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:

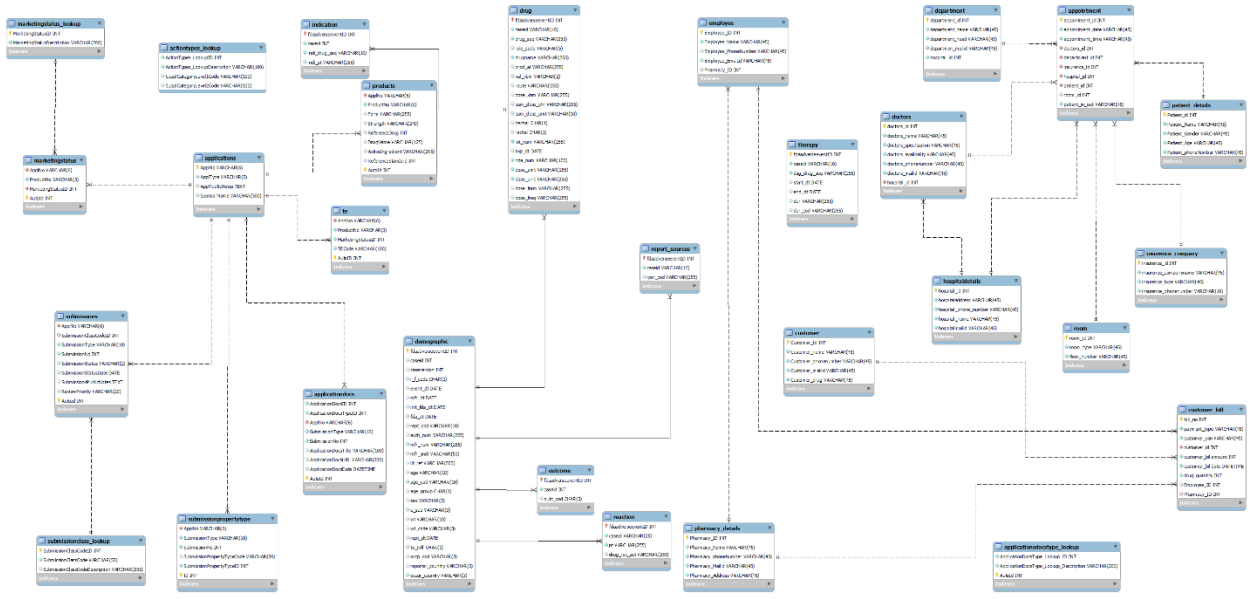
Question 1:

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 in 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:

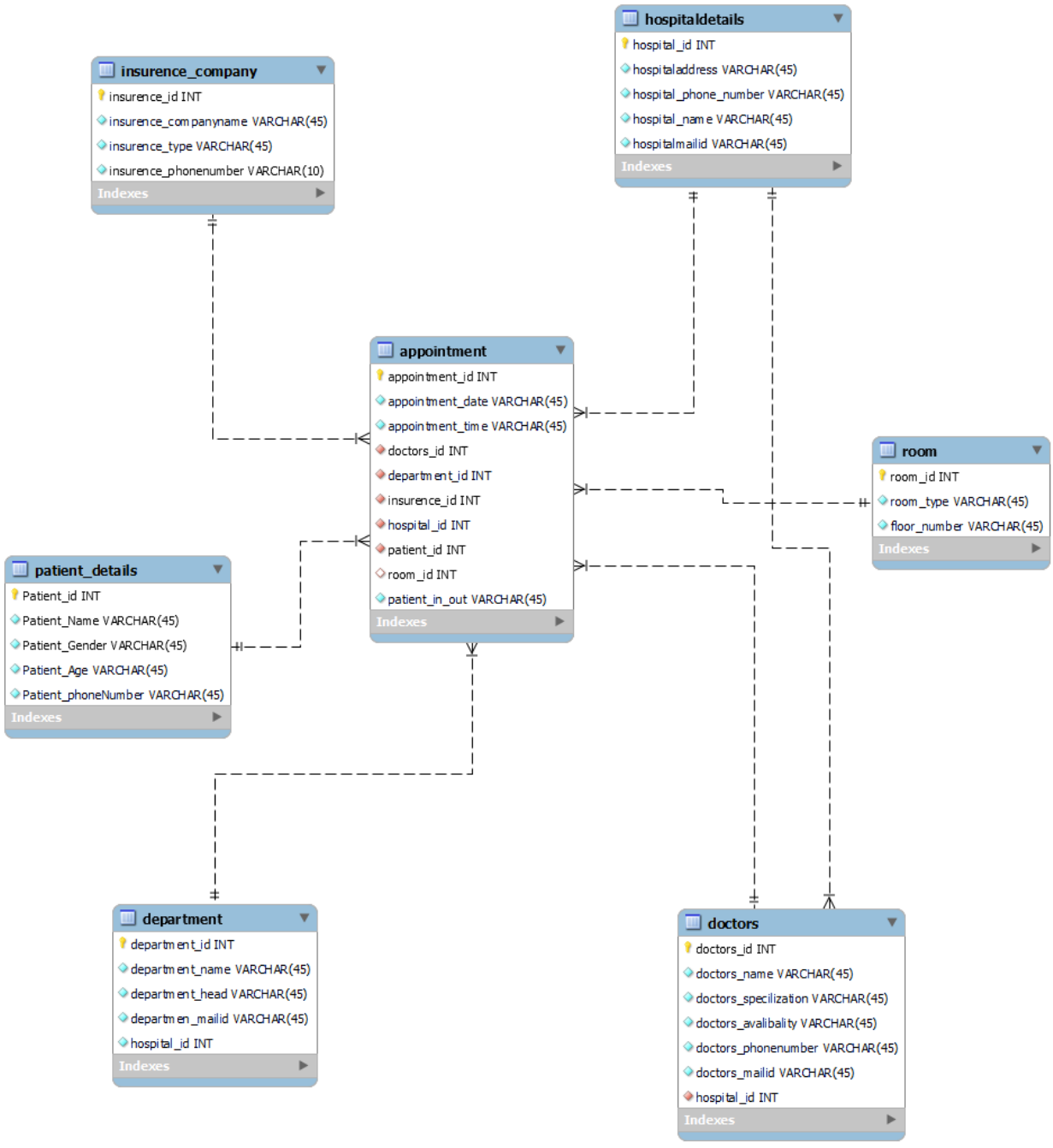
Assume we have a "patients" database with information on each hospital patient. The table may have columns with names like "patient_id," "first_name," "last_name," "gender," "phone number," "email address," "address," and others.

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.


insurence_company
insurence_id INT
insurence_companyname VARCHAR(45)
insurence_type VARCHAR(45)
insurence_phonenumber VARCHAR(10)
Indexes
hospitaldetails
hospital_id INT
hospitaladdress VARCHAR(45)
hospital_phone_number VARCHAR(45)
hospital_name VARCHAR(45)
hospitalmailid VARCHAR(45)
Indexes
appointment
appointment_id INT
appointment_date VARCHAR(45)
appointment_time VARCHAR(45)
doctors_id INT
department_id INT
insurence_id INT
hospital_id INT
patient_id INT
room_id INT
patient_in_out VARCHAR(45)
Indexes
room
room_id INT
room_type VARCHAR(45)
floor_number VARCHAR(45)
Indexes
patient_details
Patient_id INT
Patient_Name VARCHAR(45)
Patient_Gender VARCHAR(45)
Patient_Age VARCHAR(45)
Patient_phoneNumber VARCHAR(45)
Indexes
department
department_id INT
department_name VARCHAR(45)
department_head VARCHAR(45)
departmen_mailid VARCHAR(45)
hospital_id INT
Indexes
doctors
doctors_id INT
doctors_name VARCHAR(45)
doctors_specilization VARCHAR(45)
doctors_avalibality VARCHAR(45)
doctors_phonenumber VARCHAR(45)
doctors_mailid VARCHAR(45)
hospital_id INT
Indexes

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.

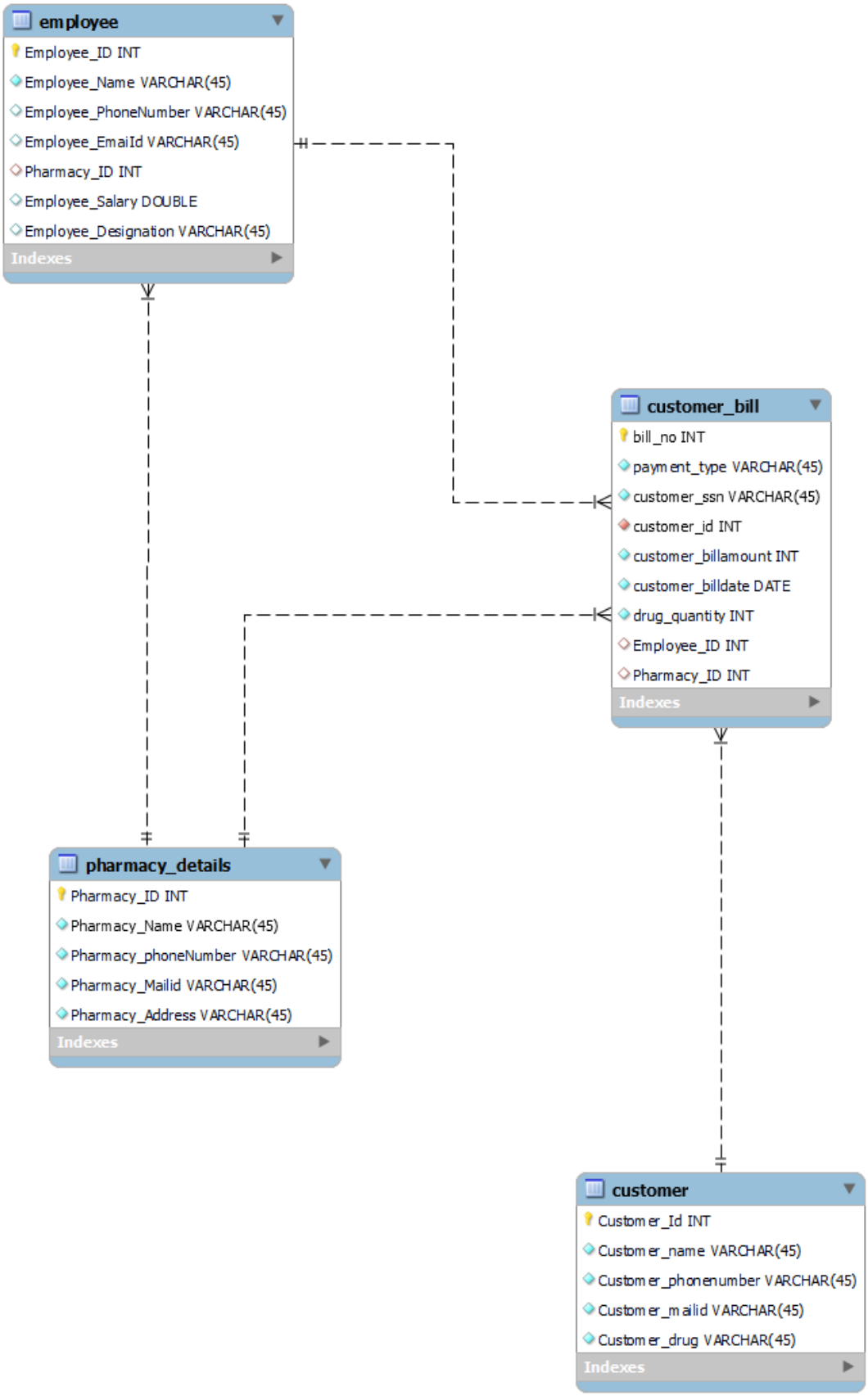
Here are some of the important tables included in a typical pharmacy management system database and an explanation of how they function:

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.


employee
Employee_ID INT
Employee_Name VARCHAR(45)
Employee_PhoneNumber VARCHAR(45)
Employee_EmaiId VARCHAR(45)
Pharmacy_ID INT
Employee_Salary DOUBLE
Employee_Designation VARCHAR(45)
Indexes
customer_bill
bill_no INT
payment_type VARCHAR(45)
customer_ssn VARCHAR(45)
customer_id INT
customer_billamount INT
customer_billdate DATE
drug_quantity INT
Employee_ID INT
Pharmacy_ID INT
Indexes
pharmacy_details
Pharmacy_ID INT
Pharmacy_Name VARCHAR(45)
Pharmacy_phoneNumber VARCHAR(45)
Pharmacy_Mailid VARCHAR(45)
Pharmacy_Address VARCHAR(45)
Indexes
customer
Customer_Id INT
Customer_name VARCHAR(45)
Customer_phonenumber VARCHAR(45)
Customer_mailid VARCHAR(45)
Customer_drug VARCHAR(45)
Indexes

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:

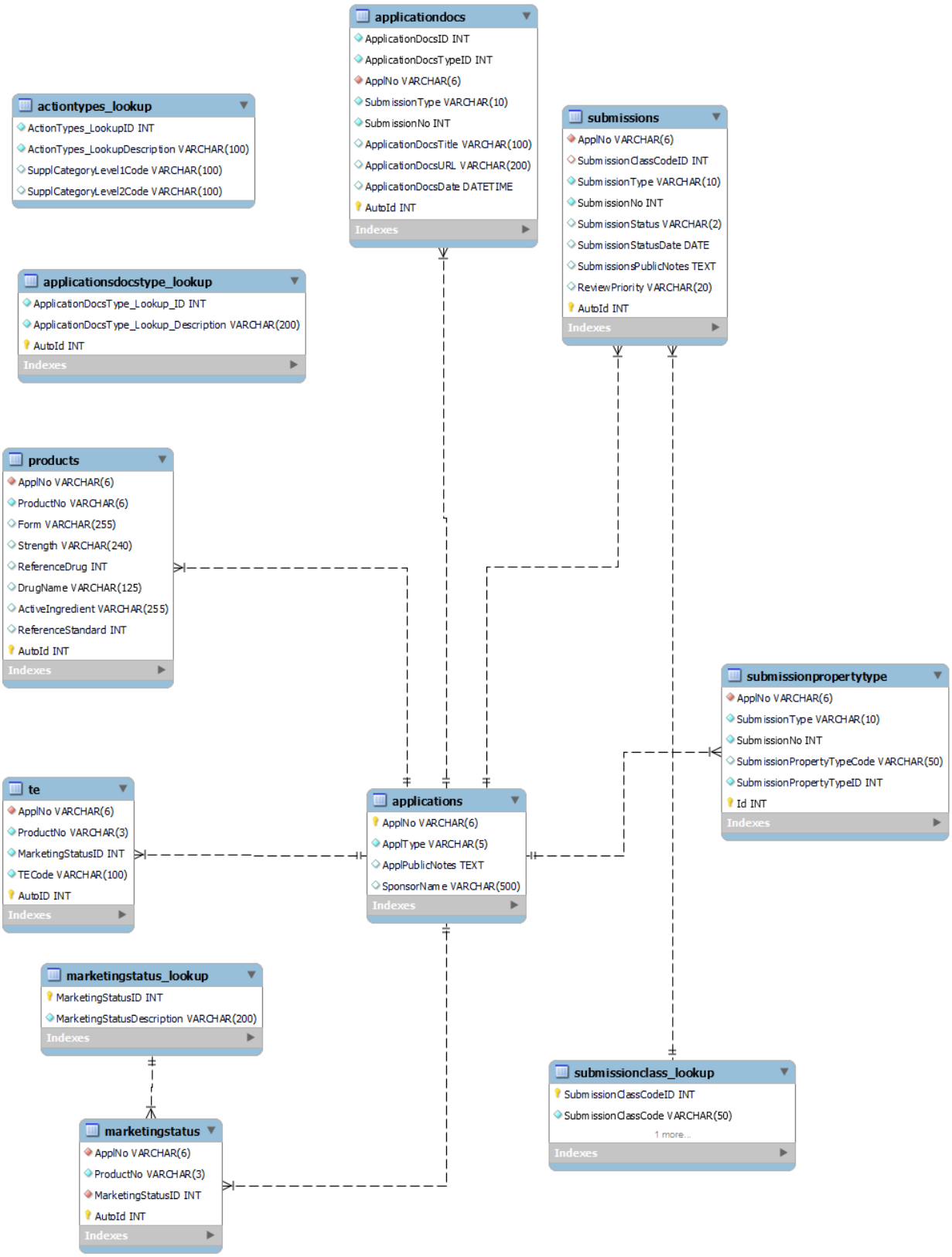
Information regarding drug applications is provided in this table, including the application number, application type, and sponsor name.

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 take, together with the application number, SubmissionClassCodeID, Submissiontype, SubmissionID, and SubmissionStatus.


applicationdocs
ApplicationDocsID INT
ApplicationDocsTypeID INT
ApplNo VARCHAR(6)
SubmissionType VARCHAR(10)
SubmissionNo INT
ApplicationDocsTitle VARCHAR(100)
ApplicationDocsURL VARCHAR(200)
ApplicationDocsDate DATETIME
AutoId INT
Indexes
actiontypes_lookup
ActionTypes_LookupID INT
ActionTypes_LookupDescription VARCHAR(100)
SupplCategoryLevel1Code VARCHAR(100)
SupplCategoryLevel2Code VARCHAR(100)
submissions
ApplNo VARCHAR(6)
SubmissionClassCodeID INT
SubmissionType VARCHAR(10)
SubmissionNo INT
SubmissionStatus VARCHAR(2)
SubmissionStatusDate DATE
SubmissionsPublicNotes TEXT
ReviewPriority VARCHAR(20)
AutoId INT
Indexes
applicationsdocstype_lookup
ApplicationDocsType_Lookup_ID INT
ApplicationDocsType_Lookup_Description VARCHAR(200)
AutoId INT
Indexes
products
ApplNo VARCHAR(6)
ProductNo VARCHAR(6)
Form VARCHAR(255)
Strength VARCHAR(240)
ReferenceDrug INT
DrugName VARCHAR(125)
ActiveIngredient VARCHAR(255)
ReferenceStandard INT
AutoId INT
Indexes
submissionpropertytype
ApplNo VARCHAR(6)
SubmissionType VARCHAR(10)
SubmissionNo INT
SubmissionPropertyTypeCode VARCHAR(50)
SubmissionPropertyTypeID INT
Id INT
Indexes
te
ApplNo VARCHAR(6)
ProductNo VARCHAR(3)
MarketingStatusID INT
TECode VARCHAR(100)
AutoID INT
Indexes
applications
ApplNo VARCHAR(6)
ApplType VARCHAR(5)
ApplPublicNotes TEXT
SponsorName VARCHAR(500)
Indexes
marketingstatus_lookup
MarketingStatusID INT
MarketingStatusDescription VARCHAR(200)
Indexes
submissionclass_lookup
SubmissionClassCodeID INT
SubmissionClassCode VARCHAR(50)
1 more...
Indexes
marketingstatus
ApplNo VARCHAR(6)
ProductNo VARCHAR(3)
MarketingStatusID INT
AutoId INT
Indexes

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:

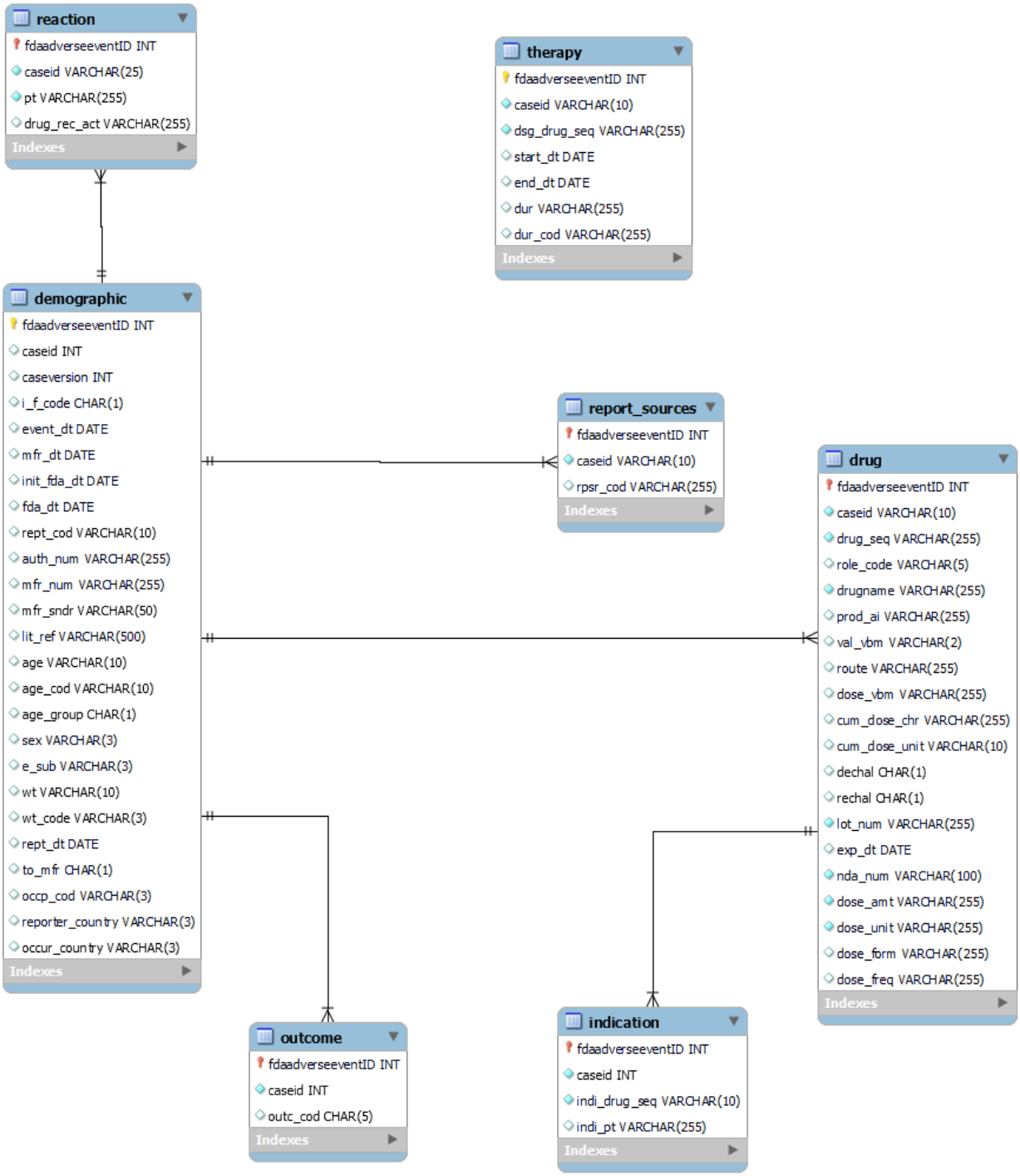
The medical problem or symptom that the medication was meant to treat is listed in this table along with other details about why it was prescribed.

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.

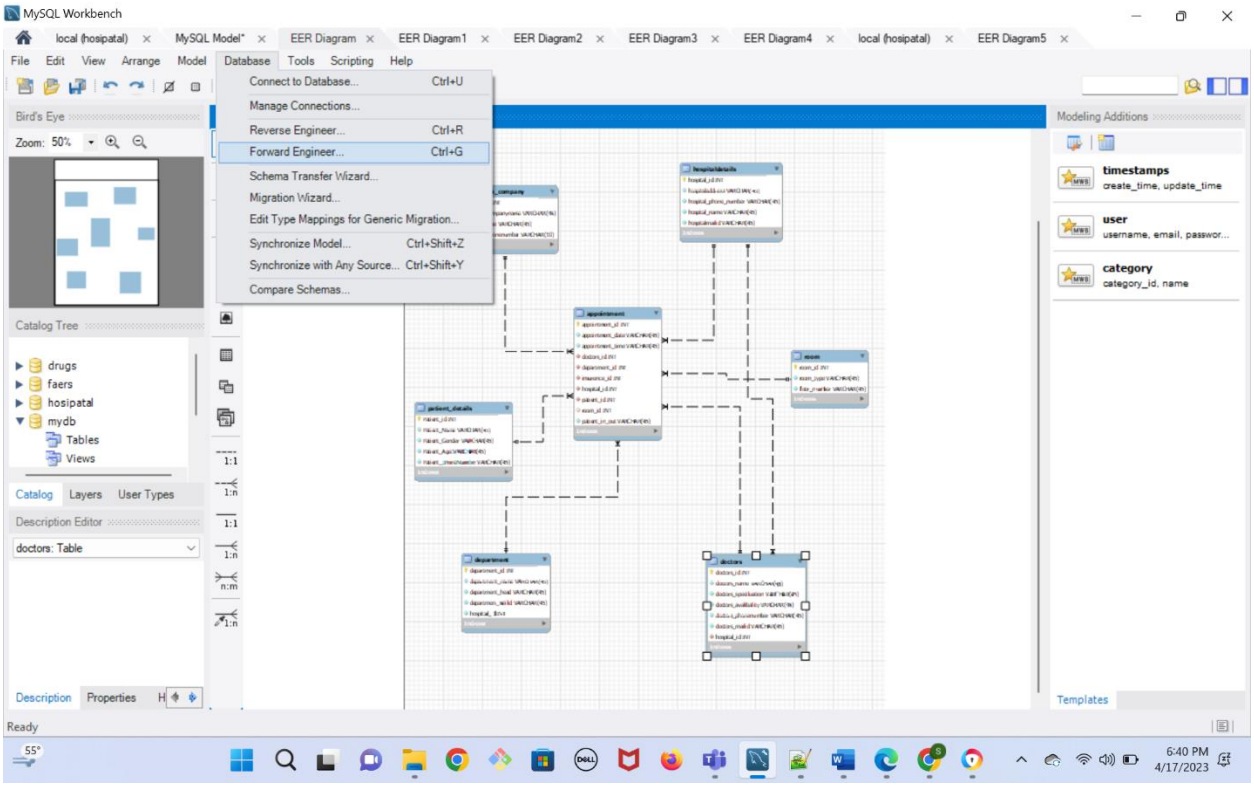

reaction
fdaadverseeventID INT
caseid VARCHAR(25)
pt VARCHAR(255)
drug_rec_act VARCHAR(255)
Indexes
therapy
fdaadverseeventID INT
caseid VARCHAR(10)
dsg_drug_seq VARCHAR(255)
start_dt DATE
end_dt DATE
dur VARCHAR(255)
dur_cod VARCHAR(255)
Indexes
demographic
fdaadverseeventID INT
caseid INT
caseversion INT
i_f_code CHAR(1)
event_dt DATE
mfr_dt DATE
init_fda_dt DATE
fda_dt DATE
rept_cod VARCHAR(10)
auth_num VARCHAR(255)
mfr_num VARCHAR(255)
mfr_sndr VARCHAR(50)
lit_ref VARCHAR(500)
age VARCHAR(10)
age_cod VARCHAR(10)
age_group CHAR(1)
sex VARCHAR(3)
e_sub VARCHAR(3)
wt VARCHAR(10)
wt_code VARCHAR(3)
rept_dt DATE
to_mfr CHAR(1)
occp_cod VARCHAR(3)
reporter_country VARCHAR(3)
occur_country VARCHAR(3)
Indexes
report_sources
fdaadverseeventID INT
caseid VARCHAR(10)
rpsr_cod VARCHAR(255)
Indexes
drug
fdaadverseeventID INT
caseid VARCHAR(10)
drug_seq VARCHAR(255)
role_code VARCHAR(5)
drugname VARCHAR(255)
prod_ai VARCHAR(255)
val_vbm VARCHAR(2)
route VARCHAR(255)
dose_vbm VARCHAR(255)
cum_dose_chr VARCHAR(255)
cum_dose_unit VARCHAR(10)
dechal CHAR(1)
rechal CHAR(1)
lot_num VARCHAR(255)
exp_dt DATE
nda_num VARCHAR(100)
dose_amt VARCHAR(255)
dose_unit VARCHAR(255)
dose_form VARCHAR(255)
dose_freq VARCHAR(255)
Indexes
outcome
fdaadverseeventID INT
caseid INT
outc_cod CHAR(5)
Indexes
indication
fdaadverseeventID INT
caseid INT
indi_drug_seq VARCHAR(10)
indi_pt VARCHAR(255)
Indexes

Question 4:

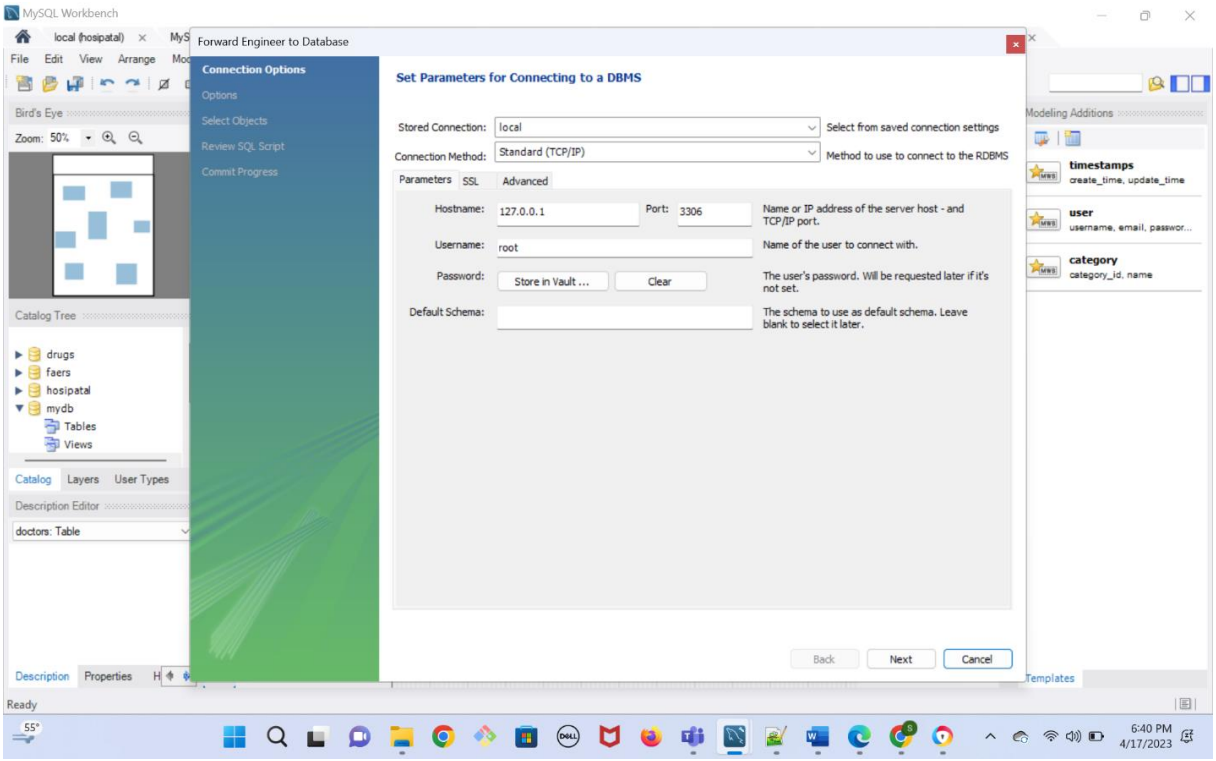
The four E-R diagrams in this question have been transformed into a relational database schema in accordance with the specifications.

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

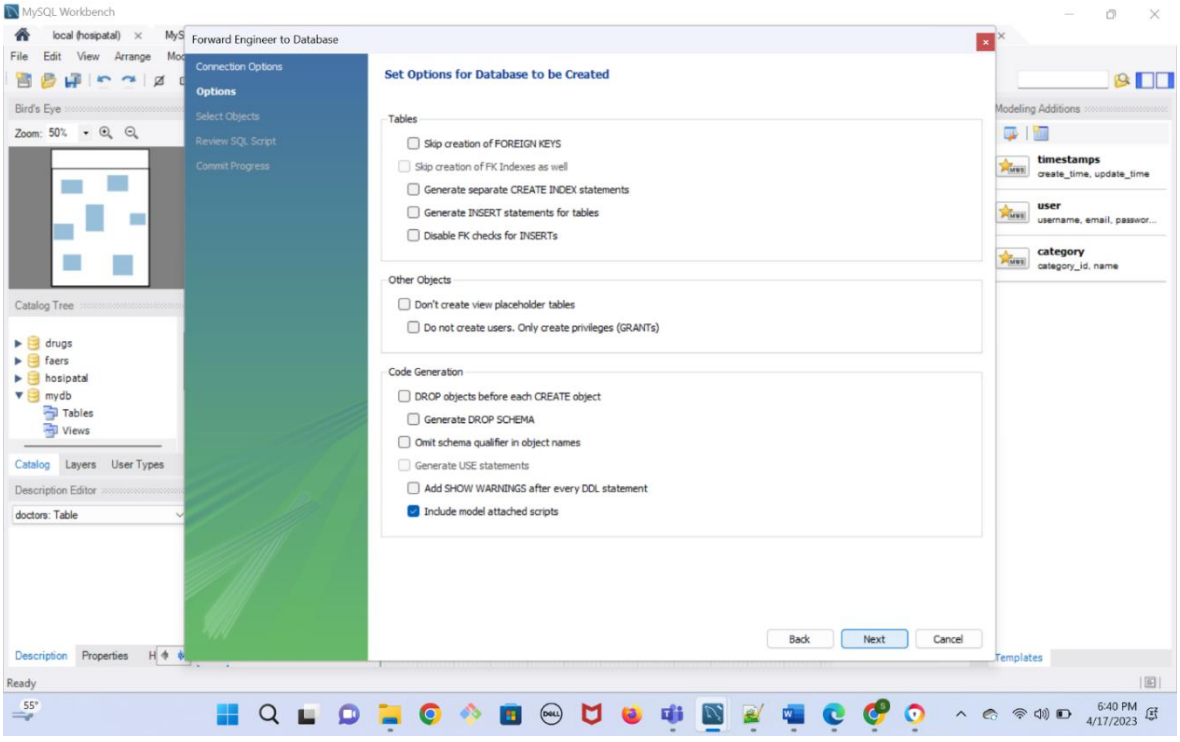
Step 1: Select Forward Engineer under Database



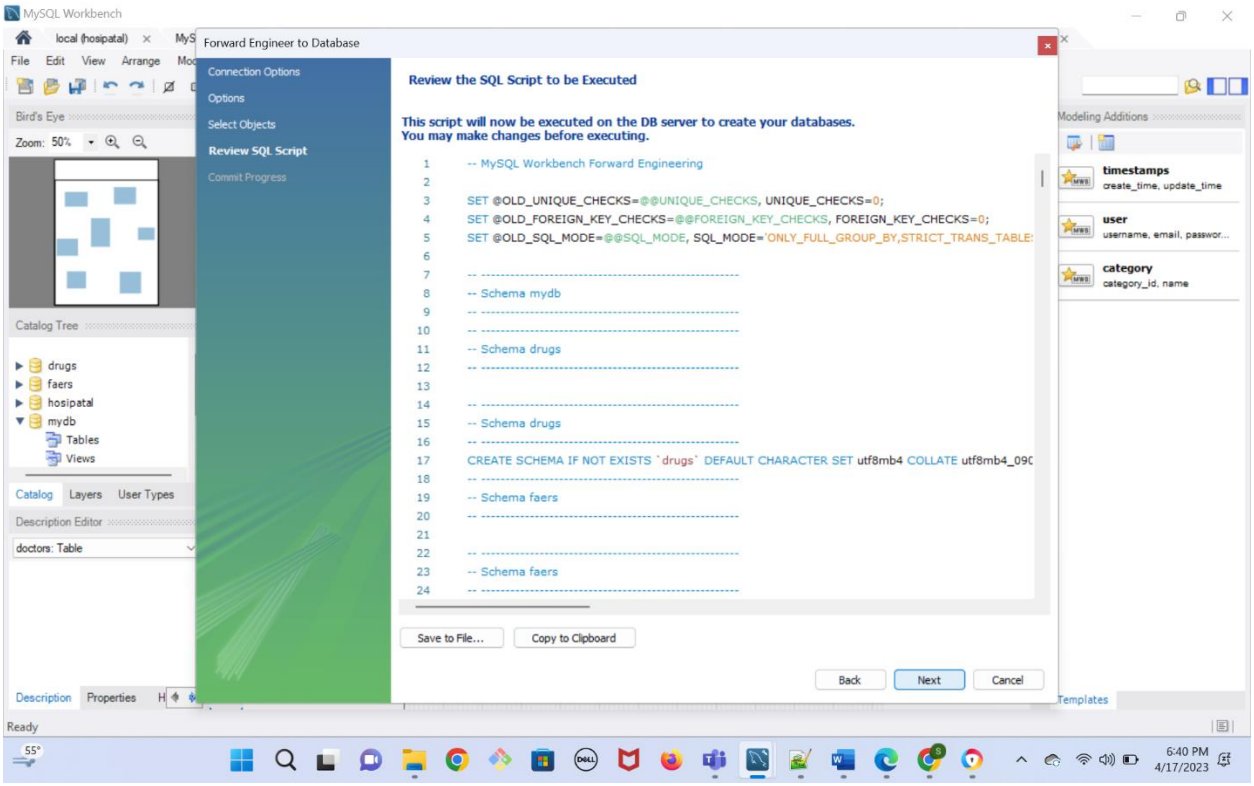
Step 2: Click Next Button



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

Hospital_Script.sql

Drug_Script.sql

Pharmacy_Script.sql

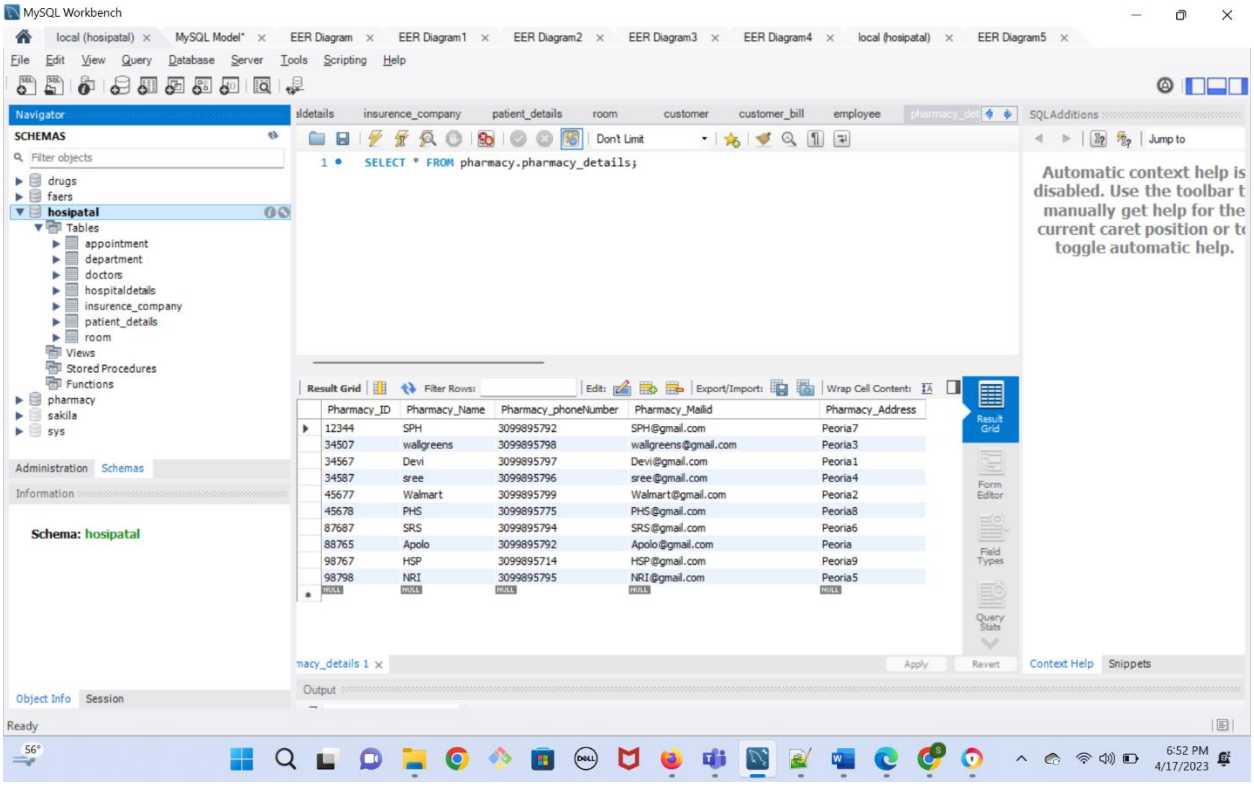
faers_Script.sql

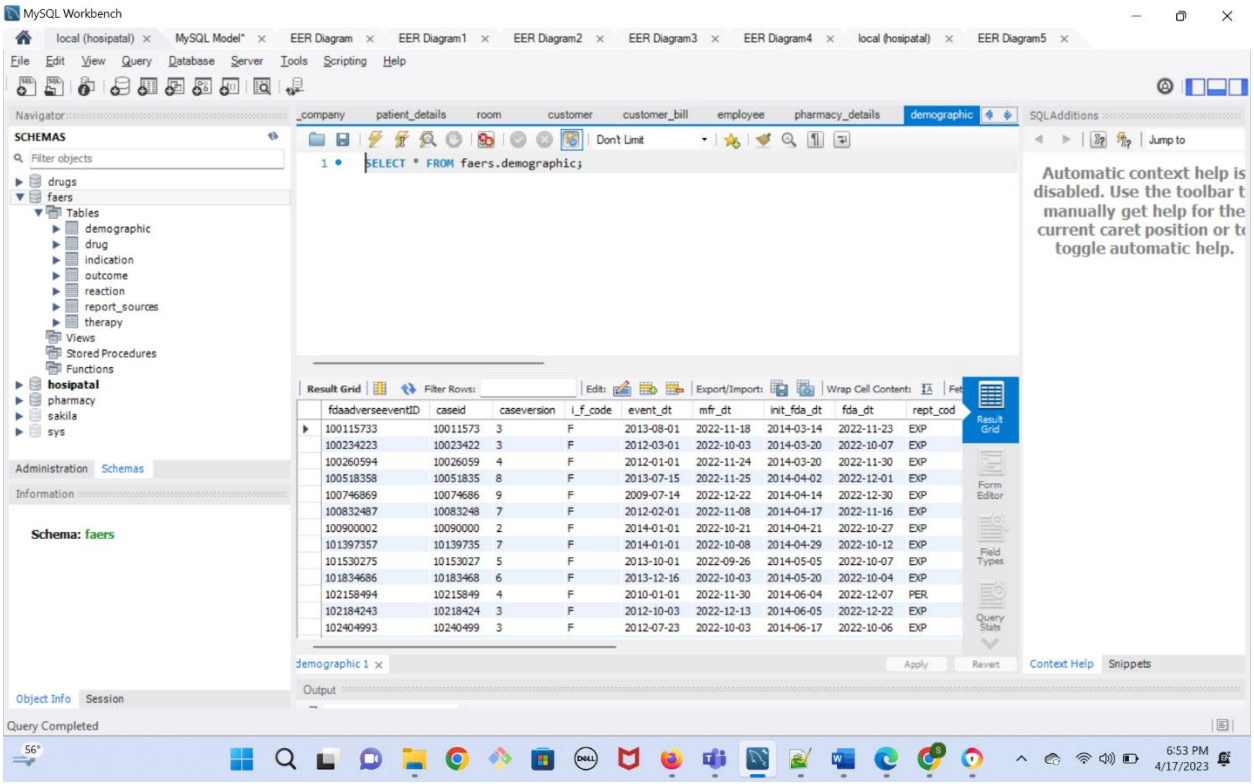
Question 5:

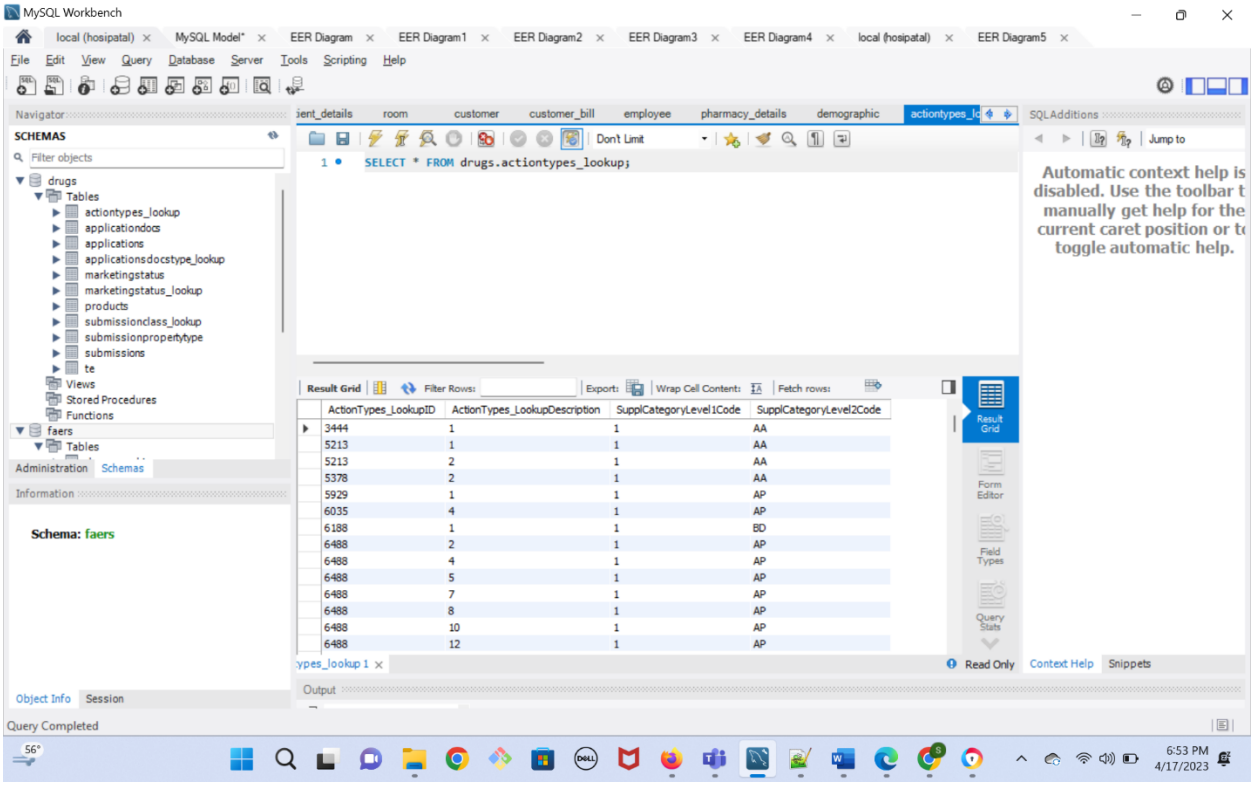
We have included information on tables that we have made for 4 Sub Systems in this query. I'm merely providing a couple screenshots of tables for reference.

Hospital, Pharmacy, Drug, FAERS Sub System

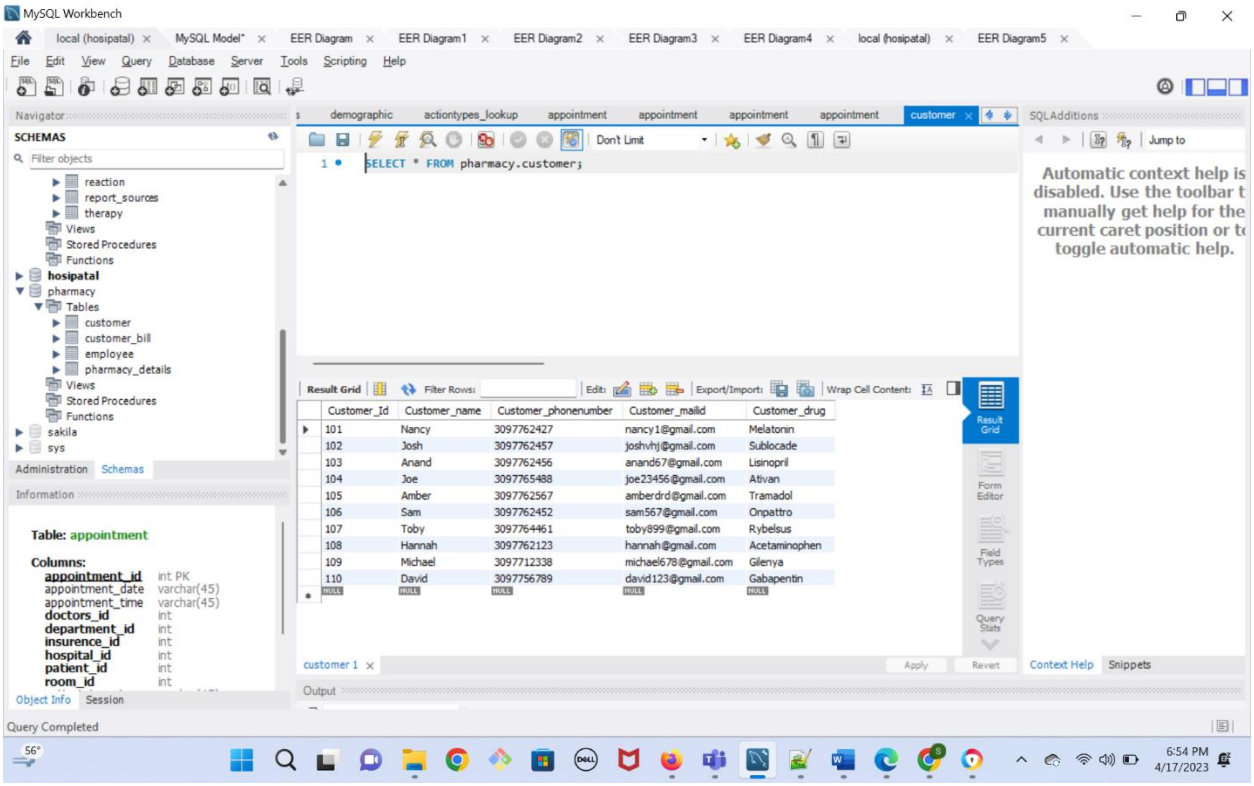
Step 1: Created a Hospital, Pharmacy, Drug, FAERS Sub Systems in MySQL

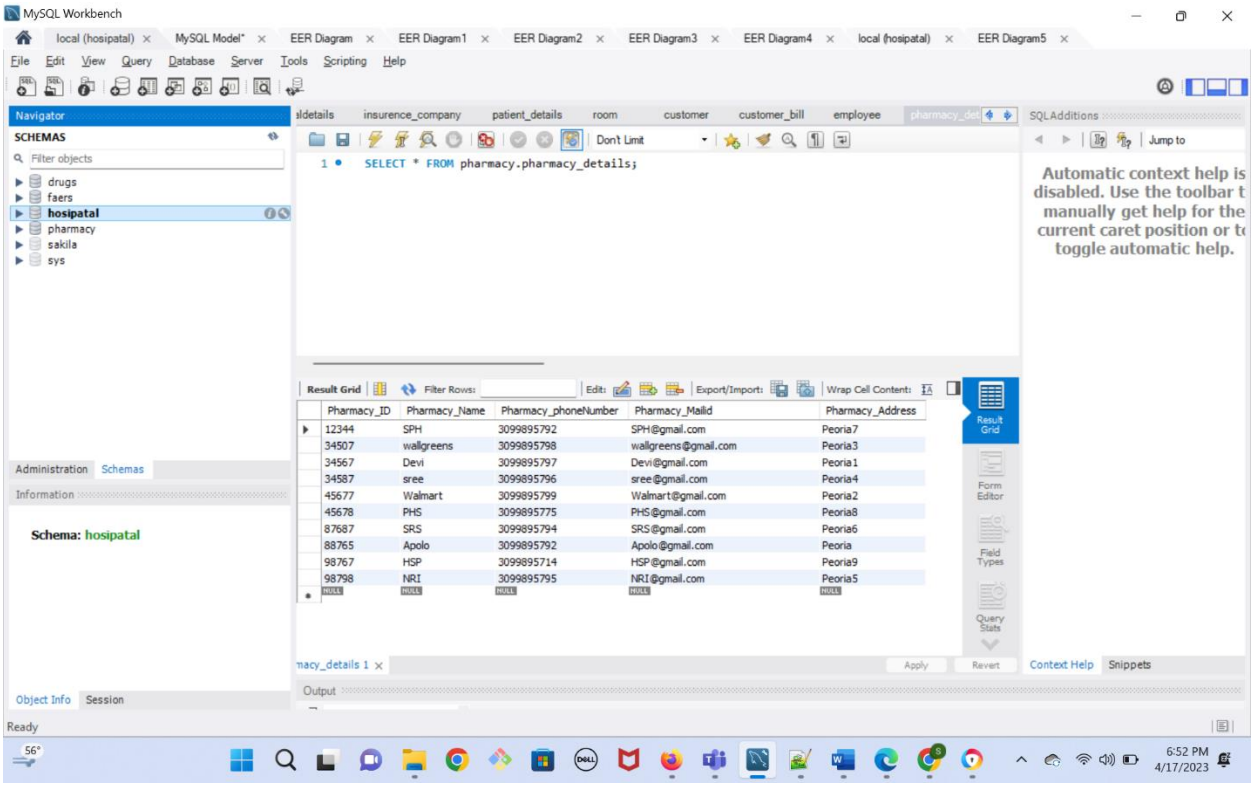



MySQL Workbench
SELECT * FROM faers.demographic;
Schema: faers
Query Completed

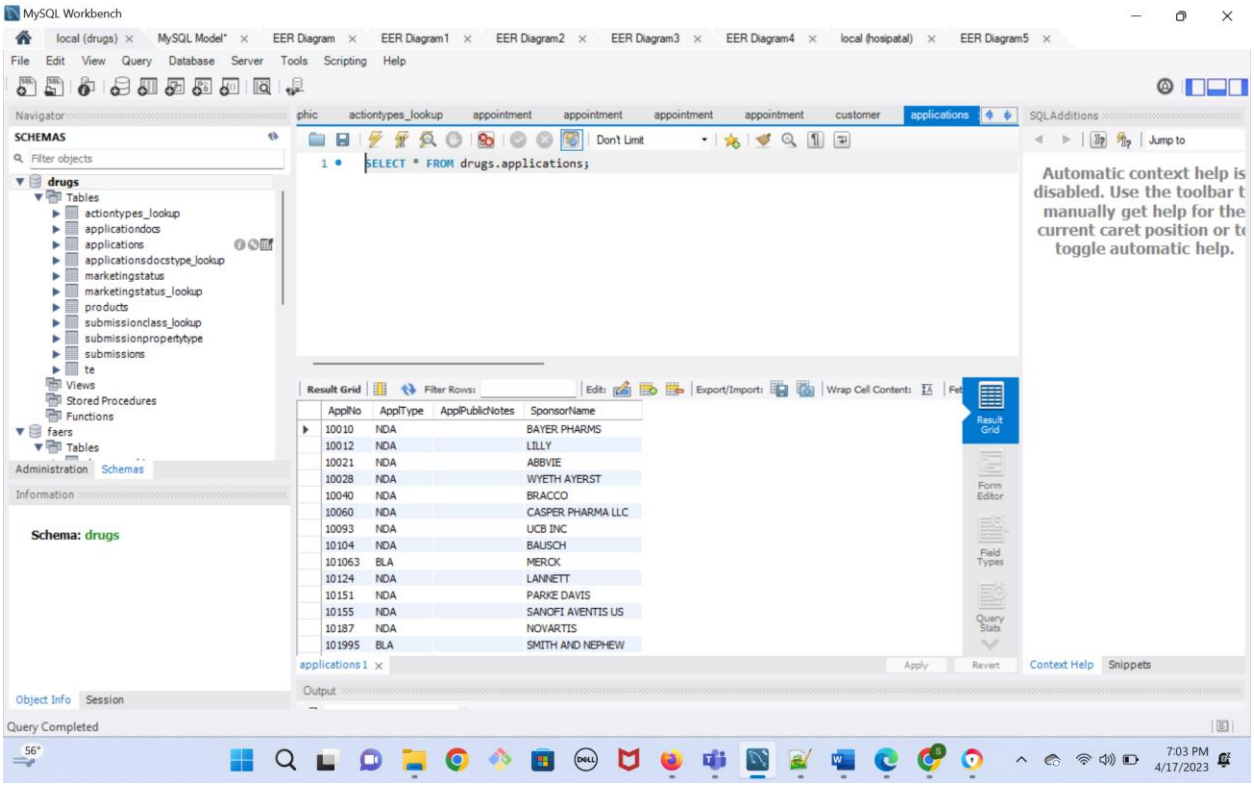

MySQL Workbench
SELECT * FROM drugs.actiontypes_lookup;
Schema: faers
Query Completed

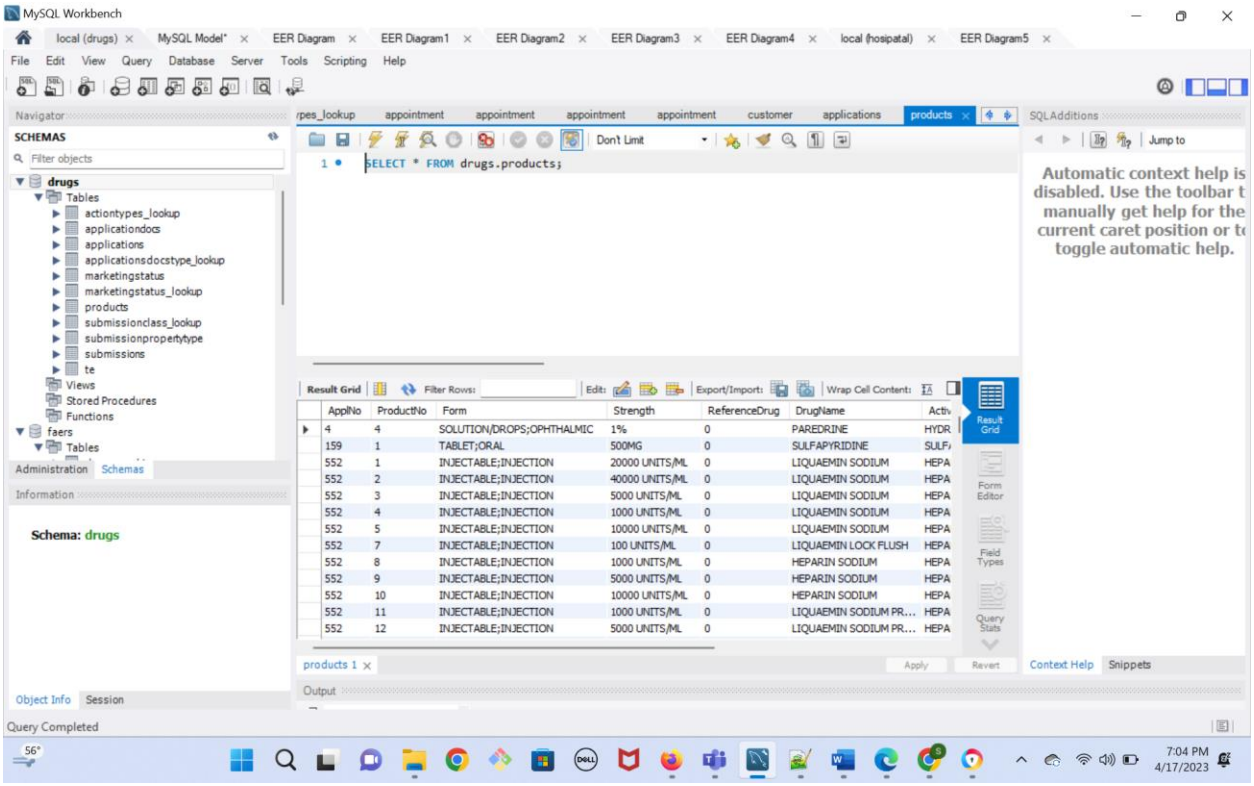
Step 2: Pharmacy Management Sub System

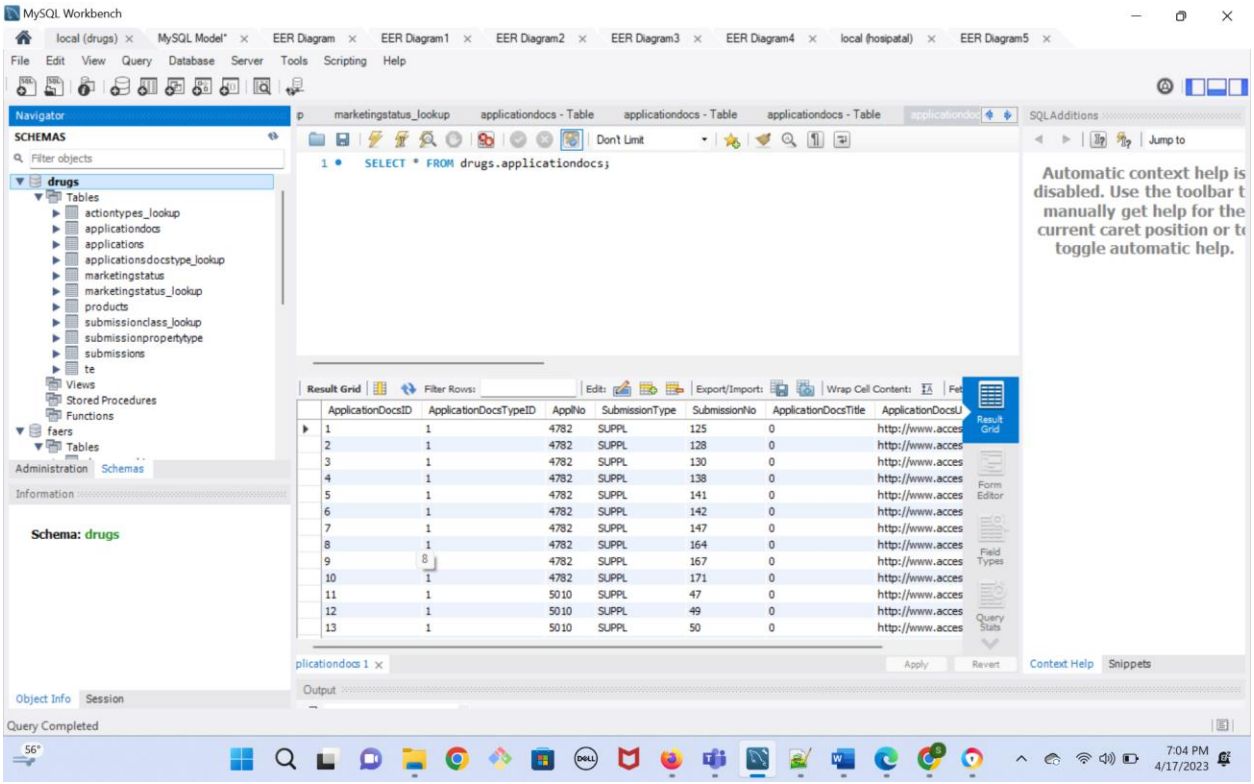




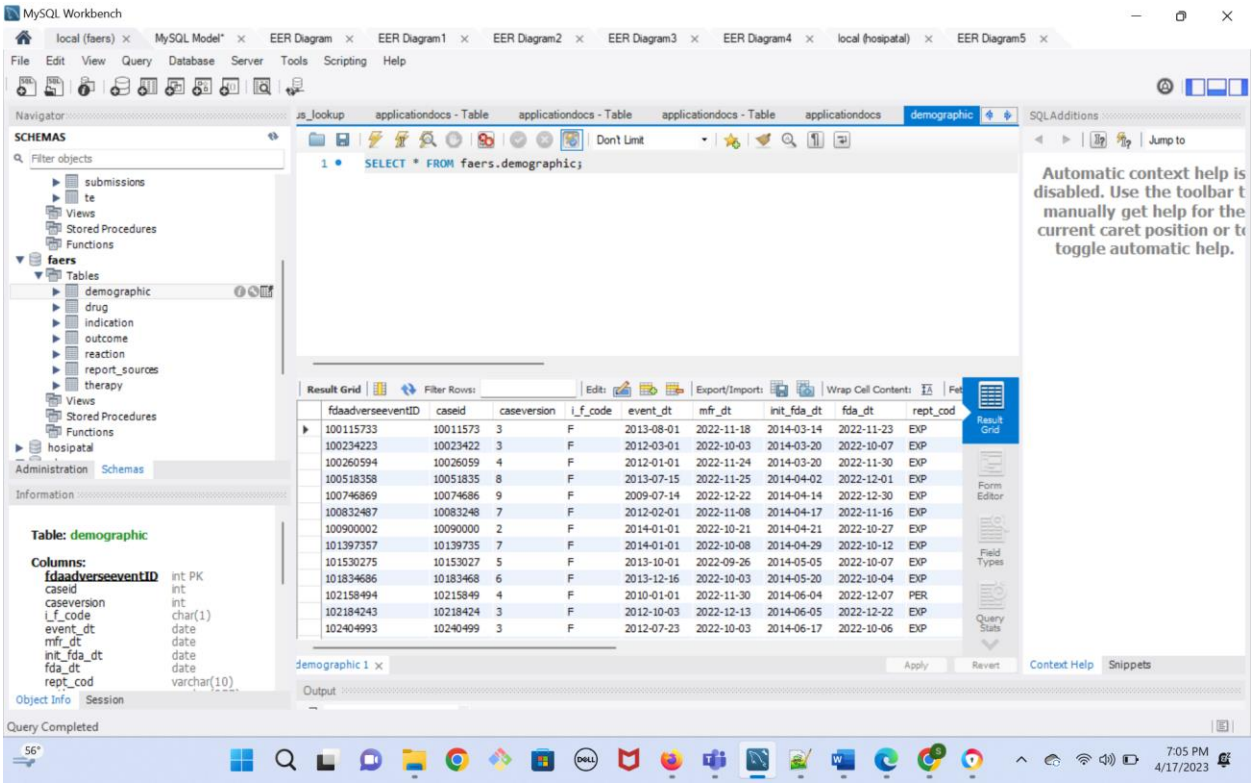
Step 3: Drug Sub System

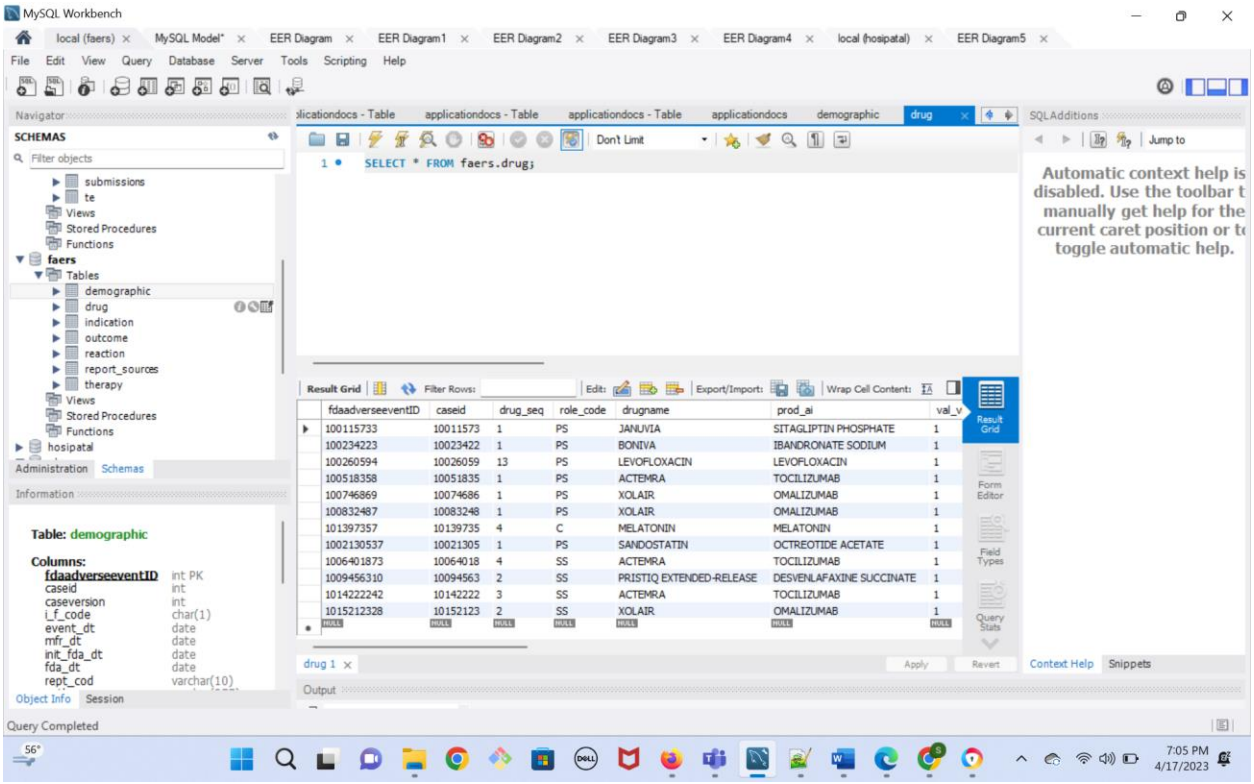


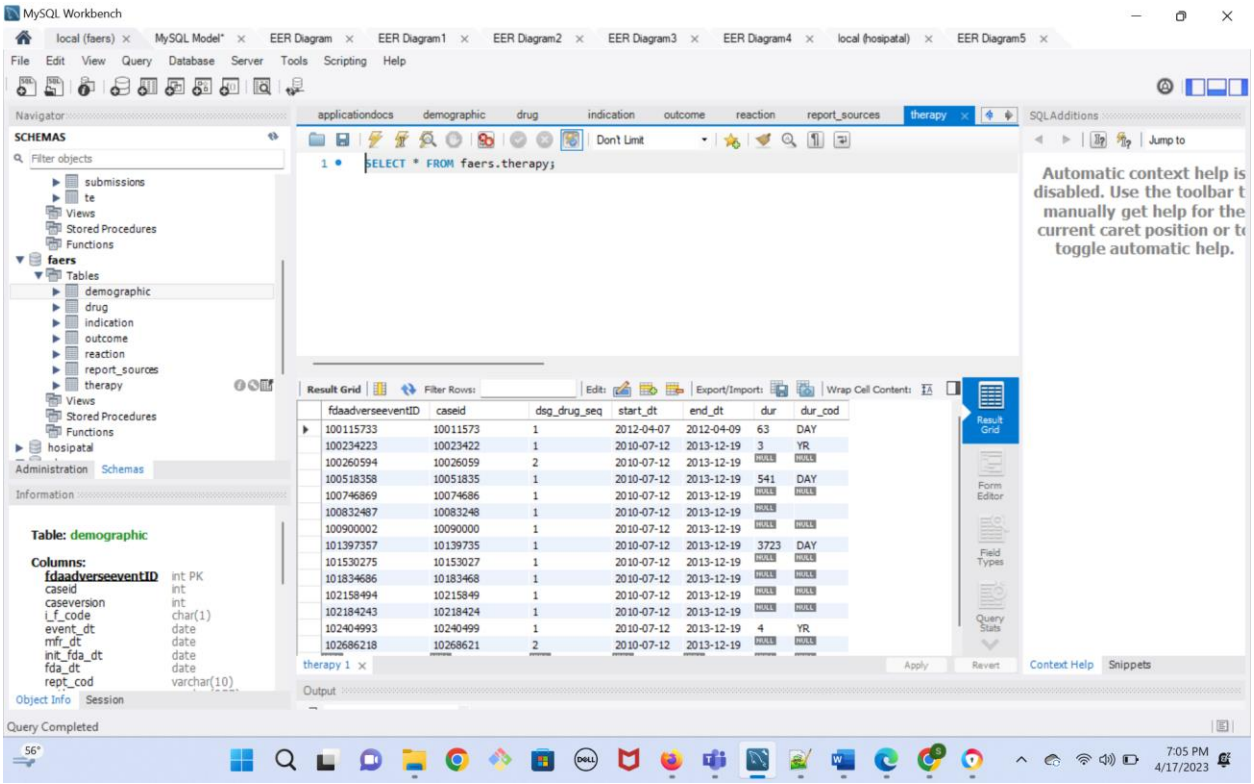




Step 3: FDA FAERS Sub System




MySQL Workbench
local (faers)
MySQL Model*
EER Diagram
EER Diagram1
EER Diagram2
EER Diagram3
EER Diagram4
local (hospatal)
EER Diagram5
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
submissions
te
Views
Stored Procedures
Functions
faers
Tables
demographic
drug
indication
outcome
reaction
report_sources
therapy
Views
Stored Procedures
Functions
hospatal
Administration Schemas
Information
Table: demographic
Columns:
fdaadverseeventID int PK
caseid int
caseversion int
i_f_code char(1)
event_dt date
mfr_dt date
init_fda_dt date
fda_dt date
rept_cod varchar(10)
Object Info Session
applicationdocs - Table
applicationdocs - Table
applicationdocs - Table
applicationdocs
demographic
drug
SELECT * FROM faers.drug;
Don't Limit
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
fdaadverseeventID caseid 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 TOCILIZUMAB 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 TOCILIZUMAB 1
1009456310 10094563 2 SS PRISTIQ EXTENDED-RELEASE DESVENLAFAXINE SUCCINATE 1
1014222242 10142222 3 SS ACTEMRA TOCILIZUMAB 1
1015212328 10152123 2 SS XOLAIR OMALIZUMAB 1
drug 1
Apply
Revert
Output
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.
Context Help Snippets
Query Completed
7:05 PM
4/17/2023


MySQL Workbench
local (faers)
MySQL Model*
EER Diagram
EER Diagram1
EER Diagram2
EER Diagram3
EER Diagram4
local (hospatal)
EER Diagram5
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
submissions
te
Views
Stored Procedures
Functions
faers
Tables
demographic
drug
indication
outcome
reaction
report_sources
therapy
Views
Stored Procedures
Functions
hospatal
Administration Schemas
Information
Table: demographic
Columns:
fdaadverseeventID int PK
caseid int
caseversion int
i_f_code char(1)
event_dt date
mfr_dt date
init_fda_dt date
fda_dt date
rept_cod varchar(10)
Object Info Session
applicationdocs
demographic
drug
indication
outcome
reaction
report_sources
therapy
SELECT * FROM faers.therapy;
Don't Limit
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
fdaadverseeventID caseid 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
100518358 10051835 1 2010-07-12 2013-12-19 541 DAY
100746869 10074686 1 2010-07-12 2013-12-19
100832487 10083248 1 2010-07-12 2013-12-19
100900002 10090000 1 2010-07-12 2013-12-19
101397357 10139735 1 2010-07-12 2013-12-19 3723 DAY
101530275 10153027 1 2010-07-12 2013-12-19
101834686 10183468 1 2010-07-12 2013-12-19
102158494 10215849 1 2010-07-12 2013-12-19
102184243 10218424 1 2010-07-12 2013-12-19
102404993 10240499 1 2010-07-12 2013-12-19 4 YR
102686218 10268621 2 2010-07-12 2013-12-19
therapy 1
Apply
Revert
Output
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.
Context Help Snippets
Query Completed
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_avalibality
FROM hosipatal.doctors doc
        where doctors_specilization = "Gynecologists";
```

Using Relational Algebra:

π doc.doctors_id, doc.doctors_name, doc.doctors_avalibality (σ doc.doctors_specilization = "Gynecologists" (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 avalibality" columns from the generated table are the only ones to be projected.

Query 2:

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

Using Relational Algebra:

π Patient_Gender, count(Patient_Gender)(γ 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_avalibality,
        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:

π doc.doctors_id, doc.doctors_name, doc.doctors_avalibality, hos.hospital_name, doc.doctors_phonenumber ((doc ⋈ doc.hospital_id = hos.hospital_id hos))

Explanation:

doc ⋈ 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_avalibality, hos.hospital_name, doc.doctors_phonenumber: Project only the "doctors_id", "doctors_name",

"doctors_avalibality", "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.insurence_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.insurence_company ins on apt.insurence_id = ins.insurence_id
left join hosipatal.room rm on apt.room_id = rm.room_id;
```

Using Relational Algebra:

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.insurence_companyname ((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.insurence_id = ins.insurence_id left join rm on apt.room_id = 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.insurence_id = ins.insurence_id: Perform another left outer join on the resulting table from the previous join with the "insurence_company" table (ins) on the condition that the "insurence_id" column in "apt" matches the "insurence_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.insurence_id = ins.insurence_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.insurence_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:

Π 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 ((σ bill.customer_id = cus.customer_id (cus ⋈ bill)) ⋈ cus.Customer_name, cus.customer_phonenumber, bill.customer_ssn, bill.customer_billamount, bill.drug_quantity (group by cus.Customer_name, cus.customer_phonenumber))

Explanation:

cus ⋈ 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".

σ bill.customer_id = cus.customer_id (cus ⋈ 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".

group by cus.Customer_name, cus.customer_phonenumber: Group the resulting table by the "Customer_name" and "customer_phonenumber" columns in "cus".

π 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: 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:

π emp.Employee_ID, Emp.Employee_Name, phm.pharmacy_name, sum(bill.customer_billamount) as Sales_Amount ((emp ⊗ phm) ⋈ bill) (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:

π phm.pharmacy_name, sum(bill.customer_billamount) as Sales_Amount (phm ⋈ bill) (group by phm.pharmacy_name)

Explanation:

phm ⋈ 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".

π phm.pharmacy_name, sum(bill.customer_billamount) 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_lk.SubmissionClassCode,
        sub_lk.SubmissionClassCodeDescription
FROM drugs.submissions sub left join  drugs.submissionclass_lookup sub_lk
on sub.SubmissionClassCodeID = sub_lk.SubmissionClassCodeID;
```

Using the Relational Algebra:

π sub.SubmissionNo, sub.SubmissionType, sub_lk.SubmissionClassCode, sub_lk.SubmissionClassCodeDescription (sub ⋈ sub_lk)

Explanation:

sub ⋈ 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".

π sub.SubmissionNo, sub.SubmissionType, sub_lk.SubmissionClassCode, sub_lk.SubmissionClassCodeDescription: 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_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;

Using the Relational Algebra:

π app.ApplNo, app.ApplType, app.SponsorName, ms.MarketingStatusID, ms_lk.MarketingStatusDescription ((drugs.applications app ⋈ drugs.marketingstatus ms) ⋈ 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 ⋈ drugs.marketingstatus ms) ⋈ 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:

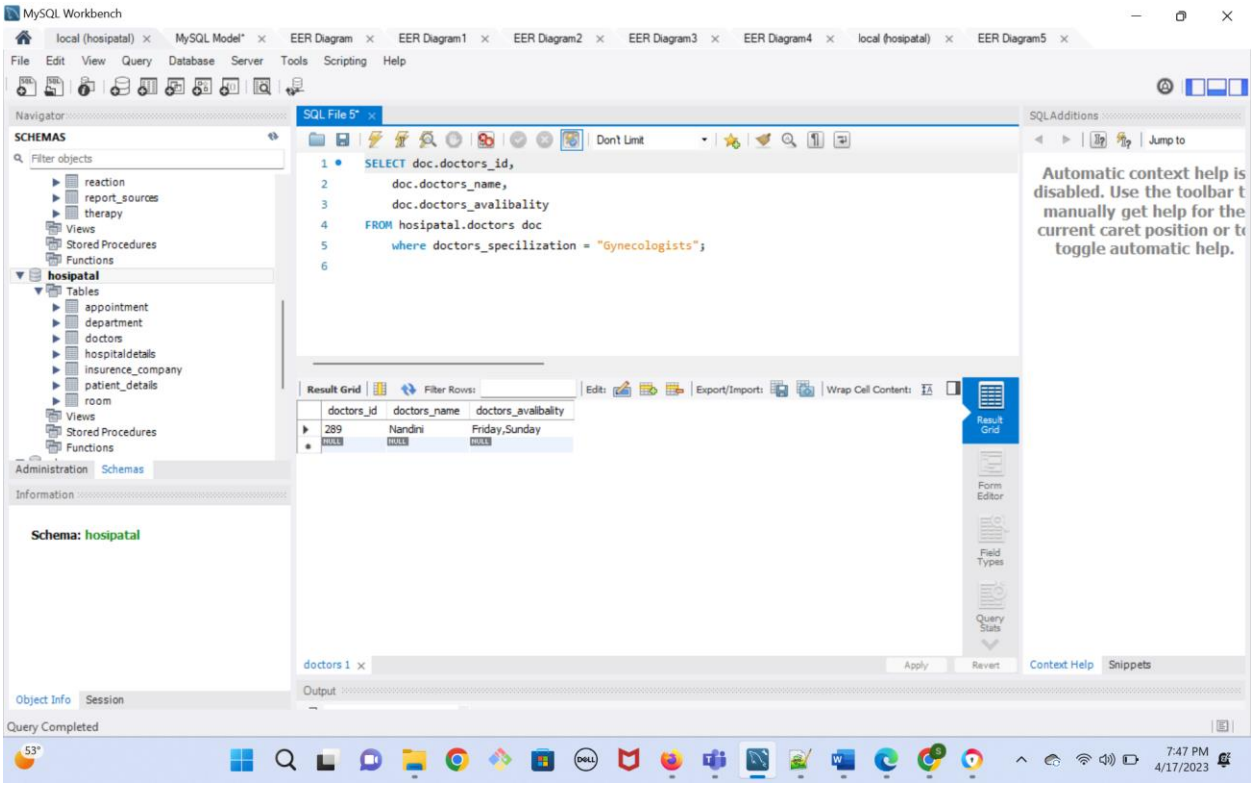
As per the requirement, we have written 10 queries and attached screenshots of the results in this question.

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_avalibality
FROM hosipatal.doctors doc
        where doctors_specilization = "Gynecologists";
```

Result:

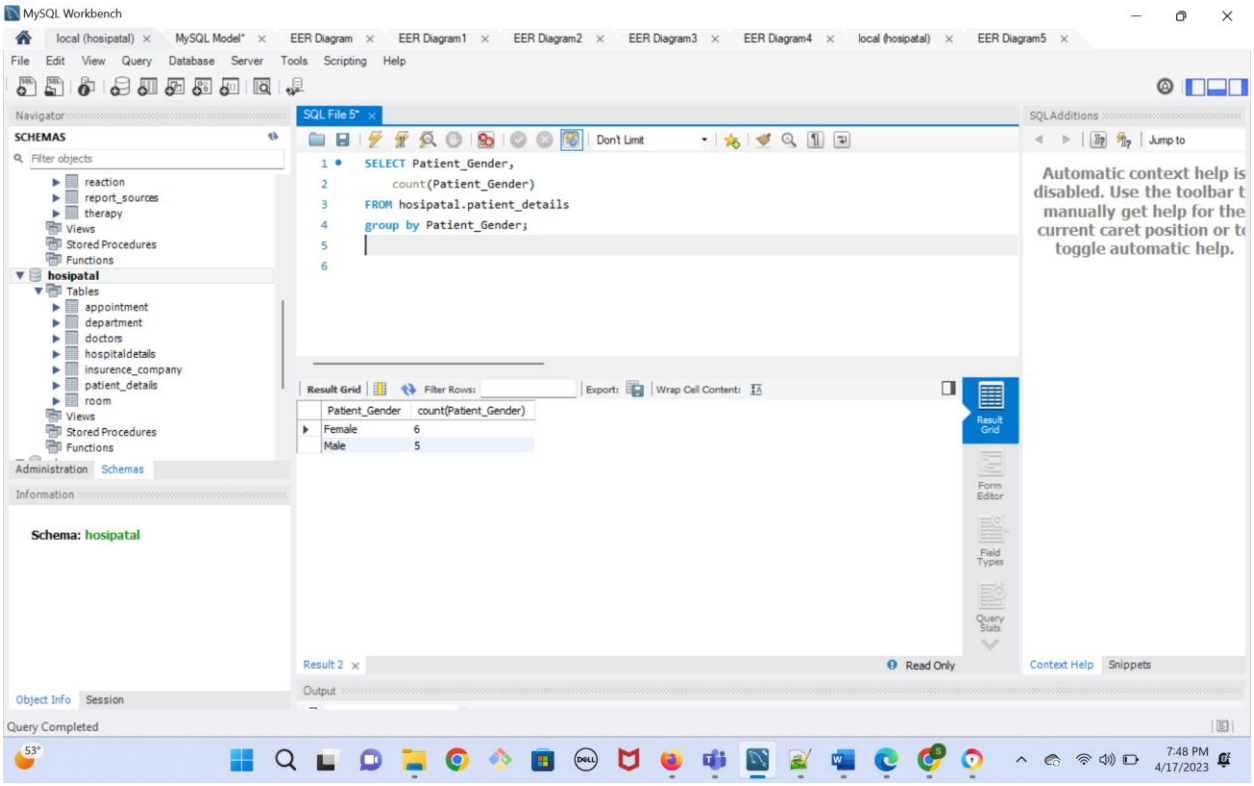


Query 2:

write a query to display total male and female patient count

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

Result:

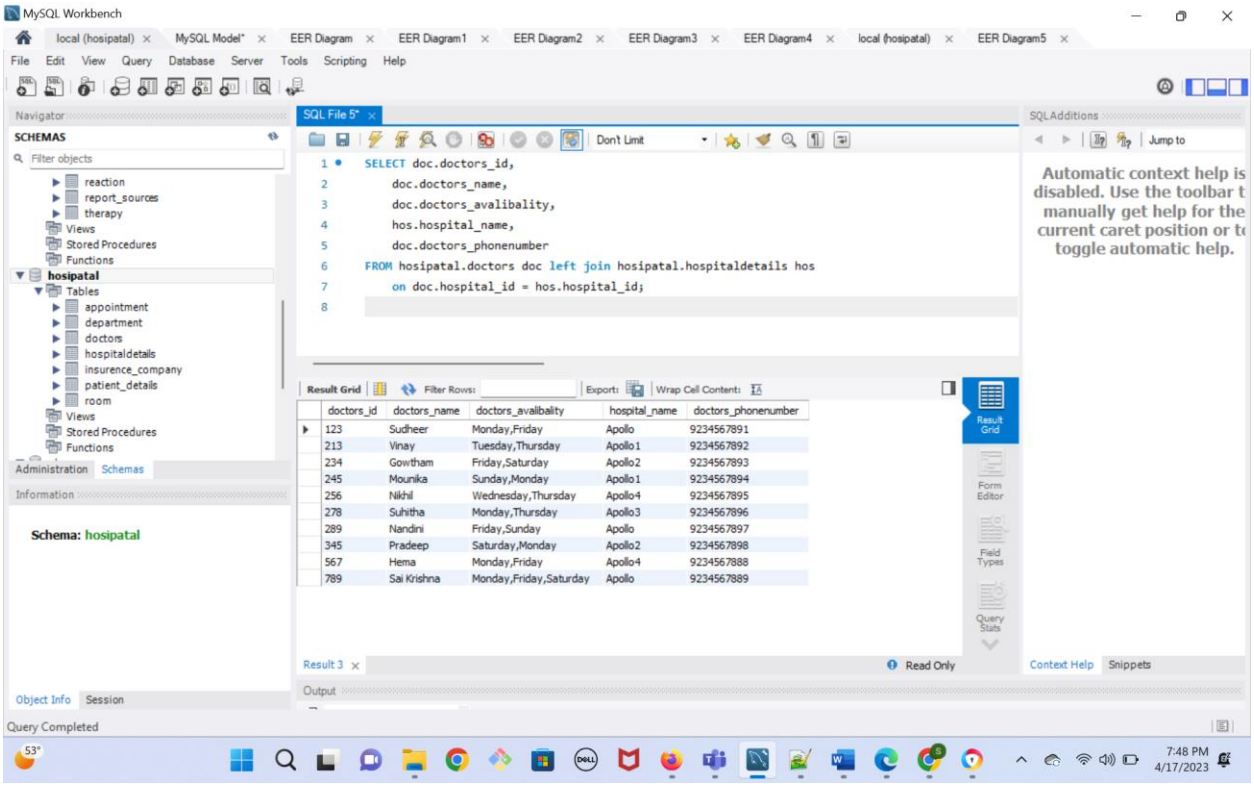


Query 3:

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

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

Result:



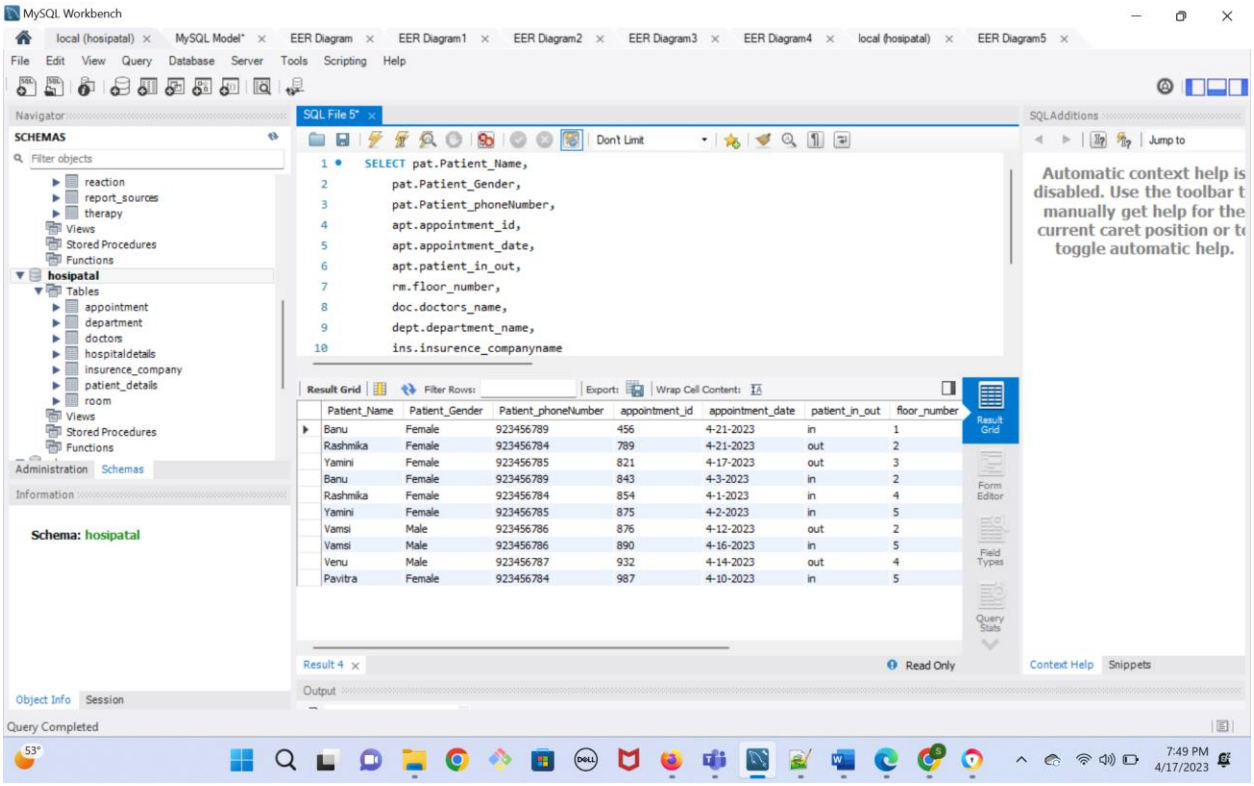
Query 4:

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.insurence_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.insurence_company ins on apt.insurence_id = ins.insurence_id
left join hosipatal.room rm on apt.room_id = rm.room_id;
```

Result:

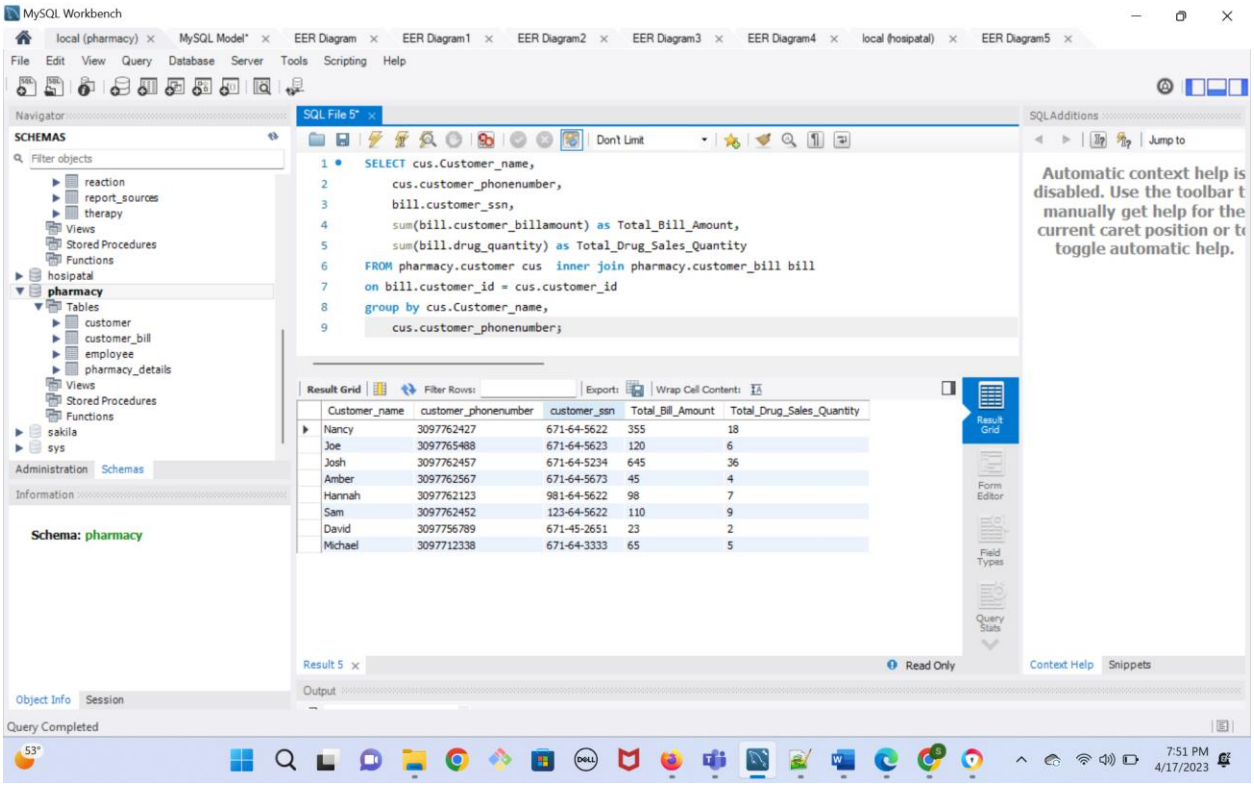


Query 5:

Write a query to display customer details along with purchaged amount and purchanged drug qunatity 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:

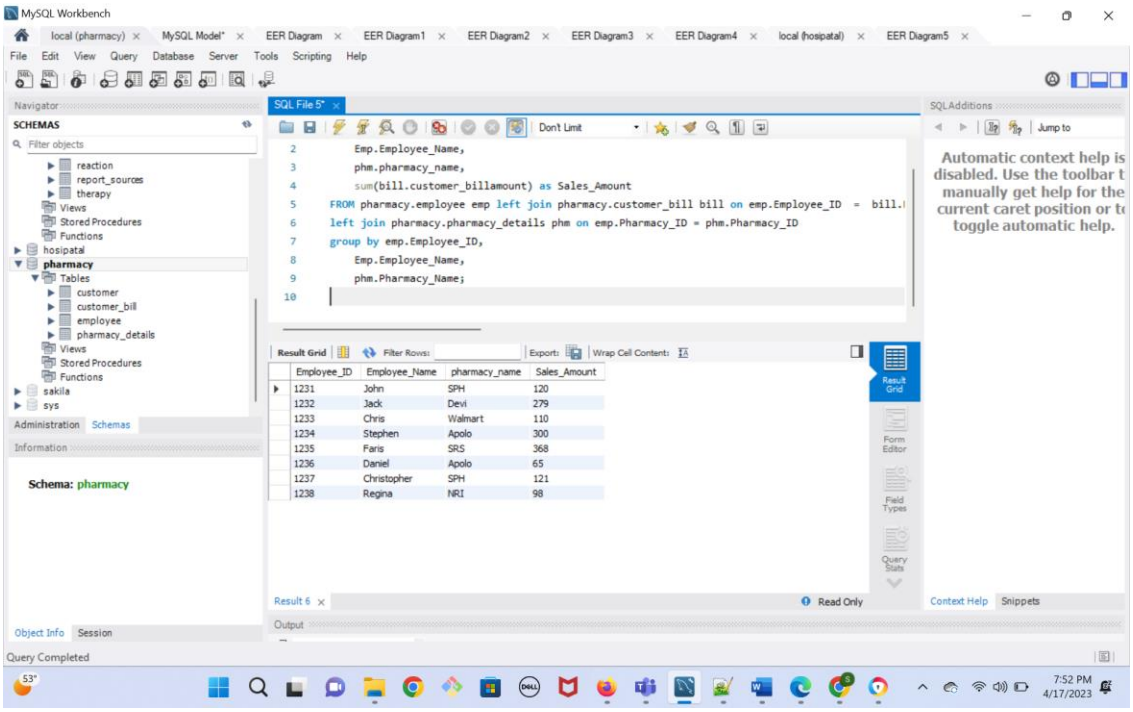


Query 6:

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:

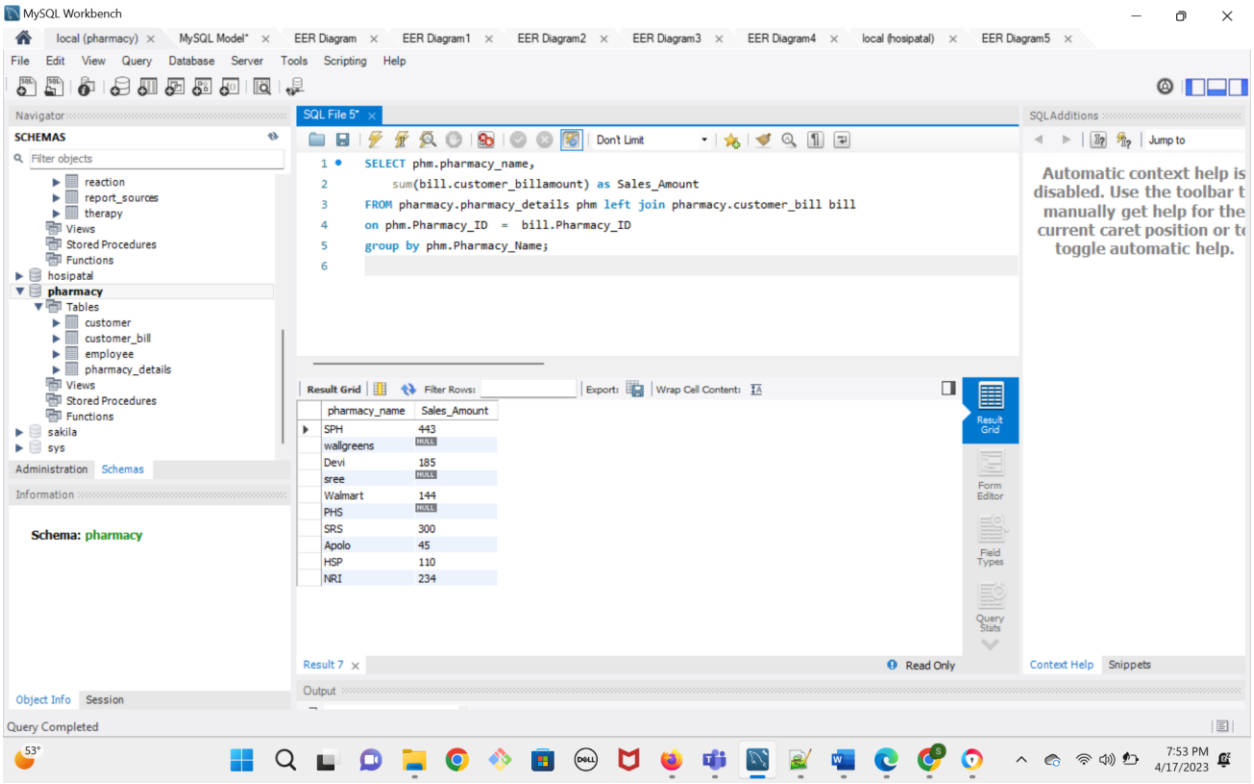


Query 7:

Write a query to diplay 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:

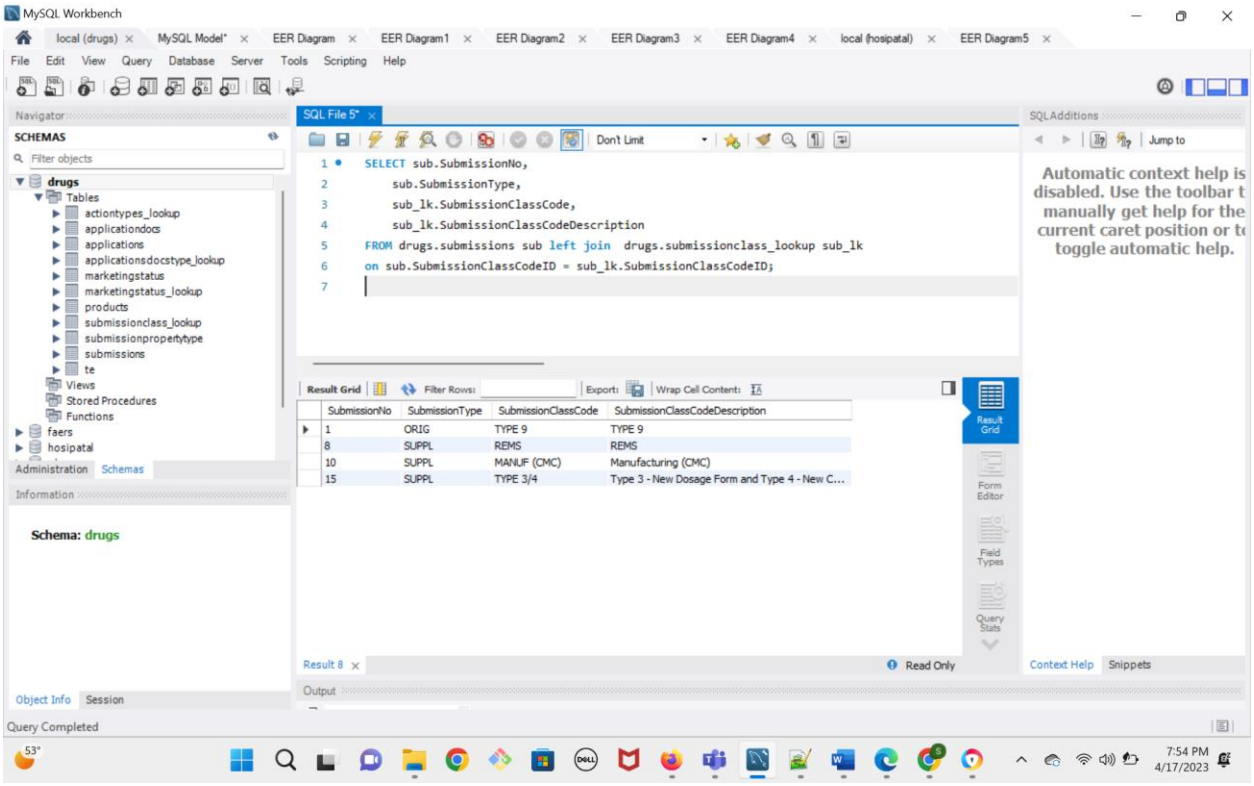


Query 8:

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

```
SELECT sub.SubmissionNo,
        sub.SubmissionType,
        sub_lk.SubmissionClassCode,
        sub_lk.SubmissionClassCodeDescription
FROM drugs.submissions sub left join  drugs.submissionclass_lookup sub_lk
on sub.SubmissionClassCodeID = sub_lk.SubmissionClassCodeID;
```

Result:

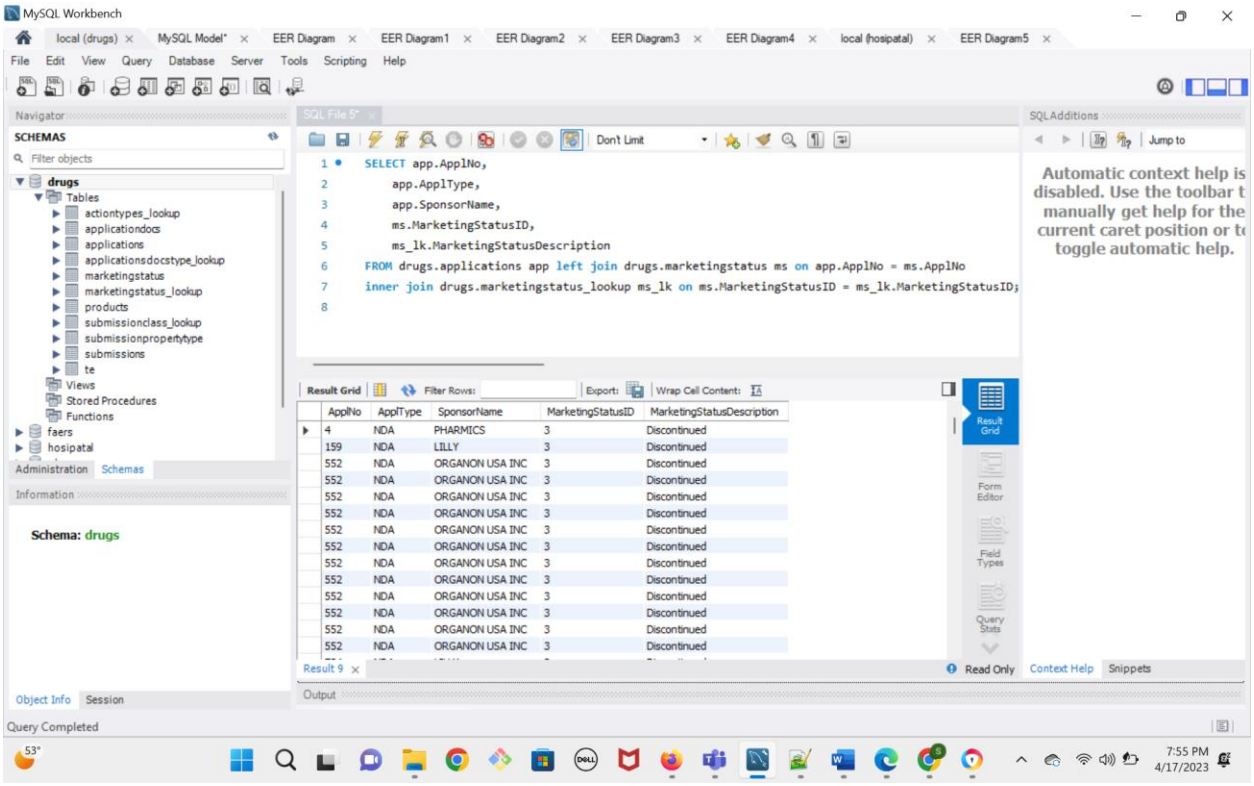


Query 9:

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:

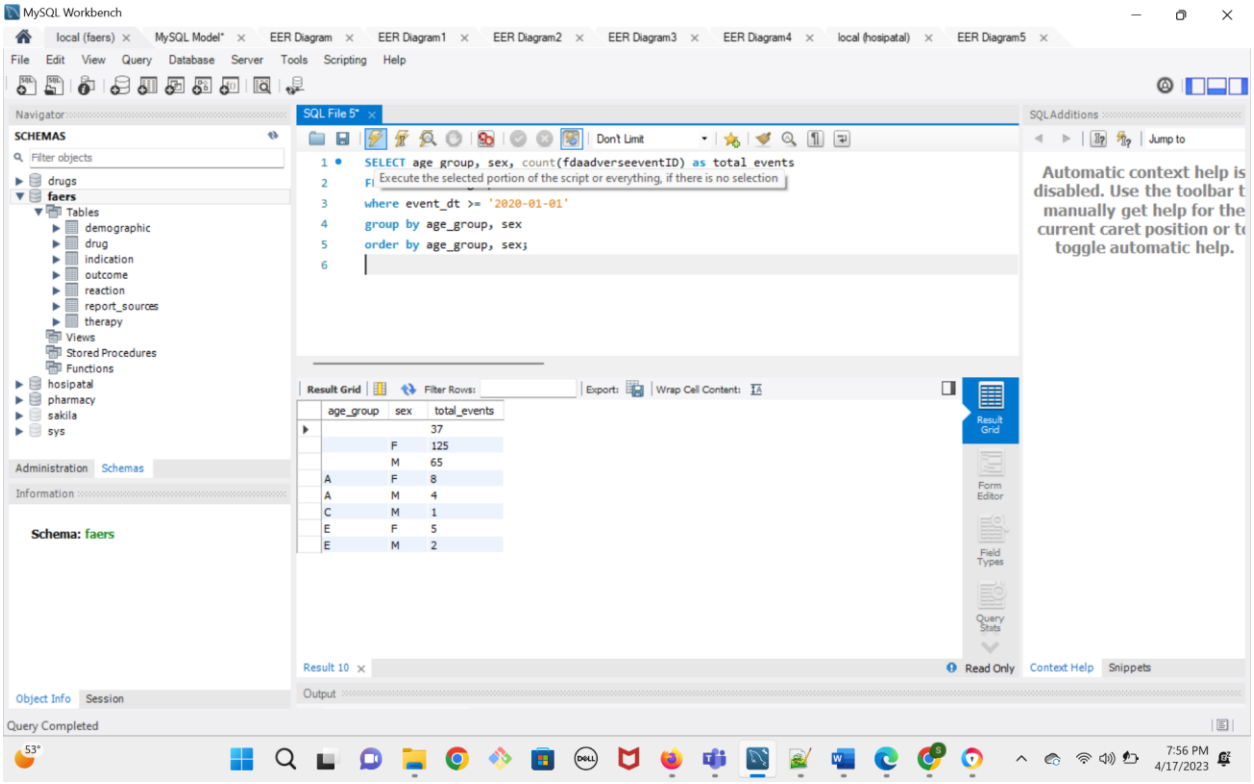


Query 10:

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:



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: