Final project

on

Introduction to Database system and processing

A final report submitted to the

Cleveland State University

Master of Science

in

Computer Science

Submitted on May'24

by

Aditya Sairam Pullabhatla, Mithul Krishna Bheemaneni Sekar

2863159, 2899485

a.pullabhatla@vikes.csuohio.edu, m.bheemanenisekar@vikes.csuohio.edu

Under the esteemed guidance of

***Prof* Aditi Singh**

Department of EECS

1. INTRODUCTION

- **Briefly introduce the Friend Book Social Network System.**

A contemporary social networking platform is called Friend Book. It makes it easier for consumers to keep in touch with their loved ones. The core of Friend Book is the Contact List Manager. This makes managing contacts easier for users. Friend Book's contact organization feature serves as the cornerstone for users to grow their networks. When creating an account, members can choose to import contacts from email accounts or phone lists. You can also fill in your phone number. Then, this function stores and manages a member's links along with data like phone numbers, email addresses, geographical coordinates, and more.

- **Emphasize the importance of the Contact List Management Subsystem.**

A solid system for managing contacts is essential to the success of any social network. Friend Book makes it easy for consumers to stay in touch with their pals in one location as a result. This makes it easier for your social network to grow naturally. The more relationships you make on Friend Book, the better your experience will be. You can meet new people there and reconnect with old pals. Your buddies can also expose you to fascinating groups and material that can suit your interests. In short, users are able to create friends on Friend Book thanks to the Contact List Management Subsystem. Your experience on Friend Book can be enjoyable and socializing thanks to its user-friendly interface. This essential tool enables users to maximize the benefits of Friend Book. Furthermore, as Friend Book expands to service millions of users, improving the system will always be a primary priority.

2. SYSTEM SPECIFICATIONS:

- **Outline user account creation details (Userid, Personal Profile)**

User Account Creation:

UserID - Primary key to uniquely identify each user.

Fname - Stores first name of user

Lname - Stores last name of user

DOB - Stores date of birth of user

Gender - Stores gender of user

Email - Stores email ID of user

Phone - Stores phone number of user

Address - Stores home address of user

ContactName – stores the name of the contact of users.

- **Define the structure of the contact List and its relationship with users.**

Structure of the Contact List:

ContactID - Primary key to uniquely identify each contact record.

UserID - Foreign key to link a contact to the Users table.

Relationship with Users:

The UserID in Contact table refers to the UserID in the Users table.

This establishes a one-to-many relationship between Users and Contact

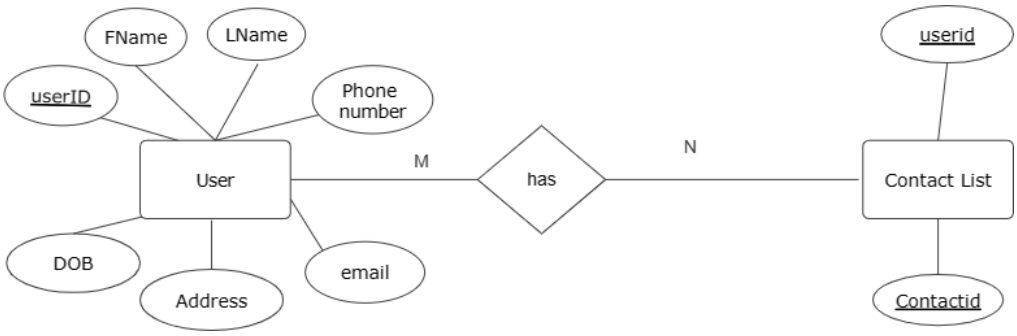
One user can have multiple contacts (stored as multiple records in Contact table)

The foreign key ensures referential integrity between the two tables.

3. DESIGN PHASE:

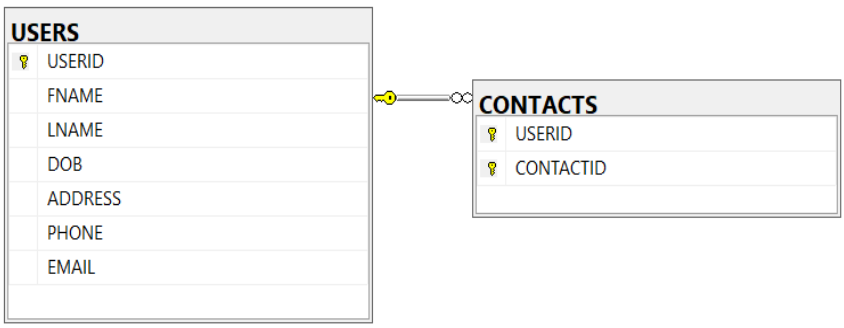
- **ER Diagram: entity relationship**

A particular kind of flowchart called an entity relationship (ER) diagram shows the relationships between "entities"—people, things, or concepts—within a system.

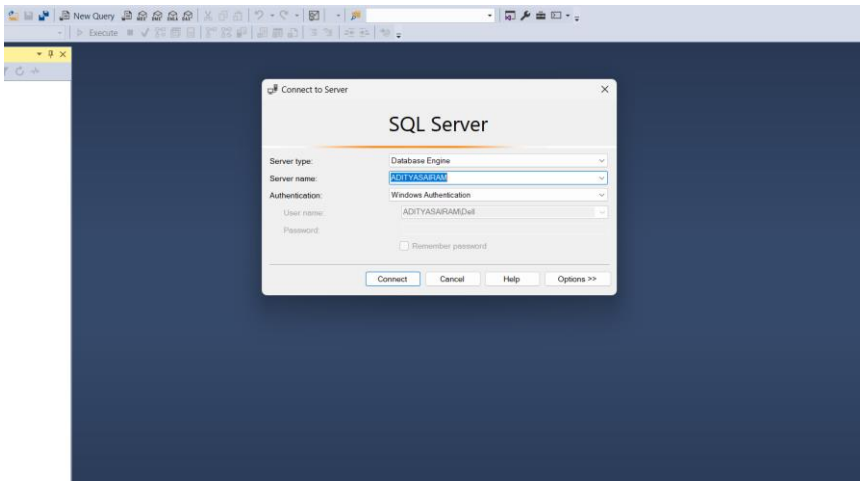


- **CLASS DIAGRAM:**

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity.



Connecting the server

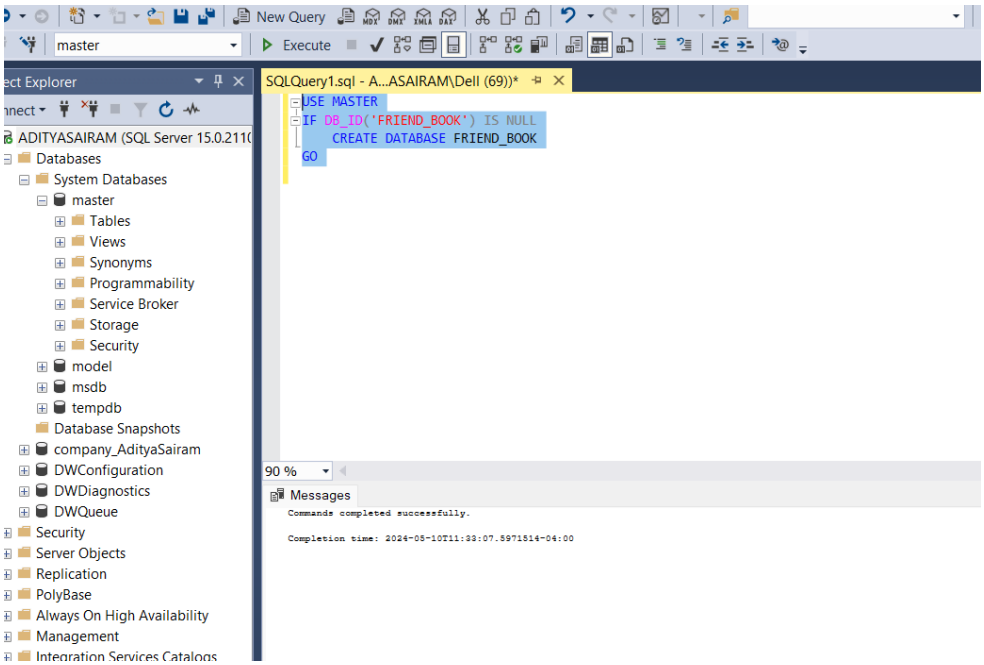


4. DATABASE DESIGN

- **Create table schemas for User and Contact List.**
- **Demonstrate M-N relationships with 3-4 tuples.**
- **Define constraints, including primary and foreign keys.**

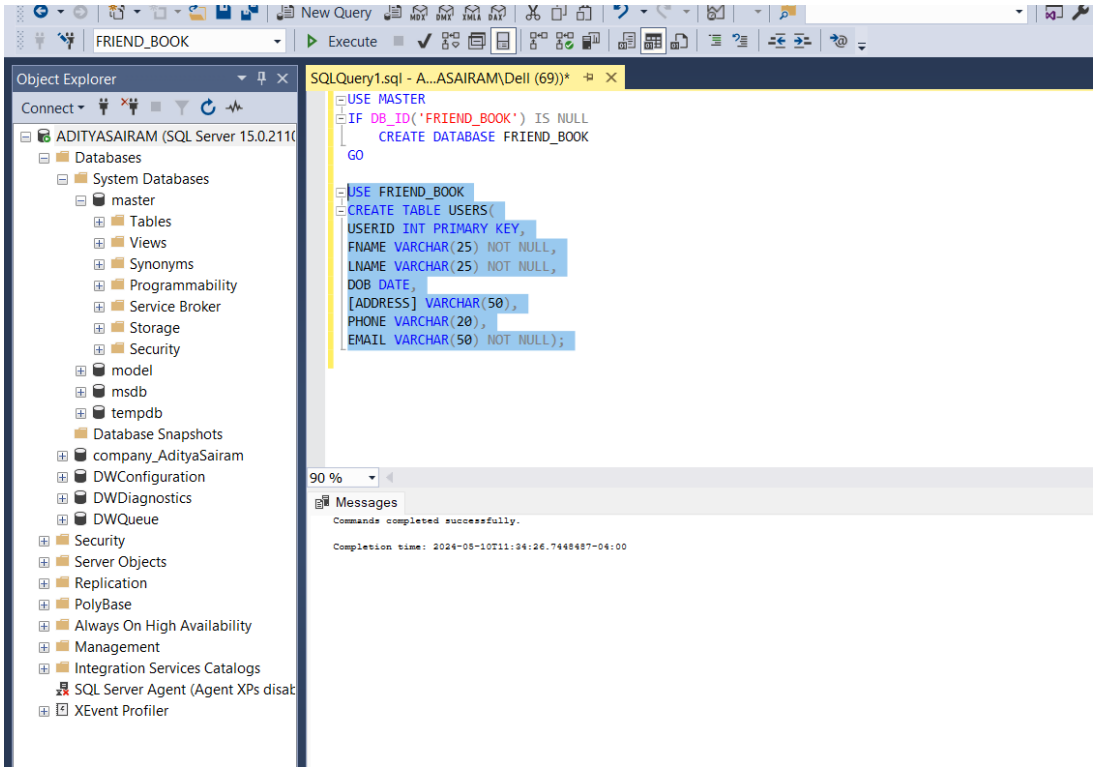
Create database

```
USE MASTER
IF DB_ID('FRIEND_BOOK') IS NULL
        CREATE DATABASE FRIEND_BOOK
GO
```

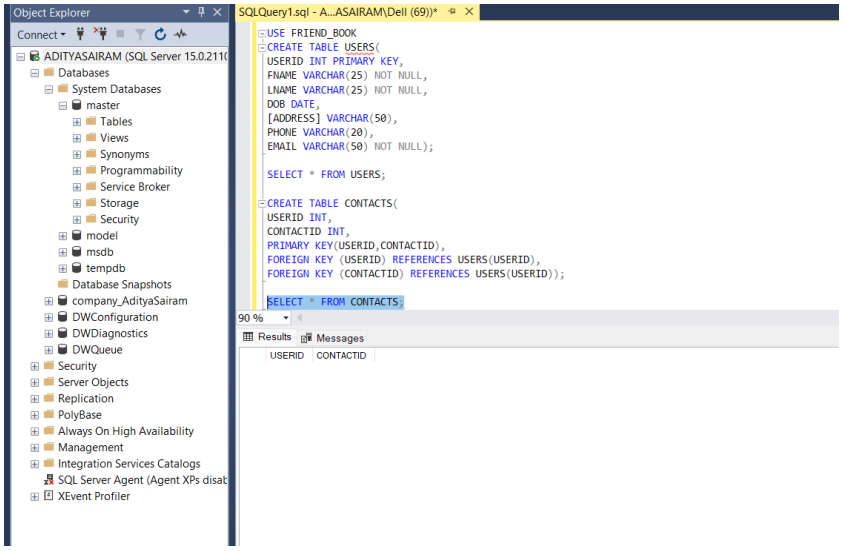


Create tables and add constraints

```
USE FRIEND_BOOK
CREATE TABLE USERS(
USERID INT PRIMARY KEY,
FNAME VARCHAR(25) NOT NULL,
LNAME VARCHAR(25) NOT NULL,
DOB DATE,
[ADDRESS] VARCHAR(50),
PHONE VARCHAR(20),
EMAIL VARCHAR(50) NOT NULL);
```



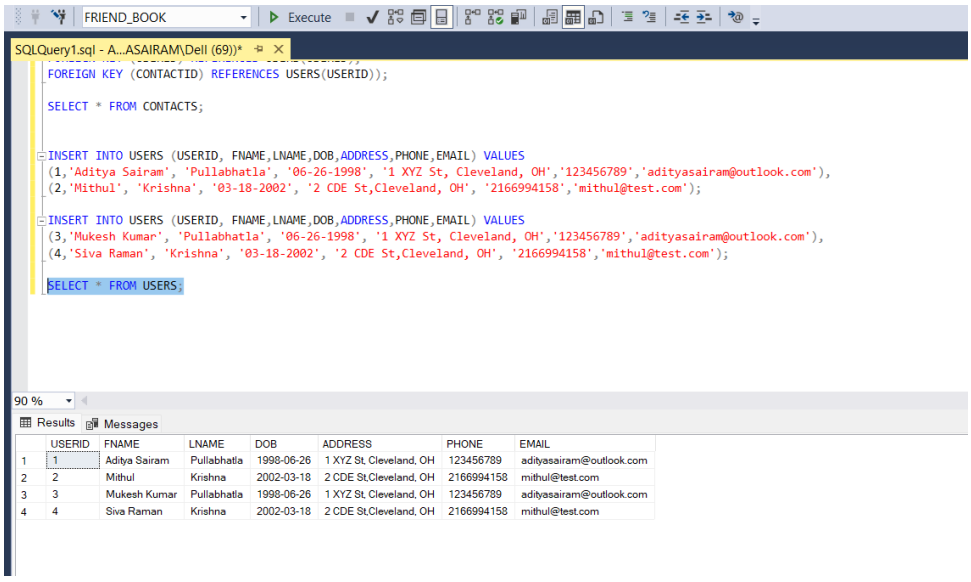

```
CREATE TABLE CONTACTS(
USERID INT,
CONTACTID INT,
PRIMARY KEY(USERID,CONTACTID),
FOREIGN KEY (USERID) REFERENCES USERS(USERID),
FOREIGN KEY (CONTACTID) REFERENCES USERS(USERID));

SELECT * FROM CONTACTS;
```



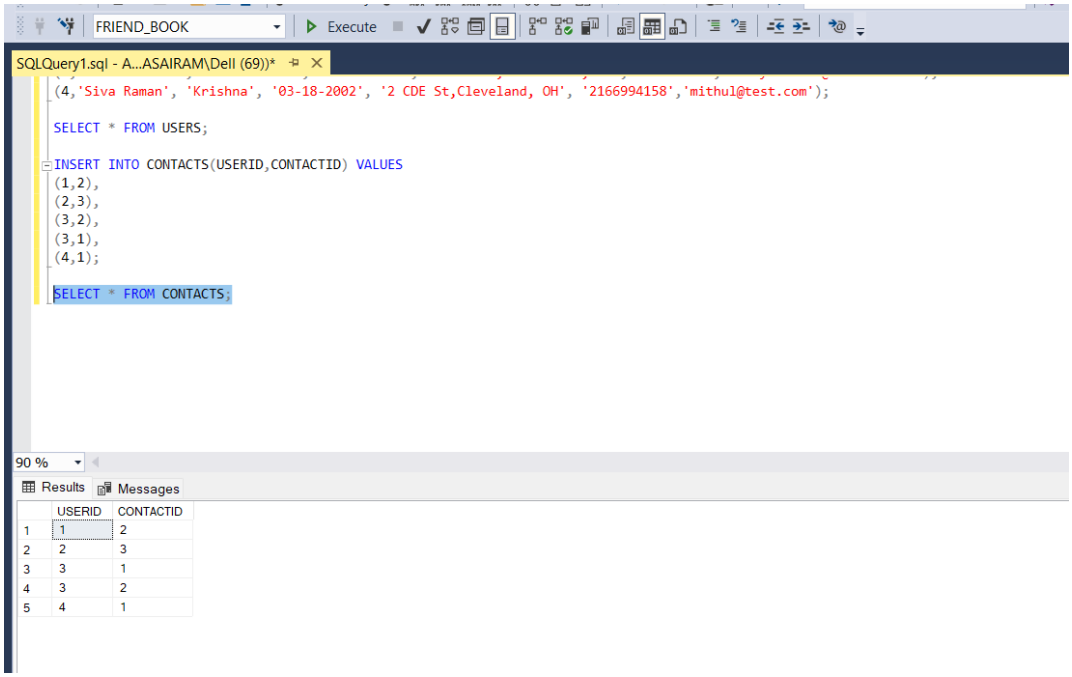
Inserting values

```
INSERT INTO USERS (USERID, FNAME,LNAME,DOB,ADDRESS,PHONE,EMAIL) VALUES
(1,'Aditya Sairam', 'Pullabhatla', '06-26-1998', '1 XYZ St, Cleveland,
OH','123456789','adityasairam@outlook.com'),
(2,'Mithul', 'Krishna', '03-18-2002', '2 CDE St,Cleveland, OH', '2166994158','mithul@test.com');
(3,'Mukesh Kumar', 'Pullabhatla', '06-26-1998', '1 XYZ St, Cleveland,
OH','123456789','adityasairam@outlook.com'),
(4,'Siva Raman', 'Krishna', '03-18-2002', '2 CDE St,Cleveland, OH', '2166994158','mithul@test.com');
```

```
SELECT * FROM USERS;
```



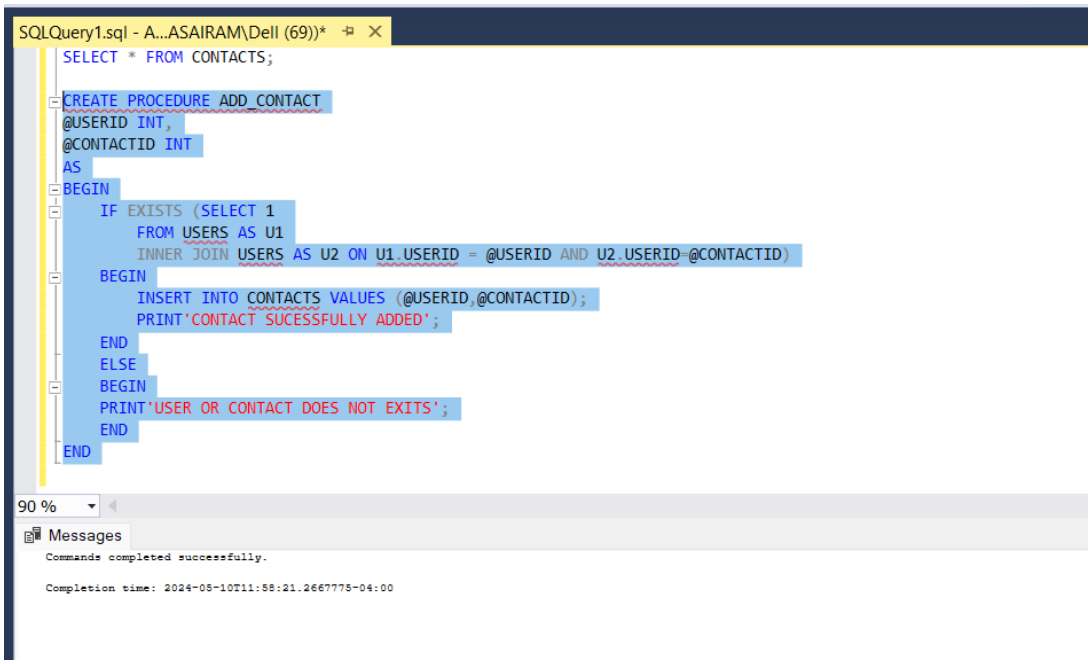
```
INSERT INTO CONTACTS(USERID,CONTACTID) VALUES
(1,2),
(2,3),
(3,2),
(3,1),
(4,1);

SELECT * FROM CONTACTS;
```



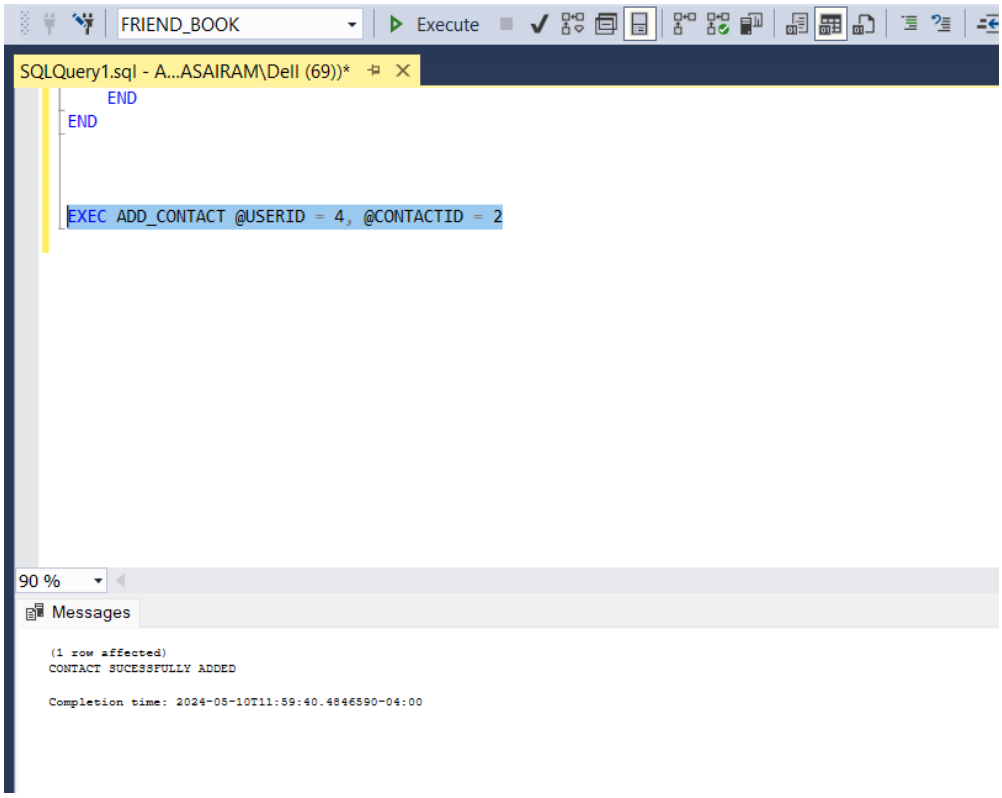
ADD CRUDE(Create, Retrieve, Update, Delete, Execute) Functions for Contact List Management

in the User Class.

```
CREATE PROCEDURE ADD_CONTACT
@USERID INT,
@CONTACTID INT
AS
BEGIN
	IF EXISTS (SELECT 1
      FROM USERS AS U1
		INNER JOIN USERS AS U2 ON U1.USERID = @USERID AND
U2.USERID=@CONTACTID)
	BEGIN
		INSERT INTO CONTACTS VALUES (@USERID,@CONTACTID);
		PRINT'CONTACT SUCESSFULLY ADDED';
	END
	ELSE
	BEGIN
	PRINT'USER OR CONTACT DOES NOT EXITS';
	END
END
```

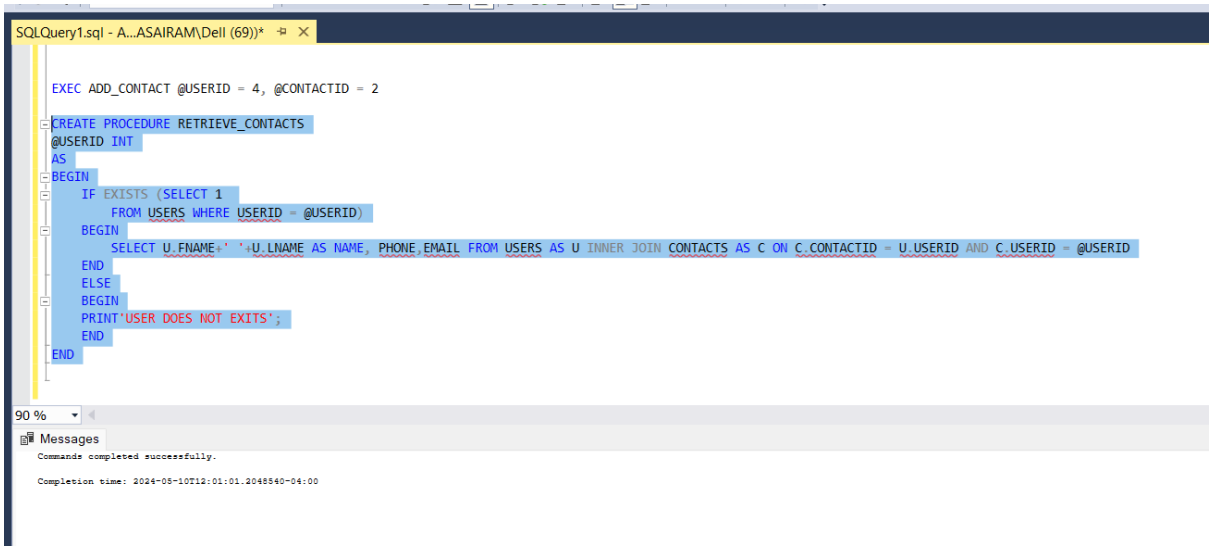


EXEC ADD_CONTACT @USERID = 4, @CONTACTID = 2

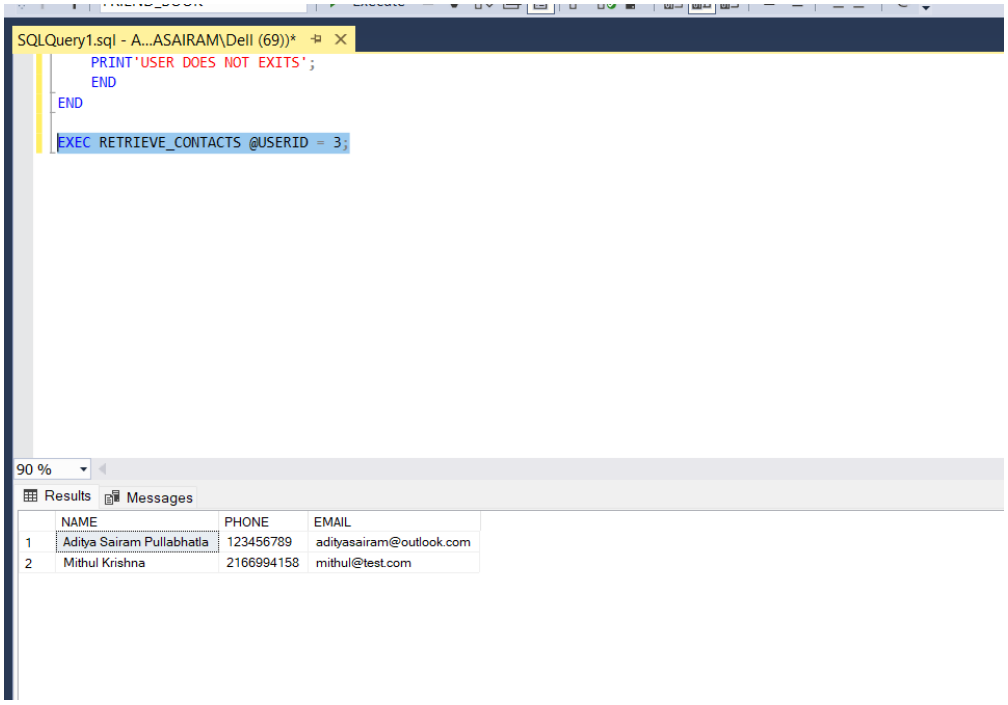



```
CREATE PROCEDURE RETRIEVE_CONTACTS
@USERID INT
AS
BEGIN
        IF EXISTS (SELECT 1
    FROM USERS WHERE USERID = @USERID)
        BEGIN
```

```
            SELECT U.FNAME+' '+U.LNAME AS NAME, PHONE,EMAIL FROM USERS
AS U INNER JOIN CONTACTS AS C ON C.CONTACTID = U.USERID AND C.USERID =
@USERID
        END
        ELSE
        BEGIN
        PRINT'USER DOES NOT EXITS';
        END
END
```

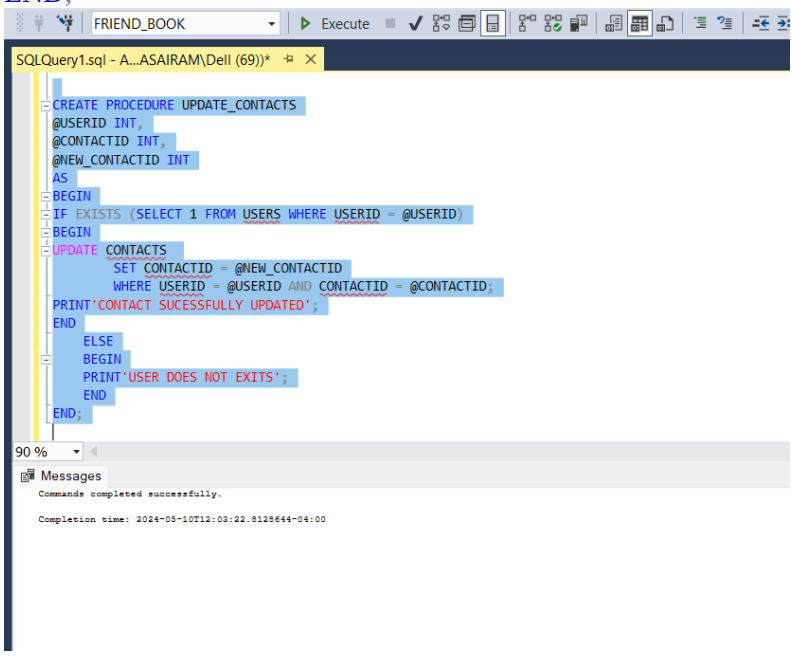



```
EXEC RETRIEVE_CONTACTS @USERID = 3;
```

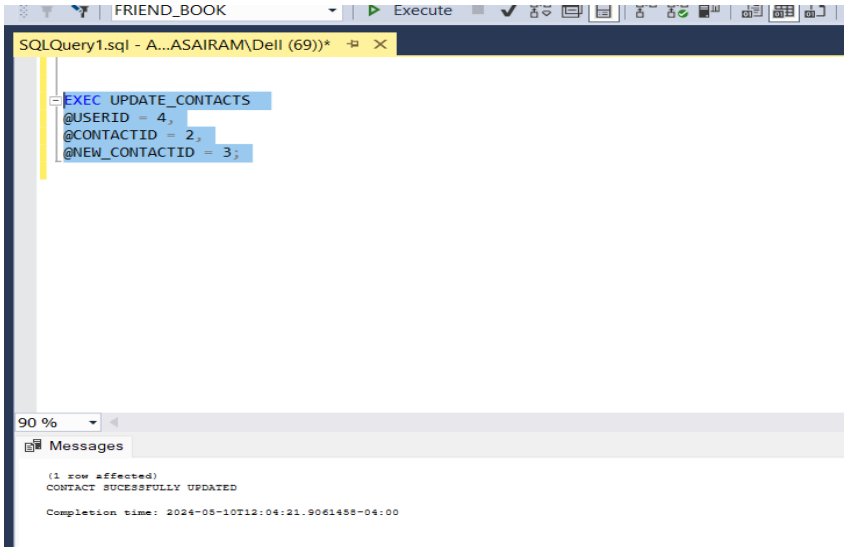



```
CREATE PROCEDURE UPDATE_CONTACTS
@USERID INT,
@CONTACTID INT,
@NEW_CONTACTID INT
AS
```

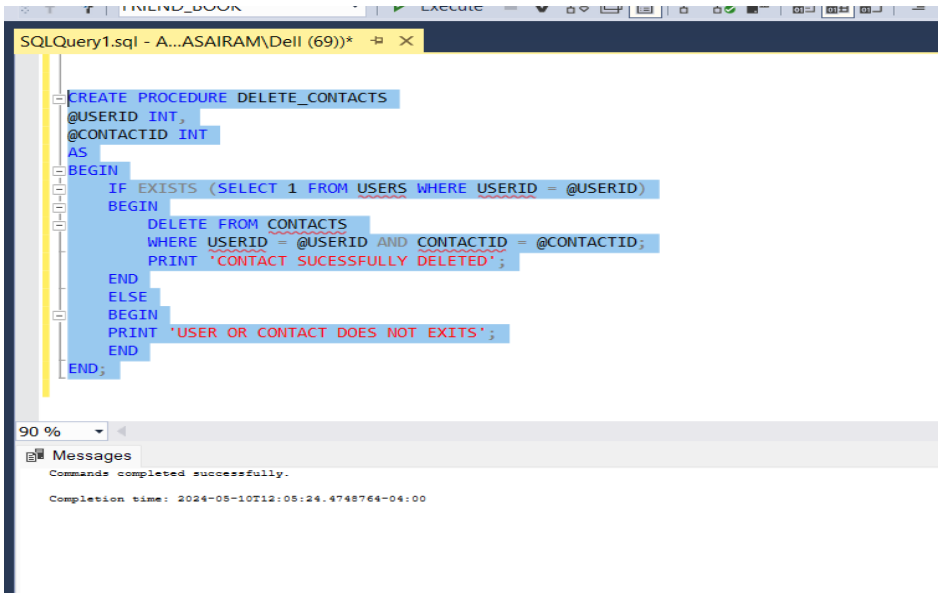
```
BEGIN
IF EXISTS (SELECT 1 FROM USERS WHERE USERID = @USERID)
BEGIN
UPDATE CONTACTS
                SET CONTACTID = @NEW_CONTACTID
                WHERE USERID = @USERID AND CONTACTID = @CONTACTID;
PRINT'CONTACT SUCESSFULLY UPDATED';
END
        ELSE
        BEGIN
        PRINT'USER DOES NOT EXITS';
        END
END;
```



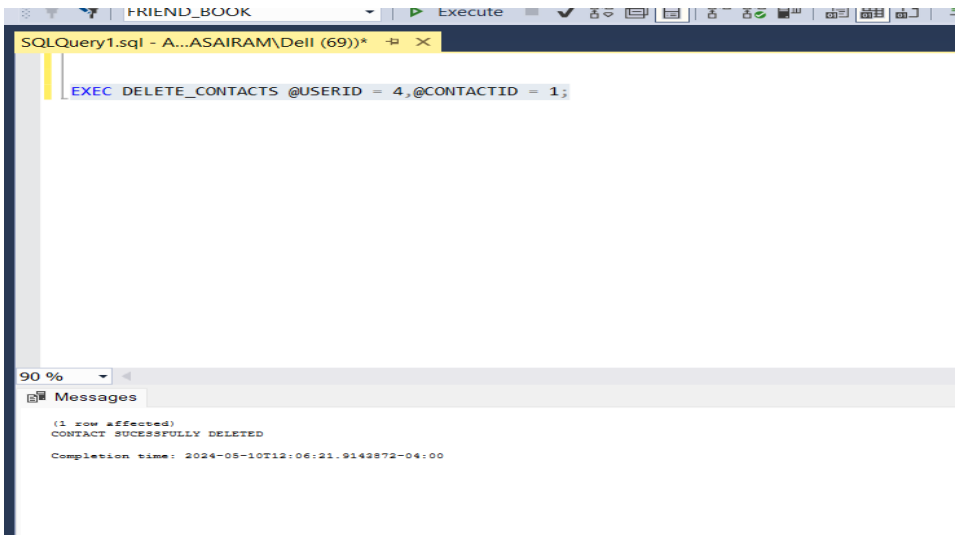

```
EXEC UPDATE_CONTACTS
@USERID = 4,
@CONTACTID = 2,
@NEW_CONTACTID = 3;
```



```
CREATE PROCEDURE DELETE_CONTACTS
@USERID INT,
@CONTACTID INT
AS
BEGIN
	IF EXISTS (SELECT 1 FROM USERS WHERE USERID = @USERID)
	BEGIN
		DELETE FROM CONTACTS
		WHERE USERID = @USERID AND CONTACTID = @CONTACTID;
		PRINT 'CONTACT SUCESSFULLY DELETED';
	END
	ELSE
	BEGIN
	PRINT 'USER OR CONTACT DOES NOT EXITS';
	END
END;
```




```
EXEC DELETE_CONTACTS @USERID = 4,@CONTACTID = 1;
```



5. TEAM DISTRIBUTION:

Aditya – Analyzing the requirement of the database and sketching, DB sketch and creating ER-diagram, Report – **testing and review**

Mithul – Creating a database, queries, stored procedure, report- **designing**

6. CONCLUSION

The Contact List Management Subsystem of the FriendBook Social Network System was thoughtfully created and engineered with consideration for database architecture, class structures, and usefulness. The Entity-Relationship (E-R) Diagram and Class Diagram provide an example of the relationships between significant system entities, such as User and Contact.

The Contact List Management CRUDE functions, which are defined in the User class, can be used to implement a variety of basic operations, including Create, Retrieve, Update, and Delete contacts. Based on these functions, the user interface and interaction elements of the system have been developed.

Potential future system enhancements could include contact grouping or other actions under the Execute function to boost user satisfaction and engagement.

Emphasize the collaborative effort in creating the FriendBook System:

7. FUTURE WORK

The FriendBook system might be improved in the following ways, given the dynamic nature of social network systems and technical advancements:

Reach out to grouping:

Give users control over creating and managing contact groups. If users could categorize their relationships, they would be able to more easily engage with certain groups within their network and share material more thoughtfully.