

Hospital Management Subsystem & Pharmacy Management Subsystem & Drug@FDA & FAERS subsystems

CS 571 Database Design Project, Spring 2023

Masters in

Department of Computer Science and Information Systems

Bradley University

By

Sudheer Kumar Kodali (349075)

Akhila Kongara (350200)

Purpose

This project's goal is to use our local MySQL to load data and implement the database tables.

By using the E-R diagram, four subsystems (databases) must be built.

We must list the functional dependencies that each relational schema satisfies by data.

With the Analysis of good data base, we have created 4 Sub- Systems in the local MYSQL.

1. Hospital Management Subsystem
2. Pharmacy Management Subsystem
3. Drug@FDA Subsystem
4. FDAAdverseEventReportingSystem(FAERS) Subsystem

INTRODUCTION

What is Database?

A database is a device for gathering and arranging data. Databases can store data about people, things, orders, and other things. Many databases begin as a list in a spreadsheet or word processing tool. Redundancies and inconsistencies start to show up in the data as the list gets longer. List style makes the data difficult to grasp, and there are few options for searching or extracting specific data sets for analysis. Transferring the data to a database made by a database management system (DBMS), such as Access, is a good solution once these issues start to arise.

- Add new data to a database, such as a new item in an inventory.
- Edit existing data in the database, such as changing the current location of an item.
- Delete information, perhaps if an item is sold or discarded.
- Organize and view the data in different ways.
- Share the data with others via reports, e-mail messages, an intranet, or the Internet.

About E-R Diagram?

An Entity Relationship (ER) Diagram is a form of flowchart that shows the relationships between "entities" like people, things, or concepts within a system. ER Diagrams are most frequently used in the disciplines of software engineering, business information systems, education, and research to build or troubleshoot relational databases. They are also known as ERDs or ER Models, and they use a predetermined collection of symbols to represent the interconnectedness of entities, relationships, and their qualities. These symbols include rectangles, diamonds, ovals, and connecting lines. They have verbs for relationships and nouns for entities, mirroring the grammatical framework.

Uses of entity relationship diagrams:

- Database design
- Database troubleshooting
- Business information systems
- Business process re-engineering (BPR)
- Education
- Research

Purpose of Project:

Our goal is to create four subsystems and combine them into a single E-R diagram. Bradley University must develop a database to manage our hospital, which includes a pharmacy, drug information (Drug@FDA), and at least one other department. local MYSQL, at least 25% of the FDA's currently available reports on adverse drug reactions (FAERS).

Report Requirements:

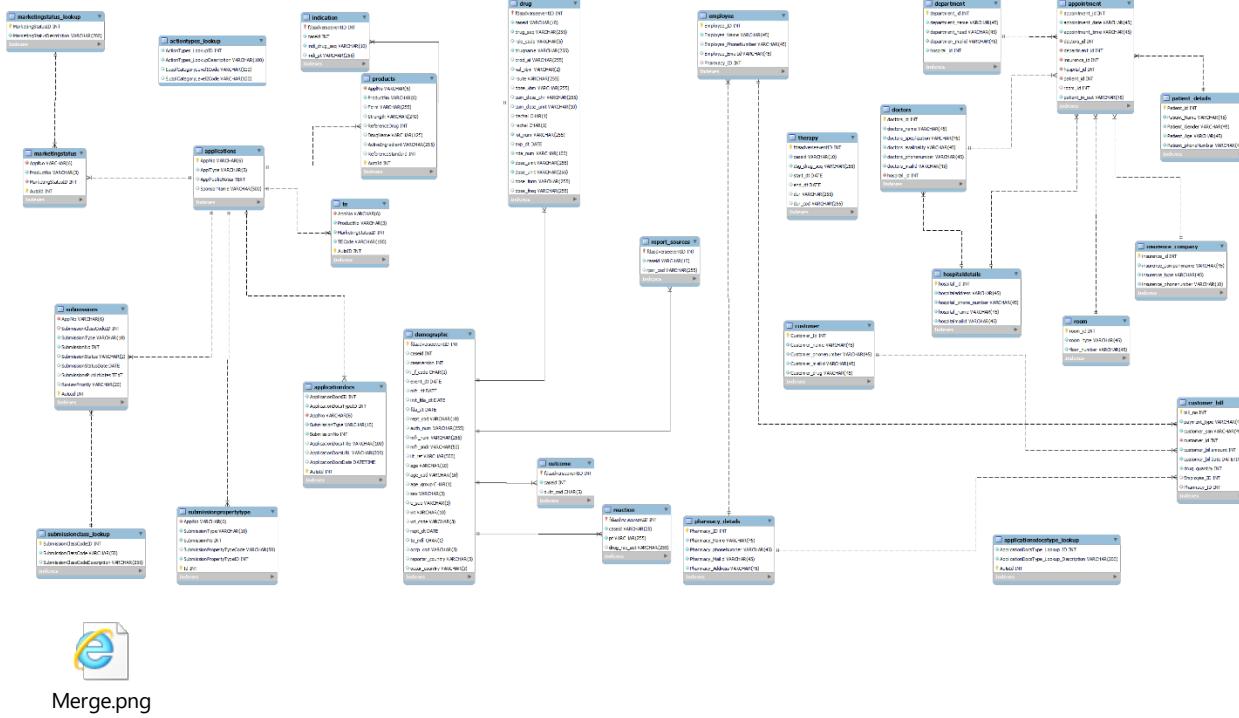
Question 1:

Initially, for the hospital management subsystem and pharmacy, I've constructed an E-R diagram with four subsystems.

How the management subsystem could be combined with the Data@FDA E-R and FAERS E-R, as well as what it might look like.

In order to combine these subsystems, we would need to establish links between the tables based on the shared entities and attributes between them. For instance, there may be a relationship between the drugs table in the pharmacy management subsystem and the drugs table in the Data@FDA E-R, as well as between the patient table in the hospital management subsystem and the adverse event table in the FAERS E-R.

With the combined E-R diagram, we would also need to make sure that there are no duplicate rows or columns. In order to do this, duplicate tables and columns across the subsystems would need to be removed, and related properties would need to be combined into a single table.



Merge.png

Question 2:

In this question, as per professor requirements we have formulated 10 realistic queries in English for every 4 Subsystems.

Hospital Management Sub System:

1. Write a query to display doctor name and their availability days who is done specialization on Gynecology?
2. write a query to display total male and female patient count who are visited hospital?
3. write a query to display doctor details and hospital name where they are working?
4. Write a query to display patient details, and their appointment details along with doctor name, department name & insurance details?
5. Write a query to display patients have appointments scheduled for a specific date?
6. Write a query to display list of patients who have consulted particular a doctor?

7. Write a query to display the patient details who visited hospital more than once?
8. Write a query to display count of patient's breakdown by consulted doctor name and hospital name?
9. Write a query to display doctor details and their availability days by hospital name?
10. Write a query to display patient details who are undergoing treatment?

Pharmacy Management Sub System:

1. Write a query to display customer details along with purchased amount and purchased drug quantity details?
2. Write a query to display employee and drug sales amount along with pharmacy name?
3. Write a query to display drugs sales amount by pharmacy name?
4. Write a query to display Employee details who is working in specific Pharmacy?
5. Write a query to display top 3 employees based on employee salary?
6. Write a query to display employee details whose salary between \$5000 to \$10000?
7. Write a query to display pharmacy sales amount by year wise?
8. Write a query to display employee details along with pharmacy details whose designation as Sales Manager?
9. Write a query to display total employee salary by employee designation?
10. Write a query to display drug sales amount by payment type in a specific month?

Drug@FDA Sub System:

1. Write a query to display all the submissions and their submission type code and description?
2. Write a query to display applications and their marketing status description?
3. Write a query to display the history of each drug products for a certain applicant?
4. Write a query to display the documents associated with a specific submission type?

5. Write a query to display the submission class codes and associated submission types?
6. Write a query to display the accessible property kinds in the database?
7. Write a query to display the application data for a certain submission ID, grouped by property type code in reverse chronological order?
8. Write a query to display what a given drug's indications are, and what dosages and strengths are available?
9. Write a query to display where all goods have a particular active component?
10. Write a query to display products with a specific application number?

FAERS Sub System:

1. Write a query to display total events by age group and gender breakdown for the events which were created from 2020?
2. Write a query to display events and drug details along with event date and patient age?
3. Write a query to display the age range and gender breakdown of individuals who had a particular medication reaction?
4. Write a query to display what age range is most frequently linked to a certain drug reaction in both male and female patients?
5. Write a query to display how many complaints deal with a certain dosage form of medication?
6. Write a query to display the most typical medication mentioned in reports of adverse events?
7. Write a query to display the particular reaction phrases that the database has on file?
8. Write a query to display the database's top 5 medication indications, stated there?
9. Write a query to display Several accounts mention the same medication and the same reaction?
10. Write a query to display how many reports came from a particular report source and a particular nation?

Question 3:

To answer this question, we established a database and tables and added data to them. From the E-R Diagram, we get the script.

Hospital:

The relationships between hospitals, patients, doctors, appointments, and other entities are shown graphically in a Hospital Management ER diagram.

The main elements of this ER diagram for the hospital management system include doctors, patients, and appointment times.

How a database table in a hospital administration system might operate:

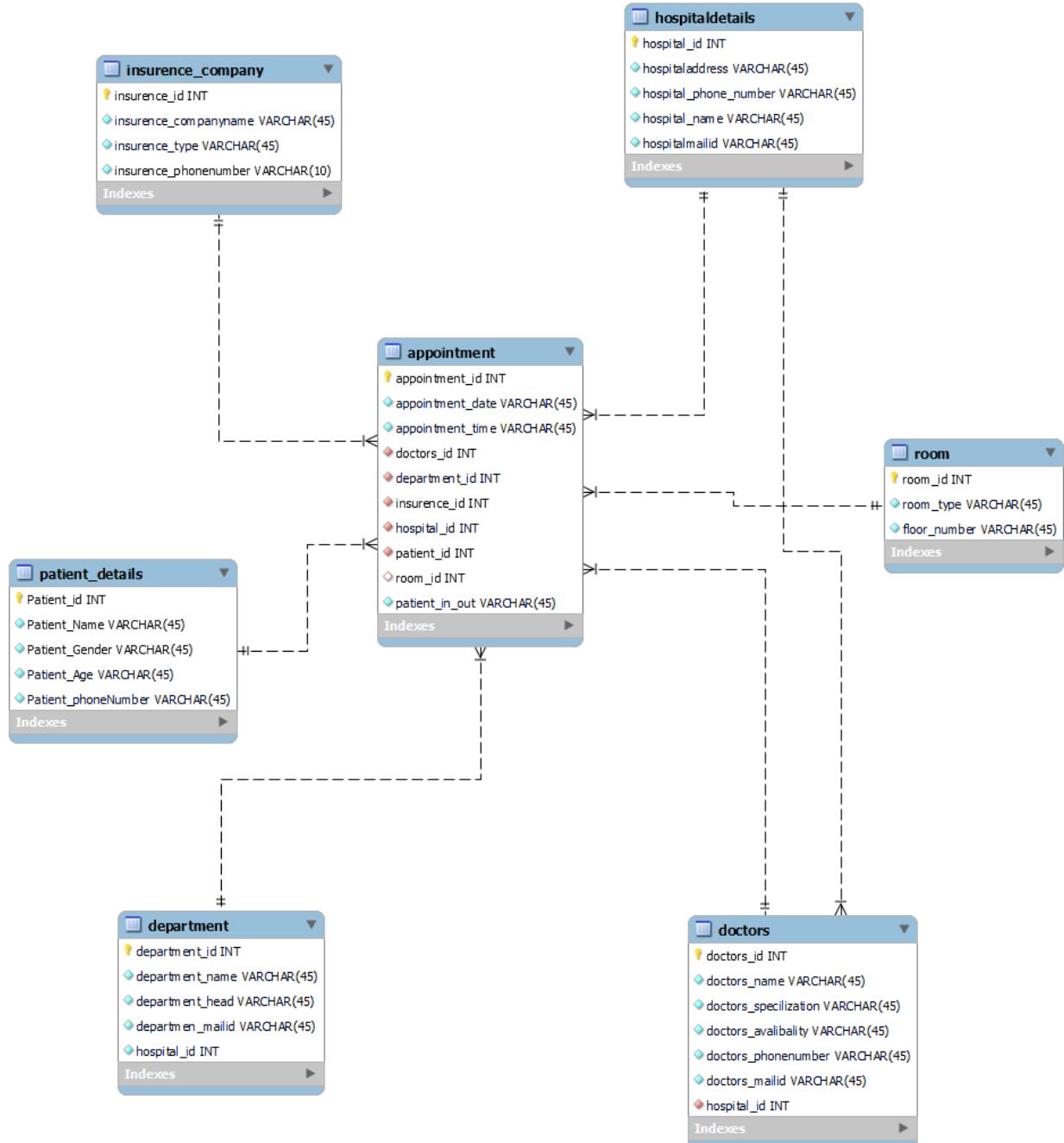
Assume we have a "patients" database with information on each hospital patient. The table may have columns with names like "patient_id," "first_name," "last_name," "gender," "phone number," "email address," "address," and others.

The hospital management system stores the patient's data in the "patients" table when a new patient is registered. In order to do this, a new row must be added to the table, and the patient's data must be entered into the appropriate columns. Each patient will have a unique identity thanks to the "patient_id" column being selected as the primary key.

The hospital administration system uses the patient's ID to find the patient's record in the "patients" table when a doctor asks information about a patient. This involves using SQL to query the table to obtain the patient data from the appropriate columns. When a patient needs to see a doctor, they can do so and enter the necessary details in the database table.

In the patient's table and prescription table, doctors can also prescribe medications and dosages for patients to take.

In general, the hospital management system database table functions by organizing and storing information on patients, staff, medical records, and other entities inside the hospital management system, and by providing tools for querying, updating, and modifying that information as necessary.



Pharmacy:

A pharmacy management system database is made up of a number of tables that work together to store and organize various types of information relevant to pharmacy operations, such as patient information, billing and payment information, prescription records, drug inventory information, and information on insurance and medications.

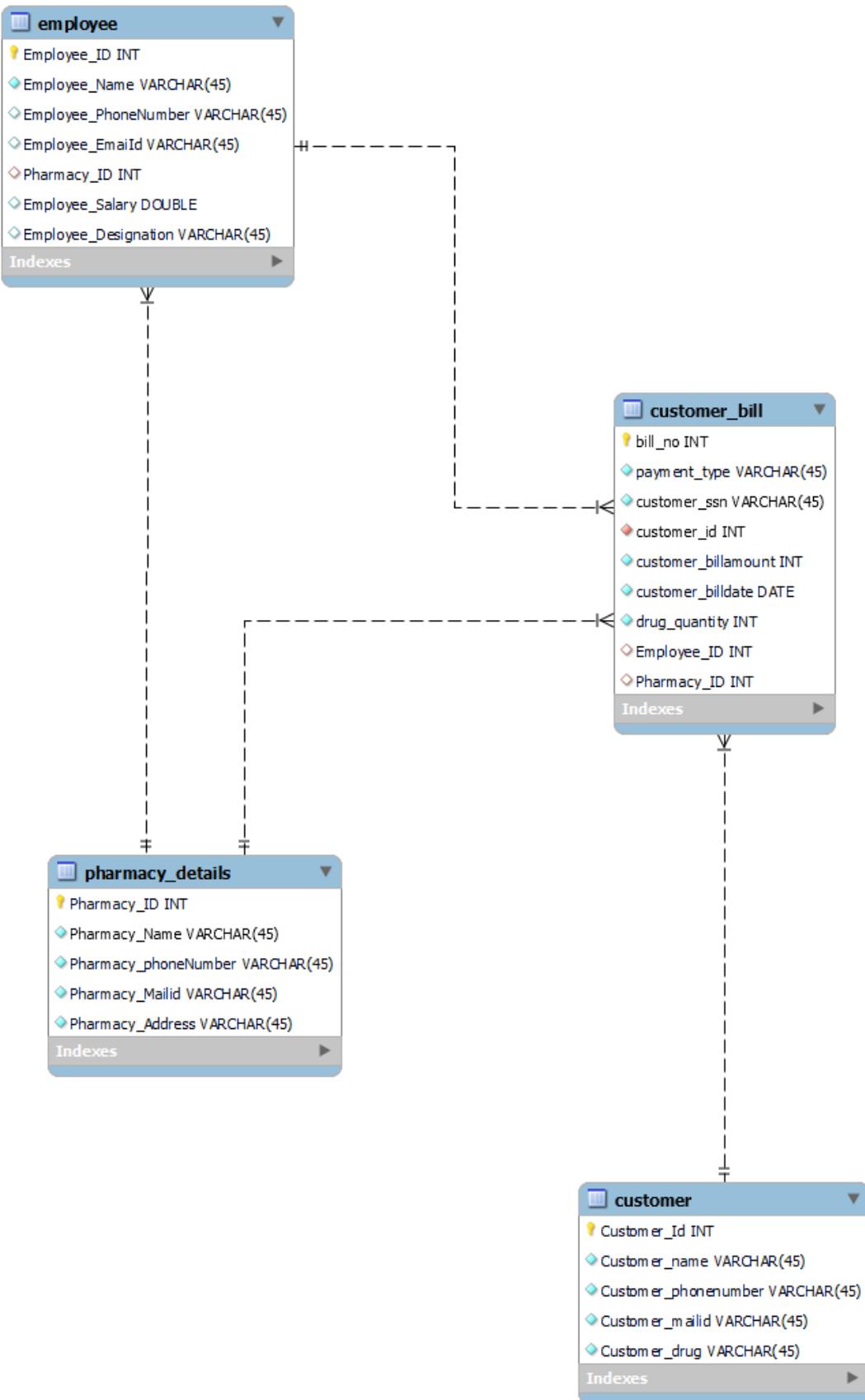
Here are some of the important tables included in a typical pharmacy management system database and an explanation of how they function:

Customer:

The names, SSNs, addresses, phone numbers, and insurance details of the patients who use the pharmacy's services are all kept in this table. This database is used to keep track of patient information and make sure they are given the right meds in the right amounts.

Customer Bill:

Information regarding pharmaceutical transactions is kept in this table, including the order ID, total amount paid, payment made to that customer, insurance payment, and customer SNN. This database is used by the pharmacy to create invoices and manage financial data.



Drug @FDA Sub System E-R Diagram:

Information about adverse events and medication errors that have been reported to the FDA can be found in a database called the FDA Adverse Event Reporting System (FAERS). It is a database including data on the patient's demographics, medical background, and history of medication use.

The data is organized in tables, and the database tables are briefly described as follows:

Applications table:

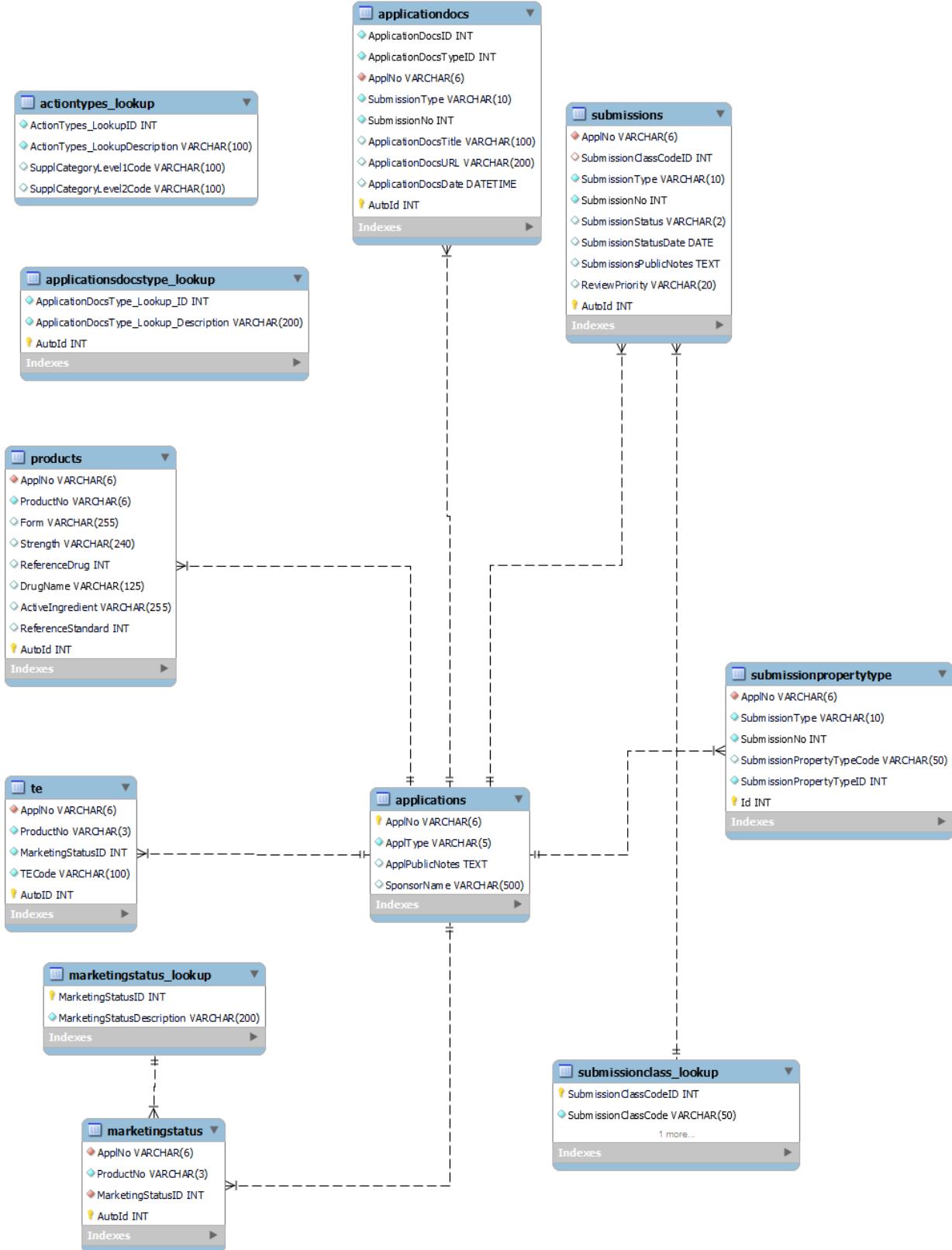
Information regarding drug applications is provided in this table, including the application number, application type, and sponsor name.

Product table:

This table contains links to the application documents, including the applicationdocsID, applicationdocstypeID, application number, submission type, submission number, applicationdocstitle, applicationdocsURL, applicationdocsdate.

Submissions table:

Links to the submissions are provided in this table, together with the application number, SubmissionClassCodeID, Submissiontype, SubmissionID, and SubmissionStatus.



FDA Adverse Event Reporting System (FAERS) Quarterly Data Subsystem

Information about adverse events and medication errors that have been reported to the FDA can be found in a database called the FDA Adverse Event Reporting System (FAERS). It is a database including data on the patient's demographics, medical background, and history of medication use.

The data is organized in tables, and the database tables are briefly described as follows:

Demographic:

The patient's age, gender, and race are all listed in this table of demographic data.

Drug:

The drug name, dosage, mode of administration, start and finish dates, and other details about the medication that was mentioned in the adverse event are included in this table.

Indication:

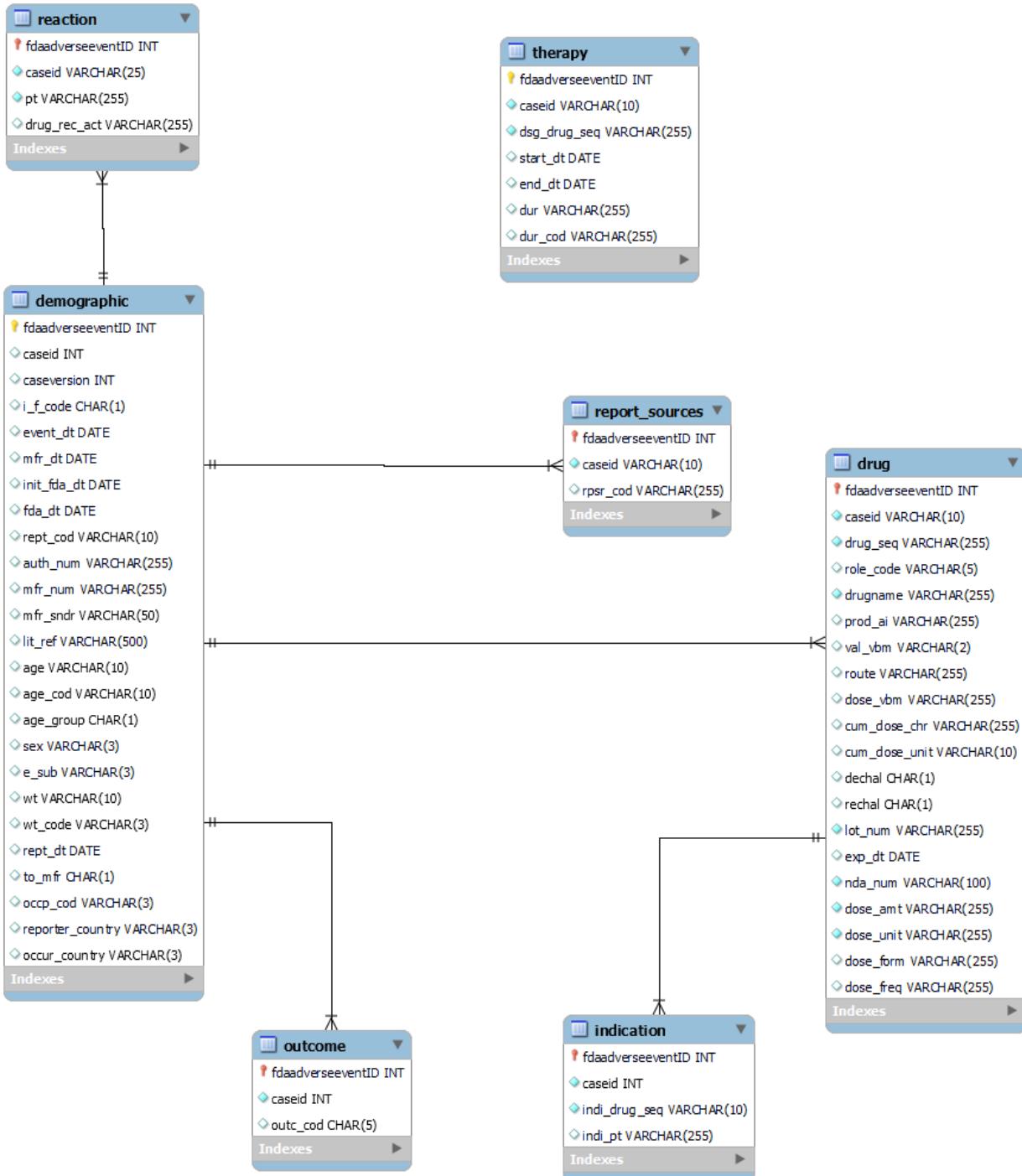
The medical problem or symptom that the medication was meant to treat is listed in this table along with other details about why it was prescribed.

Outcome:

The outcome of the adverse event is detailed in this table, including whether the patient recovered, continued to experience symptoms, or passed away.

Reaction:

This table includes details about the adverse event itself, such as the patient's symptoms, the severity of the incident, and the time it happened. With regard to adverse events and medication errors, the FAERS Quarterly Data Subsystem 4 in MySQL offers a plethora of data that may be used to enhance patient care and drug safety.

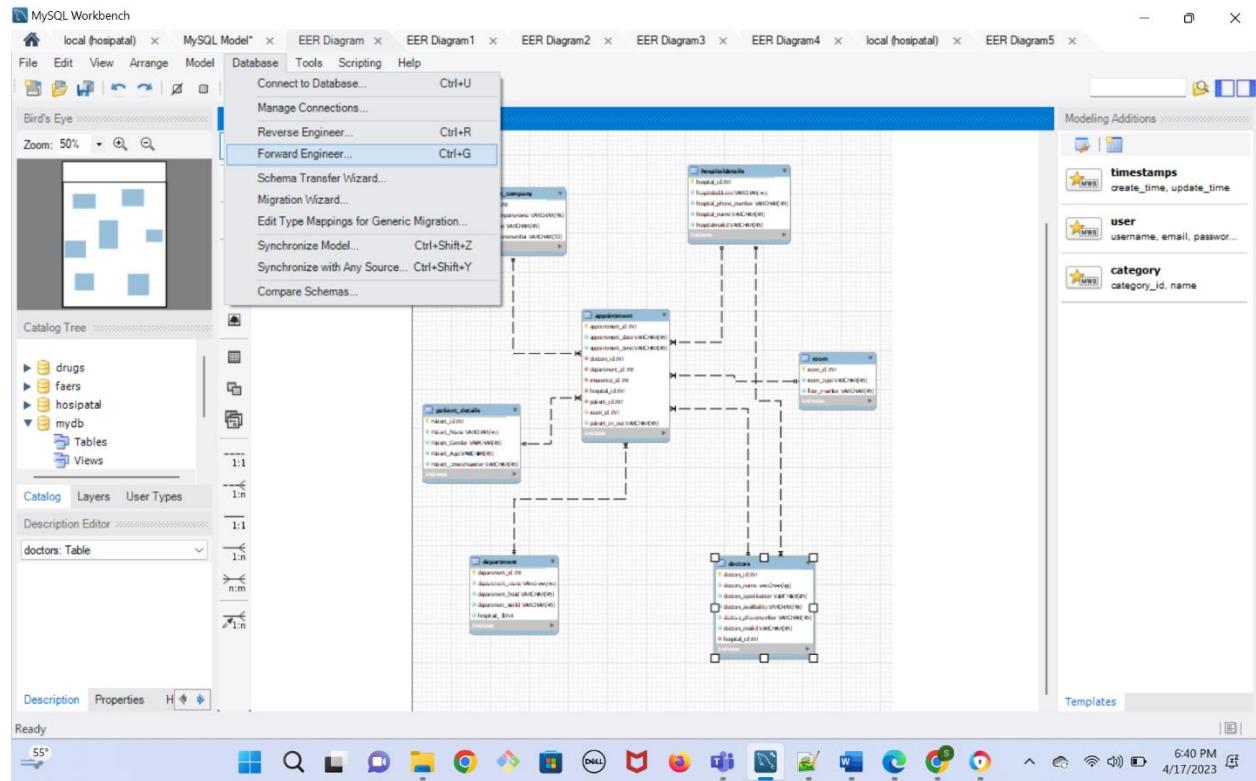


Question 4:

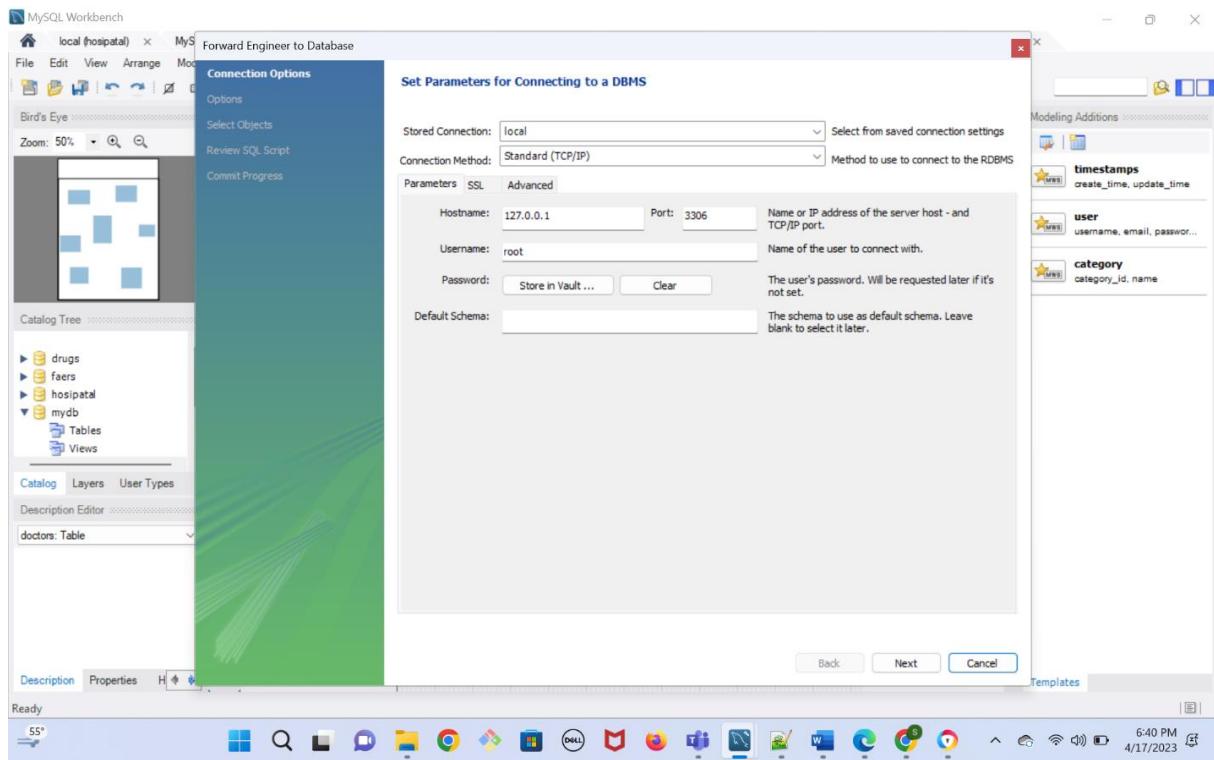
The four E-R diagrams in this question have been transformed into a relational database schema in accordance with the specifications.

Relational DB Schema to E-R Diagram for Hospital Management Sub System

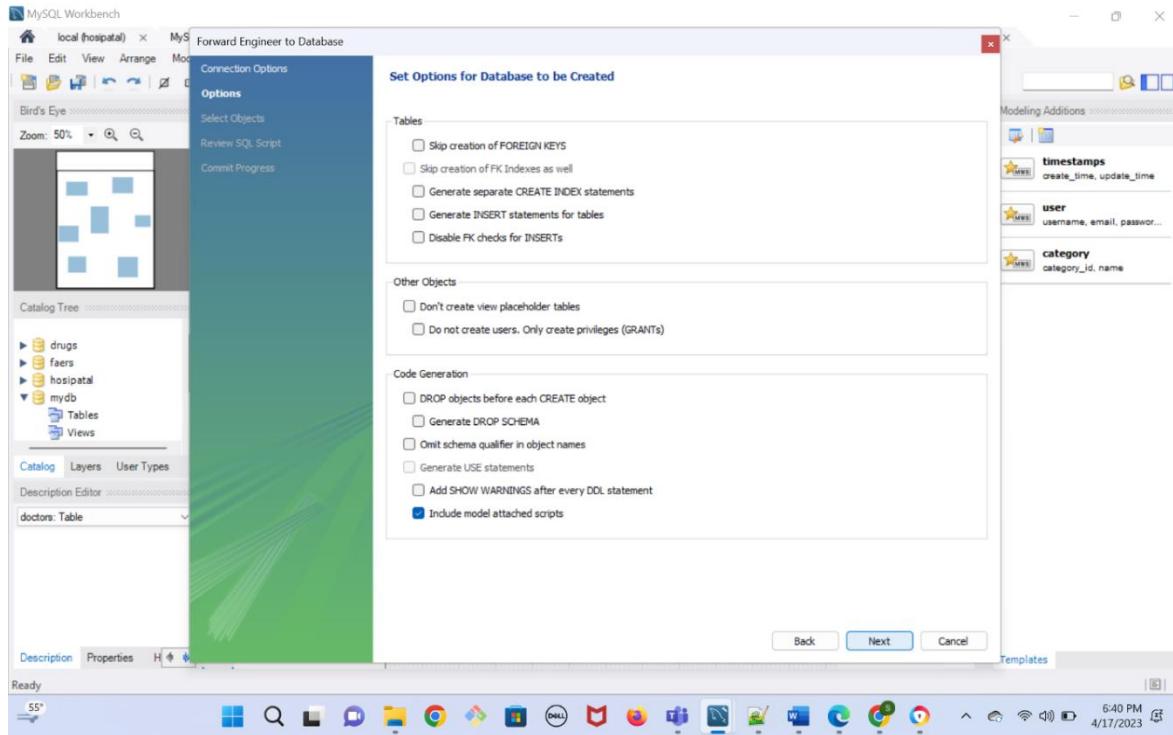
Step 1: Select Forward Engineer under Database



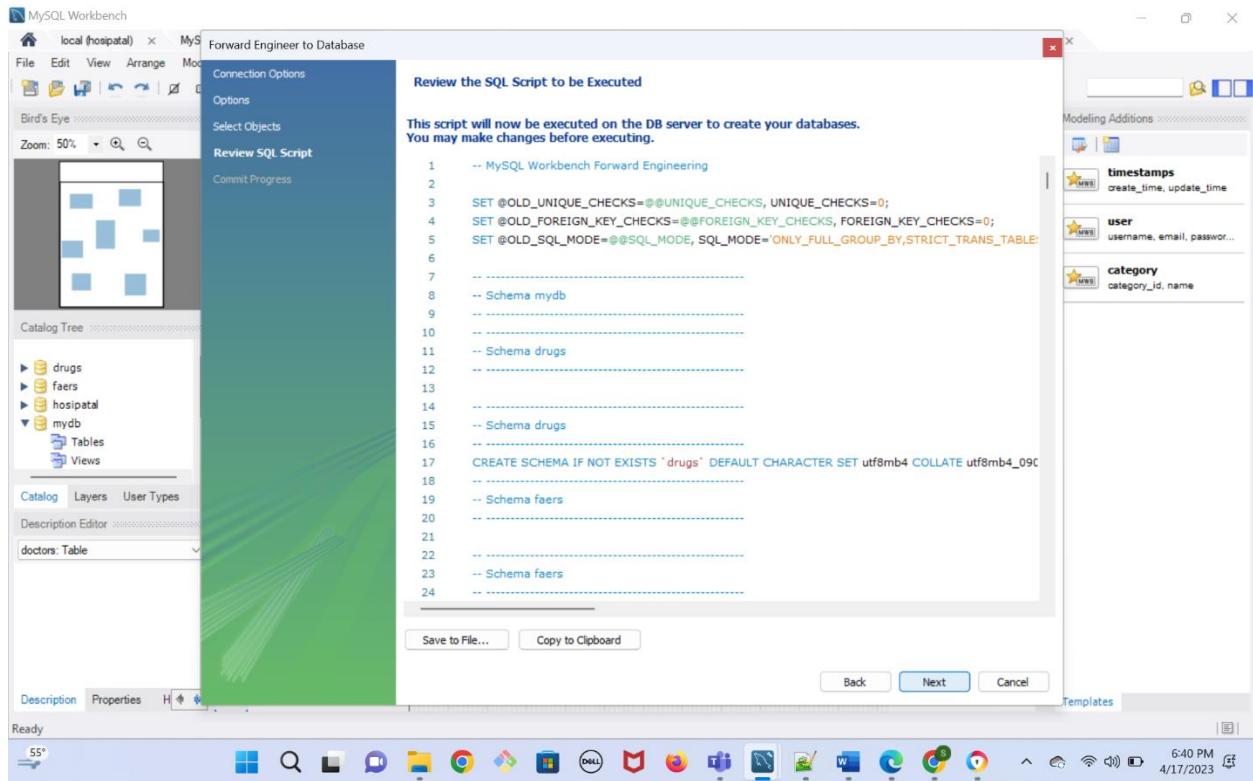
Step 2: Click Next Button



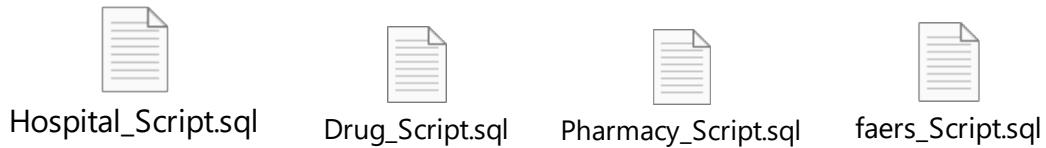
Step 3: We need only table objects so need to checkout that option



Step 4: Now we can review the SQL Script and Run it in the SQL



We need to Repeat the process for all data bases like Pharmacy, Drug and Fares



Question 5:

We have included information on tables that we have made for 4 Sub Systems in this query. I'm merely providing a couple screenshots of tables for reference.

Hospital, Pharmacy, Drug, FAERS Sub System

Step 1: Created a Hospital, Pharmacy, Drug, FAERS Sub Systems in MySQL

The screenshot shows the MySQL Workbench interface with the 'pharmacy_details' table selected in the Navigator pane. The main pane displays the results of the query: 'SELECT * FROM pharmacy.pharmacy_details;'. The results grid shows 10 rows of data with columns: Pharmacy_ID, Pharmacy_Name, Pharmacy_phoneNumber, Pharmacy_MailId, and Pharmacy_Address. The data includes various pharmacies like SPH, walgreens, Devi, sree, Walmart, PHS, SRS, Apolo, HSP, and NRI, located in Peoria1 through Peoria9. The bottom status bar shows the date and time as 4/17/2023 6:52 PM.

Pharmacy_ID	Pharmacy_Name	Pharmacy_phoneNumber	Pharmacy_MailId	Pharmacy_Address
12344	SPH	3099895792	SPH@gmail.com	Peoria7
34507	walgreens	3099895798	walgreens@gmail.com	Peoria3
34567	Devi	3099895797	Devi@gmail.com	Peoria1
34587	sree	3099895796	sree@gmail.com	Peoria4
45677	Walmart	3099895799	Walmart@gmail.com	Peoria2
45678	PHS	3099895775	PHS@gmail.com	Peoria8
87687	SRS	3099895794	SRS@gmail.com	Peoria6
88765	Apolo	3099895792	Apolo@gmail.com	Peoria
98767	HSP	3099895714	HSP@gmail.com	Peoria9
98798	NRI	3099895795	NRI@gmail.com	Peoria5
*	NULL	NULL	NULL	NULL

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- local (hospital)
- MySQL Model*
- EER Diagram
- EER Diagram1
- EER Diagram2
- EER Diagram3
- EER Diagram4
- local (hospital)
- EER Diagram5

Tables

faers

- Tables
- Views
- Stored Procedures
- Functions

drugs

pharmacy

sakila

sys

Administration Schemas

Information

Schema: faers

demographic

Result Grid

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```
1 • SELECT * FROM faers.demographic;
```

fdaadverseeventID	casid	caseversion	lf_code	event_dt	mfr_dt	init_fda_dt	fda_dt	rept_cod
100115733	10011573	3	F	2013-08-01	2022-11-18	2014-03-14	2022-11-23	EXP
100234223	10023422	3	F	2012-03-01	2022-10-03	2014-03-20	2022-10-07	EXP
100260594	10026059	4	F	2012-01-01	2022-11-24	2014-03-20	2022-11-30	EXP
100518358	10051835	8	F	2013-07-15	2022-11-25	2014-04-02	2022-12-01	EXP
100746869	10074686	9	F	2009-07-14	2022-12-22	2014-04-14	2022-12-30	EXP
100832487	10083248	7	F	2012-02-01	2022-11-08	2014-04-17	2022-11-16	EXP
100900002	10090000	2	F	2014-01-01	2022-10-21	2014-04-21	2022-10-27	EXP
101397357	10139735	7	F	2014-01-01	2022-10-08	2014-04-29	2022-10-12	EXP
101530275	10153027	5	F	2013-10-01	2022-09-26	2014-05-05	2022-10-07	EXP
101834686	10183468	6	F	2013-12-16	2022-10-03	2014-05-20	2022-10-04	EXP
102158494	10215849	4	F	2010-01-01	2022-11-30	2014-06-04	2022-12-07	PER
102184243	10218424	3	F	2012-10-03	2022-12-13	2014-06-05	2022-12-22	EXP
102404993	10240499	3	F	2012-07-23	2022-10-03	2014-06-17	2022-10-06	EXP

demographic 1 ×

Object Info Session

Output

Query Completed

6:53 PM 4/17/2023

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- local (hospital)
- MySQL Model*
- EER Diagram
- EER Diagram1
- EER Diagram2
- EER Diagram3
- EER Diagram4
- local (hospital)
- EER Diagram5

Tables

faers

- Tables
- Views
- Stored Procedures
- Functions

drugs

pharmacy

sakila

sys

Administration Schemas

Information

Schema: faers

actiontypes_id

Result Grid

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```
1 • SELECT * FROM drugs.actiontypes_lookup;
```

ActionTypes_LookupID	ActionTypes_LookupDescription	SuplCategoryLevel1Code	SuplCategoryLevel2Code
3444	1	1	AA
5213	1	1	AA
5213	2	1	AA
5378	2	1	AA
5929	1	1	AP
6035	4	1	AP
6188	1	1	BD
6488	2	1	AP
6488	4	1	AP
6488	5	1	AP
6488	7	1	AP
6488	8	1	AP
6488	10	1	AP
6488	12	1	AP

types_lookup 1 ×

Object Info Session

Output

Query Completed

6:53 PM 4/17/2023

Step 2: Pharmacy Management Sub System

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- reaction
- report_sources
- therapy
- Views
- Stored Procedures
- Functions
- hospital**
- pharmacy
- Tables
- customer
- customer_bill
- employee
- pharmacy_details
- Views
- Stored Procedures
- Functions
- sakila
- sys

Administration Schemas

Information

Table: appointment

Columns:

- appointment_id int PK
- appointment_date varchar(45)
- appointment_time varchar(45)
- doctors_id int
- department_id int
- insurance_id int
- hospital_id int
- patient_id int
- room_id int

Object Info Session

Query Completed

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

customer 1 | Output | Context Help | Snippets

Customer_ID Customer_Name Customer_PhoneNumber Customer_MailId Customer_Drug

101	Nancy	3097762427	nancy1@gmail.com	Melatonin
102	Josh	3097762457	joshvhj@gmail.com	Sublocade
103	Anand	3097762456	anands7@gmail.com	Lisinopril
104	Joe	3097765488	joe23456@gmail.com	Ativan
105	Amber	3097762567	amberdrd@gmail.com	Tramadol
106	Sam	3097762452	san567@gmail.com	Onpatro
107	Toby	3097764461	toby899@gmail.com	Rybelus
108	Hannah	3097762123	hannah@gmail.com	Acetaminophen
109	Michael	3097712338	michael678@gmail.com	Glenya
110	David	3097756789	david123@gmail.com	Gabapentin
*	NULL	NULL	NULL	NULL

customer 1 | Output | Context Help | Snippets

6:54 PM 4/17/2023

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- drugs
- faers
- hospital**
- pharmacy
- sakila
- sys

Administration Schemas

Information

Schema: hospital

Table: pharmacy_details

Columns:

- Pharmacy_ID int PK
- Pharmacy_Name varchar(45)
- Pharmacy_PhoneNumber varchar(45)
- Pharmacy_MailId varchar(45)
- Pharmacy_Address varchar(45)

Object Info Session

Query Completed

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

pharmacy_details 1 | Output | Context Help | Snippets

Pharmacy_ID Pharmacy_Name Pharmacy_PhoneNumber Pharmacy_MailId Pharmacy_Address

12344	SPH	3099895792	SPH@gmail.com	Peoria7
34507	walgreens	3099895798	walgreens@gmail.com	Peoria3
34567	Devi	3099895797	Devi@gmail.com	Peoria1
34587	sree	3099895796	sree@gmail.com	Peoria4
45677	Walmart	3099895799	Walmart@gmail.com	Peoria2
45678	PHS	3099895775	PHS@gmail.com	Peoria8
87687	SRS	3099895794	SRS@gmail.com	Peoria6
88765	Apolo	3099895792	Apolo@gmail.com	Peoria
98767	HSP	3099895714	HSP@gmail.com	Peoria9
98798	NRI	3099895795	NRI@gmail.com	Peoria5
*	NULL	NULL	NULL	NULL

pharmacy_details 1 | Output | Context Help | Snippets

6:52 PM 4/17/2023

Step 3: Drug Sub System

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

drugs

- Tables
 - actiontypes_lookup
 - applicationdocs
 - applications
 - applicationsdoctype_lookup
 - marketingstatus
 - marketingstatus_lookup
 - products
 - submissionclass_lookup
 - submissionpropertytype
 - submissions
 - te
- Views
- Stored Procedures
- Functions

Fees

Administration Schemas

Information

Schema: drugs

Object Info Session

Query Completed

Result Grid

SQL Additions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```
1 • SELECT * FROM drugs.applications;
```

AppNo	AppType	AppPublicNotes	SponsorName
10010	NDA		BAAYER PHARMS
10012	NDA		LILLY
10021	NDA		ABBVIE
10028	NDA		WYETH AYERST
10040	NDA		BRACCO
10060	NDA		CASPER PHARMA LLC
10093	NDA		UCB INC
10104	NDA		BAUSCH
101063	BLA		MERCK
10124	NDA		LANNETT
10151	NDA		PARKER DAVIS
10155	NDA		SANOFI AVENTIS US
10187	NDA		NOVARTIS
101995	BLA		SMITH AND NEPHEW

Context Help Snippets

Output

Object Info Session

Query Completed

56° 7:03 PM 4/17/2023

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

drugs

- Tables
 - actiontypes_lookup
 - applicationdocs
 - applications
 - applicationsdoctype_lookup
 - marketingstatus
 - marketingstatus_lookup
 - products
 - submissionclass_lookup
 - submissionpropertytype
 - submissions
 - te
- Views
- Stored Procedures
- Functions

Fees

Administration Schemas

Information

Schema: drugs

Object Info Session

Query Completed

Result Grid

SQL Additions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```
1 • SELECT * FROM drugs.products;
```

AppNo	ProductNo	Form	Strength	ReferenceDrug	DrugName	Actv
4	4	SOLUTION/DROPS;OPHTHALMIC	1%	0	PAREDRINE	HYDR
159	1	TABLET;ORAL	500MG	0	SULFAPYRIDINE	SULF
552	1	INJECTABLE;INJECTION	20000 UNITS/ML	0	LIQUAMIN SODIUM	HEPA
552	2	INJECTABLE;INJECTION	40000 UNITS/ML	0	LIQUAMIN SODIUM	HEPA
552	3	INJECTABLE;INJECTION	5000 UNITS/ML	0	LIQUAMIN SODIUM	HEPA
552	4	INJECTABLE;INJECTION	1000 UNITS/ML	0	LIQUAMIN SODIUM	HEPA
552	5	INJECTABLE;INJECTION	10000 UNITS/ML	0	LIQUAMIN SODIUM	HEPA
552	7	INJECTABLE;INJECTION	100 UNITS/ML	0	LIQUAMIN LOCK FLUSH	HEPA
552	8	INJECTABLE;INJECTION	1000 UNITS/ML	0	HEPARIN SODIUM	HEPA
552	9	INJECTABLE;INJECTION	5000 UNITS/ML	0	HEPARIN SODIUM	HEPA
552	10	INJECTABLE;INJECTION	10000 UNITS/ML	0	HEPARIN SODIUM	HEPA
552	11	INJECTABLE;INJECTION	1000 UNITS/ML	0	LIQUAMIN SODIUM PR...	HEPA
552	12	INJECTABLE;INJECTION	5000 UNITS/ML	0	LIQUAMIN SODIUM PR...	HEPA

Context Help Snippets

Output

Object Info Session

Query Completed

56° 7:04 PM 4/17/2023

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Marketingstatus_lookup applicationdocs - Table applicationdocs - Table applicationdocs - Table applicationdocs - Table applicationdocs - Table

Don't Limit

1 • SELECT * FROM drugs.applicationdocs;

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | SQL Additions | Result Grid

ApplicationDocsID	ApplicationDocsTypeID	AppNo	SubmissionType	SubmissionNo	ApplicationDocTitle	ApplicationDocURL
1	1	4782	SUPPL	125	0	http://www.accessdata.fda.gov/fdacards/
2	1	4782	SUPPL	128	0	http://www.accessdata.fda.gov/fdacards/
3	1	4782	SUPPL	130	0	http://www.accessdata.fda.gov/fdacards/
4	1	4782	SUPPL	138	0	http://www.accessdata.fda.gov/fdacards/
5	1	4782	SUPPL	141	0	http://www.accessdata.fda.gov/fdacards/
6	1	4782	SUPPL	142	0	http://www.accessdata.fda.gov/fdacards/
7	1	4782	SUPPL	147	0	http://www.accessdata.fda.gov/fdacards/
8	1	4782	SUPPL	164	0	http://www.accessdata.fda.gov/fdacards/
9	8	4782	SUPPL	167	0	http://www.accessdata.fda.gov/fdacards/
10	1	4782	SUPPL	171	0	http://www.accessdata.fda.gov/fdacards/
11	1	5010	SUPPL	47	0	http://www.accessdata.fda.gov/fdacards/
12	1	5010	SUPPL	49	0	http://www.accessdata.fda.gov/fdacards/
13	1	5010	SUPPL	50	0	http://www.accessdata.fda.gov/fdacards/

applicationdocs 1 × Apply Revert Context Help Snippets

Object Info Session

Output

Query Completed

56° 7:04 PM 4/17/2023

Step 3: FDA FAERS Sub System

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

faers

Tables

demographic

Marketingstatus_lookup applicationdocs - Table applicationdocs - Table applicationdocs - Table applicationdocs - Table applicationdocs - Table

Don't Limit

1 • SELECT * FROM faers.demographic;

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | SQL Additions | Result Grid

faadverseeventID	cased	caseversion	lf_code	event_dt	mfr_dt	init_fda_dt	fda_dt	rept_cod
10011573	10011573	3	F	2013-08-01	2022-11-18	2014-03-14	2022-11-23	EXP
100234223	100234223	3	F	2012-03-01	2022-10-03	2014-03-20	2022-10-07	EXP
100260594	100260594	4	F	2012-01-01	2022-11-24	2014-03-20	2022-11-30	EXP
100518358	100518358	8	F	2013-07-15	2022-11-25	2014-04-02	2022-12-01	EXP
100746869	100746869	9	F	2009-07-14	2022-12-22	2014-04-14	2022-12-30	EXP
100832487	100832487	7	F	2012-02-01	2022-11-08	2014-04-17	2022-11-16	EXP
100900002	100900002	2	F	2014-01-01	2022-10-21	2014-04-21	2022-10-27	EXP
101397357	101397357	7	F	2014-01-01	2022-10-08	2014-04-29	2022-10-12	EXP
101530275	101530275	5	F	2013-10-01	2022-09-26	2014-05-05	2022-10-07	EXP
101834686	101834686	6	F	2013-12-16	2022-10-03	2014-05-20	2022-10-04	EXP
102158494	102158494	4	F	2010-01-01	2022-11-30	2014-06-04	2022-12-07	PER
102184243	102184243	3	F	2012-10-03	2022-12-13	2014-06-05	2022-12-22	EXP
102404993	102404993	3	F	2012-07-23	2022-10-03	2014-06-17	2022-10-06	EXP

demographic 1 × Apply Revert Context Help Snippets

Object Info Session

Output

Query Completed

56° 7:05 PM 4/17/2023

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- Submissions
- te
- Views
- Stored Procedures
- Functions
- faers**
- Tables
- demographic
- drug
- indication
- outcome
- reaction
- report_sources
- therapy
- Views
- Stored Procedures
- Functions

applicationdocs - Table applicationdocs - Table applicationdocs - Table applicationdocs demographic drug

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

1 • SELECT * FROM faers.drug;

fdaadverseeventID	cased	drug_seq	role_code	drugname	prod_ai	val_v
100115733	10011573	1	PS	JANUVIA	SITAGLIPTIN PHOSPHATE	1
100234223	10023422	1	PS	BONIVA	IBANDRONATE SODIUM	1
100260594	10026059	13	PS	LEVOFLOXACIN	LEVOFLOXACIN	1
100518358	10051835	1	PS	ACTEMRA	TOCLIZUMAB	1
100746869	10074686	1	PS	XOLAIR	OMALIZUMAB	1
100832487	10083248	1	PS	XOLAIR	OMALIZUMAB	1
101397357	10139735	4	C	MELATONIN	MELATONIN	1
1002130537	10021305	1	PS	SANDOSTATIN	OCTREOTIDE ACETATE	1
1006401873	10064018	4	SS	ACTEMRA	TOCLIZUMAB	1
1009456310	10094563	2	SS	PRISTIQ EXTENDED-RELEASE	DESVENLAFAKINE SUCCINATE	1
1014222242	10142222	3	SS	ACTEMRA	TOCLIZUMAB	1
1015212328	10152123	2	SS	XOLAIR	OMALIZUMAB	1

drug 1 x

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Result Grid | Form Editor | Field Types | Query Stats | Context Help | Snippets

Output

Query Completed

56° 7:05 PM 4/17/2023

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- Submissions
- te
- Views
- Stored Procedures
- Functions
- faers**
- Tables
- demographic
- drug
- indication
- outcome
- reaction
- report_sources
- therapy
- Views
- Stored Procedures
- Functions

applicationdocs demographic drug indication outcome reaction report_sources therapy

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

1 • SELECT * FROM faers.therapy;

fdaadverseeventID	cased	dsg_drug_seq	start_dt	end_dt	dur	dur_cod
100115733	10011573	1	2012-04-07	2012-04-09	63	DAY
100234223	10023422	1	2010-07-12	2013-12-19	3	YR
100260594	10026059	2	2010-07-12	2013-12-19	HULL	HULL
100518358	10051835	1	2010-07-12	2013-12-19	541	DAY
100746869	10074686	1	2010-07-12	2013-12-19	HULL	HULL
100832487	10083248	1	2010-07-12	2013-12-19	HULL	HULL
100900002	10090000	1	2010-07-12	2013-12-19	HULL	HULL
101397357	10139735	1	2010-07-12	2013-12-19	3723	DAY
101530275	10153027	1	2010-07-12	2013-12-19	HULL	HULL
101834686	10183468	1	2010-07-12	2013-12-19	HULL	HULL
102158494	10215849	1	2010-07-12	2013-12-19	HULL	HULL
102184243	10218424	1	2010-07-12	2013-12-19	HULL	HULL
102404993	10240499	1	2010-07-12	2013-12-19	4	YR
102686218	10268621	2	2010-07-12	2013-12-19	HULL	HULL

therapy 1 x

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Result Grid | Form Editor | Field Types | Query Stats | Context Help | Snippets

Output

Query Completed

56° 7:05 PM 4/17/2023

Question 6:

All the queries in this question have been written using relational algebra.

Query 1:

```
SELECT doc.doctors_id,  
       doc.doctors_name,  
       doc.doctors_availability  
FROM hospital.doctors doc  
      where doctors_specilization = "Gynecologists";
```

Using Relational Algebra:

```
 $\pi_{\text{doc.doctors\_id}, \text{doc.doctors\_name}, \text{doc.doctors\_availability}}(\sigma_{\text{doc.doctors\_specilization} = \text{"Gynecologists"}(\text{doc})})$ 
```

Explanation:

σ Gynecologists (doc.doctors specilization = "Gynecologists"): Choose every row from the "doc" table where "Gynecologists" is the value of the "doctors specilization" column.

π The "doctors id," "doctors name," and "doctors availability" columns from the generated table are the only ones to be projected.

Query 2:

```
SELECT Patient_Gender,  
       count(Patient_Gender)  
FROM hospital.patient_details  
      group by Patient_Gender;
```

Using Relational Algebra:

```
 $\pi$  Patient_Gender, count(Patient_Gender)( $\gamma$  Patient_Gender;  
count(Patient_Gender): (hosipatal.patient_details))
```

Explanation:

γ Patient Details.patient details; count(Patient Details); Patient Gender Count the instances of each gender in the "patient details" table by grouping by the "Patient Gender" column.

π the patient's gender, count(Patient Gender): From the generated table, only the "Patient Gender" and "count(Patient Gender)" columns should be projected.

Query 3:

```
SELECT doc.doctors_id,
```

```
    doc.doctors_name,
```

```
    doc.doctors_availability,
```

```
    hos.hospital_name,
```

```
    doc.doctors_phonenumber
```

```
FROM hosipatal.doctors doc left join hosipatal.hospitaldetails hos
```

```
    on doc.hospital_id = hos.hospital_id;
```

Using the Relational Algebra:

```
 $\pi$  doc.doctors_id, doc.doctors_name, doc.doctors_availability, hos.hospital_name,  
doc.doctors_phonenumber ((doc  $\bowtie$  doc.hospital_id = hos.hospital_id hos))
```

Explanation:

$doc \bowtie doc.hospital_id = hos.hospital_id hos$: Perform a left outer join between the "doctors" table (doc) and the "hospitaldetails" table (hos) on the condition that the "hospital_id" column in "doc" matches the "hospital_id" column in "hos".

π doc.doctors_id, doc.doctors_name, doc.doctors_availability, hos.hospital_name, doc.doctors_phonenumber: Project only the "doctors_id", "doctors_name",

"doctors_availability", "hospital_name", and "doctors_phonenumber" columns from the resulting table.

Query 4:

```
SELECT pat.Patient_Name,  
       pat.Patient_Gender,  
       pat.Patient_phoneNumber,  
       apt.appointment_id,  
       apt.appointment_date,  
       apt.patient_in_out,  
       rm.floor_number,  
       doc.doctors_name,  
       dept.department_name,  
       ins.insurance_companyname  
  
FROM hosipatal.appointment apt left join hosipatal.patient_details pat on  
apt.patient_id = pat.Patient_id  
  
left join hosipatal.doctors doc on apt.doctors_id = doc.doctors_id  
  
left join hosipatal.department dept on apt.department_id = dept.department_id  
  
left join hosipatal.insurance_company ins on apt.insurance_id = ins.insurance_id  
  
left join hosipatal.room rm on apt.room_id = rm.room_id;
```

Using Relational Algebra:

```

$$\pi_{\text{pat.Patient\_Name}, \text{pat.Patient\_Gender}, \text{pat.Patient\_phoneNumber}, \text{apt.appointment\_id}, \text{apt.appointment\_date}, \text{apt.patient\_in\_out}, \text{rm.floor\_number}, \text{doc.doctors\_name}, \text{dept.department\_name}, \text{ins.insurance\_companyname}} ((\text{apt} \text{ left join } \text{pat} \text{ on } \text{apt.patient\_id} = \text{pat.Patient\_id}) \text{ left join } \text{doc} \text{ on } \text{apt.doctors\_id} = \text{doc.doctors\_id} \text{ left join } \text{dept} \text{ on } \text{apt.department\_id} = \text{dept.department\_id} \text{ left join } \text{ins} \text{ on } \text{apt.insurance\_id} = \text{ins.insurance\_id} \text{ left join } \text{rm} \text{ on } \text{apt.room\_id} = \text{rm.room\_id}))$$

```

Explanation:

apt left join pat on apt.patient_id = pat.Patient_id: Perform a left outer join between the "appointment" table (apt) and the "patient_details" table (pat) on the condition that the "patient_id" column in "apt" matches the "Patient_id" column in "pat".

(apt left join pat on apt.patient_id = pat.Patient_id) left join doc on apt.doctors_id = doc.doctors_id: Perform another left outer join on the resulting table from the previous join with the "doctors" table (doc) on the condition that the "doctors_id" column in "apt" matches the "doctors_id" column in "doc".

((apt left join pat on apt.patient_id = pat.Patient_id) left join doc on apt.doctors_id = doc.doctors_id) left join dept on apt.department_id = dept.department_id: Perform another left outer join on the resulting table from the previous join with the "department" table (dept) on the condition that the "department_id" column in "apt" matches the "department_id" column in "dept".

(((apt left join pat on apt.patient_id = pat.Patient_id) left join doc on apt.doctors_id = doc.doctors_id) left join dept on apt.department_id = dept.department_id) left join ins on apt.insurance_id = ins.insurance_id: Perform another left outer join on the resulting table from the previous join with the "insurance_company" table (ins) on the condition that the "insurance_id" column in "apt" matches the "insurance_id" column in "ins".

(((apt left join pat on apt.patient_id = pat.Patient_id) left join doc on apt.doctors_id = doc.doctors_id) left join dept on apt.department_id = dept.department_id) left join ins on apt.insurance_id = ins.insurance_id) left join rm on apt.room_id = rm.room_id: Perform another left outer join on the resulting table from the previous join with the "room" table (rm) on the condition that the "room_id" column in "apt" matches the "room_id" column in "rm".

π pat.Patient_Name, pat.Patient_Gender, pat.Patient_phoneNumber, apt.appointment_id, apt.appointment_date, apt.patient_in_out, rm.floor_number, doc.doctors_name, dept.department_name, ins.insurance_companyname: Project only the required columns from the resulting table.

Query 5:

```
SELECT cus.Customer_name,
       cus.customer_phonenumber,
       bill.customer_ssn,
       sum(bill.customer_billamount) as Total_Bill_Amount,
       sum(bill.drug_quantity) as Total_Drug_Sales_Quantity
FROM pharmacy.customer cus inner join pharmacy.customer_bill bill
on bill.customer_id = cus.customer_id
group by cus.Customer_name,
         cus.customer_phonenumber;
```

Using the Relational Algebra:

```

$$\Pi_{\text{cus.Customer\_name}, \text{cus.customer\_phonenumber}, \text{bill.customer\_ssn}, \sum(\text{bill.customer\_billamount}) \text{ as Total\_Bill\_Amount}, \sum(\text{bill.drug\_quantity}) \text{ as Total\_Drug\_Sales\_Quantity}} ((\sigma_{\text{bill.customer\_id} = \text{cus.customer\_id}} (\text{cus} \bowtie \text{bill})) \bowtie \text{cus.Customer\_name}, \text{cus.customer\_phonenumber}, \text{bill.customer\_ssn}, \text{bill.customer\_billamount}, \text{bill.drug\_quantity} \text{ (group by cus.Customer\_name, cus.customer\_phonenumber)})$$

```

Explanation:

$\text{cus} \bowtie \text{bill}$: Perform an inner join between the "customer" table (cus) and the "customer_bill" table (bill) on the condition that the "customer_id" column in "bill" matches the "customer_id" column in "cus".

$\sigma_{\text{bill.customer_id} = \text{cus.customer_id}} (\text{cus} \bowtie \text{bill})$: Filter the resulting table from the join to only include rows where the "customer_id" column in "bill" matches the "customer_id" column in "cus".

$\text{group by cus.Customer_name, cus.customer_phonenumber}$: Group the resulting table by the "Customer_name" and "customer_phonenumber" columns in "cus".

$\pi_{\text{cus.Customer_name}, \text{cus.customer_phonenumber}, \text{bill.customer_ssn}, \sum(\text{bill.customer_billamount}) \text{ as Total_Bill_Amount}, \sum(\text{bill.drug_quantity}) \text{ as Total_Drug_Sales_Quantity}}$: Project the required columns from the resulting table,

including the "Customer_name" and "customer_phonenumber" columns in "cus", the "customer_ssn" column in "bill", and the sum of the "customer_billamount" and "drug_quantity" columns in "bill" as "Total_Bill_Amount" and "Total_Drug_Sales_Quantity", respectively.

Query 6:

```
SELECT emp.Employee_ID,
       Emp.Employee_Name,
       phm.pharmacy_name,
       sum(bill.customer_billamount) as Sales_Amount
FROM pharmacy.employee emp left join pharmacy.customer_bill bill on
emp.Employee_ID = bill.Employee_ID
left join pharmacy.pharmacy_details phm on emp.Pharmacy_ID =
phm.Pharmacy_ID
group by emp.Employee_ID,
       Emp.Employee_Name,
       phm.Pharmacy_Name;
```

Using the Relational Algebra:

$$\pi_{\text{emp.Employee_ID}, \text{Emp.Employee_Name}, \text{phm.pharmacy_name}, \text{sum}(\text{bill.customer_billamount}) \text{ as Sales_Amount}} ((\text{emp} \otimes \text{phm}) \bowtie \text{bill}) \text{ (group by emp.Employee_ID, Emp.Employee_Name, phm.Pharmacy_Name)}$$

Explanation:

Do a natural join on the "Pharmacy ID" column between the "employee" table (emp) and the "pharmacy details" table (phm).

If the "Employee ID" column in "emp" matches the "Employee ID" column in "bill," then do a left outer join between the table created by the join above and the "customer bill" table.

Group the resulting table by the "Employee ID," "Employee Name," and "Pharmacy Name" columns in the appropriate rows for "emp," "Emp," and "phm."

sum(bill.customer_billamount) as Sales_Amount, emp.Employee_ID, emp.Employee_Name, phm.pharmacy_name: The "Employee ID" and "Employee Name" columns in "emp" and "Emp," the "pharmacy name" column in "phm," and the total of the "customer_billamount" column in "bill" as "Sales_Amount" are all projected from the resultant table to provide the necessary columns.

Query 7:

```
SELECT phm.pharmacy_name,  
       sum(bill.customer_billamount) as Sales_Amount  
FROM pharmacy.pharmacy_details phm left join pharmacy.customer_bill bill  
on phm.Pharmacy_ID = bill.Pharmacy_ID  
group by phm.Pharmacy_Name;
```

Using the Relational Algebra:

$$\pi_{phm.pharmacy_name, \sum(bill.customer_billamount)} \text{as Sales_Amount} (phm \bowtie bill) (\text{group by } phm.pharmacy_name)$$

Explanation:

$phm \bowtie bill$: Perform a left outer join between the "pharmacy_details" table (phm) and the "customer_bill" table (bill) on the condition that the "Pharmacy_ID" column in "phm" matches the "Pharmacy_ID" column in "bill".

group by phm.pharmacy_name: Group the resulting table by the "pharmacy_name" column in "phm".

$\pi_{phm.pharmacy_name, \sum(bill.customer_billamount)} \text{as Sales_Amount}$: Project the required columns from the resulting table, including the "pharmacy_name" column in "phm" and the sum of the "customer_billamount" column in "bill" as "Sales_Amount".

Query 8:

```
SELECT sub.SubmissionNo,  
       sub.SubmissionType,  
       sub_lkSubmissionClassCode,  
       sub_lkSubmissionClassCodeDescription  
FROM drugs.submissions sub left join drugs.submissionclass_lookup sub_lk  
on sub.SubmissionClassCodeID = sub_lkSubmissionClassCodeID;
```

Using the Relational Algebra:

$\pi_{\text{sub.SubmissionNo, sub.SubmissionType, sub_lkSubmissionClassCode, sub_lkSubmissionClassCodeDescription}}(\text{sub} \bowtie \text{sub_lk})$

Explanation:

$\text{sub} \bowtie \text{sub_lk}$: Perform a left outer join between the "submissions" table (sub) and the "submissionclass_lookup" table (sub_lk) on the condition that the "SubmissionClassCodeID" column in "sub" matches the "SubmissionClassCodeID" column in "sub_lk".

$\pi_{\text{sub.SubmissionNo, sub.SubmissionType, sub_lkSubmissionClassCode, sub_lkSubmissionClassCodeDescription}}$: Project the required columns from the resulting table, including the "SubmissionNo" and "SubmissionType" columns from "sub", and the "SubmissionClassCode" and "SubmissionClassCodeDescription" columns from "sub_lk".

Query 9:

```
SELECT app.ApplNo,  
       app.ApplType,  
       app.SponsorName,  
       ms.MarketingStatusID,  
       ms_lkMarketingStatusDescription
```

```
FROM drugs.applications app left join drugs.marketingstatus ms on app.ApplNo =  
ms.ApplNo
```

```
inner join drugs.marketingstatus_lookup ms_lk on ms.MarketingStatusID =  
ms_lk.MarketingStatusID;
```

Using the Relational Algebra:

```
 $\pi$  app.ApplNo, app.ApplType, app.SponsorName, ms.MarketingStatusID,  
ms_lk.MarketingStatusDescription ((drugs.applications app  $\bowtie$   
drugs.marketingstatus ms)  $\bowtie$  drugs.marketingstatus_lookup ms_lk)
```

Explanation:

π app.ApplNo, app.ApplType, app.SponsorName, ms.MarketingStatusID, ms_lk.MarketingStatusDescription: Project the required columns from the resulting table, including the "ApplNo", "ApplType", and "SponsorName" columns from "app", the "MarketingStatusID" column from "ms", and the "MarketingStatusDescription" column from "ms_lk".

(drugs.applications app \bowtie drugs.marketingstatus ms) \bowtie drugs.marketingstatus_lookup ms_lk: Perform an inner join between the resulting table from the previous join and the "marketingstatus_lookup" table (ms_lk) on the condition that the "MarketingStatusID" column in "ms" matches the "MarketingStatusID" column in "ms_lk".

Query 10:

```
SELECT age_group, sex, count(fdaadverseeventID) as total_events  
FROM faers.demographic  
where event_dt >= '2020-01-01'  
group by age_group, sex  
order by age_group, sex;
```

Using the Relational Algebra:

```
σ event_dt >= '2020-01-01' (demographic) ⋈ (π age_group, sex,  
fdaadverseeventID (demographic))  
→ (ρ age_group, sex, total_events (γ age_group, sex; count(fdaadverseeventID)  
as total_events))
```

Explanation:

Using the selection operator, we first restrict our selection to only those tuples in the demographic relation where the event dt is greater than or equal to '2020-01-01'.

The projection operator is then used to project the age group, sex, and fdaadverseeventID attributes from the resulting relation.

Using the natural join operator, we join this projection to the demographic relation.

Then, using the grouping operator, we divide the resulting relation into groups according to age group and sex, counting the number of fdaadverseeventIDs for each group. Using the renaming operator, we rename the resulting attribute to total events and sort the relation by age group and sex.

Query 11:

```
SELECT distinct dm.fdaadverseeventID,  
drg.drugname,  
drg.prod_ai,  
dm.event_dt,  
dm.age as Patient_age  
  
FROM faers.drug drg left join faers.demographic dm  
on drg.fdaadverseeventID = dm.fdaadverseeventID;
```

Using the Relational Algebra:

```
π dm.fdaadverseeventID, drg.drugname, drg.prod_ai, dm.event_dt, dm.age as  
Patient_age ((drg ⋈ drg.fdaadverseeventID = dm.fdaadverseeventID dm))
```

Explanation:

Using the fdaadverseeventID attribute, join the drug and demographic tables.

To choose the desired attributes, do projection on the resulting table: fdaadverseeventID, drugname, prod ai, event dt, and age (renamed as patient age).

Question 7:

As per the requirement, we have written 10 queries and attached screenshots of the results in this question.

Query 1:

Write a query to display doctor's name and their availability days who is done specialization on Gynecology.

```
SELECT doc.doctors_id,  
       doc.doctors_name,  
       doc.doctors_availability  
FROM hospital.doctors doc  
      where doctors_specilization = "Gynecologists";
```

Result:

The screenshot shows the MySQL Workbench interface. In the central SQL editor window, the following query is displayed:

```
1 • SELECT doc.doctors_id,
2         doc.doctors_name,
3         doc.doctors_availability
4     FROM hospatal.doctors doc
5     WHERE doctors_specilization = "Gynecologists";
```

The result grid shows one row of data:

doctors_id	doctors_name	doctors_availability
289	Nandini	Friday,Sunday

Query 2:

write a query to display total male and female patient count

SELECT Patient_Gender,

count(Patient_Gender)

FROM hospatal.patient_details

group by Patient_Gender;

Result:

The screenshot shows the MySQL Workbench interface. In the top navigation bar, there are multiple tabs: local (hosipatal), MySQL Model*, EER Diagram, EER Diagram1, EER Diagram2, EER Diagram3, EER Diagram4, local (hosipatal), and EER Diagram5. The main area displays a SQL query in the SQL Editor tab:

```
1 • SELECT Patient_Gender,
2         count(Patient_Gender)
3     FROM hosipatal.patient_details
4     group by Patient_Gender;
5
6
```

The Result Grid shows the following data:

Patient_Gender	count(Patient_Gender)
Female	6
Male	5

On the right side of the interface, there is a message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

Query 3:

Write a query to display doctor details and hospital name where they are working

```
SELECT doc.doctors_id,
       doc.doctors_name,
       doc.doctors_availability,
       hos.hospital_name,
       doc.doctors_phonenumber
FROM hosipatal.doctors doc left join hosipatal.hospitaldetails hos
on doc.hospital_id = hos.hospital_id;
```

Result:

The screenshot shows the MySQL Workbench interface. The left sidebar displays the Navigator with the 'SCHEMAS' section expanded, showing the 'hosipatal' schema with its tables: appointment, department, doctor, hospitaldetails, insurance_company, patient_details, and room. Below this is the 'Information' section. The bottom left shows 'Object Info' and 'Session' tabs, with 'Query Completed' status.

The main area contains a SQL editor tab titled 'SQL File 5*' with the following query:

```
1 • SELECT doc.doctors_id,
2     doc.doctors_name,
3     doc.doctors_availability,
4     hos.hospital_name,
5     doc.doctors_phonenumber
6 FROM hosipatal.doctors doc LEFT JOIN hosipatal.hospitaldetails hos
7 ON doc.hospital_id = hos.hospital_id;
```

Below the SQL editor is a 'Result Grid' tab showing the query results:

doctors_id	doctors_name	doctors_availability	hospital_name	doctors_phonenumber
123	Sudheer	Monday,Friday	Apollo	9234567891
213	Vinay	Tuesday,Thursday	Apollo1	9234567892
234	Gowtham	Friday,Saturday	Apollo2	9234567893
245	Mounika	Sunday,Monday	Apollo1	9234567894
256	Nikhil	Wednesday,Thursday	Apollo4	9234567895
278	Suitha	Monday,Thursday	Apollo3	9234567896
289	Nandini	Friday,Sunday	Apollo	9234567897
345	Pradeep	Saturday,Monday	Apollo2	9234567898
567	Hema	Monday,Friday	Apollo4	9234567888
789	Sai Krishna	Monday,Friday,Saturday	Apollo	9234567889

The bottom right corner of the interface shows system icons and the date/time: 7:48 PM 4/17/2023.

Query 4:

write a query to display patient details, and their appointment details along with doctor name, department name & insurance details

```
SELECT pat.Patient_Name,  
       pat.Patient_Gender,  
       pat.Patient_phoneNumber,  
       apt.appointment_id,  
       apt.appointment_date,  
       apt.patient_in_out,  
       rm.floor_number,  
       doc.doctors_name,  
       dept.department_name,
```

ins.insurance_companynam

```
FROM hospatal.appointment apt left join hospatal.patient_details pat on
apt.patient_id = pat.Patient_id

left join hospatal.doctors doc on apt.doctors_id = doc.doctors_id

left join hospatal.department dept on apt.department_id = dept.department_id

left join hospatal.insurence_company ins on apt.insurence_id = ins.insurence_id

left join hospatal.room rm on apt.room_id = rm.room_id;
```

Result:

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

- Navigator:** Shows the **SCHEMAS** section with **hospatal** selected, displaying tables like appointment, department, doctors, hospitaldetails, insurance_company, patient_details, and room.
- SQL Editor:** Contains the following SQL query:

```
1 •  SELECT pat.Patient_Name,
2      pat.Patient_Gender,
3      pat.Patient_phoneNumber,
4      apt.appointment_id,
5      apt.appointment_date,
6      apt.patient_in_out,
7      rm.floor_number,
8      doc.doctors_name,
9      dept.department_name,
10     ins.insurance_companynam
```
- Result Grid:** Displays the results of the query in a tabular format. The columns are Patient_Name, Patient_Gender, Patient_phoneNumber, appointment_id, appointment_date, patient_in_out, and floor_number. The data includes rows for patients like Banu, Rashmika, Yamini, etc., across different floors and appointment dates.
- Status Bar:** Shows "Query Completed" and the system tray with various icons.

Query 5:

Write a query to display customer details along with purchased amount and purchased drug quantity details

```
SELECT cus.Customer_name,  
       cus.customer_phonenumber,  
       bill.customer_ssn,  
       sum(bill.customer_billamount) as Total_Bill_Amount,  
       sum(bill.drug_quantity) as Total_Drug_Sales_Quantity  
FROM pharmacy.customer cus inner join pharmacy.customer_bill bill  
on bill.customer_id = cus.customer_id  
group by cus.Customer_name,  
       cus.customer_phonenumber;
```

Result:

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

- Navigator:** Shows the database schema with the **pharmacy** schema selected.
- SQL Editor:** Displays the query code:

```
1 •  SELECT cus.Customer_name,  
2      cus.customer_phonenumber,  
3      bill.customer_ssn,  
4      sum(bill.customer_billamount) as Total_Bill_Amount,  
5      sum(bill.drug_quantity) as Total_Drug_Sales_Quantity  
6  FROM pharmacy.customer cus inner join pharmacy.customer_bill bill  
7  on bill.customer_id = cus.customer_id  
8  group by cus.Customer_name,  
9      cus.customer_phonenumber;
```
- Result Grid:** Shows the query results in a tabular format:

Customer_name	customer_phonenumber	customer_ssn	Total_Bill_Amount	Total_Drug_Sales_Quantity
Nancy	3097762427	671-64-5622	355	18
Joe	3097765488	671-64-5623	120	6
Josh	3097762457	671-64-5234	645	36
Amber	3097762567	671-64-5673	45	4
Hannah	3097762123	981-64-5622	98	7
Sam	3097762452	123-64-5622	110	9
David	3097756789	671-45-2651	23	2
Michael	3097712338	671-64-3333	65	5
- Message Bar:** Shows "Query Completed".
- System Tray:** Shows the Windows taskbar with various application icons and system status.

Query 6:

Write a query to display employee and drug sales amount along with pharmacy name

```
SELECT emp.Employee_ID,
       Emp.Employee_Name,
       phm.pharmacy_name,
       sum(bill.customer_billamount) as Sales_Amount
FROM pharmacy.employee emp left join pharmacy.customer_bill bill on
emp.Employee_ID = bill.Employee_ID
left join pharmacy.pharmacy_details phm on emp.Pharmacy_ID = phm.Pharmacy_ID
group by emp.Employee_ID,
       Emp.Employee_Name,
       phm.Pharmacy_Name;
```

Result:

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

- SQL Editor:** The query is pasted into the SQL Editor window.
- Result Grid:** The results of the query are displayed in a grid format. The columns are Employee_ID, Employee_Name, pharmacy_name, and Sales_Amount. The data is as follows:

Employee_ID	Employee_Name	pharmacy_name	Sales_Amount
1231	John	SPH	120
1232	Jack	Devil	279
1233	Chris	Walmart	110
1234	Stephen	Apolo	300
1235	Faris	SRS	368
1236	Daniel	Apolo	65
1237	Christopher	SPH	121
1238	Regina	NRI	98

- Information Panel:** A panel on the right provides context help for the current caret position.
- System Bar:** The bottom bar shows the system tray with icons for battery, signal, and time (7:52 PM).

Query 7:

Write a query to display drugs sales amount by pharmacy name

```
SELECT phm.pharmacy_name,
       sum(bill.customer_billamount) as Sales_Amount
  FROM pharmacy.pharmacy_details phm left join pharmacy.customer_bill bill
    on phm.Pharmacy_ID = bill.Pharmacy_ID
 group by phm.Pharmacy_Name;
```

Result:

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

- SQL Editor:** Contains the query:

```
1 •  SELECT phm.pharmacy_name,
2           sum(bill.customer_billamount) as Sales_Amount
3      FROM pharmacy.pharmacy_details phm left join pharmacy.customer_bill bill
4        on phm.Pharmacy_ID = bill.Pharmacy_ID
5   group by phm.Pharmacy_Name;
6
```
- Result Grid:** Displays the results of the query:

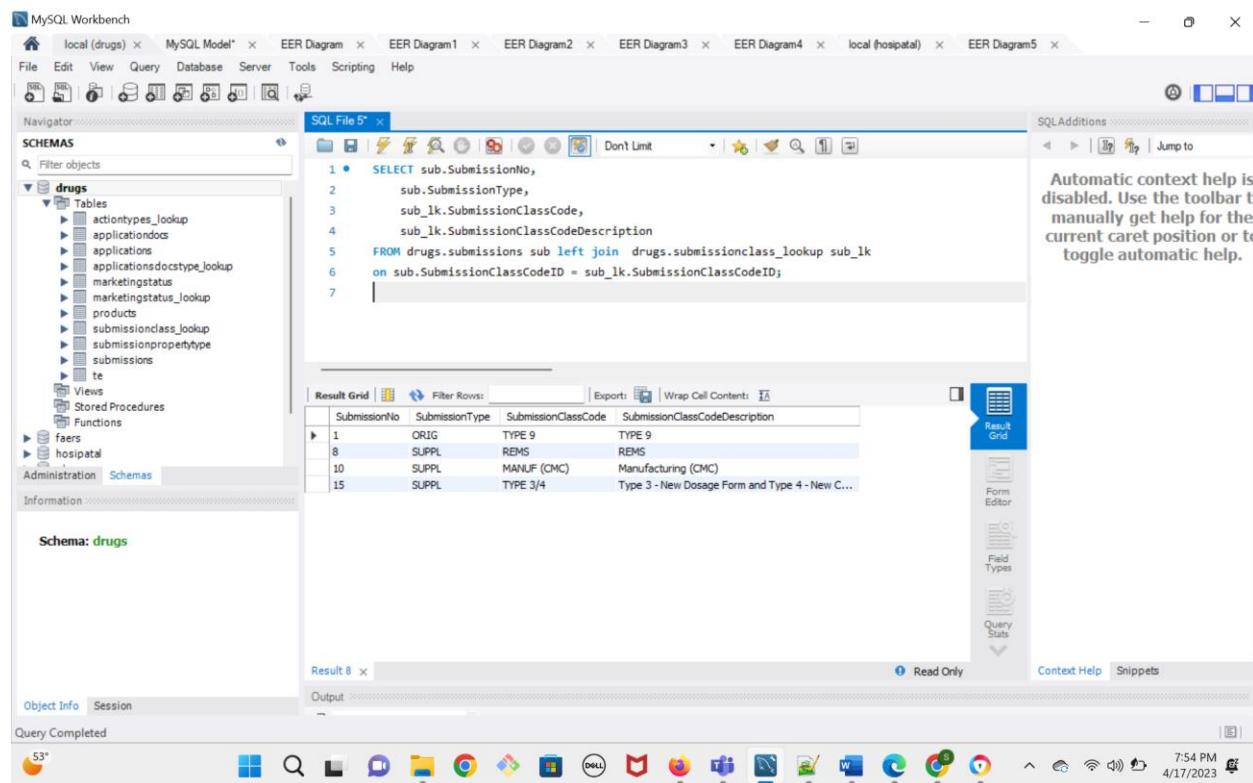
pharmacy_name	Sales_Amount
SPH	443
walgreens	NULL
Devi	185
sree	NULL
Walmart	144
PHS	NULL
SRS	300
Apolo	45
HSP	110
NRI	234
- Information Panel:** Shows the schema: **pharmacy**.
- System Bar:** Includes icons for various applications like File Explorer, Task View, and Start.
- Bottom Status:** Shows the date and time: 7:53 PM 4/17/2023.

Query 8:

Write a query to display all the submissions and their submission type code and description

```
SELECT sub.SubmissionNo,  
       sub.SubmissionType,  
       sub_lkSubmissionClassCode,  
       sub_lkSubmissionClassCodeDescription  
  
FROM drugs.submissions sub left join drugs.submissionclass_lookup sub_lk  
on sub.SubmissionClassCodeID = sub_lkSubmissionClassCodeID;
```

Result:



The screenshot shows the MySQL Workbench interface with a query editor window open. The query is:

```
1 •   SELECT sub.SubmissionNo,  
2       sub.SubmissionType,  
3       sub_lkSubmissionClassCode,  
4       sub_lkSubmissionClassCodeDescription  
5   FROM drugs.submissions sub left join drugs.submissionclass_lookup sub_lk  
6   on sub.SubmissionClassCodeID = sub_lkSubmissionClassCodeID;  
7
```

The results grid displays the following data:

SubmissionNo	SubmissionType	SubmissionClassCode	SubmissionClassCodeDescription
1	ORIG	TYPE 9	TYPE 9
8	SUPPL	REMS	REMS
10	SUPPL	MANUF (CMC)	Manufacturing (CMC)
15	SUPPL	TYPE 3/4	Type 3 - New Dosage Form and Type 4 - New C...

Query 9:

Write a query to display applications and their marketing status description

```
SELECT app.ApplNo,
```

```
    app.ApplType,
```

```
    app.SponsorName,
```

```
    ms.MarketingStatusID,
```

```
    ms_lk.MarketingStatusDescription
```

```
FROM drugs.applications app left join drugs.marketingstatus ms on app.ApplNo =  
ms.ApplNo
```

```
inner join drugs.marketingstatus_lookup ms_lk on ms.MarketingStatusID =  
ms_lk.MarketingStatusID;
```

Result:

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

- Navigator:** Shows the database schema for the "drugs" database, including tables like "actiotype", "applications", "marketingstatus", etc.
- SQL Editor:** Displays the query code:

```
1 •   SELECT app.ApplNo,  
2       app.ApplType,  
3       app.SponsorName,  
4       ms.MarketingStatusID,  
5       ms_lk.MarketingStatusDescription  
6   FROM drugs.applications app left join drugs.marketingstatus ms on app.ApplNo = ms.ApplNo  
7   inner join drugs.marketingstatus_lookup ms_lk on ms.MarketingStatusID = ms_lk.MarketingStatusID;
```
- Result Grid:** Shows the query results in a grid format. The columns are: ApplNo, ApplType, SponsorName, MarketingStatusID, and MarketingStatusDescription. The data consists of multiple rows where ApplNo is 4, ApplType is NDA, SponsorName is PHARMICS or LILLY, MarketingStatusID is 3, and MarketingStatusDescription is Discontinued.
- Status Bar:** Shows the system tray with icons for battery, signal, and time (7:55 PM, 4/17/2023).

Query 10:

Write a query to display total events by age group and gender breakdown for the events which were created from 2020

```
SELECT age_group, sex, count(fdaadverseeventID) as total_events  
FROM faers.demographic  
where event_dt >= '2020-01-01'  
group by age_group, sex  
order by age_group, sex;
```

Result:

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

- Navigator:** Shows the database schema with the **faers** schema selected. Under **Tables**, it lists: demographic, drug, indication, outcome, reaction, report_sources, and therapy.
- SQL Editor:** Displays the query code:

```
1 • SELECT age_group, sex, count(fdaadverseeventID) as total_events  
2   | Execute the selected portion of the script or everything, if there is no selection  
3   where event_dt >= '2020-01-01'  
4   group by age_group, sex  
5   order by age_group, sex;  
6
```
- Result Grid:** Shows the query results in a grid format:

age_group	sex	total_events
	F	37
	M	125
A	F	8
A	M	4
C	M	1
E	F	5
E	M	2
- Status Bar:** Shows "Query Completed" and the system tray with various icons.

Query 11:

Write a query to display events and drug details along with event date and patient age

```
SELECT distinct dm.fdaadverseeventID,
drg.drugname,
drg.prod_ai,
dm.event_dt,
dm.age as Patient_age
FROM faers.drug drg left join faers.demographic dm
on drg.fdaadverseeventID = dm.fdaadverseeventID;
```

Result:

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

- Navigator:** Shows the database schema with the 'faers' database selected. Under 'Tables', it lists 'demographic', 'drug', 'indication', 'outcome', 'reaction', 'report_sources', and 'therapy'. Under 'Views', 'Stored Procedures', and 'Functions', there are no entries.
- SQL Editor:** Displays the SQL query:

```
1 • SELECT distinct dm.fdaadverseeventID,
2   drg.drugname,
3   drg.prod_ai,
4   dm.event_dt,
5   dm.age as Patient_age
6   FROM faers.drug drg left join faers.demographic dm
7   on drg.fdaadverseeventID = dm.fdaadverseeventID;
```
- Result Grid:** Shows the query results in a tabular format. The columns are 'fdaadverseeventID', 'drugname', 'prod_ai', 'event_dt', and 'Patient_age'. The data includes various drugs and their associated details.
- Information:** Shows the schema 'faers' is selected.
- Object Info:** Shows the session information.
- Output:** Shows the query completed successfully.
- System Bar:** Shows the system tray with icons for battery, search, taskbar, and network status.

fdaadverseeventID	drugname	prod_ai	event_dt	Patient_age
100115733	JANUVIA	SITAGLIPTIN PHOSPHATE	2013-08-01	75
100234223	BONIVA	IBANDRONATE SODIUM	2012-03-01	67
100260594	LEVOFLOXACIN	LEVOFLOXACIN	2012-01-01	63
100518358	ACTEMRA	TOCILIZUMAB	2013-07-15	50
100746869	XOLAIR	OMALIZUMAB	2009-07-14	52
100832487	XOLAIR	OMALIZUMAB	2012-02-01	46
101397357	MELATONIN	MELATONIN	2014-01-01	16
1002130537	SANDOSTATIN	OCTREOTIDE ACETATE	2021-04-15	43
1006401873	ACTEMRA	TOCILIZUMAB	2012-12-01	71
1009456310	PRISTIQ EXTENDED-RELEASE	DESVENLAFAXINE SUCCINATE	2013-01-01	46
1014222242	ACTEMRA	TOCILIZUMAB	2013-12-01	52
1015212328	XOLAIR	OMALIZUMAB	2013-12-05	38