

FAERS

Dayana Alberto

Kim Ngan Thai

Luis Franco

Overview

This project is a collaboration with Dr. Feng Cheng, a research professor of the Department of Pharmaceutical Sciences, College of Pharmacy and Department of Biostatistics, College of Public Health, to help develop an enterprise management system giving the FDA Adverse Event Reporting System (FAERS) database.

A simple search of FAERS data cannot be performed with these files, hence this management system, FAERS Search Tool as we called it, was designed and developed. This search tool helps the process of looking up raw data, extracted from the AERS data files a lot easier when going through the large database.

Client Functionality

In order to start using the search feature of the website, a person must first register in our system. After that, users database will be established and will be divided into two types: admin and regular user.

1. Regular Users
 - Regular users can look up and search for specific drug name, and adverse event. They can view, and print the results in pdf form.
2. Admin
 - Admin can do everything that any regular users can, but with few additional features. User with a role of an admin will have the user page available to them. On when they login as admin, that page will show. They can view the list of users, and make changes to the list such as assigning the roles to different users, and deleting anyone out of the database, but themselves.

Of the seven total page types shown below, **Regular Users** can view 6 of these pages, while an **Admin** can view all of the pages mentioned above and an additional page called Users Page:

1. Home Page (2x)
2. Register
3. Login Page
4. Search Page - This page shows the demographic table with links to select each primary ID and display each ID's associated drug information.
5. Drug Page
6. Adverse Events Page
7. Users Page

The following section will provide detailed information on each and every page, their functionality, and how the user can interact with each one.

Page Description

Before Login

Regular Users can only view the homepage. They will only interactive with the register page and login page.

Page 1: Home Page (2x)

The first homepage (Figure 1) contains all the information about FAERS, links to visit for other additional information, and instructions on how to use the FAERS search tool on our web server (Figure 2). This page contains the Register and Login Page.

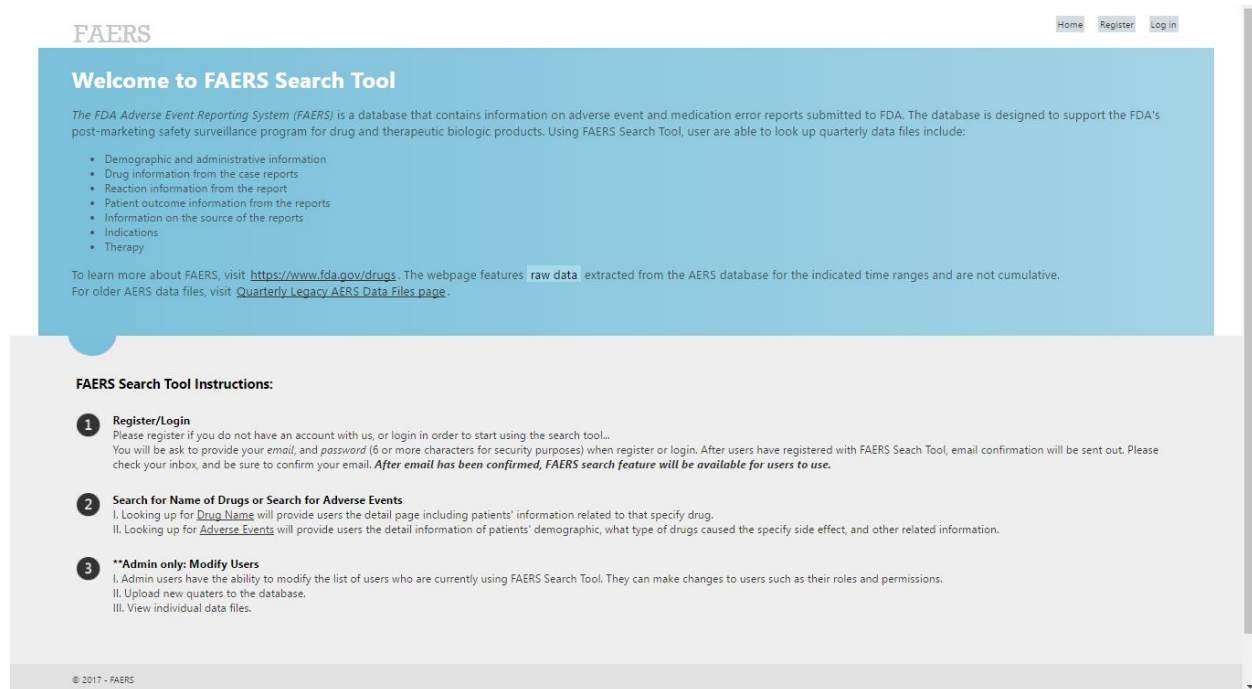


Figure 1 displays FAERS homepage when users first entered the site

FAERS Search Tool Instructions:

- 1 Register/Login**
 Please register if you do not have an account with us, or login in order to start using the search tool...
 You will be ask to provide your *email*, and *password* (6 or more characters for security purposes) when register or login. After users have registered with FAERS Seach Tool, email confirmation will be sent out. Please check your inbox, and be sure to confirm your email. ***After email has been confirmed, FAERS search feature will be available for users to use.***
- 2 Search for Name of Drugs or Search for Adverse Events**
 I. Looking up for Drug Name will provide users the detail page including patients' information related to that specify drug.
 II. Looking up for Adverse Events will provide users the detail information of patients' demographic, what type of drugs caused the specify side effect, and other related information.
- 3 **Admin only: Modify Users**
 I. Admin users have the ability to modify the list of users who are currently using FAERS Search Tool. They can make changes to users such as their roles and permissions.
 II. Upload new quaters to the database.
 III. View individual data files.

© 2017 - FAERS

Figure 2 displays the instructions prior to use FAERS Search Tool

Page 2: Register

The register page (Figure 3) will ask a user to create a unique username, and a secure password.

FAERS Home Register Log in

Register. Create a new account.

Email address

Password

Confirm password

© 2017 - FAERS

Figure 3 displays registration page of first time users

Page 3: Login

The login page (Figure 4) will ask a user for their username and password to use to log into the system. The application layer will query the database with the information the user has provided and check to see if it is valid in the system.

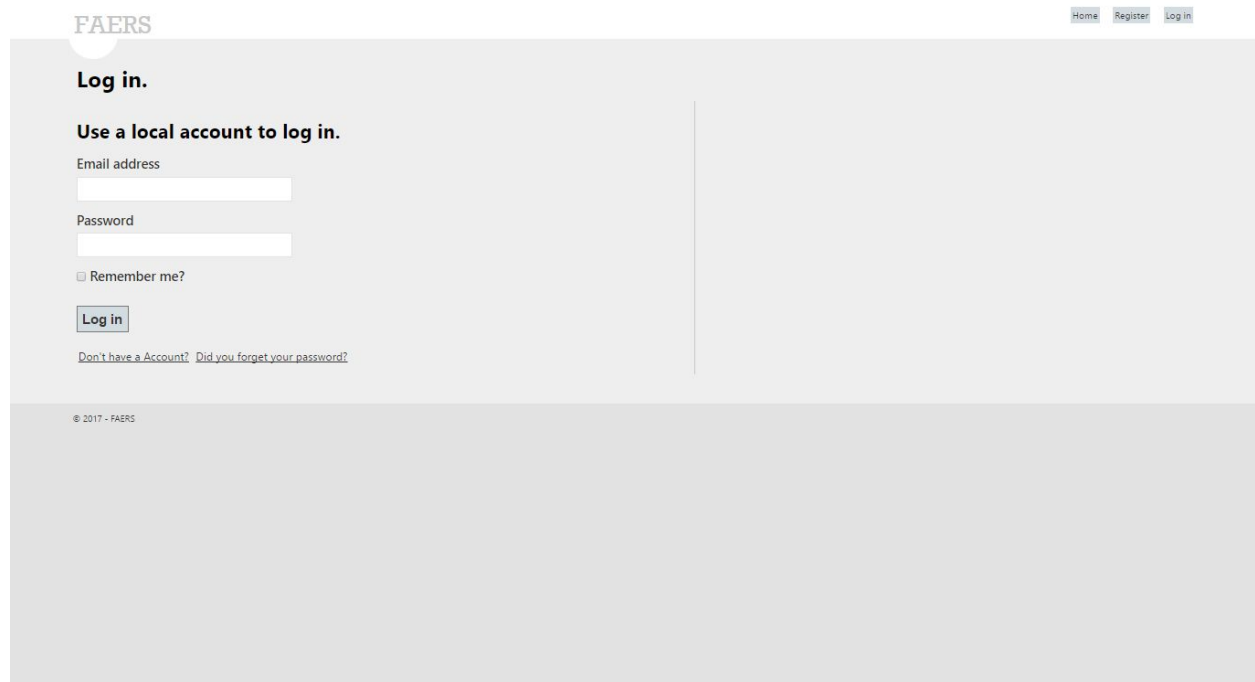


Figure 4 displays the login page for returning users

After Login Home Page

After user has successfully login, they will be directed to a different homepage immediately (Figure 5). This homepage is a welcome page with a greeting at the top left. The greeting will change depending on the type of user, admin vs regular user. Below the greeting, it will show you the instruction and show a message saying that your login/registered steps is finished, and you can now start using FAERS search features. Every page will have a menu bar at the top. This will allow users to move fluidly around our system from any page. Most users will have access to the same menu bar although some of the web pages may not be available unless a user is an admin.

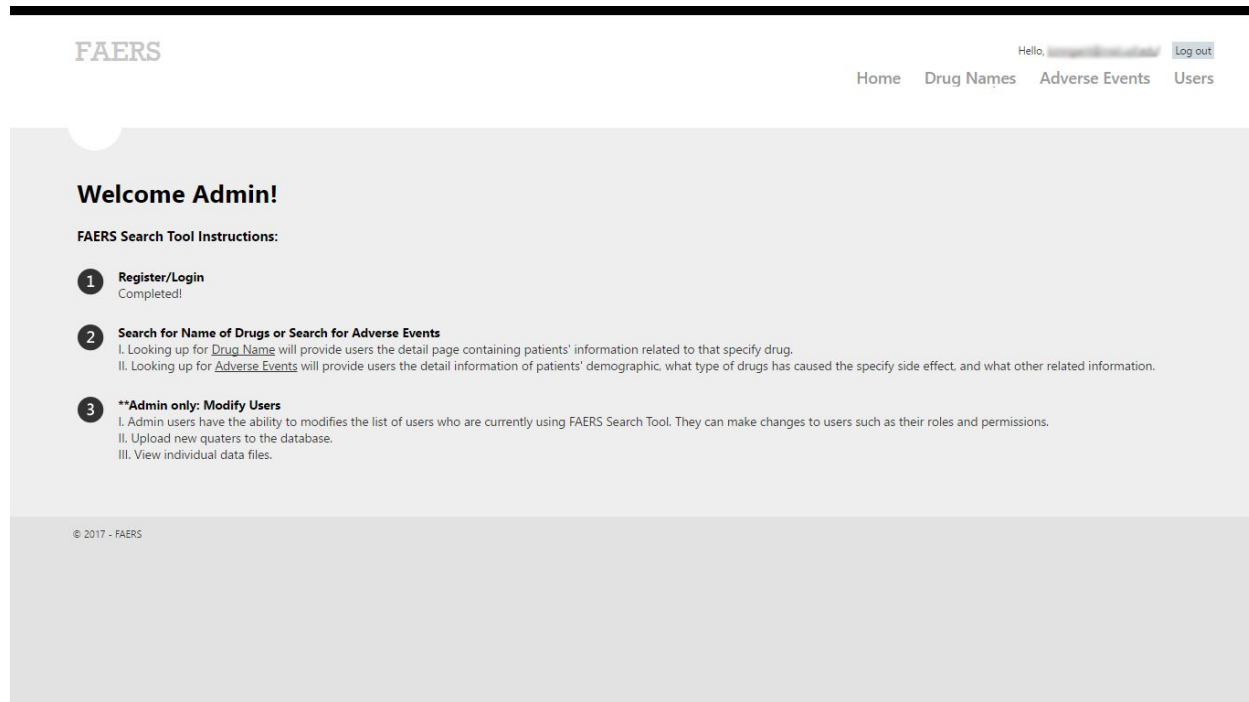


Figure 5 displays home page after user has login as an Admin

Page 4: Search Page (2x)

- a. Drug Name (Figure 6)
- b. Adverse Events (Figure 7)

This page consist of two modes. One is to search for drug names, and the second one is the search for adverse events.

- Search for drug names allow users to look up a name of the drug, choose from the list of drugs containing that keyword and view the result base by clicking on the case id.

Search Drug Names

- [RUXOLITINIB](#)
- [RUXOLITINIB INCYTE](#)
- [RUXOLITINIB PHOSPHATE](#)

Figure 6 displays the list of drugs containing the lookup keywords

- Similarly, user will see the same procedure when searching for adverse events. They can look up adverse event, choose from the list containing that keyword and view the result base by clicking on the case id.

Search Adverse Events

- [Anaemia](#)
- [Anaemia folate deficiency](#)
- [Anaemia macrocytic](#)
- [Anaemia megaloblastic](#)
- [Anaemia neonatal](#)
- [Anaemia of chronic disease](#)

Figure 7 displays the list of adverse events cases containing the lookup keywords

Page 5: Result (Figure 8)

This page will display the results of either the drug search or the adverse event search in a table format, each item having a unique id to click on that will take you to the Detail Information Page(lightbox)

The main component of the database is in the Results page. Figure 8 shows the Results page for a Search Drugs.

Results Page (Search Drugs Result)

Results: RUXOLITINIB								
Case	Patient	Seq	Code	Ingredient	Route	Dose	Dechallenge	Rechallenge
123224412	09/10/2015 F 46 YR 138 LB US	4	SS	RUXOLITINIB	Oral		Y	
123240642	09/10/2015 F 46 YR 138 LB US	4	SS	RUXOLITINIB	Oral	15 MG, BID		
123516933	05/19/2016 DE	2	SS	RUXOLITINIB	Unknown	20MG	D	U
		3	SS	RUXOLITINIB	Unknown	40MG	D	U
		4	SS	RUXOLITINIB	Unknown	NOT PROVIDED	D	U
125812022	02/08/2016 M 59 YR 256 LB US	4	SS	RUXOLITINIB	Oral	10 MG, BID		
		5	SS	RUXOLITINIB	Oral	1 DF, QD		
		6	SS	RUXOLITINIB	Oral	15 MG, BID		

Figure 8 displays the Results table for Search Drugs, with each Primary ID (primary key) , date, age, sex, weight, and country.

Page 6 Detail information

If you click on one of the primary ID of this page, you will be sent to the Detailed Information page as shown in Figure 8. It is page is a lightbox that appears when you click on the case id. In Figure 9, queries were used along with HTML code to show every drug, along with its main ingredient, patient info, and reactions associated with each primary ID. The Search Adverse Events works the same way as Search Drugs and it should follow the same process.

Case: 123224412

Patient Information:

Date	Age	Sex	Weight	Country
09/10/2015	46.60 YR	F	138.01 LB	US

Drug(s):

Name	Active Ingredient	Route	Dose	Form	Start	End	PT
RIVAROXABAN	RIVAROXABAN	Oral		TABLET		20150912	Product used for unknown indication
RIVAROXABAN	RIVAROXABAN	Oral		TABLET	20150919		Product used for unknown indication
REGORAFENIB	REGORAFENIB	Oral		TABLET	20150303	20150911	Colon cancer
RUXOLITINIB	RUXOLITINIB	Oral		TABLET	20150303	20150911	Colon cancer

Adverse Event(s):

- Arthritis infective
- Embolism
- Oedema peripheral

Outcome(s):


- HO

Print

Figure 9 shows the patient info, drugs taken, reactions along other details that correspond to the ID "123224412".

Page 7: User page (Figure 10)

This is an **Admin** only page in which the user can see a list of all currently registered users and whether each user is an **Admin** or a **Regular** user.



The screenshot shows the FAERS web application interface. At the top left is the 'FAERS' logo. At the top right, there is a user greeting 'Hello, [redacted]@outlook.com!' and a 'Log out' button. Below the greeting are three navigation links: 'Drug Names', 'Adverse Events', and 'Users'. The 'Users' link is highlighted. The main content area displays a table with the following data:

Email	Role	Delete
[redacted]@gmail.com		X
[redacted]@gmail.com	user	X
[redacted]@outlook.com	admin	X
[redacted]@mail.usf.edu	admin	X

Figure 10 displays the Users Page with the list of the registered users and their roles

Database Tables

FAERS Tables

1. **Demographic** (primaryid, caseid, caseversion, i_f_cod, event_dt, mfr_dt, init_fda_dt, fda_dt, rept_cod, auth_num, mfr_num, mfr_sndr, lit_ref, age, age_cod, age_grp, sex, e_sub, wt, wt_cod, rept_dt, to_mfr, occp_cod, reporter_country, occr_country)
 - Primary key: primaryid
 - Foreign key: N/AHolds demographic information of each case, such as the patient's age, sex, country, etc.
2. **Reaction** (primaryid, caseid, pt, drug_rec_act)
 - Primary key: N/A
 - Foreign key: Demographic(primaryid)Holds the reactions that each case has experienced, like nausea for example.
3. **Outcome** (primaryid, caseid, outc_cod)
 - Primary key: N/A
 - Foreign key: Demographic(primaryid)Holds the outcomes for each patient case.
4. **Drug** (primaryid, caseid, drug_seq, role_cod, drugname, prod_ai, val_vbm, route, dose_vbm, cum_dose_chr, cum_dose_unit, dechal, rechal, lot_num, exp_dt, nda_num, dose_amt, dose_unit, dose_form, dose_freq)
 - Primary key: primaryid, drug_seq
 - Foreign key: Demographic(primaryid)Holds the drug information for each case, such as the name of drug, and doses consumed.
5. **Report_Sources** (primaryid, caseid, rpsr_cod)
 - Primary key: N/A
 - Foreign key: Demographic(primaryid)Holds the report sources of each case.
6. **Therapy** (primaryid, caseid, dsq_drug_seq, start_dt, end_dt, dur, dur_cod)
 - Primary key: N/A
 - Foreign key: N/AHolds the duration of each patient's drug test.
7. **Indication** (primaryid, caseid, indi_drug_seg, indi_pt)
 - Primary key: N/A
 - Foreign key: N/AHolds the indication of each case.

Tables Used for Web Application

1. **UserProfile** (UserId, email)
 - Primary key: UserId
 - Foreign key: N/AHolds the UserProfile(email) of each user.

2. **Membership** (UserId, CreateDate, ConfirmationToken, IsConfirmed, LastPasswordFailureDate, PassWordFailuresSinceLastSuccess, Password, PasswordChangedDate, PasswordSalt, PasswordVerificationToken, PasswordVerificationTokenExpirationDate)
 - Primary key: N/A
 - Foreign key: UserId (UserProfile)Holds the password of each user.
3. **OAuthMembersip** (Provider, ProviderUserId, UserId)
 - Primary key: Provider, ProviderUserId
 - Foreign key: UserId (UserProfile)Holds Provider Information for each user.
4. **Roles** (RoleId, RoleName)
 - Primary key: RoleId
 - Foreign key: N/AHolds the role types that a user can have (User and Admin)
5. **UsersInRoles** (UserId, RoleId)
 - Primary key: UserId, RoleId
 - Foreign key: UserId (UserProfile), RoleId (Roles)Holds the user with the user's assigned role

Entity-Relationship Diagram (Figure 11)

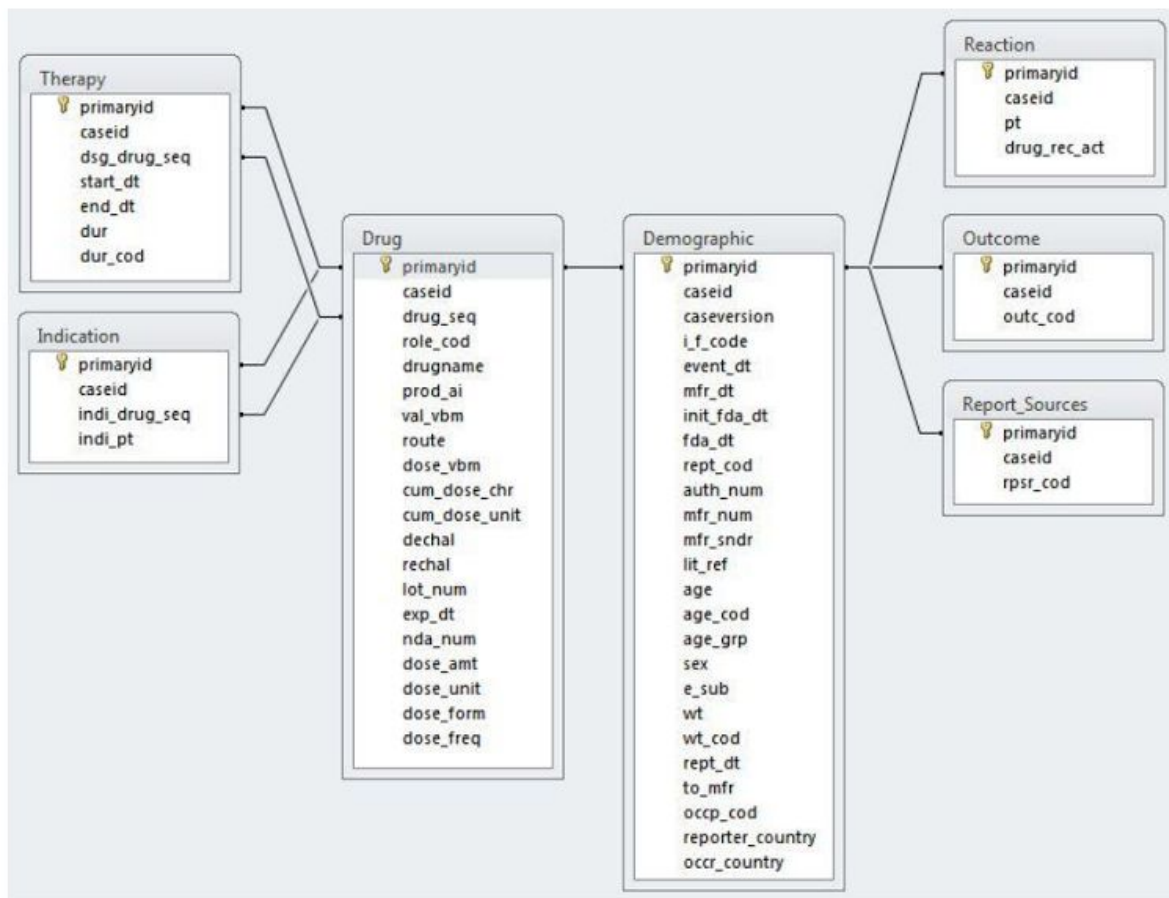


Figure 11: Entity Relationship Diagram

SQL Queries:

Create Tables and Relation Queries

Once the database was created, we used a Create Tables query for each Table. The following is one of the queries used to create the tables in the FAERS database and build its relations, as seen in Figure 12 down below.

```

-- 2. Create Reaction Table

IF OBJECT_ID('Reaction', 'U') IS NOT NULL
    DROP TABLE Reaction
GO

CREATE TABLE Reaction(
    primaryid BIGINT,
    caseid INT,
    pt VARCHAR(500), |
    drug_rec_act VARCHAR(500)
    CONSTRAINT FK_Reaction_ID FOREIGN KEY (primaryid) REFERENCES Demographic(primaryid)
)

```

Figure 12 displays the create table query for the Reaction table.

Database Initialization Queries

The following SQL queries were used to bulk insert the text files into our FAERS database.

```

USE FAERS
GO

```

-- Load Demographic Data

```

BULK INSERT FAERS.dbo.Demographic
FROM 'C:\Users\Luisman\Desktop\faers_ascii_2016q4\ascii\DEMO16Q4.txt'
WITH (
    FIELDTERMINATOR = '$',
    ROWTERMINATOR = '0x0a',
    FIRSTROW = 2
);

```

-- Load Drug Data

```

BULK INSERT FAERS.dbo.Drug
FROM 'C:\Users\Luisman\Desktop\faers_ascii_2016q4\ascii\DRUG16Q4.txt'
WITH (
    FIELDTERMINATOR = '$',
    ROWTERMINATOR = '0x0a',
    FIRSTROW = 2
)

```

```
);
```

```
-- Load Indication Data
```

```
BULK INSERT FAERS.dbo.Indication
```

```
FROM 'C:\Users\Luisman\Desktop\faers_ascii_2016q4\ascii\INDI16Q4.txt'
```

```
WITH (
```

```
    FIELDTERMINATOR = '$',
```

```
    ROWTERMINATOR = '0x0a',
```

```
    FIRSTROW = 2
```

```
);
```

```
-- Load Outcome Data
```

```
BULK INSERT FAERS.dbo.Outcome
```

```
FROM 'C:\Users\Luisman\Desktop\faers_ascii_2016q4\ascii\OUTC16Q4.txt'
```

```
WITH (
```

```
    FIELDTERMINATOR = '$',
```

```
    ROWTERMINATOR = '0x0a',
```

```
    FIRSTROW = 2
```

```
);
```

```
-- Load Reaction Data
```

```
BULK INSERT FAERS.dbo.Reaction
```

```
FROM 'C:\Users\Luisman\Desktop\faers_ascii_2016q4\ascii\REAC16Q4.txt'
```

```
WITH (
```

```
    FIELDTERMINATOR = '$',
```

```
    ROWTERMINATOR = '0x0a',
```

```
    FIRSTROW = 2
```

```
);
```

```

-- Load Report_Sources Data

BULK INSERT FAERS.dbo.Report_Sources
FROM 'C:\Users\Luisman\Desktop\faers_ascii_2016q4\ascii\RPSR16Q4.txt'
WITH (
    FIELDTERMINATOR = '$',
    ROWTERMINATOR = '0x0a',
    FIRSTROW = 2
);

-- Load Therapy Data

BULK INSERT FAERS.dbo.Therapy
FROM 'C:\Users\Luisman\Desktop\faers_ascii_2016q4\ascii\THER16Q4.txt'
WITH (
    FIELDTERMINATOR = '$',
    ROWTERMINATOR = '0x0a',
    FIRSTROW = 2
);

```

Figure 13 displays the bulk insert queries used in our database.

Display Records

These queries were used to display each record of the selected table, as seen below in Figure # with the Report Sources table in HTML format.

```

var report_sources = db.Query(
    "SELECT *
    FROM Report_Sources
    WHERE primaryid = @0",
    primaryid);

```

Figure 14 displays code that will show all the records contained in the Report Sources table.

To create our database, we created a web configuration code as shown below. This uses the database's name in order to connect to our database in SQL Server. Then in order to create the detailed information page, we also used HTML code combined with SQL queries to search for the reaction, outcome, sources and drug that are connected to each primary ID as seen in Figure 15.

Database Connection Code

```
<configuration>

  <system.web>

    <compilation debug="true" targetFramework="4.5.2"/>

    <httpRuntime targetFramework="4.5.2"/>

  </system.web>

  <connectionStrings>

    <add connectionString="Server=KIM\SQLEXPRESS;Trusted_Connection=True;database=FAERS"

      name="FAERS_db" providerName="System.Data.SqlClient" />

  </connectionStrings>
```

Figure 15: Database Connection Code used to connect the database to the web application.

Search Queries

Below is the code used in the search pages so it would determine whether to search for reactions with the drug name, or search for drugs using a reaction name.

(Code for Search Drugs/Search Adverse Events)

```
// check mode [drug or event]

if (mode == "event") {

  subheader = "Adverse Events";

  sql = "SELECT DISTINCT pt FROM Reaction WHERE CHARINDEX(LOWER(@0), LOWER(pt)) > 0 ORDER BY pt";

} else {

  sql = "SELECT DISTINCT drugname FROM Drug WHERE CHARINDEX(LOWER(@0), LOWER(drugname)) > 0 ORDER BY drugname";

}
```

Say for example, the user selects Search Drug. Then we apply this code in Figure # to find the reactions of said drug.

Find Reaction with Drug

```
var drug = db.Query(  
"SELECT *  
  
FROM Drug D FULL OUTER JOIN Therapy T ON T.primaryid = D.primaryid AND T.dsg_drug_seq = D.drug_seq "  
+"FULL OUTER JOIN Indication I ON I.primaryid = D.primaryid AND I.indi_drug_seq = D.drug_seq " +  
  
"WHERE D.primaryid = @0",  
primaryid);
```

If the opposite happens to say for example, the user selects Search Drug. Then we apply this code in Figure # to find the reactions of said drug.

Find Drug with Reaction

```
var drug = db.Query(  
"SELECT *  
  
FROM Drug D FULL OUTER JOIN Therapy T ON T.primaryid = D.primaryid AND T.dsg_drug_seq = D.drug_seq "  
+"FULL OUTER JOIN Indication I ON I.primaryid = D.primaryid AND I.indi_drug_seq = D.drug_seq " +  
  
"WHERE D.primaryid = @0",  
primaryid);
```

Results Code (Find Patient Demographic Using Drugs/Reactions)

```
//get results  
  
sql = "SELECT DISTINCT TOP 100 primaryid FROM Reaction WHERE pt = @0 ORDER BY primaryid";  
  
sql = "SELECT DISTINCT primaryid, drugname FROM Drug WHERE drugname = @0 ORDER BY primaryid";  
  
var patient_query = db.QuerySingle("SELECT primaryid, event_dt, rept_cod, age, age_cod, sex, wt, wt_cod, occr_country  
  
FROM Demographic  
  
WHERE primaryid = @0",  
row.primaryid);  
  
if (mode == "event") { sql = "SELECT primaryid, drug_seq, role_cod, drugname, prod_ai, route, dose_vbm, dechal, rechal FROM Drug WHERE primaryid = @1 ORDER BY drug_seq";
```

```

}else {sql = "SELECT primaryid, drug_seq, role_cod, drugname, prod_ai, route, dose_vbm, dechal, rechal " +
"FROM Drug WHERE drugname = @0 AND primaryid = @1 ORDER BY drug_seq";

}

```

Figure 16 shows the queries to look up information within FAERS database

View Queries

The following queries were used to create the views that you see in our web application. These were created using SQL as seen below with Drug View, Patient Info View and User View. First we ran a query that would enable our database to create those views. Then we added the content that would be shown in each view by running specific queries.

```

-- Run this query to create all Views needed to run website
USE FAERS
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

```

Drug View

```

-- check if already exists and remove

IF OBJECT_ID('Drug_Info', 'V') IS NOT NULL

    DROP VIEW Drug_Info;

GO

-- create new view

CREATE VIEW Drug_Info AS

    SELECT D.primaryid, D.caseid, D.drug_seq, D.drugname, D.prod_ai, D.val_vbm, D.route, D.dose_vbm,
D.dose_form, D.rechal, I.indi_pt, T.dur, T.dur_cod,

        dbo.format_date(T.start_dt) start_dt, T.start_dt original_start_dt, dbo.format_date(T.end_dt)
end_dt, end_dt original_end_dt

    FROM Drug D FULL OUTER JOIN Therapy T ON T.primaryid = D.primaryid AND T.dsg_drug_seq =
D.drug_seq

        FULL OUTER JOIN Indication I ON I.primaryid = D.primaryid AND I.indi_drug_seq
= D.drug_seq

GO

```

Patient Info View

```

-- check if already exists and remove
IF OBJECT_ID('Patient_Info', 'V') IS NOT NULL
    DROP VIEW Patient_Info;
GO

-- create new view
CREATE VIEW Patient_Info AS
SELECT primaryid, caseid, caseversion, i_f_code, dbo.format_date(event_dt) event_dt,
       event_dt original_event_dt, -- keep original event date in separate column as given in data
       dbo.format_age(age, age_cod) age,
       age original_age, -- keep original age in separate column as given in data
       age_cod, age_grp, sex, dbo.format_weight(wt, wt_cod) wt, wt original_wt, -- keep original weight in
separate column as given in data
       wt_cod, dbo.format_date(rept_dt) rept_dt, rept_dt original_rept_dt,
       occp_cod, occr_country
FROM Demographic
GO

```

User View

```

IF OBJECT_ID('Users', 'V') IS NOT NULL
    DROP VIEW Users;
GO

CREATE VIEW Users AS
SELECT U.UserId, U.Email, R.RoleName, R.RoleId
FROM UserProfile AS U LEFT OUTER JOIN
    webpages_UsersInRoles AS UR ON U.UserId = UR.UserId LEFT OUTER JOIN
    webpages_Roles AS R ON UR.RoleId = R.RoleId
GO

```

Figure 17 shows the queries used to create the view shown in our web application and the query used to enable their creation.

User Control Queries

These queries were used to create the user roles (Admin and Regular), assign a role to a user, and the ability for admins to delete other regular users.

Create Roles

```

-- Run this script to insert Roles into webpages_Roles
INSERT INTO webpages_Roles (RoleName)
VALUES ('admin'), ('user');

```

Admin Role Designation

```

-- Run this script to make your UserID an admin
INSERT INTO webpages_UsersInRoles (UserId, RoleId)
VALUES (( [TYPE USERID HERE] ), ( SELECT RoleId FROM webpages_Roles WHERE RoleName = 'admin' ));

```

Regular User Deletion

-- Run this script so that admin users can delete other users

```
ALTER TABLE webpages_UsersInRoles
DROP CONSTRAINT fk_UserId
GO
ALTER TABLE webpages_UsersInRoles
ADD CONSTRAINT fk_UserId FOREIGN KEY (UserId)
REFERENCES UserProfile (UserId)
ON DELETE CASCADE
GO
```

Figure 18 shows the queries used to create user roles, assign them to users(Admin) and delete non-Admin users.

Functions

These are functions (Figure 19) that were used to modify the views that appear in our web application.

Format Age

```
CREATE FUNCTION format_age (@age NUMERIC(12,2), @code CHAR(7))
RETURNS NUMERIC(12,2) AS BEGIN
    RETURN(
        CASE @code -- use age_code to convert all ages to years
            WHEN 'DY' THEN @age/365
            WHEN 'WK' THEN @age/52
            WHEN 'MON' THEN @age/12
            WHEN 'HR' THEN @age/8760
            WHEN 'DEC' THEN @age*10
            WHEN 'YR' THEN @age
        END
    )
END
```

Format Weight

```
CREATE FUNCTION format_weight (@weight NUMERIC(14,5), @code CHAR(20))
RETURNS NUMERIC(14,2) AS BEGIN
    RETURN(
        CASE @code -- convert all weight to Kilograms
            WHEN 'LBS' THEN @weight / 2.20462
            WHEN 'KG' THEN @weight
        END
    )
END
```

Format Date

```
CREATE FUNCTION format_date (@date CHAR(8))
```

RETURNS DATE AS BEGIN

```
        RETURN(  
            CASE LEN( LTRIM(RTRIM(@date)) ) -- Trim column and get length of value  
                WHEN 6 THEN TRY_CONVERT(DATE, LTRIM(RTRIM(@date)) + '01', 101) -- when only  
year and month is available, add day and convert to date format  
                ELSE TRY_CONVERT(DATE, @date, 101) -- convert to date format  
            END  
        )  
END
```

Figure 19 shows the functions used to modify certain attributes such as age, weight and date, to be viewed properly in the Views.