DEFAULT NULL,
  `department_id` varchar(10) DEFAULT NULL,
  KEY `department_id` (`department_id`),
  CONSTRAINT `ward_ibfk_1` FOREIGN KEY (`department_id`) REFERENCES `department` (`Department_id`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

The 'Result 66' window shows the execution history with six entries related to table creation:

#	Time	Action	Data Length
132	12:01:46	SHOW CREATE TABLE Nurse	411 bytes
133	12:02:20	SHOW CREATE TABLE Nurse_contact_relation	1 row(s) returned
134	12:02:55	SHOW CREATE TABLE NursePat_relation	1 row(s) returned
135	12:04:35	SHOW CREATE TABLE Bill	1 row(s) returned
136	12:05:13	SHOW CREATE TABLE Ward	1 row(s) returned

3.4.29 Ward_staff_relation Table

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

- Schemas:** Local_Instance(1) is selected.
- Tables:** The `hospital_information_system` schema is expanded, showing tables like `app_reason`, `appointment`, `bill`, `department`, etc.
- Query Editor:** The query `SHOW CREATE TABLE Ward_staff_relation` is run, and the results are displayed in the Result Grid.
- Create Table Dialog:** A modal window titled "Edit Data for Create Table (VARCHAR)" shows the SQL code for creating the `ward_staff_relation` table. The code includes columns `Ward_number`, `Dept_id`, and `Staff_id`, along with various constraints and engine settings.
- Output Window:** The Action Output section shows the execution history of the queries, including the creation of the `Ward_staff_relation` table.

3.4.30 Ward_nurse_relation Table

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

- Schemas:** Local_Instance(1) is selected.
- Tables:** The `hospital_information_system` schema is expanded, showing tables like `app_reason`, `appointment`, `bill`, `department`, etc.
- Query Editor:** The query `SHOW CREATE TABLE Ward_nurse_relation` is run, and the results are displayed in the Result Grid.
- Create Table Dialog:** A modal window titled "Edit Data for Create Table (VARCHAR)" shows the SQL code for creating the `ward_nurse_relation` table. The code includes columns `Ward_number`, `Dept_id`, and `Nurse_id`, along with various constraints and engine settings.
- Output Window:** The Action Output section shows the execution history of the queries, including the creation of the `Ward_nurse_relation` table.

3.4.31 Ward_patient_relation Table

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

- Schemas:** Local_Instance(1) is selected.
- Tables:** The `hospital_information_system` schema is expanded, showing tables like `app_reason`, `appointment`, `bill`, `department`, `department_doc_relation`, `department_staff_relation`, `doctor`, `doctor_phonenumber`, `medicine`, `non_medical_staff`, `nurse`, `nurse_contact_relation`, `nursepat_relation`, `patient`, `patient_contact_relation`, `patient_doctor_relation`, and `patient_med_relation`.
- Query Editor:** The query `CREATE TABLE `ward_patient_relation` (` is being typed into the main query window.
- Result Grid:** Shows the execution history of previous commands, including:
 - 135 12:04:35 SHOW CREATE TABLE Bill
 - 136 12:05:13 SHOW CREATE TABLE Ward
 - 137 12:06:00 SHOW CREATE TABLE Ward_staff_re
 - 138 12:06:50 SHOW CREATE TABLE Ward_nurse_relati
 - 139 12:07:19 SHOW CREATE TABLE Ward_patient_relati
- Output:** Action Output panel showing the creation of the `Ward_patient_relation` table.
- Information:** Shows the `app_reason` table definition.
- Object Info:** Shows the session information.

3.5 Updating & Deleting data of tables

3.5.1 Doctor Table

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

- Schemas:** Local instance MySQL80, hospital_information_system.
- Query Editor:** Contains the following SQL code:

```
-- Updating and deleting data for the tables:
SELECT * FROM DOCTOR;
UPDATE Doctor SET City = "Kilinochchi" WHERE Doctor_id = "D01";
UPDATE Doctor SET City = "Kadawatha" WHERE Doctor_id = "D02";
DELETE FROM Doctor WHERE Doctor_id = "D05";
SELECT * FROM DOCTORS;
```
- Result Grid:** Displays the Doctor table data with 6 rows.

Doctor_id	First_name	Last_name	Date_of_birth	Street	City	Province	Specialization
D01	Sanath	Kumara	1980-10-28	Clock tower junction	Kilinochchi	Northern	Orthopedics
D02	Lahiru	Kumara	1980-10-28	Kurumankadu	Kadawatha	Western	Pediatrics
D03	Madushan	Lahiru	1975-09-20	Wakwella	Galle	Southern	Neurology
D04	Madushan	Thilanka	1975-09-20	Wellawatta	Colombo	Western	Dermatology
D06	Madushan	Thilanka	1972-03-18	Kithul Road	Hatton	Central	Cardiology
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
- Output:** Shows the execution log for the actions performed on the Doctor table.

3.5.2 Doctor_PhoneNumber

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

- Schemas:** Local instance MySQL80, hospital_information_system.
- Query Editor:** Contains the same SQL code as the previous screenshot:

```
-- Updating and deleting data for the tables:
SELECT * FROM DOCTOR;
UPDATE Doctor SET City = "Kilinochchi" WHERE Doctor_id = "D01";
UPDATE Doctor SET City = "Kadawatha" WHERE Doctor_id = "D02";
DELETE FROM Doctor WHERE Doctor_id = "D05";
SELECT * FROM DOCTORS;
```
- Result Grid:** Displays the Doctor table data with 6 rows.

Doctor_id	First_name	Last_name	Date_of_birth	Street	City	Province	Specialization
D01	Sanath	Kumara	1980-10-28	Clock tower junction	Kilinochchi	Northern	Orthopedics
D02	Lahiru	Kumara	1980-10-28	Kurumankadu	Kadawatha	Western	Pediatrics
D03	Madushan	Lahiru	1975-09-20	Wakwella	Galle	Southern	Neurology
D04	Madushan	Thilanka	1975-09-20	Wellawatta	Colombo	Western	Dermatology
D06	Madushan	Thilanka	1972-03-18	Kithul Road	Hatton	Central	Cardiology
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
- Output:** Shows the execution log for the actions performed on the Doctor table.

3.5.3 Medicine

The screenshot shows the MySQL Workbench interface with the database set to 'Local instance MySQL80'. In the 'Navigator' pane, the schema 'hospital_information_system' is selected. A query editor window titled 'rerection_3' contains the following SQL code:

```
622 -- Updating and deleting data for the tables:
623
624 • SELECT * FROM DOCTOR;
625 • UPDATE Doctor SET City = "Kilinochchi" WHERE Doctor_id = "D01";
626 • UPDATE Doctor SET City = "Kadawatha" WHERE Doctor_id = "D02";
627 • DELETE FROM Doctor WHERE Doctor_id = "D05";
628 • SELECT * FROM DOCTORS;
629
```

The 'Result Grid' shows the following data for the DOCTOR table:

Doctor_id	First_name	Last_name	Date_of_birth	Street	City	Province	Specialization
D01	Sanath	Kumara	1980-10-28	Clock tower junction	Kilinochchi	Northern	Orthopedics
D02	Lahiru	Kumara	1980-10-28	Kurumankadu	Kadawatha	Western	Pediatrics
D03	Madushan	Lahiru	1975-09-20	Wakwella	Galle	Southern	Neurology
D04	Madushan	Thilanka	1975-09-20	Welavatta	Colombo	Western	Dermatology
D06	Madushan	Thilanka	1972-03-18	Kithul Road	Hatton	Central	Cardiology
• HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

The 'DOCTOR 36' output pane shows the results of the executed actions:

#	Time	Action	Message	Duration / Fetch
103	11:41:11	UPDATE Doctor SET City = "Kadawatha". WHERE Doctor_id = "D02"	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0 0.000 sec	
104	11:41:11	DELETE FROM Doctor WHERE Doctor_id = "D05"	1 row(s) affected	0.000 sec
105	11:41:11	SELECT * FROM DOCTOR LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

3.5.4 Prescription

The screenshot shows the MySQL Workbench interface with the database set to 'Local instance MySQL80'. In the 'Navigator' pane, the schema 'hospital_information_system' is selected. A query editor window titled 'rerection_3' contains the following SQL code:

```
649
650
651 • SELECT * FROM Prescription ;
652 • UPDATE Prescription SET Dosage = 1 WHERE Prescription_id = "PR001";
653 • UPDATE Prescription SET Dosage = 2 WHERE Prescription_id = "PR002";
654 • DELETE FROM Prescription WHERE Prescription_id = "PR006";
655 • SELECT * FROM Prescription ;
656
```

The 'Result Grid' shows the following data for the Prescription table:

Prescription_id	Dosage	No_of_days	Prescribed_date	Doc_id
PR001	1	7	2023-08-15	D01
PR002	2	14	2023-08-20	D01
PR003	3	5	2023-08-25	D01
PR004	2	10	2023-08-28	D01
PR005	1	7	2023-08-30	D01
• HULL	HULL	HULL	HULL	HULL

The 'Prescription 44' output pane shows the results of the executed actions:

#	Time	Action	Message	Duration / Fetch
123	12:07:28	UPDATE Prescription SET Dosage = 2 WHERE Prescription_id = "PR002"	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0 0.000 sec	
124	12:07:28	DELETE FROM Prescription WHERE Prescription_id = "PR006"	1 row(s) affected	0.000 sec
125	12:07:28	SELECT * FROM Prescription LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
126	12:07:45	SELECT * FROM Prescription LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

3.5.5 Pres_Medi_Relation

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

- Schemas:** Local instance MySQL80, hospital_information_system
- Query Editor:** A window titled "rerection_3" containing the following SQL code:

```
658
659
660 • SELECT * FROM Pres_Medi_Relation ;
661 • UPDATE Pres_Medi_Relation SET Med_id = "M0003" WHERE Prescription_id = "PR001" AND Med_id = "M0001";
662 • UPDATE Pres_Medi_Relation SET Med_id = "M0002" WHERE Prescription_id = "PR002" AND Med_id = "M0001";
663 • DELETE FROM Pres_Medi_Relation WHERE Prescription_id = "PR002" AND Med_id = "M0004";
664 • SELECT * FROM Pres_Medi_Relation ;
```
- Result Grid:** Shows the results of the SELECT query, listing rows with Prescription_id and Med_id.
- Action Output:** Shows the log of actions taken:

#	Time	Action	Message	Duration / Fetch
128	12:09:44	UPDATE Pres_Medi_Relation SET Med_id = "M0003" WHERE Prescription_id = "PR001" AND Med_id = "M0001";	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.015 sec
129	12:09:44	UPDATE Pres_Medi_Relation SET Med_id = "M0002" WHERE Prescription_id = "PR002" AND Med_id = "M0001";	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
130	12:09:44	DELETE FROM Pres_Medi_Relation WHERE Prescription_id = "PR002" AND Med_id = "M0004";	1 row(s) affected	0.000 sec
131	12:09:44	SELECT * FROM Pres_Medi_Relation LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec

3.5.6 Non_medical_staff

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

- Schemas:** Local instance MySQL80, hospital_information_system
- Query Editor:** A window titled "rerection_3" containing the following SQL code:

```
667
668
669 • SELECT * FROM Non_medical_staff ;
670 • UPDATE Non_medical_staff SET Street = "321 Main St" WHERE Staff_id = "S001";
671 • UPDATE Non_medical_staff SET Street = "654 Elm St" WHERE Staff_id = "S002";
672 • DELETE FROM Non_medical_staff WHERE Staff_id = "S005";
673 • SELECT * FROM Non_medical_staff ;
```
- Result Grid:** Shows the results of the SELECT query, listing rows with Staff_id, Manager_id, First_name, Last_name, Date_of_birth, Gender, Job_title, Street, City, and Province.
- Action Output:** Shows the log of actions taken:

#	Time	Action	Message	Duration / Fetch
134	12:23:31	UPDATE Non_medical_staff SET Street = "321 Main St" WHERE Staff_id = "S001"	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
135	12:23:31	UPDATE Non_medical_staff SET Street = "654 Elm St" WHERE Staff_id = "S002"	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
136	12:23:31	DELETE FROM Non_medical_staff WHERE Staff_id = "S005"	1 row(s) affected	0.016 sec
137	12:23:31	SELECT * FROM Non_medical_staff LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

3.5.7 Staff_contact_relation

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

- Schemas:** hospital_information_system
- Queries:** A script named "recorrection_3" is open, containing the following SQL statements:

```
676
677
678 •  SELECT * FROM Staff_contact_relation ;
679 •  UPDATE Staff_contact_relation SET Phone_number = "0728321412" WHERE Staff_id = "S001" AND Phone_number = "0728393412";
680 •  UPDATE Staff_contact_relation SET Phone_number = "0728322412" WHERE Staff_id = "S001" AND Phone_number = "0768993412";
681 •  DELETE FROM Staff_contact_relation WHERE Staff_id = "S006" AND Phone_number = "0740993412";
682 •  SELECT * FROM Staff_contact_relation ;
683
```
- Result Grid:** Shows the current state of the Staff_contact_relation table with the following data:

Staff_id	Phone_number
S001	0728321412
S001	0728322412
S002	0776493412
S003	0769341412
S004	0728972812
S006	0728123412
NULL	NULL
- Output:** Action Output pane shows the execution history:

#	Time	Action	Message	Duration / Fetch
140	12:46:46	UPDATE Staff_contact_relation SET Phone_number = "0728322412" WHERE Staff_id ...	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
141	12:46:49	SELECT * FROM Staff_contact_relation LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec
142	12:47:16	DELETE FROM Staff_contact_relation WHERE Staff_id = "S006" AND Phone_number ...	1 row(s) affected	0.000 sec
143	12:47:20	SELECT * FROM Staff_contact_relation LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

3.5.8 Department

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

- Schemas:** hospital_information_system
- Queries:** A script named "recorrection_3" is open, containing the following SQL statements:

```
683
684
685
686
687 •  SELECT * FROM Department ;
688 •  UPDATE Department SET Head_id = "D01" WHERE Department_id = "DE01" ;
689 •  UPDATE Department SET Head_id = "D02" WHERE Department_id = "DE02" ;
690 •  DELETE FROM Department WHERE Department_id = "DE06" ;
691 •  SELECT * FROM Department ;
```
- Result Grid:** Shows the current state of the Department table with the following data:

Department_id	Department_name	Head_id
D001	Pediatrics	D01
D002	Orthopaedic	D02
D004	Neurology	D03
D005	Dermatology	D04
NULL	NULL	NULL
- Output:** Action Output pane shows the execution history:

#	Time	Action	Message	Duration / Fetch
320	13:17:46	UPDATE Department SET Head_id = "D01" WHERE Department_id = "DE01"	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
321	13:17:46	UPDATE Department SET Head_id = "D02" WHERE Department_id = "DE02"	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
322	13:17:46	DELETE FROM Department WHERE Department_id = "DE06"	1 row(s) affected	0.000 sec
323	13:17:46	SELECT * FROM Department LIMIT 0, 1000	4 row(s) returned	0.016 sec / 0.000 sec

3.5.9 Department_Staff_Relation

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

- Schemas:** hospital_information_system
- Tables:** department
- Query Editor:** Contains the following SQL code:

```
692
693
694
695
696
697 • SELECT * FROM Department_Staff_Relation ;
698 • UPDATE Department_Staff_Relation SET Dep_id = "DE02" WHERE Dep_id = "DE01" AND Staff_id = "S001";
699 • UPDATE Department_Staff_Relation SET Staff_id = "S002" WHERE Dep_id = "DE04" AND Staff_id = "S001";
700 • DELETE FROM Department_Staff_Relation WHERE Dep_id = "DE02" AND Staff_id = "S004";
701 • SELECT * FROM Department_Staff_Relation ;
```
- Result Grid:** Shows the results of the SELECT query, listing staff members associated with department DE02.
- Action Output:** Displays the log of executed statements, including their time, action, message, and duration.

3.5.10 Department_Doc_Relation

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

- Schemas:** hospital_information_system
- Tables:** department
- Query Editor:** Contains the following SQL code:

```
704
705
706
707
708 • SELECT * FROM Department_Doc_Relation ;
709 • UPDATE Department_Doc_Relation SET Dep_id = "DE02" WHERE Dep_id = "DE01" AND Doc_id = "D02";
710 • UPDATE Department_Doc_Relation SET Doc_id = "D02" WHERE Dep_id = "DE01" AND Doc_id = "D01";
711 • DELETE FROM Department_Doc_Relation WHERE Dep_id = "DE05" AND Doc_id = "D06";
712 • SELECT * FROM Department_Doc_Relation ;
```
- Result Grid:** Shows the results of the SELECT query, listing documents associated with department DE02.
- Action Output:** Displays the log of executed statements, including their time, action, message, and duration.

3.5.11 Test_report

```
recorrection_3 x department
713
714
715
716
717
718 •   SELECT * FROM Test_report ;
719 •   UPDATE Test_report SET Tested_time = "15:30:00" WHERE Test_id = "TR0001" ;
720 •   UPDATE Test_report SET Tested_time = "11:30:00" WHERE Test_id = "TR0002" ;
721 •   DELETE FROM Test_report WHERE Test_id = "TR0006" ;
722 •   SELECT * FROM Test_report ;
723

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Result Grid | Form Editor | Revert |
Test_id Test_name Tested_date Tested_time result
TR0001 Blood test 2023-07-13 15:30:00 All are normal
TR0002 hematocrit 2023-07-14 11:30:00 All are normal
TR0003 Liver function test 2023-07-14 09:00:00 All are normal
TR0004 Pregnancy test 2023-07-16 15:00:00 positive
TR0005 Kidney function test 2023-07-18 14:00:00 In the middle level of kidney failure
NULL NULL NULL NULL NULL

Test_report 129 x
Output
Action Output
# Time Action Message Duration / Fetch
336 13:23:29 UPDATE Test_report SET Tested_time = "15:30:00" WHERE Test_id = "TR0001" 1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0 0.000 sec
337 13:23:29 UPDATE Test_report SET Tested_time = "11:30:00" WHERE Test_id = "TR0002" 1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0 0.016 sec
338 13:23:29 DELETE FROM Test_report WHERE Test_id = "TR0006" 1 row(s) affected 0.000 sec
339 13:23:29 SELECT * FROM Test_report LIMIT 0, 1000 5 row(s) returned 0.000 sec / 0.000 sec

Object Info Session
```

3.5.12 Patient

```
recorrection_3 x department
723
724
725
726
727
728 •   SELECT * FROM Patient ;
729 •   UPDATE Patient SET Last_name = "Jackson" WHERE Patient_id = "P001" ;
730 •   UPDATE Patient SET Last_name = "Clinton" WHERE Patient_id = "P002" ;
731 •   DELETE FROM Patient WHERE Patient_id = "P006" ;
732 •   SELECT * FROM Patient ;
733

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Result Grid | Form Editor | Revert |
patient_id First_name Last_name Gender Age Street City Province
P001 John Jackson M 45 123 Main St Anytown California
P002 Jane Clinton F 32 456 Elm Ave Springfield Illinois
P003 Michael Johnson M 28 789 Oak Rd Oakville New York
P004 Emily Williams F 60 101 Pine St Pineville Louisiana
P005 David Brown M 50 222 Maple Lane Maple City Michigan
NULL NULL NULL NULL NULL NULL NULL NULL

Patient 131 x
Output
Action Output
# Time Action Message Duration / Fetch
341 13:24:38 UPDATE Patient SET Last_name = "Jackson" WHERE Patient_id = "P001" 1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0 0.000 sec
342 13:24:38 UPDATE Patient SET Last_name = "Clinton" WHERE Patient_id = "P002" 1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0 0.000 sec
343 13:24:38 DELETE FROM Patient WHERE Patient_id = "P006" 1 row(s) affected 0.000 sec
344 13:24:38 SELECT * FROM Patient LIMIT 0, 1000 5 row(s) returned 0.000 sec / 0.000 sec

Object Info Session
```

3.5.13 Patient_Doctor_relation

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

- Schemas:** hospital_information_system
- Tables:** Patient_Doctor_relation
- Query Editor:** Displays the following SQL code:

```
738 • SELECT * FROM Patient_Doctor_relation ;
739 • UPDATE Patient_Doctor_relation SET Doctor_id = "D02" WHERE Patient_id = "P002" AND Doctor_id = "D01";
740 • UPDATE Patient_Doctor_relation SET Doctor_id = "D02" WHERE Patient_id = "P004" AND Doctor_id = "D01";
741 • DELETE FROM Patient_Doctor_relation WHERE Patient_id = "P005" AND Doctor_id = "D03";
742 • SELECT * FROM Patient_Doctor_relation ;
```
- Result Grid:** Shows the current state of the Patient_Doctor_relation table with the following data:

Patient_id	Doctor_id
P001	D01
P001	D02
P002	D02
P004	D02
P005	D02
P001	D03
P003	D03
*	*
- Action Output:** Shows the log of executed actions:

#	Time	Action	Message	Duration / Fetch
349	13:30:38	UPDATE Patient_Doctor_relation SET Doctor_id = "D02" WHERE Patient_id = "P002" ... 0 rows affected Rows matched: 0 Changed: 0 Warnings: 0	0.000 sec	
350	13:30:38	UPDATE Patient_Doctor_relation SET Doctor_id = "D02" WHERE Patient_id = "P004" ... 0 rows affected Rows matched: 0 Changed: 0 Warnings: 0	0.000 sec	
351	13:30:38	DELETE FROM Patient_Doctor_relation WHERE Patient_id = "P005" AND Doctor_id = ... 1 row(s) affected	0.016 sec	
352	13:30:38	SELECT * FROM Patient_Doctor_relation LIMIT 0,1000	7 row(s) returned	0.000 sec / 0.000 sec

3.5.14 Patient_Med_relation

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

- Schemas:** hospital_information_system
- Tables:** Patient_Med_relation
- Query Editor:** Displays the following SQL code:

```
745 •
746 •
747 • SELECT * FROM Patient_Med_relation ;
748 • UPDATE Patient_Med_relation SET Med_id = "M0003" WHERE Patient_id = "P002" AND Med_id = "M0001";
749 • UPDATE Patient_Med_relation SET Med_id = "M0003" WHERE Patient_id = "P003" AND Med_id = "M0001";
750 • DELETE FROM Patient_Med_relation WHERE Patient_id = "P003" AND Med_id = "M0005";
751 • SELECT * FROM Patient_Med_relation ;
```
- Result Grid:** Shows the current state of the Patient_Med_relation table with the following data:

Patient_id	Med_id
P001	M0001
P002	M0002
P002	M0003
P003	M0003
P002	M0004
P005	M0004
*	*
- Action Output:** Shows the log of executed actions:

#	Time	Action	Message	Duration / Fetch
354	13:31:59	UPDATE Patient_Med_relation SET Med_id = "M0003" WHERE Patient_id = "P002" A... 1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec	
355	13:31:59	UPDATE Patient_Med_relation SET Med_id = "M0003" WHERE Patient_id = "P003" A... 1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec	
356	13:31:59	DELETE FROM Patient_Med_relation WHERE Patient_id = "P003" AND Med_id = "M0... 1 row(s) affected	0.000 sec	
357	13:31:59	SELECT * FROM Patient_Med_relation LIMIT 0,1000	6 row(s) returned	0.000 sec / 0.000 sec

3.5.15 Patient_contact_relation

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

- Navigator:** Shows the schema `hospital_information_system` and its tables: `app_reason`, `appointment`, `bill`, `department`, `department_doc_relation`, `department_staff_relation`, `doctor`, `doctor_phonenumber`, `medicine`, `non_medical_staff`, `nurse`, `nurse_contact_relation`, and `nursespat_relation`.
- SQL Editor:** Contains the following SQL code:

```
754
755
756 • SELECT * FROM Patient_contact_relation ;
757 • UPDATE Patient_contact_relation SET Phone_number = "0721393412" WHERE Pat_id = "P001" AND Phone_number = "0728393412";
758 • UPDATE Patient_contact_relation SET Phone_number = "0722393412" WHERE Pat_id = "P001" AND Phone_number = "0768993412";
759 • DELETE FROM Patient_contact_relation WHERE Pat_id = "P005" AND Phone_number = "0765778684";
760 • SELECT * FROM Patient_contact_relation ;
```
- Result Grid:** Displays the results of the query:

Pat_id	Phone_number
P001	0721393412
P001	0722393412
P002	0724787912
P003	0769001412
P004	0728972812
HULL	HULL
- Output:** Shows the action log with the following entries:

#	Time	Action	Message	Duration / Fetch
359	13:33:56	UPDATE Patient_contact_relation SET Phone_number = "0721393412" WHERE Pat_id = "P001" AND Phone_number = "0728393412";	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.015 sec
360	13:33:56	UPDATE Patient_contact_relation SET Phone_number = "0722393412" WHERE Pat_id = "P001" AND Phone_number = "0768993412";	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
361	13:33:56	DELETE FROM Patient_contact_relation WHERE Pat_id = "P005" AND Phone_number = "0765778684";	1 row(s) affected	0.000 sec
362	13:33:56	SELECT * FROM Patient_contact_relation LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

3.5.16 Patient_Test_relation

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

- Navigator:** Shows the schema `hospital_information_system` and its tables: `app_reason`, `appointment`, `bill`, `department`, `department_doc_relation`, `department_staff_relation`, `doctor`, `doctor_phonenumber`, `medicine`, `non_medical_staff`, `nurse`, `nurse_contact_relation`, and `nursespat_relation`.
- SQL Editor:** Contains the following SQL code:

```
765 • SELECT * FROM Patient_Test_relation ;
766 • UPDATE Patient_Test_relation SET Patient_id = "P003" WHERE Patient_id = "P001" AND Test_id = "TR0003";
767 • UPDATE Patient_Test_relation SET Patient_id = "P004" WHERE Patient_id = "P002" AND Test_id = "TR0004";
768 • DELETE FROM Patient_Test_relation WHERE Patient_id = "P003" AND Test_id = "TR0005";
769 • SELECT * FROM Patient_Test_relation ;
```
- Result Grid:** Displays the results of the query:

Patient_id	Test_id
P001	TR0001
P002	TR0002
P003	TR0003
P004	TR0004
HULL	HULL
- Output:** Shows the action log with the following entries:

#	Time	Action	Message	Duration / Fetch
364	13:34:50	UPDATE Patient_Test_relation SET Patient_id = "P003" WHERE Patient_id = "P001" AND Test_id = "TR0003";	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
365	13:34:50	UPDATE Patient_Test_relation SET Patient_id = "P004" WHERE Patient_id = "P002" AND Test_id = "TR0004";	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
366	13:34:50	DELETE FROM Patient_Test_relation WHERE Patient_id = "P003" AND Test_id = "TR0005";	1 row(s) affected	0.000 sec
367	13:34:50	SELECT * FROM Patient_Test_relation LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

3.5.17 Patient_Prescription_relation

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

- Schemas:** hospital_information_system
- Tables:** Patient_Prescription_relation
- SQL Editor:** A list of executed SQL statements:


```

772
773
774 • SELECT * FROM Patient_Prescription_relation ;
775 • UPDATE Patient_Prescription_relation SET Patient_id = "P002" WHERE Patient_id = "P001" AND Pres_id = "PR002";
776 • UPDATE Patient_Prescription_relation SET Patient_id = "P003" WHERE Patient_id = "P001" AND Pres_id = "PR003";
777 • DELETE FROM Patient_Prescription_relation WHERE Patient_id = "P003" AND Pres_id = "PR005";
778 • SELECT * FROM Patient_Prescription_relation ;
            
```
- Result Grid:** Shows the current state of the Patient_Prescription_relation table with the following data:

Patient_id	Pres_id
P001	PR001
P002	PR002
P003	PR003
P002	PR004
	HULL
- Action Output:** Shows the log of actions taken:

#	Time	Action	Message	Duration / Fetch
369	13:35:57	UPDATE Patient_Prescription_relation SET Patient_id = "P002" WHERE Patient_id = "P001" ...	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
370	13:35:57	UPDATE Patient_Prescription_relation SET Patient_id = "P003" WHERE Patient_id = "P001" ...	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
371	13:35:57	DELETE FROM Patient_Prescription_relation WHERE Patient_id = "P003" AND Pres_id = "PR005"	1 row(s) affected	0.000 sec
372	13:35:57	SELECT * FROM Patient_Prescription_relation LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

3.5.18 Appointment

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

- Schemas:** GP_10_Hospital_Information_S...
- Tables:** Appointment
- SQL Editor:** A list of executed SQL statements:


```

788 • SELECT * FROM Appointment ;
789 • UPDATE Appointment SET Apointment_time = "14:00:00" WHERE Apointment_date = "2023-08-20" AND Apointment_time = "17:00:00" AND Patient_id = "P001" ;
790 • UPDATE Appointment SET Apointment_time = "2023-08-28" WHERE Apointment_date = "2023-08-23" AND Apointment_time = "15:00:00" AND Patient_id = "P001" ;
791 • DELETE FROM Appointment WHERE Apointment_date = "2023-08-21" AND Apointment_time = "17:00:00" AND Patient_id = "P004" ;
792 • SELECT * FROM Appointment ;
            
```
- Result Grid:** Shows the current state of the Appointment table with the following data:

Apointment_date	Apointment_time	current_status	patient_id
2023-08-20	14:00:00	pending	P001
2023-08-21	15:00:00	confirmed	P001
2023-08-28	15:00:00	completed	P001
2023-08-20	15:00:00	canceled	P001
2023-08-20	15:00:00	confirmed	P005
- Action Output:** Shows the log of actions taken:

#	Time	Action	Message	Duration / Fetch
282	19:07:44	SELECT * FROM Ward_nurse_relation LIMIT 0, 1000	OK	0.000 sec
283	19:07:44	SELECT * FROM Ward_patient_relation LIMIT 0, 1000	OK	0.000 sec
284	19:07:44	UPDATE Ward_patient_relation SET Ward_number = 2 WHERE Ward_number = 1 AND Dept_id = "DE01" AND Pati...	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
285	19:07:44	UPDATE Ward_patient_relation SET Ward_number = 2 WHERE Ward_number = 1 AND Dept_id = "DE02" AND Pati...	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
286	19:07:44	DELETE FROM Ward_patient_relation WHERE Ward_number = 1 AND Dept_id = "DE02" AND Patient_id = "P003"	1 row(s) affected	0.000 sec
287	19:07:44	SELECT * FROM Ward_patient_relation LIMIT 0, 1000	OK	0.000 sec
288	19:08:45	SELECT * FROM Appointment LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
289	19:11:30	SELECT * FROM Appointment LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

3.5.19 App_reason

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

- Schemas:** GP_18_Hospital_Information_S
- Script:**

```

797
798 •   SELECT * FROM App_reason ;
799 •   UPDATE App_reason SET Reason = "Back pain" WHERE Appintment_date = "2023-08-21" AND Appintment_time = "15:00:00" AND Patient_id = "P001" ;
800 •   UPDATE App_reason SET Reason = "Neck pain" WHERE Appintment_date = "2023-08-23" AND Appintment_time = "15:00:00" AND Patient_id = "P001" ;
801 •   DELETE FROM App_reason WHERE Appintment_date = "2023-08-21" AND Appintment_time = "17:00:00" AND Patient_id = "P004" ;
802 •   SELECT * FROM App_reason ;
803

```
- Result Grid:** Shows the current state of the 'App_reason' table with columns: Appointment_date, Appointment_time, patient_id, reason.
- Action Output:** Displays the history of database operations with columns: Time, Action, Message, Duration / Fetch.

3.5.20 Record

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

- Schemas:** recorrection_3
- Script:**

```

799
800
801 •   SELECT * FROM Record ;
802 •   UPDATE Record SET Date_admitted = "2023-08-22" WHERE Record_id = "R0001" ;
803 •   UPDATE Record SET Date_admitted = "2023-08-23" WHERE Record_id = "R0002" ;
804 •   DELETE FROM Record WHERE Record_id = "R0006" ;
805 •   SELECT * FROM Record ;
806

```
- Result Grid:** Shows the current state of the 'Record' table with columns: Record_id, Date_admitted, Date_discharged, Patient_id.
- Action Output:** Displays the history of database operations with columns: Time, Action, Message, Duration / Fetch.

3.5.21 Record_Doctor_relation

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

- Schemas:** hospital_information_system
- Tables:** department, Record_Doctor_relation
- Query Editor:** Displays the following SQL code:

```
807
808
809
810 • SELECT * FROM Record_Doctor_relation ;
811 • UPDATE Record_Doctor_relation SET Record_id = "R0002" WHERE Record_id = "R0001" AND Doctor_id = "D02";
812 • UPDATE Record_Doctor_relation SET Record_id = "R0003" WHERE Record_id = "R0002" AND Doctor_id = "D03";
813 • DELETE FROM Record_Doctor_relation WHERE Record_id = "R0004" AND Doctor_id = "D06";
814 • SELECT * FROM Record_Doctor_relation ;
```
- Result Grid:** Shows the results of the SELECT query, displaying Record_id and Doctor_id.
- Action Output:** Shows the log of actions taken:

#	Time	Action	Message	Duration / Fetch
395	13:49:52	UPDATE Record_Doctor_relation SET Record_id = "R0002" WHERE Record_id = "R0001" AND Doctor_id = "D02";	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
396	13:49:52	UPDATE Record_Doctor_relation SET Record_id = "R0003" WHERE Record_id = "R0002" AND Doctor_id = "D03";	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
397	13:49:52	DELETE FROM Record_Doctor_relation WHERE Record_id = "R0004" AND Doctor_id = "D06";	1 row(s) affected	0.016 sec
398	13:49:52	SELECT * FROM Record_Doctor_relation LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

3.5.22 Record_Test_relation

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

- Schemas:** hospital_information_system
- Tables:** department, Record_Test_relation
- Query Editor:** Displays the following SQL code:

```
816
817
818
819 • SELECT * FROM Record_Test_relation ;
820 • UPDATE Record_Test_relation SET Rec_id = "R0003" WHERE Rec_id = "R0001" AND Test_id = "TR0003";
821 • UPDATE Record_Test_relation SET Rec_id = "R0002" WHERE Rec_id = "R0003" AND Test_id = "TR0004";
822 • DELETE FROM Record_Test_relation WHERE Rec_id = "R0003" AND Test_id = "TR0005";
823 • SELECT * FROM Record_Test_relation ;
```
- Result Grid:** Shows the results of the SELECT query, displaying Rec_id and Test_id.
- Action Output:** Shows the log of actions taken:

#	Time	Action	Message	Duration / Fetch
400	13:51:25	UPDATE Record_Test_relation SET Rec_id = "R0003" WHERE Rec_id = "R0001" AND Test_id = "TR0003";	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
401	13:51:25	UPDATE Record_Test_relation SET Rec_id = "R0002" WHERE Rec_id = "R0003" AND Test_id = "TR0004";	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
402	13:51:25	DELETE FROM Record_Test_relation WHERE Rec_id = "R0003" AND Test_id = "TR0005";	1 row(s) affected	0.000 sec
403	13:51:25	SELECT * FROM Record_Test_relation LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

3.5.23 Record_Med_relation

The screenshot shows the MySQL Workbench interface with a query editor window titled "reconnection_3" containing the following SQL code:

```

825
826
827
828 • SELECT * FROM Record_Med_relation ;
829 • UPDATE Record_Med_relation SET Record_id = "R0003" WHERE Record_id = "R0002" AND Med_id = "M0003";
830 • UPDATE Record_Med_relation SET Record_id = "R0001" WHERE Record_id = "R0005" AND Med_id = "M0003";
831 • DELETE FROM Record_Med_relation WHERE Record_id = "R0003" AND Med_id = "M0005";
832 • SELECT * FROM Record_Med_relation ;

```

The "Result Grid" pane displays the following data:

Record_id	Med_id
R0001	M0001
R0002	M0002
R0001	M0003
R0003	M0003
R0002	M0004
NULL	NULL

The "Output" pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
405	13:53:12	UPDATE Record_Med_relation SET Record_id = "R0003" WHERE Record_id = "R0002"	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
406	13:53:12	UPDATE Record_Med_relation SET Record_id = "R0001" WHERE Record_id = "R0005"	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
407	13:53:12	DELETE FROM Record_Med_relation WHERE Record_id = "R0003" AND Med_id = "M0003"	1 row(s) affected	0.000 sec
408	13:53:12	SELECT * FROM Record_Med_relation LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

3.5.24 Nurse

The screenshot shows the MySQL Workbench interface with a query editor window titled "reconnection_3" containing the following SQL code:

```

834
835
836
837 • SELECT * FROM Nurse ;
838 • UPDATE Nurse SET Last_name = "Jane" WHERE Nurse_id = "N001" ;
839 • UPDATE Nurse SET Last_name = "Carl" WHERE Nurse_id = "N002" ;
840 • DELETE FROM Nurse WHERE Nurse_id = "N006" ;
841 • SELECT * FROM Nurse ;

```

The "Result Grid" pane displays the following data:

Nurse_id	First_name	Last_name	Date_of_birth	Gender	Age
N001	Mary	Jane	1985-07-15	F	38
N002	Robert	Carl	1990-02-20	M	33
N003	Jennifer	Smith	1988-11-05	F	34
N004	William	Brown	1982-09-10	M	40
N005	Jessica	Williams	1995-06-25	F	28
NULL	NULL	NULL	NULL	NULL	NULL

The "Output" pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
410	14:02:59	UPDATE Nurse SET Last_name = "Jane" WHERE Nurse_id = "N001"	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
411	14:02:59	UPDATE Nurse SET Last_name = "Carl" WHERE Nurse_id = "N002"	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
412	14:02:59	DELETE FROM Nurse WHERE Nurse_id = "N006"	1 row(s) affected	0.000 sec
413	14:02:59	SELECT * FROM Nurse LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

3.5.25 Nurse_contact_relation

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

- Schemas:** hospital_information_system
- Tables:** nurse_contact_relation
- Queries:**

```
843
844
845
846 •  SELECT * FROM Nurse_contact_relation ;
847 •  UPDATE Nurse_contact_relation SET Phone_number = "0778893412" WHERE Nurse_id = "N001" AND Phone_number = "0728393412";
848 •  UPDATE Nurse_contact_relation SET Phone_number = "0778993412" WHERE Nurse_id = "N001" AND Phone_number = "0768993412";
849 •  DELETE FROM Nurse_contact_relation WHERE Nurse_id = "N005" AND Phone_number = "0765778684";
850 •  SELECT * FROM Nurse_contact_relation ;
```
- Result Grid:**

Nurse_id	Phone_number
N001	0778893412
N001	0778993412
N002	0724787912
N003	0769001412
N004	0728972812
*	NULL
- Action Output:**

#	Time	Action	Message	Duration / Fetch
417	14:05:00	UPDATE Nurse_contact_relation SET Phone_number = "0778993412" WHERE Nurse_id = "N001" AND Phone_number = "0728393412";	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
418	14:05:02	SELECT * FROM Nurse_contact_relation LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
419	14:05:31	DELETE FROM Nurse_contact_relation WHERE Nurse_id = "N005" AND Phone_number = "0765778684";	1 row(s) affected	0.000 sec
420	14:05:38	SELECT * FROM Nurse_contact_relation LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

3.5.26 NursePat_relation

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

- Schemas:** hospital_information_system
- Tables:** nursepat_relation
- Queries:**

```
852
853
854
855 •  SELECT * FROM NursePat_relation ;
856 •  UPDATE NursePat_relation SET Nurse_id = "N002" WHERE Nurse_id = "N005" AND Pat_id = "P001";
857 •  UPDATE NursePat_relation SET Pat_id = "P002" WHERE Nurse_id = "N001" AND Pat_id = "P001";
858 •  DELETE FROM NursePat_relation WHERE Nurse_id = "N003" AND Pat_id = "P004";
859 •  SELECT * FROM NursePat_relation ;
```
- Result Grid:**

Nurse_id	Pat_id
N002	P001
N001	P002
N002	P002
N003	P003
*	NULL
- Action Output:**

#	Time	Action	Message	Duration / Fetch
424	14:11:08	UPDATE NursePat_relation SET Pat_id = "P002" WHERE Nurse_id = "N001" AND Pat_id = "P001";	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
425	14:11:11	SELECT * FROM NursePat_relation LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
426	14:11:33	DELETE FROM NursePat_relation WHERE Nurse_id = "N003" AND Pat_id = "P004";	1 row(s) affected	0.000 sec
427	14:11:35	SELECT * FROM NursePat_relation LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

3.5.27 Bill

The screenshot shows the MySQL Workbench interface with the 'recorrection_3' database selected. The 'Tables' section under the 'hospital_information_system' schema shows the 'Bill' table. A query editor window displays several SQL statements:

```
861
862
863
864 • SELECT * FROM Bill ;
865 • UPDATE Bill SET Medicine_charge = 2000 WHERE Bill_number = 1 ;
866 • UPDATE Bill SET Medicine_charge = 1000 WHERE Bill_number = 2 ;
867 • DELETE FROM Bill WHERE Bill_number = 6 ;
868 • SELECT * FROM Bill ;
```

The 'Result Grid' shows the following data for the Bill table:

Bill_number	Ward_charge	Medicine_charge	Test_charge	bill_date	patient_id
1	1000	2000	2000	2023-04-23	P001
2	500	1000	2500	2023-02-13	P002
3	1000	1500	1000	2023-04-03	P001
4	1000	2500	1500	2023-03-15	P003
5	900	1500	2000	2023-01-30	P003
ALL	NULL	NULL	NULL	NULL	NULL

The 'Output' pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
429	14:12:58	UPDATE Bill SET Medicine_charge = 2000 WHERE Bill_number = 1	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
430	14:12:58	UPDATE Bill SET Medicine_charge = 1000 WHERE Bill_number = 2	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
431	14:12:58	DELETE FROM Bill WHERE Bill_number = 6	1 row(s) affected	0.000 sec
432	14:12:58	SELECT * FROM Bill LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

3.5.28 Ward

The screenshot shows the MySQL Workbench interface with the 'recorrection_3' database selected. The 'Tables' section under the 'hospital_information_system' schema shows the 'Ward' table. A query editor window displays several SQL statements:

```
872
873 • SELECT * FROM Ward ;
874 • UPDATE Ward SET Capacity = 15 WHERE Ward_number = 1 AND Department_id = "DE01" ;
875 • UPDATE Ward SET Capacity = 10 WHERE Ward_number = 2 AND Department_id = "DE01" ;
876 • DELETE FROM Ward WHERE Ward_number = 3 AND Department_id = "DE02" ;
877 • SELECT * FROM Ward ;
```

The 'Result Grid' shows the following data for the Ward table:

Ward_number	Capacity	Availability	department_id
1	15	1	DE01
1	20	0	DE02
2	10	1	DE01

The 'Output' pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
456	14:20:45	SELECT * FROM Ward_patient_relation LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
457	14:22:11	DELETE FROM Ward_patient_relation WHERE Ward_number = 1 AND Dept_id = "DE0..."	1 row(s) affected	0.000 sec
458	14:22:15	SELECT * FROM Ward_patient_relation LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
459	14:27:23	SELECT * FROM Ward LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

3.5.29 Ward_staff_relation

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

- Navigator:** Shows the schema `hospital_information_system` under `Tables`, which includes `ward_staff_relation`.
- Query Editor:** A tab named `rerection_3` contains the following SQL code:

```
879  
880  
881  
882 • SELECT * FROM Ward_staff_relation ;  
883 • UPDATE Ward_staff_relation SET Ward_number = 1 WHERE Ward_number = 2 AND Dept_id = "DE01" AND Staff_id = "S002" ;  
884 • UPDATE Ward_staff_relation SET Ward_number = 2 WHERE Ward_number = 1 AND Dept_id = "DE02" AND Staff_id = "S002" ;  
885 • DELETE FROM Ward_staff_relation WHERE Ward_number = 3 AND Dept_id = "DE02" AND Staff_id = "S004" ;  
886 • SELECT * FROM Ward_staff_relation ;
```
- Result Grid:** Displays the data from the `Ward_staff_relation` table:

Ward_number	Dept_id	Staff_id
1	DE01	S002
2	DE01	S003
1	DE02	S001
2	DE02	S002
*	NULL	NULL
- Action Output:** Shows the execution log:

#	Time	Action	Message	Duration / Fetch
441	14:16:15	UPDATE Ward_staff_relation SET Ward_number = 2 WHERE Ward_number = 1 AND ...	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
442	14:16:18	SELECT * FROM Ward_staff_relation LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
443	14:16:38	DELETE FROM Ward_staff_relation WHERE Ward_number = 3 AND Dept_id = "DE02..."	1 row(s) affected	0.016 sec
444	14:16:41	SELECT * FROM Ward_staff_relation LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

3.5.30 Ward_nurse_relation

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

- Navigator:** Shows the schema `hospital_information_system` under `Tables`, which includes `ward_nurse_relation`.
- Query Editor:** A tab named `rerection_3` contains the following SQL code:

```
887  
888  
889  
890  
891 • SELECT * FROM Ward_nurse_relation ;  
892 • UPDATE Ward_nurse_relation SET Ward_number = 1 WHERE Ward_number = 2 AND Dept_id = "DE01" AND Nurse_id = "N004" ;  
893 • UPDATE Ward_nurse_relation SET Ward_number = 2 WHERE Ward_number = 1 AND Dept_id = "DE02" AND Nurse_id = "N001" ;  
894 • DELETE FROM Ward_nurse_relation WHERE Ward_number = 1 AND Dept_id = "DE02" AND Nurse_id = "N003" ;  
895 • SELECT * FROM Ward_nurse_relation ;
```
- Result Grid:** Displays the data from the `Ward_nurse_relation` table:

Ward_number	Dept_id	Nurse_id
1	DE01	N004
1	DE02	N002
2	DE02	N001
*	NULL	NULL
- Action Output:** Shows the execution log:

#	Time	Action	Message	Duration / Fetch
448	14:18:28	UPDATE Ward_nurse_relation SET Ward_number = 2 WHERE Ward_number = 1 AN...	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
449	14:18:31	SELECT * FROM Ward_nurse_relation LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
450	14:18:59	DELETE FROM Ward_nurse_relation WHERE Ward_number = 1 AND Dept_id = "DE..."	1 row(s) affected	0.000 sec
451	14:19:03	SELECT * FROM Ward_nurse_relation LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

3.5.31 Ward_patient_relation

The screenshot shows the MySQL Workbench interface. In the top navigation bar, the database is set to "Local instance MySQL80". The main area displays a query editor window titled "recorrection_3" with the following SQL code:

```
899
900 • SELECT * FROM Ward_patient_relation ;
901 • UPDATE Ward_patient_relation SET Ward_number = 1 WHERE Ward_number = 2 AND Dept_id = "DE01" AND Patient_id = "P004" ;
902 • UPDATE Ward_patient_relation SET Ward_number = 2 WHERE Ward_number = 1 AND Dept_id = "DE02" AND Patient_id = "P002" ;
903 • DELETE FROM Ward_patient_relation WHERE Ward_number = 1 AND Dept_id = "DE02" AND Patient_id = "P003" ;
904 • SELECT * FROM Ward_patient_relation ;
```

The "Result Grid" tab is selected, showing the following data:

Ward_number	Dept_id	Patient_id
1	DE01	P004
1	DE02	P001
2	DE02	P002
•	DE02	P003

The "Output" tab shows the "Action Output" section with the following log entries:

#	Time	Action	Message	Duration / Fetch
455	14:20:40	UPDATE Ward_patient_relation SET Ward_number = 2 WHERE Ward_number = 1 AND ...	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
456	14:20:45	SELECT * FROM Ward_patient_relation LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
457	14:22:11	DELETE FROM Ward_patient_relation WHERE Ward_number = 1 AND Dept_id = "DE0... 1 row(s) affected		0.000 sec
458	14:22:15	SELECT * FROM Ward_patient_relation LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

Chapter 4 : Transactions

Below shown screenshots consist seven simple queries and thirteen complex queries to retrieve data from the database.

4.1 Simple Queries

➤ Select Operation

The screenshot shows the MySQL Workbench interface. The SQL tab contains the query: `1 • Select * from patient where Gender = 'M';`. The Result Grid shows the following data:

patient_id	First_name	Last_name	Gender	Age	Street	City	Province
P001	John	Jackson	M	45	123 Main St	Anytown	California
P003	Michael	Johnson	M	28	789 Oak Rd	Oakville	New York
P005	David	Brown	M	50	222 Maple Lane	Maple City	Michigan
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

The Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
84	02:59:56	select patient_id, First_name, Last_name from patient	except (select patient_id, First_name, Last_name from p... 0 row(s) returned	0.000 sec / 0.000 sec
85	03:00:23	(select patient_id, First_name, Last_name from patient) except (select patient_id, First_name, Last_name from p... 4 row(s) returned		0.015 sec / 0.000 sec
86	03:00:31	(select patient_id, First_name, Last_name from patient) except (select patient_id, First_name, Last_name from p... 3 row(s) returned		0.000 sec / 0.000 sec
87	03:10:54	select * from patient where Gender = 'M' LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

➤ Project Operation

The screenshot shows the MySQL Workbench interface. The SQL tab contains the query: `1 • select First_name, Last_name, Specialization from doctor;`. The Result Grid shows the following data:

First_name	Last_name	Specialization
Saritha	Kumara	Orthopedics
Lahiru	Kumara	Pediatrics
Madushan	Lahiru	Neurology
Madushan	Thilanka	Dermatology
Madushan	Thilanka	Cardiology

The Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
1	01:37:05	select * from patient LIMIT 0, 1000	5 row(s) returned	0.063 sec / 0.000 sec
2	01:41:56	select Last_name, Speciality from doctor LIMIT 0, 1000	Error Code: 1054. Unknown column 'Speciality' in field list'	0.125 sec
3	01:42:09	SELECT * FROM hospital_information_system.doctor LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
4	01:42:41	select First_name, Last_name, Specialization from doctor LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

➤ Cartesian Product Operation

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Standard MySQL icons.
- Navigator:** Local instance MySQL80, Database01*, Workshop03*, Workshop_5, GP_18_Hospital_Information_S_, Transaction.
- SQL Editor:** Contains the query: `1 * select * from patient cross join appointment;`
- Result Grid:** Displays the results of the Cartesian product. The columns are patient_id, First_name, Last_name, Gender, Age, Street, City, Province, Appointment_date, Appointment_time, current_status, and patient_id. The data shows every patient listed against every appointment record.
- Output Panel:** Shows the execution details: Action Output, # 7, Time 01:45:33, Action select * from patient cross join appointment LIMIT 0, 1000, Message 25 row(s) returned, Duration / Fetch 0.016 sec / 0.000 sec.
- System Bar:** Shows the taskbar with various application icons and the system clock at 1:48 AM on 9/6/2023.

➤ Create a user view

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Standard MySQL icons.
- Navigator:** Local instance MySQL80, Database01*, Workshop03*, Workshop_5, GP_18_Hospital_Information_S_, Transaction*.
- SQL Editor:** Contains the SQL code for creating a view:12
13 * create view UV1 as select patient_id, Last_name from patient;
14 * select * from UV1;
15
16
17
18
19
- Result Grid:** Displays the results of the view creation. The columns are patient_id and Last_name. The data shows the last names of all patients: Jackson, Clinton, Johnson, Williams, and Brown.
- Output Panel:** Shows the execution details: Action Output, # 4, Time 13:43:02, Action select * from UV1 LIMIT 0, 1000, Message 5 row(s) returned, Duration / Fetch 0.313 sec / 0.000 sec; # 5, Time 13:43:36, Action SELECT * FROM hospital_information_system.app_reason LIMIT 0, 1000, Message 6 row(s) returned, Duration / Fetch 0.000 sec / 0.000 sec; # 6, Time 13:43:38, Action SELECT * FROM hospital_information_system.appointment LIMIT 0, 1000, Message 5 row(s) returned, Duration / Fetch 0.000 sec / 0.000 sec; # 7, Time 13:43:44, Action select * from UV1 LIMIT 0, 1000, Message 5 row(s) returned, Duration / Fetch 0.000 sec / 0.000 sec.
- System Bar:** Shows the taskbar with various application icons and the system clock at 1:43 PM on 9/4/2023.

➤ Renaming Operation

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (employee_directory, fo, hospital_information_sys). The **hospital_information_sys** schema is selected.
- SQL Editor:** SQL File 2* (Transaction). The query is:


```
1  rename table nursep_at_relation to Nurse_patient_relation;
2  • show tables;
```
- Result Grid:** Shows the results of the `show tables;` command, listing various tables including `nurse_patient_relation`.
- Output:** Action Output shows two queries:
 - Query 5: `SELECT * FROM hospital_information_system.appointment LIMIT 0, 1000` - Message: 5 row(s) returned, Duration / Fetch: 0.000 sec / 0.000 sec
 - Query 6: `SELECT * FROM hospital_information_system.bill LIMIT 0, 1000` - Message: 5 row(s) returned, Duration / Fetch: 0.031 sec / 0.000 sec
- System Bar:** Shows the date and time as 1:53 AM 9/6/2023.

➤ Use of an aggregation function

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (bill, department, department_doc_rel, department_staff_rel, doctor, doctor_phonenumber, medicine, non_medical_staff, nurse, nurse_contact_relation, nursep_at_relation, patient, patient_contact_relation, patient_doctor_relation, patient_med_relation, patient_prescription, patient_test_relation, pres_medi_relation, ...). The **bill** schema is selected.
- SQL Editor:** SQL File 2* (Transaction). The query is:


```
12
13
14
15
16
17 •  select * from bill where Test_charge = (select max(Test_charge) from bill);
```
- Result Grid:** Shows the results of the query, listing a single row with Bill_number 2, Ward_charge 500, Medicine_charge 1000, Test_charge 2500, bill_date 2023-02-13, and patient_id P002.
- Output:** Action Output shows two queries:
 - Query 508: `select * from bill where Test_charge = (select Avg(Test_charge) from bill) LIMIT 0, 1000` - Message: 0 row(s) returned, Duration / Fetch: 0.070 sec / 0.000 sec
 - Query 509: `select * from bill where Test_charge = (select max(Test_charge) from bill) LIMIT 0, 1000` - Message: 1 row(s) returned, Duration / Fetch: 0.000 sec / 0.000 sec
- System Bar:** Shows the date and time as 3:03 PM 9/3/2023.

➤ Use of LIKE keyword

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

- Navigator:** Shows the schema `hospital_information_system` and its tables: doctor, doctor_phonenumber, medicine, non_medical_staff, nurse, nurse_contact_relativ, nurse_patient_relativ, patient, patient_contact_relativ, patient_doctor_relativ, patient_med_relativ, patient_prescription, patient_test_relativ, pres_medi_relativ, prescription, record.
- SQL Editor:** Contains the following SQL code:


```
1 • select * from non_medical_staff
2   where Job_title like "%Asst";
```
- Result Grid:** Displays the results of the query, showing two rows of data from the `non_medical_staff` table.
- Output:** Shows the execution log with four entries, each detailing a query execution step.
- System Bar:** Shows the Windows taskbar with various application icons and the date/time as 9/6/2023 1:59 AM.

4.2 Complex Queries

4.2.1 Basic Set Operations without User views

➤ Union Operation

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

- Navigator:** Shows the schema `hospital_information_system` and its tables: doctor, doctor_phonenumber, medicine, non_medical_staff, nurse, nurse_contact_relativ, nurse_patient_relativ, patient, patient_contact_relativ, patient_doctor_relativ, patient_med_relativ, patient_prescription, patient_test_relativ, pres_medi_relativ, prescription, record.
- SQL Editor:** Contains the following SQL code:


```
1 • (select Doctor_id, Last_name as Doctor_name, Province from doctor where Province = 'Western')
2 union
3   (select Doctor_id, Last_name as Doctor_name, Province from doctor where Province = 'Northern');
```
- Result Grid:** Displays the results of the union query, showing three rows of data from the `doctor` table.
- Output:** Shows the execution log with four entries, each detailing a query execution step.
- System Bar:** Shows the Windows taskbar with various application icons and the date/time as 9/6/2023 2:07 AM.

➤ Intersection Operation

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

- Navigator:** Shows the schema `hospital_information_system` with various tables like `app_reason`, `appointment`, `bill`, etc.
- SQL Editor:** A query titled "GP_18_Hospital_Information_S..." is running under Transaction 1*. It includes a note: "Since INTERSECT keyword is not working in our version, we have used this query to intersect two queries." The query itself is:


```
937 -- Since INTERSECT keyword is not working in our version ,we have used this query to intersect two queries.
938
939 • select * from appointment natural join app_reason
940 where reason = 'ill' and current_status ='canceled';
941
```
- Result Grid:** Displays the result of the query, showing one row:

Appointment_date	Appointment_time	Patient_id	Current_Status	Reason
2023-08-20	15:00:00	P001	canceled	ill
- Output:** Shows the execution log with 6 actions, all completed quickly:

Action	Time	Message	Duration / Fetch
1. 12:01:55 USE Hospital_Information_System		0 row(s) affected	0.000 sec / 0.000 sec
2. 12:03:03 SELECT * FROM hospital_information_system.appointment LIMIT 0, 1000		5 row(s) returned	0.000 sec / 0.000 sec
3. 12:03:18 SELECT * FROM hospital_information_system.app_reason LIMIT 0, 1000		6 row(s) returned	0.015 sec / 0.000 sec
4. 12:08:40 select appointment_date,appointment_time,current_status from appointment natural join app_reason ...		1 row(s) returned	0.000 sec / 0.000 sec
5. 12:08:51 select * from appointment natural join app_reason where reason = 'ill' and current_status = 'cancel...'		1 row(s) returned	0.000 sec / 0.000 sec
6. 12:11:46 select * from appointment natural join app_reason where reason = 'ill' and current_status = 'cancel...'		1 row(s) returned	0.000 sec / 0.000 sec

➤ Set Difference Operation

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

- Navigator:** Shows the schema `hospital_information_system` with various tables like `employee_directory`, `doctor`, `patient`, etc.
- SQL Editor:** A query titled "GP_18_Hospital_Information_S..." is running under Transaction 1*. It includes a note: "Since EXCEPT keyword is not working in our version, we have used this query to execute the 'set difference' operation." The query itself is:


```
946 -- Since EXCEPT keyword is not working in our version ,we have used this query to execute the "set difference" operation.
947
948 -- Here we are selecting all columns from appointment table except the tables which are containing the current_status values as 'confirmed', 'completed'.
949
950 • SELECT *
951   FROM appointment
952   WHERE current_status NOT IN ('confirmed', 'completed');
953
954 /*
955   (select * from appointment) except
956   (select * from appointment where current_status = 'confirmed' or current_status = 'completed');
957 */
```
- Result Grid:** Displays the result of the query, showing four rows of doctor information:

Doctor_id	First_name	Last_name
D01	Sanath	Kumara
D02	Lahiru	Kumara
D03	Madushan	Lahiru
D06	Madushan	Thilanka
- Output:** Shows the execution log with 6 actions, all completed quickly:

Action	Time	Message	Duration / Fetch
13 09:02:11 SELECT * FROM hospital_information_system.department LIMIT 0, 1000		4 row(s) returned	0.016 sec / 0.000 sec
14 09:02:17 SELECT * FROM hospital_information_system.doctor LIMIT 0, 1000		5 row(s) returned	0.000 sec / 0.000 sec
15 09:08:28 select d.doctor_id,first_name,last_name from doctor as d where not exists (select de.department_name from de...)		4 row(s) returned	0.000 sec / 0.000 sec
16 09:59:45 select d.doctor_id,first_name,last_name from doctor as d where not exists (select de.department_name from de...)		4 row(s) returned	0.000 sec / 0.000 sec

➤ Division Operation

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

- Schemas:** Local instance MySQL80, SQL_Lec_01, Group 10_Hospital_Information_5.
- SQL Editor:** Contains a query for division operation:

```

882
883 -- Division operation
884
885 •  SELECT p.patient_id, p.First_name,p.Last_name
886   FROM Patient p
887   WHERE NOT EXISTS (
888     SELECT m.Medicine_name
889       FROM Medicine m
890      WHERE NOT EXISTS (
891        SELECT 1
892          FROM patient_med_relation as m2
893         WHERE m2.Patient_id = p.Patient_id
894      )
895    );
896
897

```

- Result Grid:** Shows the result of the query:

patient_id	First_name	Last_name
P001	John	Jackson
P002	Jane	Clinton
P003	Michael	Johnson
P005	David	Brown
P005	David	Brown
- Output:** Action Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
1	20 20:26:47	SELECT p.patient_id, p.First_name,p.Last_name FROM Patient p WHERE NOT EXISTS (SELECT m.Medicine_name	0.000 sec / 0.000 sec
2		FROM Medicine m	4 row(s) returned	
3		WHERE NOT EXISTS (
4		SELECT 1		
5		FROM patient_med_relation as m2		
6		WHERE m2.Patient_id = p.Patient_id		
7)		

4.2.2 Relational algebraic operations by using user views

➤ Inner Join

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

- Schemas:** Local instance MySQL80, SQL_Lec_01, Group 10_Hospital_Information_5.
- SQL Editor:** Contains an inner join query:

```

966 -- inner join
967 •  create view DP as
968   (select D.Doctor_Id, D.First_name, D.Last_name from doctor as D
969   inner join prescription as P on D.Doctor_Id = P.Doc_id)
970
971 •  select * from DP;
972

```

- Result Grid:** Shows the result of the query:

Doctor_Id	First_name	Last_name
D01	Sanath	Kumara
- Output:** Action Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
1	21:03:48	USE Hospital_Information_System		0.000 sec
2	21:04:09	drop view App1	0 row(s) affected	0.000 sec
3	21:04:35	create view App1 as (select D.Doctor_Id, D.First_name, D.Last_name from doctor as D inner join prescription as P on D.Doctor_Id = P.Doc_id)	0 row(s) affected	0.000 sec
4	21:13:02	drop view App1	0 row(s) affected	0.000 sec
5	21:13:29	create view DP as (select D.Doctor_Id, D.First_name, D.Last_name from doctor as D inner join prescription as P on D.Doctor_Id = P.Doc_id)	0 row(s) affected	0.016 sec
6	21:13:30	select * from DP LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

➤ Natural Join

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

- Schemas:** The current schema is "GP_18_Hospital_Information_5".
- Code Editor:** The code being run is:

```

974 -- natural join
975 • create view Ward_Nurse as
976 select Ward_number, Availability, Nurse_id, department_id from ward
977 natural join ward_nurse_relation where Capacity<15;
978
979 • select * from Ward_Nurse;
980

```
- Result Grid:** The result grid shows one row of data:

Ward_number	Availability	Nurse_id	department_id
2	1	N001	DE01
- Output Window:** The output window displays the execution log:

Action	Time	Action	Message	Duration / Fetch
52	22:13:14	SELECT * FROM hospital_information_system.ward LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
53	22:14:40	drop view Ward_Nurse	0 row(s) affected	0.000 sec
54	22:15:17	create view Ward_Nurse as select Ward_number, Availability, Nurse_id, department_id from ward natural join ward_nurse_relation where Capacity<15;	0 row(s) affected	0.000 sec
55	22:15:17	select * from Ward_Nurse LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
56	22:17:45	select * from Ward_Nurse LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

➤ Left Outer Join

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

- Schemas:** The current schema is "GP_18_Hospital_Information_5".
- Code Editor:** The code being run is:

```

982
983
984 -- left outer join
985 • create view D_Pres2 as
986 select * from prescription as P
987 left outer join doctor as D
988 on P.Doc_Id = D.Doctor_Id where No_of_days<=7;
989
990 • select * from D_Pres2;
991

```
- Result Grid:** The result grid shows three rows of data:

Prescription_id	Dosage	No_of_days	Prescribed_date	Doc_id	Doctor_id	First_name	Last_name	Date_of_birth	Street	City	Province	Specialization
PR003	3	5	2023-08-25	D01	D01	Sanath	Kumara	1980-10-28	Clock tower junction	Kilinochchi	Northern	Orthopedics
PR001	1	7	2023-08-15	D01	D01	Sanath	Kumara	1980-10-28	Clock tower junction	Kilinochchi	Northern	Orthopedics
PR005	1	7	2023-08-30	D01	D01	Sanath	Kumara	1980-10-28	Clock tower junction	Kilinochchi	Northern	Orthopedics
- Output Window:** The output window displays the execution log:

Action	Time	Action	Message	Duration / Fetch
62	22:43:59	select * from D_Pres2 LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
63	22:46:30	create view D_Pres2 as select * from prescription as P left outer join doctor as D on P.Doc_Id = D.Doctor_Id where No_of_days<=7;	Error Code: 1050. Table 'D_Pres2' already exists	0.000 sec
64	22:46:46	drop view D_Pres2	0 row(s) affected	0.016 sec
65	22:46:59	create view D_Pres2 as select * from prescription as P left outer join doctor as D on P.Doc_Id = D.Doctor_Id where No_of_days<=7;	0 row(s) affected	0.000 sec
66	22:46:59	select * from D_Pres2 LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

➤ Right Outer Join

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

- Navigator:** Shows various database objects like patient_contact, patient_doctor, patient_med_rel, etc.
- SQL Editor:** Contains the SQL code for creating a right outer join view:


```

991
992
993 -- right outer join
994 • create view D_Pres1 as
995 select * from prescription as P
996 right outer join doctor as D
997 on P.Doc_id = D.Doctor_id where No_of_days<=7;
998
999 • select * from D_Pres1;
1000
      
```
- Result Grid:** Displays the results of the query, showing three rows of data:

Prescription_id	Dosage	No_of_days	Prescribed_date	Doc_id	Doctor_id	First_name	Last_name	Date_of_birth	Street	City	Province	Specialization
PR001	1	7	2023-08-15	D01	D01	Sanath	Kumara	1980-10-28	Clock tower junction	Kilinochchi	Northern	Orthopedics
PR003	3	5	2023-08-25	D01	D01	Sanath	Kumara	1980-10-28	Clock tower junction	Kilinochchi	Northern	Orthopedics
PR005	1	7	2023-08-30	D01	D01	Sanath	Kumara	1980-10-28	Clock tower junction	Kilinochchi	Northern	Orthopedics
- Output:** Shows the execution log for the session D_Pres1_30:

Action	Time	Action	Message	Duration / Fetch
57	22:34:19	SELECT * FROM hospital_information_system.prescription LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
58	22:36:15	drop view D_Pres1	0 row(s) affected	0.015 sec
59	22:36:22	create view D_Pres1 as select * from prescription as P right outer join doctor as D on P.Doc_id = D.Doctor_id wh...	0 row(s) affected	0.016 sec
60	22:36:22	select * from D_Pres1 LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
61	22:37:37	select * from D_Pres1 LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

➤ Full Outer Join

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

- Navigator:** Shows various database objects like patient_contact, patient_doctor, patient_med_rel, etc.
- SQL Editor:** Contains the SQL code for creating a full outer join view:


```

1000
1001
1002
1003
1004 -- Full outer join
1005 • select * from D_Pres1
1006 union
1007 select * from D_Pres2
1008
1009
      
```
- Result Grid:** Displays the results of the query, showing three rows of data:

Prescription_id	Dosage	No_of_days	Prescribed_date	Doc_id	Doctor_id	First_name	Last_name	Date_of_birth	Street	City	Province	Specialization
PR003	3	5	2023-08-25	D01	D01	Sanath	Kumara	1980-10-28	Clock tower junction	Kilinochchi	Northern	Orthopedics
PR001	1	7	2023-08-15	D01	D01	Sanath	Kumara	1980-10-28	Clock tower junction	Kilinochchi	Northern	Orthopedics
PR005	1	7	2023-08-30	D01	D01	Sanath	Kumara	1980-10-28	Clock tower junction	Kilinochchi	Northern	Orthopedics
- Output:** Shows the execution log for the session Result 33:

Action	Time	Action	Message	Duration / Fetch
63	22:46:30	create view D_Pres2 as select * from prescription as P left outer join doctor as D on P.Doc_id = D.Doctor_id wh...	Error Code: 1050. Table 'D_Pres2' already exists	0.000 sec
64	22:46:46	drop view D_Pres2	0 row(s) affected	0.016 sec
65	22:46:59	create view D_Pres2 as select * from prescription as P left outer join doctor as D on P.Doc_id = D.Doctor_id wh...	0 row(s) affected	0.000 sec
66	22:46:59	select * from D_Pres2 LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
67	22:52:53	select * from D_Pres1 union select * from D_Pres2	3 row(s) returned	0.000 sec / 0.000 sec

➤ Outer Union

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

- Navigator:** Shows the database schema with tables like employee_directory, foe, hos, and hospital_information_5.
- SQL Editor:** Contains the following SQL code:


```

1005
1006 -- Outer union
1007
1008 • SELECT * from doctor as d
1009 inner join department as de ON d.doctor_id = de.head_id
1010 UNION
1011 SELECT doctor_id,First_name,Last_name,Date_of_birth,Street,City,Province,Specialization,null,null,null from doctor ;
      
```
- Result Grid:** Displays the results of the query, showing columns: Doctor_id, First_name, Last_name, Date_of_birth, Street, City, Province, Specialization, Department_id, Department_name, Head_id. The data includes rows for doctors D01 through D06 across different departments.
- Information:** Shows the definition of the app_reason table.
- Output:** Shows the execution log with actions, times, messages, and durations.

4.2.3 Nested Queries

❖ Nested Query 1

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

- Navigator:** Shows the database schema with tables like bill, department, department_doc, department_staff, doctor, doctor_phonenum, medicine, and patient.
- SQL Editor:** Contains the following SQL code:


```

1016
1017 -- Nested Queries
1018
1019
1020 • select Department_id from department where department_name
1021 in(select specialization from doctor);
      
```
- Result Grid:** Displays the results of the query, showing the Department_id column. The data includes rows for DE05 and H001.
- Information:** Shows the definition of the medicine table.
- Output:** Shows the execution log with actions, times, messages, and durations.

❖ Nested Query 2

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

- Navigator:** Shows the database schema with tables like bill, department, doctor, etc., and a specific table named medicine.
- SQL Editor:** Displays the following SQL code:


```

1024 -- Query 2
1025   select nurse_id, Last_name, Gender from nurse
1026   where age > All(
1027     select age from nurse where Gender = 'F');
1028
      
```
- Result Grid:** Shows the results of the query, which is empty (0 rows).
- Output Window:** Shows the execution log with the following entries:

Action	Time	Message	Duration / Fetch
98 01:21:48	select nurse_id, Last_name, Gender from nurse where age > All(select age from nurse where Gender = 'M') LIMIT 0, 0 row(s) returned	0.000 sec / 0.000 sec	
99 01:21:55	select nurse_id, Last_name, Gender from nurse where age > All(select age from nurse where Gender < 'F') LIMIT 0, 1 row(s) returned	0.000 sec / 0.000 sec	
100 01:22:03	select nurse_id, Last_name, Gender from nurse where age > All(select age from nurse where Gender > 'F') LIMIT 0, 1 row(s) returned	0.000 sec / 0.000 sec	
101 01:30:22	select nurse_id, Last_name, Gender from nurse where age > All(select age from nurse where Gender = 'F') LIMIT 0, 1 row(s) returned	0.000 sec / 0.000 sec	
102 01:31:04	select nurse_id, Last_name, Gender from nurse where age > All(select age from nurse where Gender < 'F') LIMIT 0, 1 row(s) returned	0.000 sec / 0.000 sec	

❖ Nested Query 3

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

- Navigator:** Shows the database schema with tables like bill, department, doctor, etc., and a specific table named medicine.
- SQL Editor:** Displays the following SQL code:


```

1029 -- Query 3
1030   select patient_id from patient where age
1031   in
1032   (select age from patient where age<40);
1033
1034
1035
1036
1037
      
```
- Result Grid:** Shows the results of the query, which includes patient IDs P003, P002, and P001.
- Output Window:** Shows the execution log with the following entries:

Action	Time	Message	Duration / Fetch
104 01:37:32	selected First_name from patient where First_name in (select First_name from non_medical_sta... 2 row(s) returned	0.000 sec / 0.000 sec	
105 01:38:27	selected First_name from nurse where First_name in (select First_name from non_medical_sta... 0 row(s) returned	0.000 sec / 0.000 sec	
106 01:38:40	selected First_name from nurse where First_name in (select First_name from patient) LIMIT 0, 1... 0 row(s) returned	0.000 sec / 0.000 sec	
107 01:41:03	selected patient_id from patient where age in (select age from patient where age<40) LIMIT 0, ... 2 row(s) returned	0.000 sec / 0.000 sec	
108 01:45:10	selected patient_id from patient where age in (select age from patient where age<40) LIMIT 0, ... 2 row(s) returned	0.000 sec / 0.000 sec	

Chapter 5 : Tuning

The indexing technique is used to selected complex queries written in Chapter 4. In order to provide a clear understanding, the following steps are taken in query tuning for each complex query.

- ✓ Drop existing indices that are externally created in used tables
- ✓ Show indexes of table before creating suitable index (using SHOW INDEX command) to know about the existing indices.
- ✓ Show number of accessed rows before creating suitable index (using EXPLAIN command) when we are retrieving some particular data.
- ✓ Create suitable index in order to tune the query .
- ✓ Show indexes of table after creating suitable index (using SHOW INDEX command) to ensure that the index was created.
- ✓ Show the number of accessed rows after creating suitable index (using EXPLAIN command) when we are retrieving some particular data.

Tunning of a query is demonstrated in this chapter by comparing the number of accessed rows in the explain table before and after creating a suitable index. If the number of accessed rows decreases after the index is created, the query has been properly tuned. All of the screenshots included below meet that requirement.

- ❖ Tuning of Union Operation
 - Before Tuning

MySQL 8.0 Command Line Cli

```
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'on doctor' at line 1
mysql> select * from doctor;
+-----+-----+-----+-----+-----+-----+-----+
| Doctor_id | First_name | Last_name | Date_of_birth | Street | City | Province | Specialization |
+-----+-----+-----+-----+-----+-----+-----+
| D01 | Sanath | Kumara | 1980-10-28 | Clock tower junction | Kilinochchi | Northern | Orthopedics
| D02 | Lahiru | Kumara | 1980-10-28 | Kurumankadu | Kadawatha | Western | Pediatrics
| D03 | Madushan | Lahiru | 1975-09-20 | Wakwella | Galle | Southern | Neurology
| D04 | Madushan | Thilanka | 1975-09-20 | Wellawatta | Colombo | Western | Dermatology
| D05 | Madushan | Thilanka | 1972-03-18 | Kithul Road | Hatton | Central | Cardiology
+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> show index from doctor;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| doctor | 0 | PRIMARY | 1 | Doctor_id | A | 5 | NULL | NULL | BTREE | | YES | NULL |
| doctor | 0 | Name_birth | 1 | First_name | A | 3 | NULL | NULL | BTREE | | YES | NULL |
| doctor | 0 | Name_birth | 2 | Last_name | A | 4 | NULL | NULL | BTREE | | YES | NULL |
| doctor | 0 | Name_birth | 3 | Date_of_birth | A | 5 | NULL | NULL | BTREE | | YES | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> explain((select Doctor_id, Last_name as Doctor_name, Province from doctor where Province = 'Western')
-> union
-> (select Doctor_id, Last_name as Doctor_name, Province from doctor where Province = 'Northern'));
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | doctor | NULL | ALL | NULL | NULL | NULL | 5 | 20.00 | Using where | |
| 2 | UNION | doctor | NULL | ALL | NULL | NULL | NULL | 5 | 20.00 | Using where |
| 3 | UNION RESULT | <union1,2> | NULL | ALL | NULL | NULL | NULL | NULL | NULL | NULL | Using temporary |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set, 1 warning (0.00 sec)
```

- After Tuning

MySQL 8.0 Command Line Cli

```
3 rows in set, 1 warning (0.00 sec)

mysql> create index Province_Ind on doctor(Province);
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show index from doctor;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| doctor | 0 | PRIMARY | 1 | Doctor_id | A | 5 | NULL | NULL | BTREE | | YES | NULL | |
| doctor | 0 | Name_birth | 1 | First_name | A | 3 | NULL | NULL | BTREE | | YES | NULL |
| doctor | 0 | Name_birth | 2 | Last_name | A | 4 | NULL | NULL | BTREE | | YES | NULL |
| doctor | 0 | Name_birth | 3 | Date_of_birth | A | 5 | NULL | NULL | BTREE | | YES | NULL |
| doctor | 1 | Province_Ind | 1 | Province | A | 4 | NULL | NULL | YES | BTREE | | YES | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> explain((select Doctor_id, Last_name as Doctor_name, Province from doctor where Province = 'Western')
-> union
-> (select Doctor_id, Last_name as Doctor_name, Province from doctor where Province = 'Northern'));
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | doctor | NULL | ref | Province_Ind | Province_Ind | 163 | const | 2 | 100.00 | NULL |
| 2 | UNION | doctor | NULL | ref | Province_Ind | Province_Ind | 163 | const | 1 | 100.00 | NULL |
| 3 | UNION RESULT | <union1,2> | NULL | ALL | NULL | NULL | NULL | NULL | NULL | NULL | Using temporary |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set, 1 warning (0.00 sec)
```

❖ Tuning of Intersection operation

- Before Tuning

```
MySQL 8.0 Command Line Cli + X

mysql> select * from appointment;
+-----+-----+-----+-----+
| Appointment_date | Appintment_time | current_status | patient_id |
+-----+-----+-----+-----+
| 2023-08-20 | 14:00:00 | pending | P001 |
| 2023-08-21 | 15:00:00 | confirmed | P001 |
| 2023-08-28 | 15:00:00 | completed | P001 |
| 2023-08-20 | 15:00:00 | canceled | P001 |
| 2023-08-20 | 15:00:00 | confirmed | P005 |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> show index from appointment;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| appointment | 1 | fk_PatApp | 1 | patient_id | A | 3 | NULL | NULL | BTREE | | | YES | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> explain(select * from appointment natural join app_reason
-> where reason = 'ill' and current_status = 'canceled' );
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | appointment | NULL | ALL | fk_PatApp | NULL | NULL | NULL | 6 | 16.67 | Using where |
| 1 | SIMPLE | app_reason | NULL | ref | fk_AppReason | fk_AppReason | 122 | hospital_information_system.appointment.patient_id | 2 | 14.29 | Using where |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.01 sec)
```

- After Tuning

```
MySQL 8.0 Command Line Cli + X

mysql> create index Cur_Status_Ind on appointment(current_status);
Query OK, 0 rows affected (0.85 sec)
Records: 0 Duplicates: 0 Warnings: 0
mysql> show index from appointment;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| appointment | 1 | fk_PatApp | 1 | patient_id | A | 3 | NULL | NULL | BTREE | | | YES | NULL |
| appointment | 1 | Cur_Status_Ind | 1 | current_status | A | 4 | NULL | NULL | YES | BTREE | | | YES | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)

mysql> explain(select * from appointment natural join app_reason
-> where reason = 'ill' and current_status = 'canceled' );
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | appointment | NULL | ref | fk_PatApp,Cur_Status_Ind | Cur_Status_Ind | 83 | const | 1 | 100.00 | NULL |
| 1 | SIMPLE | app_reason | NULL | ref | fk_AppReason | fk_AppReason | 122 | hospital_information_system.appointment.patient_id | 2 | 14.29 | Using where |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)

mysql>
```

❖ Tuning of Set Difference Operation

- Before Tuning

```
MySQL 8.0 Command Line Cli + X

mysql> select * from appointment;
+-----+-----+-----+-----+
| Appointment_date | Appintment_time | current_status | patient_id |
+-----+-----+-----+-----+
| 2023-08-20 | 14:00:00 | pending | P001 |
| 2023-08-21 | 15:00:00 | confirmed | P001 |
| 2023-08-28 | 15:00:00 | completed | P001 |
| 2023-08-20 | 15:00:00 | canceled | P001 |
| 2023-08-20 | 15:00:00 | confirmed | P005 |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> show index from appointment;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| appointment | 1 | fk_PatApp | 1 | patient_id | A | 3 | NULL | NULL | BTREE | | | YES | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> explain(SELECT *
-> FROM appointment
-> WHERE current_status NOT IN ('confirmed', 'completed'));
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | appointment | NULL | ALL | NULL | NULL | NULL | NULL | 6 | 66.67 | Using where |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)
```

- After Tuning

```
MySQL 8.0 Command Line Cli + X

mysql> create index Cur_status_Ind on appointment(current_status);
Query OK, 0 rows affected (0.05 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> show index from appointment;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| appointment | 1 | fk_PatApp | 1 | patient_id | A | 3 | NULL | NULL | BTREE | | | YES | NULL |
| appointment | 1 | Cur_status_Ind | 1 | current_status | A | 4 | NULL | NULL | BTREE | | | YES | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> explain(SELECT *
->   FROM appointment
->   WHERE current_status NOT IN ('confirmed', 'completed'));
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | appointment | NULL | range | Cur_status_Ind | Cur_status_Ind | 83 | NULL | 3 | 100.00 | Using index condition |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)

mysql>
```

❖ Tuning of Natural join operation

- Before Tuning

```
MySQL 8.0 Command Line Cli + X

mysql> select * from Ward_Nurse;
+-----+-----+-----+
| Ward_number | Availability | Nurse_id | department_id |
+-----+-----+-----+
| 2 | 1 | N001 | DE01 |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql> show index from Ward_Nurse;
Empty set (0.00 sec)

mysql> show index from Ward;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ward | 1 | dep_Ind | 1 | department_id | A | 2 | NULL | NULL | BTREE | | | YES | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> show index from ward_nurse_relation;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ward_nurse_relation | 0 | PRIMARY | 1 | Ward_number | A | 2 | NULL | NULL | BTREE | | | YES | NULL |
| ward_nurse_relation | 0 | PRIMARY | 2 | Dept_id | A | 3 | NULL | NULL | BTREE | | | YES | NULL |
| ward_nurse_relation | 0 | PRIMARY | 3 | Nurse_id | A | 3 | NULL | NULL | BTREE | | | YES | NULL |
| ward_nurse_relation | 1 | Dept_id | 1 | Dept_id | A | 2 | NULL | NULL | BTREE | | | YES | NULL |
| ward_nurse_relation | 1 | Nurse_id | 1 | Nurse_id | A | 3 | NULL | NULL | BTREE | | | YES | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> explain(select * from Ward_Nurse);
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | ward | NULL | ALL | NULL | NULL | NULL | hospital_information_system.ward.Ward_number | 3 | 33.33 | Using where |
| 1 | SIMPLE | ward_nurse_relation | NULL | ref | PRIMARY | PRIMARY | 4 | NULL | 1 | 100.00 | Using index |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)
```

- After Tuning

```
MySQL 8.0 Command Line Cli + X

mysql>
mysql> create index Cap_Ind on ward(Capacity);
Query OK, 0 rows affected (0.03 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> show index from Ward;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ward | 1 | dep_Ind | 1 | department_id | A | 2 | NULL | NULL | BTREE | | | YES | NULL |
| ward | 1 | Cap_Ind | 1 | Capacity | A | 3 | NULL | NULL | BTREE | | | YES | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> explain(select * from Ward_Nurse);
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | ward | NULL | range | Cap_Ind | Cap_Ind | 5 | NULL | 1 | 100.00 | Using index condition |
| 1 | SIMPLE | ward_nurse_relation | NULL | ref | PRIMARY | PRIMARY | 4 | hospital_information_system.ward.Ward_number | 1 | 100.00 | Using index |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)
```

❖ Tuning of Right outer join operation

- Before Tuning

```
MySQL 8.0 Command Line Cli X + ×

mysql> select * from D_Pres1;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Prescription_id | Dosage | No_of_days | Prescribed_date | Doc_id | Doctor_id | First_name | Last_name | Date_of_birth | Street          |
| City           | Province | Specialization |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| PRB01          | 1      | 7       | 2023-08-15   | D01   | D01     | Sanath    | Kumara   | 1988-10-28  | Clock tower junction | Kilinochchi | Northern | Orthopedics |
| PRB03          | 3      | 5       | 2023-08-25   | D01   | D01     | Sanath    | Kumara   | 1988-10-28  | Clock tower junction | Kilinochchi | Northern | Orthopedics |
| PRB05          | 1      | 7       | 2023-08-30   | D01   | D01     | Sanath    | Kumara   | 1988-10-28  | Clock tower junction | Kilinochchi | Northern | Orthopedics |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> show index from D_Pres1;
Empty set (0.00 sec)

mysql> show index from Doctor;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| doctor | 0 | PRIMARY | 1 | Doctor_id | A | 5 | NULL | NULL | BTREE | | YES | NULL | |
| doctor | 0 | Name_birth | 1 | First_name | A | 3 | NULL | NULL | BTREE | | YES | NULL |
| doctor | 0 | Name_birth | 2 | Last_name | A | 4 | NULL | NULL | BTREE | | YES | NULL |
| doctor | 0 | Name_birth | 3 | Date_of_birth | A | 5 | NULL | NULL | BTREE | | YES | NULL |
| doctor | 1 | Province_Ind | 1 | Province | A | 4 | NULL | NULL | YES | BTREE | | YES | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> show index from Prescription;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| prescription | 0 | PRIMARY | 1 | Prescription_id | A | 6 | NULL | NULL | BTREE | | YES | NULL |
| prescription | 1 | fk_PresDoc | 1 | Doc_id | A | 1 | NULL | NULL | BTREE | | YES | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> explain(select * from D_Pres1);
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | p | NULL | ALL | fk_PresDoc | NULL | NULL | NULL | 6 | 33.33 | Using where |
| 1 | SIMPLE | d | NULL | eq_ref | PRIMARY | PRIMARY | 62 | hospital_information_system.p.Doc_id | 1 | 100.00 | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)
```

- After Tuning

```
MySQL 8.0 Command Line Cli X + ×

2 rows in set, 1 warning (0.00 sec)

mysql> create index days_Ind on Prescription(No_of_days);
Query OK, 0 rows affected (0.00 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show index from Prescription;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| prescription | 0 | PRIMARY | 1 | Prescription_id | A | 6 | NULL | NULL | BTREE | | YES | NULL | |
| prescription | 1 | fk_PresDoc | 1 | Doc_id | A | 1 | NULL | NULL | BTREE | | YES | NULL |
| prescription | 1 | days_Ind | 1 | No_of_days | A | 4 | NULL | NULL | YES | BTREE | | YES | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)

mysql> explain(select * from D_Pres1);
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | p | NULL | range | fk_PresDoc,days_Ind | days_Ind | 5 | NULL | 3 | 100.00 | Using index condition; Using where |
| 1 | SIMPLE | d | NULL | eq_ref | PRIMARY | PRIMARY | 62 | hospital_information_system.p.Doc_id | 1 | 100.00 | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)
```

❖ Tuning of Left outer join operation

- Before Tuning

```
MySQL 8.0 Command Line Cli X + ×

mysql> select * from D_Pres2;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Prescription_id | Dosage | No_of_days | Prescribed_date | Doc_id | Doctor_id | First_name | Last_name | Date_of_birth | Street          |
| City           | Province | Specialization |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| PRB03          | 3      | 5       | 2023-08-25   | D01   | D01     | Sanath    | Kumara   | 1988-10-28  | Clock tower junction | Kilinochchi | Northern | Orthopedics |
| PRB01          | 1      | 7       | 2023-08-15   | D01   | D01     | Sanath    | Kumara   | 1988-10-28  | Clock tower junction | Kilinochchi | Northern | Orthopedics |
| PRB05          | 1      | 7       | 2023-08-30   | D01   | D01     | Sanath    | Kumara   | 1988-10-28  | Clock tower junction | Kilinochchi | Northern | Orthopedics |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> drop index days_Ind on Prescription;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show index from Prescription;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| prescription | 0 | PRIMARY | 1 | Prescription_id | A | 6 | NULL | NULL | BTREE | | YES | NULL |
| prescription | 1 | fk_PresDoc | 1 | Doc_id | A | 1 | NULL | NULL | BTREE | | YES | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> show index from Doctor;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| doctor | 0 | PRIMARY | 1 | Doctor_id | A | 5 | NULL | NULL | BTREE | | YES | NULL | |
| doctor | 0 | Name_birth | 1 | First_name | A | 3 | NULL | NULL | BTREE | | YES | NULL |
| doctor | 0 | Name_birth | 2 | Last_name | A | 4 | NULL | NULL | BTREE | | YES | NULL |
| doctor | 0 | Name_birth | 3 | Date_of_birth | A | 5 | NULL | NULL | BTREE | | YES | NULL |
| doctor | 1 | Province_Ind | 1 | Province | A | 4 | NULL | NULL | YES | BTREE | | YES | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> explain(select * from D_Pres2);
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | p | NULL | ALL | NULL | NULL | NULL | NULL | 6 | 33.33 | Using where |
| 1 | SIMPLE | d | NULL | eq_ref | PRIMARY | PRIMARY | 62 | hospital_information_system.p.Doc_id | 1 | 100.00 | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)
```

- After Tuning

```
MySQL 8.0 Command Line Cli X + ×

mysql>
mysql>
mysql> create index days_Ind on Prescription(No_of_days);
Query OK, 0 rows affected (0.03 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> show index from Prescription;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_Index | Column_name | collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| prescription | 0 | PRIMARY | 1 | Prescription_id | A | 6 | NULL | NULL | BTREE | | YES | NULL | |
| prescription | 1 | fk_PresDoc | 1 | Doc_id | A | 1 | NULL | NULL | BTREE | | YES | NULL |
| prescription | 1 | days_Ind | 1 | No_of_days | A | 4 | NULL | NULL | YES | BTREE | | YES | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> explain(select * from D_Pres);
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | d | NULL | range | days_Ind | days_Ind | 5 | NULL | 3 | 100.00 | Using index condition |
| 1 | SIMPLE | d | NULL | eq_ref | PRIMARY | PRIMARY | 62 | hospital_information_system.p.Doc_id | 1 | 100.00 | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)

mysql>
```

❖ Tuning of Full outer join operation

- Before Tuning

```
MySQL 8.0 Command Line Cli X + ×

mysql> drop index days_Ind on Prescription;
Query OK, 0 rows affected (0.01 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> select * from D_Pres
-> union
-> select * from D_Pres2;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Prescription_id | Dosage | No_of_days | Prescribed_date | Doc_id | Doctor_id | First_name | Last_name | Date_of_birth | Street | City | Province | Specialization |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| PR801 | 1 | 7 | 2823-08-15 | D01 | D01 | Sanath | Kumara | 1980-10-28 | Clock tower junction | Kilinochchi | Northern | Orthopedics |
| PR803 | 3 | 5 | 2823-08-25 | D01 | D01 | Sanath | Kumara | 1980-10-28 | Clock tower junction | Kilinochchi | Northern | Orthopedics |
| PR805 | 1 | 7 | 2823-08-30 | D01 | D01 | Sanath | Kumara | 1980-10-28 | Clock tower junction | Kilinochchi | Northern | Orthopedics |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> explain(select * from D_Pres
-> union
-> select * from D_Pres2);
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | p | NULL | ALL | fk_PresDoc | PRIMARY | 62 | hospital_information_system.p.Doc_id | 6 | 33.33 | Using where |
| 1 | PRIMARY | d | NULL | eq_ref | PRIMARY | PRIMARY | 62 | hospital_information_system.p.Doc_id | 1 | 100.00 | NULL |
| 2 | UNION | p | NULL | ALL | NULL | NULL | NULL | NULL | 6 | 33.33 | Using where |
| 2 | UNION | d | NULL | eq_ref | PRIMARY | PRIMARY | 62 | hospital_information_system.p.Doc_id | 1 | 100.00 | NULL |
| 3 | UNION RESULT | <union1,2> | NULL | ALL | NULL | NULL | NULL | NULL | NULL | NULL | Using temporary |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set, 1 warning (0.00 sec)
```

- After Tuning

```
MySQL 8.0 Command Line Cli X + ×

5 rows in set, 1 warning (0.00 sec)

mysql> create index days_Ind on Prescription(No_of_days);
Query OK, 0 rows affected (0.03 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> show index from Prescription;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_Index | Column_name | collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| prescription | 0 | PRIMARY | 1 | Prescription_id | A | 6 | NULL | NULL | BTREE | | YES | NULL | |
| prescription | 1 | fk_PresDoc | 1 | Doc_id | A | 1 | NULL | NULL | BTREE | | YES | NULL |
| prescription | 1 | days_Ind | 1 | No_of_days | A | 4 | NULL | NULL | YES | BTREE | | YES | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> explain(select * from D_Pres
-> union
-> select * from D_Pres2);
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | p | NULL | range | fk_PresDoc,days_Ind | days_Ind | 5 | NULL | 3 | 100.00 | Using index condition; Using where |
| 1 | PRIMARY | d | NULL | eq_ref | PRIMARY | PRIMARY | 62 | hospital_information_system.p.Doc_id | 1 | 100.00 | NULL |
| 2 | UNION | p | NULL | range | days_Ind | days_Ind | 5 | NULL | 3 | 100.00 | Using index condition |
| 2 | UNION | d | NULL | eq_ref | PRIMARY | PRIMARY | 62 | hospital_information_system.p.Doc_id | 1 | 100.00 | NULL |
| 3 | UNION RESULT | <union1,2> | NULL | ALL | NULL | NULL | NULL | NULL | NULL | NULL | Using temporary |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set, 1 warning (0.00 sec)

mysql>
```

❖ Tuning of Nested Query 1

- Before Tuning

```
MySQL 8.0 Command Line Cli X + ×
Records: 0 Duplicates: 0 Warnings: 0
mysql> select Department_id from department where department_name
-> in(select specialization from doctor);
+-----+
| Department_id |
+-----+
| DE85         |
+-----+
1 row in set (0.00 sec)

mysql> show index from doctor;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| doctor | 0 | PRIMARY | 1 | Doctor_id | A | 5 | NULL | NULL | BTREE | | | YES | NULL | |
| doctor | 0 | Name_birth | 1 | First_name | A | 3 | NULL | NULL | BTREE | | | YES | NULL |
| doctor | 0 | Name_birth | 2 | Last_name | A | 4 | NULL | NULL | BTREE | | | YES | NULL |
| doctor | 0 | Name_birth | 3 | Date_of_birth | A | 5 | NULL | NULL | BTREE | | | YES | NULL |
| doctor | 1 | Province_Ind | 1 | Province | A | 4 | NULL | NULL | YES | BTREE | | | YES | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> explain(select Department_id from department where department_name
-> in(select specialization from doctor));
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | department | NULL | ALL | NULL | NULL | NULL | NULL | 4 | 100.00 | NULL |
| 1 | SIMPLE | doctor | NULL | ALL | NULL | NULL | NULL | spec_ind | 5 | 20.00 | Using where; FirstMatch(department); Using join buffer (hash join) |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)
```

- After Tuning

```
MySQL 8.0 Command Line Cli X + ×
2 rows in set, 1 warning (0.00 sec)
mysql> create index spec_ind on doctor(specialization);
Query OK, 0 rows affected (0.03 sec)
Records: 0 Duplicates: 0 Warnings: 0
mysql> show index from doctor;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| doctor | 0 | PRIMARY | 1 | Doctor_id | A | 5 | NULL | NULL | BTREE | | | YES | NULL | |
| doctor | 0 | Name_birth | 1 | First_name | A | 3 | NULL | NULL | BTREE | | | YES | NULL |
| doctor | 0 | Name_birth | 2 | Last_name | A | 4 | NULL | NULL | BTREE | | | YES | NULL |
| doctor | 0 | Name_birth | 3 | Date_of_birth | A | 5 | NULL | NULL | BTREE | | | YES | NULL |
| doctor | 1 | Province_Ind | 1 | Province | A | 4 | NULL | NULL | YES | BTREE | | | YES | NULL |
| doctor | 1 | spec_ind | 1 | Specialization | A | 5 | NULL | NULL | YES | BTREE | | | YES | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> explain(select Department_id from department where department_name
-> in(select specialization from doctor));
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | department | NULL | ALL | NULL | NULL | NULL | spec_ind | 163 | hospital_information_system.department.Department_name | 100.00 | Using where |
| 1 | SIMPLE | doctor | NULL | ref | spec_ind | spec_ind | 163 | hospital_information_system.department.Department_name | 1 | 100.00 | Using where; Using index; FirstMatch(department) |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)
mysql>
```

❖ Tuning of Nested Query 2

- Before Tuning

```
MySQL 8.0 Command Line Cli X + ×
mysql>
mysql> show index from nurse;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| nurse | 0 | PRIMARY | 1 | Nurse_id | A | 6 | NULL | NULL | BTREE | | | YES | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> select nurse_id, Last_name, Gender from nurse
-> where age > All(
-> select age from nurse where Gender = 'F');
+-----+-----+-----+
| nurse_id | Last_name | Gender |
+-----+-----+-----+
| N004 | Brown | M |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql> explain(select nurse_id, Last_name, Gender from nurse
-> where age > All(
-> select age from nurse where Gender = 'F'));
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | nurse | NULL | ALL | NULL | NULL | NULL | NULL | 6 | 66.67 | Using where |
| 2 | SUBQUERY | nurse | NULL | ALL | NULL | NULL | NULL | NULL | 6 | 16.67 | Using where |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)
```

- After Tuning

```
MySQL 8.0 Command Line Cli + X
Records: 0 Duplicates: 0 Warnings: 0
mysql> create index Gender_ind on nurse(Gender);
Query OK, 0 rows affected (0.03 sec)
Records: 0 Duplicates: 0 Warnings: 0
mysql> show index from nurse;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| nurse | 0 | PRIMARY | 1 | Nurse_id | A | 6 | NULL | NULL | YES | BTREE | | | YES | NULL |
| nurse | 1 | Gender_ind | 1 | Gender | A | 2 | NULL | NULL | YES | BTREE | | | YES | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)

mysql> explain(select nurse_id, Last_name, Gender from nurse
-> where age > All(
-> select age from nurse where Gender = 'F');
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | nurse | NULL | ALL | NULL | NULL | NULL | NULL | 6 | 66.67 | Using where |
| 2 | SUBQUERY | nurse | NULL | ref | Gender_ind | Gender_ind | 7 | const | 3 | 100.00 | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)
```

❖ Tuning of Nested Query 3

- Before Tuning

```
MySQL 8.0 Command Line Cli + X
Records: 0 Duplicates: 0 Warnings: 0
mysql> explain (select patient_id from patient where age
-> in
-> (select age from patient where age<=48));
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | patient | NULL | ALL | NULL | NULL | NULL | NULL | 6 | 100.00 | Using where |
| 1 | SIMPLE | patient | NULL | ALL | NULL | NULL | NULL | NULL | 6 | 16.67 | Using where; FirstMatch(patient); Using join buffer (hash join) |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)

mysql> show index from patient;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| patient | 0 | PRIMARY | 1 | patient_id | A | 6 | NULL | NULL | YES | BTREE | | | YES | NULL |
| patient | 1 | fname_ind | 1 | First_name | A | 5 | NULL | NULL | YES | BTREE | | | YES | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

- After Tuning

```
MySQL 8.0 Command Line Cli + X
mysql> create index age_ind on patient(age);
Query OK, 0 rows affected (0.05 sec)
Records: 0 Duplicates: 0 Warnings: 0
mysql> show index from patient;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| patient | 0 | PRIMARY | 1 | patient_id | A | 6 | NULL | NULL | YES | BTREE | | | YES | NULL |
| patient | 1 | fname_ind | 1 | First_name | A | 5 | NULL | NULL | YES | BTREE | | | YES | NULL |
| patient | 1 | age_ind | 1 | Age | A | 5 | NULL | NULL | YES | BTREE | | | YES | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)

mysql> explain (select patient_id from patient where age
-> in
-> (select age from patient where age<=48));
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | patient | NULL | range | age_ind | age_ind | 5 | NULL | 2 | 100.00 | Using where; Using index |
| 1 | SIMPLE | patient | NULL | ref | age_ind | age_ind | 5 | hospital_information_system.patient.Age | 1 | 100.00 | Using index; FirstMatch(patient) |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)
```